Towards Semantic Data Interoperability in oneM2M Standard

Mahdi Ben Alaya, Samir Medjiah, Thierry Monteil, Khalil Drira

Towards Semantic Data Interoperability in oneM2M Standard

Mahdi Ben Alaya [a,b], Samir Medjiah [a,c], Thierry Monteil [a,d], Khalil Drira [a]

[a] CNRS, LAAS, 7 avenue du colonel Roche, F-31400, Toulouse, France
[b] Univ de Toulouse, LAAS, F-31400, Toulouse, France
[c] Univ de Toulouse, UPS, LAAS, F-31400, Toulouse, France
[d] Univ de Toulouse, INSA, LAAS, F-31400, Toulouse, France
{ben.alaya | medjiah | monteil | drira} @ laas.fr

Abstract– OneM2M standard is a global initiative led jointly by major standardization organizations around the world in order to come up with a unique standard for M2M communications. Prior standards, but also oneM2M, while focusing on achieving interoperability at the communication level, fail or lack to achieve full interoperability at the semantic data level. An expressive ontology for IoT called IoT-O has been defined making best use of already defined ontologies in specific domains such as sensor, observation, service, quantity kind, units, or time. IoT-O defines also some missing concepts relevant for IoT such as thing, actuator, actuation, or manager. The extension of the oneM2M standard to support semantic data interoperability based on IoT-O is discussed. Finally, through comprehensive real use cases, benefits of the augmented standard are demonstrated ranging from heterogeneous devices interoperability to autonomic behavior achieved by automated reasoning.

Keywords: IoT, M2M, oneM2M, interoperability, semantic, ontology, autonomic, reconfiguration, reasoning, IoT-O.

I. INTRODUCTION

In the recent few years, M2M systems have witnessed the emergence of various and different standardization initiatives. Indeed, different applications sectors are pushing standards that are often targeting mainly a specific application domain such as smart meters standards developed by IEC or IEEE (EN 13757, IEEE 1888-2011, etc.). Different SDOs have tackled this problem by focusing on the definition of a horizontal service platform that fits different verticals. This work has been consolidated later on into a global initiative aka oneM2M.

It is worth to notice that all these initiatives have focused on the communication interoperability between system entities (servers, devices, applications, etc.). Indeed, these standards have defined a horizontal service layer that enables seamless communication between heterogeneous entities independently of the underlined network and vendor-specific device technologies. It thus possible to reach any entity in the system and deliver a message to it. However, no standard has tackled the "meaning" of the message content being exchanged. Although SmartM2M standard has introduced some recommendations for supporting semantics [1], a generic data model has not been specified. This has been let to the appreciation to the service provider, system developer, or the system user. Such standards have achieved interoperability at the communication level only and lack to provide interoperability at the data level too. This has led to inefficient systems, since actual autonomic systems could not be achieved.

Semantic data is brought through the definition of a common set of ontologies that describe the entire system's entities but also the data items produced, exchanged and consumed by these entities. Various information models have been defined for IoT ranging from specialized models such as the Zigbee or KNX data models to more general ones such as W3C SSN **Erreur ! Source du renvoi introuvable.**. These solutions suffer from two main issues. They are even too specialized and focused in a specific application domain, or lack from some concepts mainly related to actuation. Indeed, in multiple information models, "control" concepts are missing. Using such models may be very challenging since in M2M systems, devices may be sensors or actuators, or both.

In this paper, we discuss and propose an ontology model (IoT-O) that handles both sensing and actuating concepts of M2M devices but also some concepts related to services. We, then, discuss the extension of oneM2M standard to support semantic data based on the proposed ontology. Finally, through comprehensive uses cases, we show the use of IoT-O along the oneM2M standard.

