Autonomous avatar-based architecture for value-added services provision

Karima Khadir, Nawal Guermouche, Thierry Monteil

Autonomous avatar-based architecture for value-added services provision

Karima KHADIR
LAAS-CNRS, Université de Toulouse, INSA.
Toulouse, France.
kkhadir@laas.fr

Nawal GUERMOUCHE
LAAS-CNRS, Université de Toulouse, INSA.
Toulouse, France.
guermouche@laas.fr

Thierry MONTEIL
LAAS-CNRS, Université de Toulouse, INSA.
Toulouse, France.
monteil@laas.fr

***Abstract*—The Internet of Things (IoT) aims to create a global infrastructure that provides a variety of value-added services resulting from the interconnection of a large number of heterogeneous devices from different domains. Web of Things (WoT) enables to implement this vision by relying on the Web to virtualize and interconnect IoT objects in a transparent way. The IoT systems often operate in dynamic, unpredictable, and mobile environments with limited resources. Therefore, virtualized IoT objects on the Web need autonomous reasoning and decision-making capabilities to adapt their behaviors to their context. In this paper, we propose to extend virtual objects with reasoning mechanisms based on semantic models. The goal is to adapt their actions to the context of their environment. This extension is called *Autonomous Avatar*. It provides an artifact of a real-world entity on the Web while providing multiple features, such as standalone reasoning, context management, and collaboration capabilities. Based on the concept of autonomous avatar, we propose Fog based architecture for IoT Systems. This architecture is a pillar for defining IoT approaches for services delivery.**

***Index Terms*—Avatar, Web of Things, virtualization, autonomous reasoning, Cloud-Fog-device architecture, Semantic, SWRL rules.**

I. Introduction

The Internet of Things (IoT) is a new paradigm that has emerged strongly in recent years thanks to the proliferation of connected devices and the revolution of Information and Communication Technologies (ICT) as well as embedded systems. The main objective of this paradigm is to allow different heterogeneous objects connected to the Internet to interact with each other and to exchange their data in order to cooperate and perform common tasks thus facilitating human life in its various aspects.

IoT objects essentially include RFID (Radio-Frequency IDentification) tags and NFC (Near Field Communication) codes to enable standardized device addressing and identification, sensors for environmental perception, and actuators for actuation. The variety of these objects allows the emergence of many applications in all areas thus allowing to revolutionize the user's lifestyle.

Although the IoT paradigm has been a success, the development of IoT applications raises great challenges. The first is that IoT objects are deployed in highly distributed and uncertain environments, thus undergoing unpredictable changes. They are also endowed with limited resources which considerably impacts the IoT devices performance. Cloud-based solutions are necessary to collect, store and process data. IoT also suffers from another serious problem that is the vertical fragmentation of its market, this problem is due to the diversity of standards used by the manufacturers of connected objects, the different data formats as well as the variety of protocols and communication standards.

The Web of Things (WoT) is a step forward in the IoT. It relies on the Web to offer a high-level representation of IoT objects. Such representation constitutes the concept of Virtual Objects (VO) adopted in several works [1]–[5]. Recently, the W3C consortium gave it the name of *Avatar* [6].

The virtualization makes the use of heterogeneous IoT objects uniform and standardized. This enables interoperability and flexibility across IoT systems. In the literature, the proposed works rely on passive avatars (resp. VO) [2], [3]. In fact, the goal of avatars is limited to expose on the Web the supported functionalities so that the represented objects can be integrated. The management of theses avatars is ensured by a central entity, usually called Manager.

In this paper, we aim to empower the concept of an avatar with autonomous capabilities. This makes avatars active and then able to fulfill intelligent decisions according, for example, to their context. Moreover, endowing avatars with intelligent reasoning presents an opportunity to overcome the limitations of existing works that rely on a central manager that handles and manages passive avatars. Indeed, central manager-based approaches are not scalable and in the case of manager dysfunction, the whole system can be compromised.

This article is organized as follows. Section II analyses the related works. In section III, we present high-level autonomous-based architecture of IoT systems and we detail the concept of autonomous avatar. Before concluding, we expose the reasoning mechanisms based on the semantic descriptions and logical rules that allow the avatar to adopt an autonomous behavior IV.

II. Related works

Many works related to the concept of virtualization of connected objects have been proposed in recent years through scientific research and industrial projects.

The work of Romer and al. in [7] uses the concept of Virtual Meta Counterparts (VMC) to represent unconnected objects tagged by RFID tags. This work is limited compared to the needs of IoT applications since it does not allow automatic discovery of the services provided by the objects object and their collaboration,

In [9], a building automation system where its devices are represented by the DPWS concept (Device Profile for Web Service) is proposed. This work is based on SOA (Service-Oriented Architecture) paradigm to fulfill dynamic service composition according to the context information collected and processed. In this architecture, all DPWSs are orchestrated and managed by a central Building Application Server (BApS) and the DPWS are passive.