A. oneM2M Standard

The oneM2M global initiative [3] is an international partnership project established in June 2009 by the seven most important standard defining organizations in the world and various alliances and industry. The main goal is to define a globally agreed M2M service platform by consolidating currently isolated M2M service layer standards activities. OneM2M is planning to boost M2M market by removing the need to develop common components, simplify development of applications by providing a common set of APIs, leverage existing worldwide networks, and provide evolution and interoperability of standard functions support. The oneM2M technical working groups are focusing on requirements, system architecture, protocols, security, management, abstraction and semantics. **Figure 1** oneM2M system architecture [3] describes the OneM2M system architecture.

The system architecture is composed of four functional entities called nodes known as application dedicated node (ADN), application service node (ASN), middle node (MN), and infrastructure node (IN). Each node contains a

common services entity (CSE), an application entity (AE), or both. An AE provides application logic, such as remote blood sugar monitoring, for end-to-end M2M solutions. A CSE comprises a set of service functions called common services functions (CSFs) that can be used by applications and other CSEs. CSFs incudes registration, Security, application and service layer management (ASM), Device Management, communication management and delivery handling, network service exposure, data management and repository, Discovery, subscription and notification, service session management, service charging and accounting, group management, and location service.

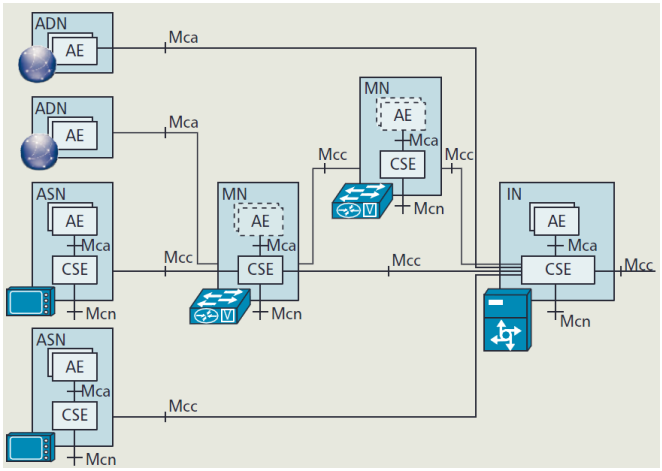

Figure 1 oneM2M system architecture [3]

The system architecture has specified three reference points called Mca, Mcc, and Mcn. The Mca interface enables AEs to use the services provided by the CSE. The Mcc interface enables inter-CSE communications. The Mcc' interface is similar to Mcc, but provides an interface to another oneM2M system. The Mcn interface is between a CSE and the service entities in the underlying networks. OneM2M adopted a RESTful architecture, thus all services are represented as resources. Resources are associated with CSFs to provide the defined functions.

B. Full interoperability challenge

Full interoperability is a desirable property to achieve in M2M systems. This will pave the way to the ultimate goal aka autonomic systems. Indeed, interoperability between heterogeneous devices, services is only a requirement to achieve autonomic behavior including self-configuration, self-healing, self-optimization, and self-repairing.

As introduced earlier, almost all standardization initiatives have not efficiently tackled the issue of full interoperability, i.e. considering both communication and data interoperability. Having treated the interoperability at the communication level through the definition of common service, resources, payload formats, and mapping to existing internet protocols or network access technologies, the different M2M standards have achieved the communications interoperability. Thanks to this interoperability, M2M systems' entities already benefit from services such as discovery, monitoring, management, etc. Although such service platform can be sufficient for the design and implementation of specific M2M systems, autonomic system behavior using automated reasoning based on top of a knowledge oriented service platform cannot be achieved.

For example, using a service platform built upon oneM2M standard, an application can discover seamlessly new devices plugged into the system. This application can subscribe to the new device events and will receive them as soon as they are triggered even if the routing path implies the crossing of multiple entities and using heterogeneous communication protocols or network technologies at any segment of the communication path. This has been made possible thanks to the interoperability at the communication level. Now that device events have been successfully reported, the application does not have any mean in order to "understand" the reports' content without prior conventions (data formats, encapsulation, and semantics) set up between the application and the device application developers.