The work presented in [10] relies on the concept of *avatar* to represent physical objects on the Web. As described, in this architecture the avatars are passive and the authors tackle the problem of dynamic deployment of avatars. The underlying architecture is suitable for centralized management of avatars.

The european project SENSEI [11], [12] aims to create an open architecture that embodies the vision of the real-world Internet (RWI). This architecture is based on the abstraction of the heterogeneous Wireless Sensor and Actuator Network (WSAN) of the various companies belonging to the consortium in a global framework via entities called resources. A SENSEI resource can represent one or more objects. It exposes their features via universal interfaces. Their solution is limited to the passive representation of IoT objects.

IoT-A (Internet of Things-Architecture) [1], [13] is an European project that also aims to provide a reference architecture and a basic set of building blocks to facilitate the implementation of IoT applications. This project extends the models proposed in the SENSEI project [11], [12] by enriching the description models of the data produced or used by the physical objects by semantic annotations via OWL-DL. These semantic descriptions include contextual information about the device environment, which gives meaning to the raw data of the sensors. A VO, in this project, is designated by a Virtual Entity (VE) rather than Resource as in SENSEI. A physical object can provide one or more services and each service is associated with an VE which. The basic building blocks provided are high-level services ready for use. They are the result of the orchestration of several VEs. The model proposed in this project is very interesting thanks to the use of semantic ontologies but VEs are passive.

iCore [14]–[16] is a framework that leverages the IoT-A project's advanced semantic modeling capabilities to represent real-world devices. The specificity of the defined architecture is that it relies on a cognitive framework to automate the processing of the collected information and the making of the corresponding decisions to the detected changes. The iCore architecture consists of three layers: i) The lower layer is called Virtual Object Level (VOL) which provides virtual representations of real-world objects that can be dynamically created and destroyed. The capabilities of objects are universally exposed to upper layers and other architectures. A VO can abstract several real objects and a real object can be represented by several VOs as in IoT-A. VOs are stored in a common VO register that provides polling functions to find the VO that best matches a request. ii) Virtual Object Level Composite (CVOL) is the layer that provides the cognitive mechanisms for the semantic mash-up of available VOs to create compound services with added value. It is based on the "event/action" logic. Constructed VOCs are also published in a CVO repository. iii) Service Level (SL) is the last layer of the iCore architecture to translate end-user requests and search for the most appropriate CVOs to satisfy them. Cognitive science and semantic reasoning mechanisms are the strength of the iCore architecture because they automate the processing of user requests. However VOs have no reasoning ability to react to internal or external events, and they are just used by another layer.

ETSI M2M and oneM2M: ETSI M2M [17] is a set of technical specifications developed in 2009 by the European Telecommunications Standards Institute (ETSI) to develop and maintain complete IoT/M2M architectures by creating a common service layer. In 2012, ETSI is fully committed to oneM2M's global initiative of seven international standardization organisms [18], which aims to provide a global standard for the construction of IoT service platforms.Within this standard, virtualization of physical objects is represented via RESTful resources. A resource is addressable in a unique way via a URI and it corresponds to a single physical object which gives the association one to one. This standard is earned by providing mechanisms for platforms to be highly extensible to integrate existing and new technologies. It also provides methods for the efficient discovery of resources and subscriptions/notifications. However, like the other works, these resources are passive and cannot be aware of their state to act as a consequence.

To summarize, Table I gives a synthesis of the studied works according to six properties:

- Cardinality of representation : that designates the number of objects assigned to a virtual representation.
- The concept of virtualization: that corresponds to the used abstraction concept
- Semantic description: that considers if the studied work relies on rich semantic descriptions
- The nature of application management: that indicates if the systems management is centralized or decentralized.
- Deployment infrastructure: that defines the architecture deployment environment
- Type of behavior: that corresponds to a passive or active behavior of the virtual representation

Most of the existing works focus only on the interoperability problem of IoT systems in their virtualization proposals. For this, we propose in this work to extend these proposals by highlighting an architecture based on virtual objects that have autonomous reasoning and decision-making capabilities without the intervention of a central third party.

TABLE I
A SYNTHESIS OF THE STUDIED WORKS

Studied work	Cardinality of representation	Concept of virtualization	Semantic Description	Application management nature	Deployment infrastructure	Type of behavior
Romer and al. [7]	One to Many	Virtual Meta Counterparts (VMC)	-	Central	Not defined	-
Mrissa and al. [10]	One to One	Avatar	+	Not defined	Device, Fog and Cloud	+
HAN and al. [9]	One to One	Device Profile for Web Service (DPWS)	+	Central database	Server (location not defined)	-
SENSEI [11], [12]	One to Many	Resource	-	Distributed	Cloud	-
IoT-A [1], [13]	Many to One	Virtual Entity (VE)	+	Distributed	Not defined	-
iCore [14]–[16]	Many to Many	Virtual Object (VO)	+	Central repository	Fog and Cloud	-
ETSI M2M and oneM2M [17], [18]	One to One	Resource	+	Distributed	Fog and Cloud	-

III. AUTONOMOUS AVATAR-BASED ARCHITECTURE FOR IOT SYSTEMS

As stated above, an avatar is defined as a virtual abstraction of a physical device in the Web [6]. It exposes the features provided by this device as services. Its aim is to abstract a concrete object to offer an interoperable interface. In this paper, in addition to exposing object functionalities, an avatar has reasoning capabilities. This enables avatars to handle their collaborations without the intervention of a third party.

Several advantages can be derived from the use of autonomous avatars in IoT systems including:

- Interoperability: the objects of the world are modeled in a uniform and standard way, whatever their nature and their technologies, notably thanks to the use of semantics, which essentially favors their automatic discovery in IoT systems.
- Autonomous behaviour: Reasoning to make decisions without the intervention of other entities promotes efficiency in terms of response time.
- Plug and Play: it represents the process of quickly and automatically detecting devices as soon as they join the network and expose their features and make their avatar active.
- Mobility management: virtualization of mobile objects makes it possible to manage them in a transparent manner regardless of their location, which will only represent contextual information relating to its data.
- Context management: avatars are able to become aware of the context (location, time, etc) in which the physical object operate which improves their functionalities.

Traditional IoT architectures that use the virtualization of real-world objects devote a whole layer above virtual objects for application management. This layer is usually at the Cloud level where the applications are managed in a centralized way so that if a crash occurs at the server where they are deployed, their data and their code will be lost. In addition, there are several limitations because of the distance between the devices and the cloud processing centers, especially in terms of latency, bandwidth and security. The aim of our work is to equip avatars with autonomous behavior and reasoning mechanisms to enable them to collaborate with each other in order to achieve common goals and build complex applications.

The proposed architecture is composed of two logical layers as shown in the Figure 1:

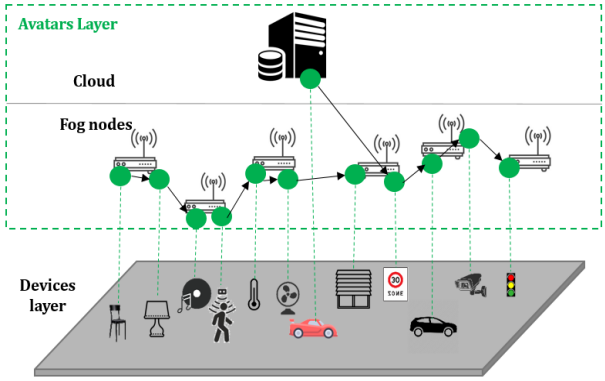

Fig. 1. Autonomous avatar-based Architecture

- Devices Layer: is a collection of real-world devices: sensors, actuators, and unconnected devices tagged with RFID tags connected via several technologies to several gateways that are in turn registered on a server. These devices are heterogeneous, they can be mobile or fixed and may be deployed in a distributed location.
- Avatars Layer: each Device Layer object is represented via an avatar that virtualizes its profile and its features

using semantic descriptions. These avatars are active entities. They have an autonomous behavior that allows them to react following events modeled by changes in their context (location for example) or at the level of their internal data (e.g., detection of an anomaly in the flow of a camera). Autonomic reaction capabilities are related to the use of semantic representation and reasoning mechanisms. Avatars can also talk to each other to form a collaboration in order to solve a common problem. They can be deployed directly at the device level if they have the necessary resources. Also, Fog nodes near the end-users or even the Cloud can be used if the processes to be performed in the avatar are gourmand in resources.

The generic architecture of an avatar is essentially composed of three components as illustrated in Figure 2: i) a Knowledge Base (KB) which contains its data and behaviors modeled as logical rules, ii) a virtual object component that ensures interoperability and iii) a component to model its autonomous behavior in decision-making.

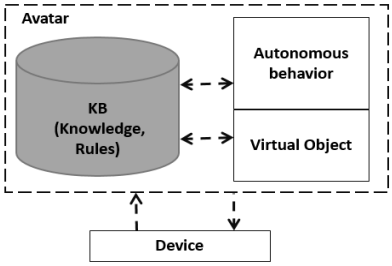

Fig. 2. Autonomous avatar Architecture

IV. Semantic knowledge model

Semantic descriptions are a key element in our work since it enable avatars to interpret their raw data to be understandable. Moreover, inference and reasoning mechanisms can be executed on the data to draw conclusions and decide what action to do by using semantic reasoning rules.