C. Semantic data, concepts and principles

Ontologies have proven beneficial for intelligent information integration, information retrieval, and knowledge management. They enable to index resources content using semantic annotations that can result in the representation of explicit knowledge that cannot be assessed and managed because of their mess. Ontologies are very popular and useful to overcome challenges fixed in the proposed study because they provide a new way of cleverly structuring a domain making use of semantic hierarchical and property/value relationships based on a vocabulary of concepts/instances [4].

The most popular language in the domain of semantic knowledge modeling making use of ontologies is the Web Ontology Language (OWL). OWL is a semantic an expressive schema language for publishing and sharing ontologies using RDF (the Resource Description Framework) extensions. OWL facilitates interoperability between entities by providing a shared understanding of the domain in question. It is an effective means for explicating implicit design decisions and underlying assumptions at system build time based on powerful deductive reasoning capabilities such as the Semantic Web Rule Language (SWRL) or the SPARQL query language.

II. AN EXPRESSIVE ONTOLOGY FOR IOT (IOT-O)

A. IoT ontology principles

Users and software agents should be able to discover, monitor and control sensors and actuators offering particular services and having particular properties with a high degree of automation. However, oneM2M did not standardized the device data model and the data they handle. In this study, we propose a semantic data model based on ontology to represent IoT device meta-data, operations, and exchanged data independently from the

underlying formalism originally used for describing them. Since ontologies are designed to be reusable and extensible, we decided to define a complete ontology for IoT by reusing existing ontologies. New concepts are designed only when needed. This approach enables to reduce the ambiguity of IoT terminology and allows to converge quickly to a common vocabulary.

In general, an ontology for IoT should represent a variety of concepts such as platform, deployment system, thing, device, manager, service, sensor, actuator, sensing and actuating capabilities, observation, operation, time, unit, kind, value, and their relationships. Since there is no single model that covers all these concepts, a set of well-defined ontologies were carefully selected in order to form an expressive ontology called IoT-O. **Figure 2** shows how the selected ontologies are merged together to form this new ontology. IoT-O consists of five main parts which are sensor, observation, actuator, actuation and service models.

B. IoT-O concepts and relationships

The DUL upper ontology is selected to describe very general concepts that are the same across all knowledge domains, and so facilitate reuse and interoperability. It is a lightweight foundational model for representing either physical or social contexts. The SSN ontology, which is aligned with DUL, is selected to represent sensors in terms of measurement capabilities and properties, observations and other related concepts, however it does not describe actuator devices. Since currently there is no ontology that accurately describes actuators, we designed a new ontology called ACT, which is inspired from SSN and aligned with DUL, to describe actuators in terms of actuating capabilities and properties, actuation, and related concepts. The QUDV ontology was selected to represent quantities, units, dimensions and values. The OWL-TIME ontology was selected to provide a vocabulary for expressing facts about topological relations among instants and intervals, together with information about duration, and about date time information.

Given that OneM2M aims to enable seamless interactions between business applications and services, it is important to represent how these services can be requested, without any ambiguity in order to reduce the amount of manual effort required for discovering and using them. The MSM ontology was selected to describe services since it provides a common vocabulary based on existing web standards able to capture the core semantics of both Web services and Web APIs in a common model. Each service is described using a number of operations that have address, method, input and output Message Content descriptions.