In our model, each avatar has its proposed knowledge base consisting of an instantiated generic ontology that mainly contains a description of the avatar and the features it can perform exposed as services, and a set of rules of type "events/actions" where events represent internal changes of data or context and actions represent a series of actions or tasks to be performed following the events produced at the level of the avatar.

We begin by presenting the ontology proposed in this work and then we detail the reasoning mechanisms based on the rules.

A. Ontologies

An ontology [21] is simply a graph that serves to provide an unambiguous vocabulary and common model about objects, their properties, and their relationships.

The proposed ontology "AvatarOnt", illustrated in the Figure 3, essentially describes the physical entities represented by the avatars and all of their features that are modeled as services. This ontology can be reused in any IoT application domain. For the design of this ontology, we have based on the NeOn [20] methodology which defines two types of requirements: conceptual to present the concepts to integrate into the ontology to create and functional regarding its general structure.

The modules that make up the AvatarOnt ontology are:

- ***A service module***: based on MSM[1] to describe service operations, their inputs and outputs, hRests [22] for service invocation REST methods and WSOnto to express their non-functional part (QoS).
- ***A sensor module***: based on SOSA[2] and IoT-O[3] to describe the sensors and their observations.
- ***An actuator module***: based on SAN[4] and IoT-O to describe the actuators and the actions to be performed on the devices.
- ***An avatar module***: to describe the avatar, its goals, its deployment node, the mobility of the device represented and its location.

Example: A Camera_Avatar description

The following example gives a semantic description of the camera avatar that represents the sensor camera device and exposes its features as services. This avatar has a purpose modeled as a process that is the detection and treatment of driver tiredness.

```
<owl:NamedIndividual rdf:about="avataront;Camera_Avatar">
  <rdf:type rdf:resource="&avataront;Avatar"/>
  <realizes rdf:resource="&DEMISA;Tiredness_Process"/>
  <represents rdf:resource="&ssn;Camera"/>
  <hasService rdf:resource="&msm;Get_Tiredness_Driver"/>
</ /owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&avataront;Camera">
  <rdf:type rdf:resource="&ssn;Sensor"/>
  <hasObservation rdf:resource="&ssn;Driver_Tiredness"/>
</ /owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&ssn;Driver_Tiredness">
  <rdf:type rdf:resource="&ssn;Observation"/>
  <hasSensorOutput rdf:about ="&ssn;Eyes_Inclination"/>
</ /owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&msm;Get_Tiredness_Driver">
  <rdf:type rdf:resource="&msm;Service"/>
</ /owl:NamedIndividual>
```

Each avatar has a set of objectives to achieve, modeled as a process. A process is defined as a continuous series of correlated actions that has dependencies in their execution.

An objective can be broken down into several tasks that can be complex or atomic. Several approaches have been proposed for this purpose. The abstract description of the workflow by

[1]http://iserve.kmi.open.ac.uk/ns/msm
[2]http://www.w3.org/ns/sosa/
[3]https://www.irit.fr/recherches/MELODI/ontologies/IoT-O
[4]http://www.w3.org/ns/ssn/

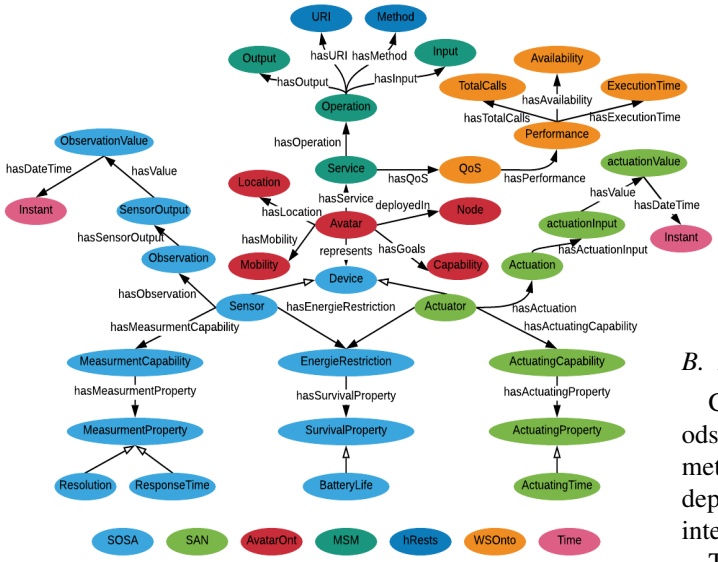

Fig. 3. AvatarOnt ontology

the designer is one of the most common approaches given its reduced cost. To do this, we use an ontology that describes the different composite and atomic tasks that make up the application to be performed and the logical dependencies between them. The semantic description of the objectives to be achieved provides an interoperable framework for understanding the messages exchanged between them during their collaboration. We use the same ontology proposed by Tiezt and al [23] while adding a Task related node to designate the process state see: Waiting, Running or Pausing, as shown in the Figure4.

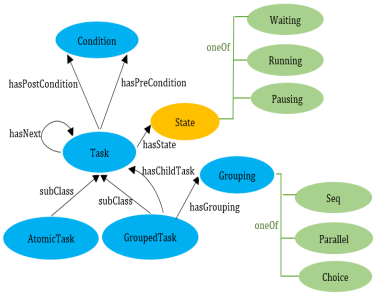

Fig. 4. DEMISA Task ontology [23]

Example: Tiredness_Process description

In this example, we describe a process that triggers once tiredness is detected. This process is composed of two atomic tasks that can performed in parallel; a vibration task that needs to receive an ON command as a precondition and a Message task that requires the text to be displayed to the driver.

<owl:NamedIndividual rdf:about="&DEMISA;Tiredness_Process">
<rdf:type rdf:resource="&DEMISA;Task"/ >
<hasGrouping rdf:resource="&DEMISA;Parallel"/ >
<hasChildTask rdf:resource="&DEMISA;Vibration"/ >
<hasChildTask rdf:resource="&DEMISA;Message"/ >
< /owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&DEMISA;Vibration">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/ >
<hasPreCondition rdf:resource="&DEMISA;op=ON"/ >
< /owl:NamedIndividual>
<owl:NamedIndividual rdf:about="&DEMISA; Message">
<rdf:type rdf:resource="&DEMISA;AtomicTask"/ >
<hasPreCondition rdf:resource="&DEMISA;op=ON&text='Attention Assist: Pause!'"/ >
< /owl:NamedIndividual>

B. Reasoning mechanisms

Concerning reasoning mechanisms, we have turned to methods based on reactive rules driven by events because these methods have proved their effectiveness in distributed systems deployed in uncertain environments, particularly in artificial intelligence (AI) and Multi-Agents Systems (MAS).

The purpose of this type of rules is to detect events in order to allow automatic reactions. They are of type $E1 \wedge E2 \wedge ... \wedge En \rightarrow Pi$, where the first part of rule represents a conjunction (or a disjunction) of events and the second part Pi represents the process to be accomplished as a result of events.

Example:

This example shows the rule which allows to trigger the TirednessProcess and which has as antecedent: the detection of driver eyes inclination by a sensor intended to observe the driver tiredness.

$$Sensor(?s) \wedge hasObservation(?s, DriverTiredness) \wedge hasSensorOutput(DriverTiredness, Eyes\_Inclinatio) \rightarrow hasState(TirednessProces, Waiting)$$

The reasoning and decision-making within an avatar are provided essentially by three components illustrated in the Figure 5 and detailed below:

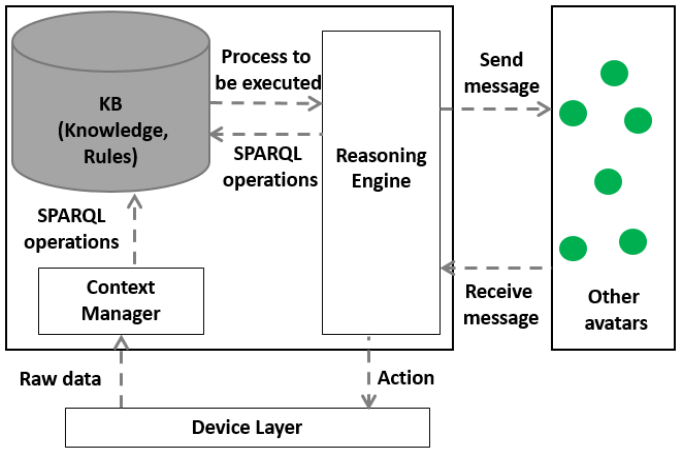

Fig. 5. Reasoning modules

- Context Manager: it allows to permanently monitor the changes that may occur in the the avatar's environment or at the device level represented by him regarding changes

in his state or data. The contextual information we are interested in is: quality of service, device features, device location, mobility, and availability. Once the changes are detected, this manager is responsible for updating the knowledge base via SPARQL operations (create, update and delete).

- Knowledge Base (KB): it consists of the instantiated AvatarOnt ontology, a set of logical rules describing the behavior of the avatar in response to events and a cache to store the latest messages received or sent by the avatar. All of these three elements constitute the knowledge of the avatar.
- Reasoning Engine: this module integrates several algorithms implemented in a programming language (Java or C++ for example) and a reasoner (JENA for example), which take care of the execution of processes (to achieve a goal) triggered by a logical rule at the level of the KB, the processing of messages received by the avatar and the update of the avatar KB with a new knowledge. This means that once a process takes the Waiting state in the KB, that engine puts the process state to Raining via an UPDATE SPARQL query and execute a SELECT SPARQL query to retrieve the complete process description. Then for each task, it executes an ASK SPARQL query to find if there is a task that corresponds to the considered task taking into account its preconditions and post-conditions. If it does, it executes it, otherwise, it solicits a UDDI[5] (Universal Description Discovery and Integration) to retrieve the list of avatars able to achieve it while considering preconditions and post conditions of the task. The avatar sends a request message to the first avatar of the list to ask him to confirm that he can perform the task. If he get an acceptance message and he has the input data of this task, he sends him a confirmation message with the inputs. Otherwise, if he will have a rejection message, he will request another avatar from the returned list, and so on. The dependencies existing between the process tasks such as the sequence, parallelism for example are specified in its semantic description (see ontology Figure 5) which allows to guide this engine in the execution stage. All of these actions are summarized in Algorithm 1 and Algorithm 2.

Example :

Let us consider a tiredness detection scenario within a connected vehicle. It is assumed that the vehicle is equipped with a camera placed in front of the driver to analyze its behavior and detect any sign of tiredness: head inclination and eyes inclination, a vibrator built into the steering wheel and an user interface to display messages to the driver.

We will now unfold the functioning of the above modules on our use case: once an inclination of the driver's eyes is detected via the video stream analysis of the camera that monitors his tiredness state thanks to an algorithm embedded in the Context Manager of the Camera avatar,this last updates the knowledge base with an UPDATE operation. Thanks to the rule of the example , this change will trigger the process expressed in the example by setting its state to Waiting. Once the reasoning engine realizes that there is a pending process through periodic checks, it sets its state to Raining with a SPARQL UPDATE operation.

[5]We assume in this first work that all available avatars publish their services in an UDDI

Algorithm 1 Process Execution

Inputs: Process i with Waiting state
Outputs: success or failure
BEGIN

```
1  Make the state of Process i to Raining with SPARQL UPDATE request
2  Get the process description with SPARQL GET request
3  Extract the first task : task
4  WHILE (task hasNext)
5    IF task is AtomicTask
       Call ExecuteAomicTask();
6    ELSE
7      IF task hasGrouping Parallel or Choice
         FOR each ChildTask of task
           JUMP to 5
8      ELSE
         JUMP to 4
       ENDIF
     ENDIF
   ENDWHILE
```

END

Then it retrieves its description, it extracts the first task which is of type GroupedTask, so it recovers the type of Grouping thanks to the relation hasGrouping and it finds that it is of parallel type. And so he gets back the subtasks through the hasChildTask relation and he finds that he has two subtasks Vibration and Message and as the relation is parallel he takes for each task apart. For the vibration task, it checks whether it is an atomic or grouped task, it finds that it is atomic and it checks if it has in its knowledge base a service to carry out this task. So, it seeks from the UDDI the avatars who can realize the task and it finds one who is Vibration_Avatar. The engine sends it a REQUEST message awaits its answer. Once the avatar receives the response ACCEPT_PROPOSAL, it checks the preconditions for the execution of the task, it finds that there are no conditions that depend on other tasks so it sends a CONFIRM message to Vibration_Aavatar with the complete description of the task. Once the Vibration_Avatar receives the confirmation, it executes the corresponding service. And the same with the message display task.

V. Implementation technologies

As we said at the beginning, our goal is to add autonomous behavior to virtualized IoT entities that ensure interoperability between different IoT domains regardless of the technologies

Algorithm 2 ExecuteAtomicTask()

Inputs: task description
Output: outputsService, success or failure
BEGIN

```
1  Run a SPARQL ASK query on the avatar KB
2  IF (avatar is able to perform the task)
     3  Execute the service matches
     4  Return his outputs
5  ELSE
     6  Search the list of avatars able to perform the task
        in the UDDI
     7  Send a message to all the avatars of the list: msg =
        [id, REQUEST, taskDescription, list];
     8  Continue = true;
     9  WHILE (Continue & (timeOut not expire) )
      13  ReceiveMessage();
        14  IF (Performative == ACCEPT_PROPOSAL)
          15  Send a message to avatar: msg = [id, CON-
              FIRM, taskDescription, avatar]
          16  Continue = false
        17  ELSE
          18  IF (Performative == REJECT_PROPOSAL)
          19  Continue= true
              ENDIF
            ENDIF
         ENDWHILE
      20  IF (timeOut expire)
        21  Return failure
   ENDIF
END
```

used. For this, we chose to use a platform based on the intentional standard oneM2M .OM2M[6]: is an open source project developed by the LAAS-CNRS[7], then distributed by the Eclipse Foundation. It provides a standardized horizontal M2M services platform implementing ETSI SmartM2M and oneM2M standards. OM2M aims to reduce the complexity of the process of developing vertical M2M applications that can work across a wide range of heterogeneous devices, protocols and networks to facilitate their deployment. This platform offers a modular architecture, which runs over the OSGI EQUINOX (Open Services Gateway initiative) layer, which makes it highly extensible via plugins.

In order to add autonomous behavior to an OM2M resource that represents a virtual representation of a single real-world object, we have studied the multi-agent frameworks as JaCaMo[8], PADE[9], FraMaS [25] and JADE[10]. Thanks to this study, we realized that JADE is the most adapted to our needs given its ease of use and compliance with the FIPA standard.

JADE[11] (Java Agent Development Framework): is a software that offers implementation facilities. It includes: i) A runtime environment where agents will run on distributed hosts. ii) A library of classes that can be used to develop agents, per-to-per-call communication via FIPA-ACL[12] (Foundation for Intelligent Physical Agents) asynchronous messages. This framework also supports the migration of agents from one host to another which makes it interesting to implement the autonomous behavior of avatars.

The combination of OM2M and JADE architectures in our solution gave rise to the architecture shown in Figure 6:

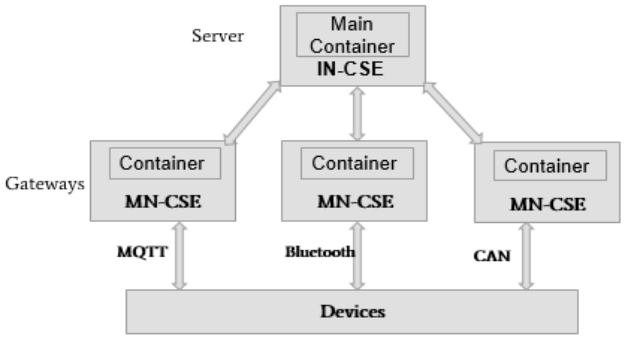

Fig. 6. Proposed avatar-based technical architecture

The distributed architecture of the proposed solution essentially consists of a set of devices based on heterogeneous technologies from different brands, connected to several OM2M gateways that ensure interoperability by exposing the features offered by the different devices via open interfaces based on a lightweight RESTful API. Each OM2M gateway (MN-CSE[13]) hosts a JADE container on which the avatars run. These gateways are connected and registered with an OM2M server (IN-CSE[14]) which allows access to the entire system and which in turn hosts the JADE Main Container. The latter contains a DF(Directory Facilitator) which represents Yellow Pages of services where avatars can publish their capabilities and search for avatars providing the capabilities they need and an AMS(Agent Management System) used for avatars life cycle management (creation, deletion and their migration).

Concerning the modeling of semantic knowledge of avatars, we have chosen OWL[15](Web Ontology Language) for the instantiation of the ontologies used and SWRL[16] (Semantic Web Rule Language) which provides a high-level syntax for modeling the logical rules of the avatar as a sequence of axioms and facts, because of their simplicity and intuitive use.

[6]https://www.eclipse.org/om2m/
[7]https://www.laas.fr/public/
[8]http://jacamo.sourceforge.net/
[9]https://pade.readthedocs.io/en/latest/
[10]http://jade.tilab.com/
[11]http://jade.tilab.com/
[12]http://www.fipa.org/
[13]MN-CSE: Middle Node Common Services Entities
[14]IN-CSE: Infrastructure Node Common Services Entities
[15]https://www.w3.org/OWL/
[16]https://www.w3.org/Submission/SWRL/

VI. Conclusion

In this paper, we presented the general context of our work and a first proposal for a modular architecture for IoT systems that relies on a software artifact called *autonomous avatar*. This later virtualizes and represents a given real-world object on the Web. This is suitable for developing interoperable complex IoT systems. The particularity of our proposition is that it relies on the artifact of autonomous and collaborative avatars, which enables decentralized and collaborative management of IoT systems. . In this contribution, we focused on the reasoning module design of this architecture and the logical rules to apply on the knowledge base.

In our ongoing works, we are extending the proposed architecture by defining a distributed and decentralized approach for goal-guided autonomous avatars collaboration. This will ensure a dynamic and distributed management of the services provided by the avatars to realize complex IoT applications. We aim to implement all of these proposals on the use case of overtaking between vehicles in collaboration with the company Continental[17].

ACKNOWLEDGMENT

This work is funded by Continental Digital Service France (CDSF) in the framework of the eHorizon project.

References

[1] Nitti, M., Pilloni, V., Colistra, G., and Atzori, L. (2015). The virtual object as a major element of the internet of things: a survey. IEEE Communications Surveys Tutorials, 18(2), 1228-1240.

[2] Terdjimi, M. (2015, October). Multi-level context adaptation in the Web of Things.

[3] Rachkidi, E. (2017). Modelling and placement optimization of compound services in a converged infrastructure of cloud computing and internet of things (Doctoral dissertation, Université Paris-Saclay; Université d'Evry-Val-d'Essonne).

[4] Farris, I., Girau, R., Militano, L., Nitti, M., Atzori, L., Iera, A., and Morabito, G. (2015). Social virtual objects in the edge cloud. IEEE Cloud Computing, 2(6), 20-28.

[5] Han, N. S. (2015). Semantic service provisioning for 6LoWPAN: powering internet of things applications on Web (Doctoral dissertation).

[6] DAVE, RAGGETT. Introduction to the Web of Things. W3C.[Online]. Available at https://www.w3.org/2015/03/intro-wot.pdf, Accessed: 2019-06-30.

[7] Römer, K., Schoch, T., Mattern, F., and Dübendorfer, T. (2004). Smart identification frameworks for ubiquitous computing applications. Wireless Networks, 10(6), 689-700.

[8] Angarita, R., Manouvrier, M., and Rukoz, M. (2016, April). An agent architecture to enable self-healing and context-aware web of things applications.

[9] Han, S. N., Lee, G. M., and Crespi, N. (2013). Semantic context-aware service composition for building automation system. IEEE Transactions on industrial informatics, 10(1), 752-761.

[10] Mrissa, M., Medini, L., Jamont, J. P., Le Sommer, N., and Laplace, J. (2015). An avatar architecture for the web of things. IEEE Internet Computing, 19(2), 30-38.

[11] Presser, M., Barnaghi, P. M., Eurich, M., and Villalonga, C. (2009). The SENSEI project: Integrating the physical world with the digital world of the network of the future. IEEE Communications Magazine, 47(4), 1-4.

[12] SENSEI project.[Online]. Available at https://www.ict-sensei.org/, Accessed: 2019-06-30.

[13] De, S., Barnaghi, P., Bauer, M., and Meissner, S. (2011, September). Service modelling for the Internet of Things. In 2011 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 949-955). IEEE.

[14] Projet iCore.[Online]. Available at http://www.iot-icore.eu, Accessed: 2019-07-01.

[15] Kibria, M. G., Kim, H. S., and Chong, I. (2016, January). IoT learning model based on virtual object cognition. In 2016 International Conference on Information Networking (ICOIN) (pp. 369-371). IEEE.

[16] Parodi, A., Maresca, M., Provera, M., and Baglietto, P. (2015, October). An iot approach for the connected vehicle. In International Internet of Things Summit (pp. 158-161). Springer, Cham.

[17] ETSI, Terms of Reference for Technical Committee (TC) Smart Machine-to-Machine communications (Smart M2M).[Online]. Available at https://portal.etsi.org/TBSiteMap/SmartM2M/SmartM2MToR.aspx, Accessed: 2019-07-01.

[18] oneM2M,Standards for M2M and the Internet of Things. [Online]. Available at http://www.onem2m.org/, Accessed: 2019-07-01.

[19] Seydoux, N., Drira, K., Hernandez, N., and Monteil, T. (2016, November). IoT-O, a core-domain IoT ontology to represent connected devices networks. In European Knowledge Acquisition Workshop (pp. 561-576). Springer, Cham.

[20] The NeOn Methodology.[Online]. Available at http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/methodologies/59-neon-methodology/, Accessed: 2019-07-02.

[21] Mizoguchi, R., and Kozaki, K. (2009). Ontology engineering environments. In Handbook on Ontologies (pp. 315-336). Springer, Berlin, Heidelberg.

[22] Roman, D., Kopecký, J., Vitvar, T., Domingue, J., and Fensel, D. (2015). WSMO-Lite and hRESTS: Lightweight semantic annotations for Web services and RESTful APIs. Journal of Web Semantics, 31, 39-58.

[23] Tietz, V., Blichmann, G., Pietschmann, S., and Meißner, K. (2011, June). Task-based recommendation of mashup components. In International Conference on Web Engineering (pp. 25-36). Springer, Berlin, Heidelberg.

[24] Swetina, J., Lu, G., Jacobs, P., Ennesser, F., and Song, J. (2014). Toward a standardized common M2M service layer platform: Introduction to oneM2M. IEEE Wireless Communications, 21(3), 20-26.

[25] Su, C. J., and Wu, C. Y. (2011). JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. Applied Soft Computing, 11(1), 315-325.

[17] https://www.societe.com/societe/continental-digital-services-france-821289675.html