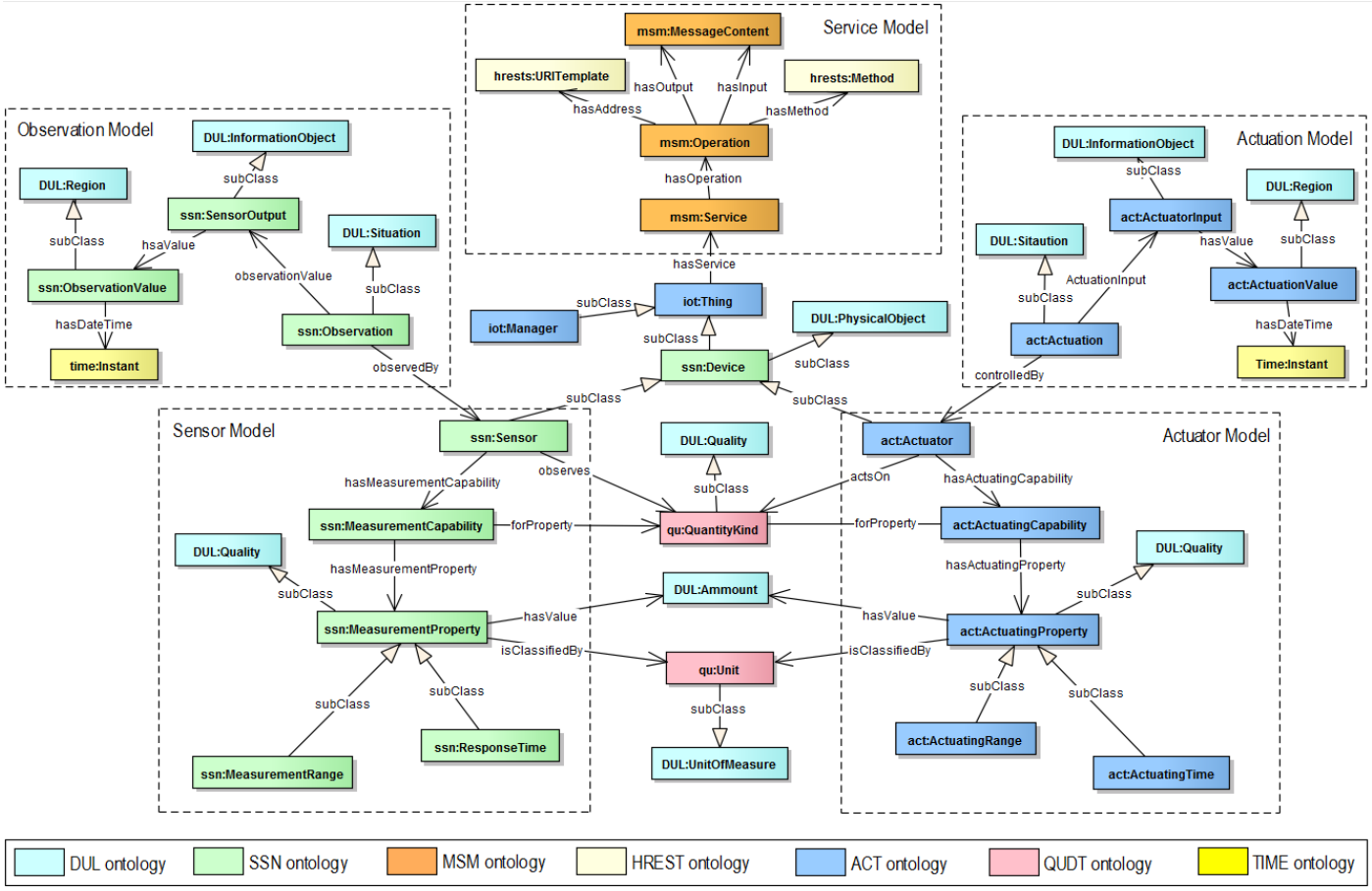

Figure 2: IoT-O ontology model

D. Actuator model instance according to IoT-O

To understand how the IoT-O ontology works, let's consider a concrete example representing a real actuator using the M2M ontology. The "HUELUX" actuator is a digitally dimmable wireless lighting bulb from Philips. It has a power range of 0 Watt to 50 Watt with a lighting time of 2 seconds. The luminance level can be dimmed by requesting the required power value. The light bulb offers a web service to enable remote luminance control. The luminance can be dimmed instantaneously by sending a create request to the address "/HUELUX_APP/dimming" with a message body containing the required power. **Figure 3** Actuator model instance according to IoT-O details the corresponding ontology instance. It shows how the actuator, actuation and service information are inserted in the IoT-O ontology. The actuator model represents the light bulb information and actuating capabilities including power range and lighting time. The actuation model represents the dimming command. The Service model represents the light bulb web service including the luminance dimming operation, address and method.

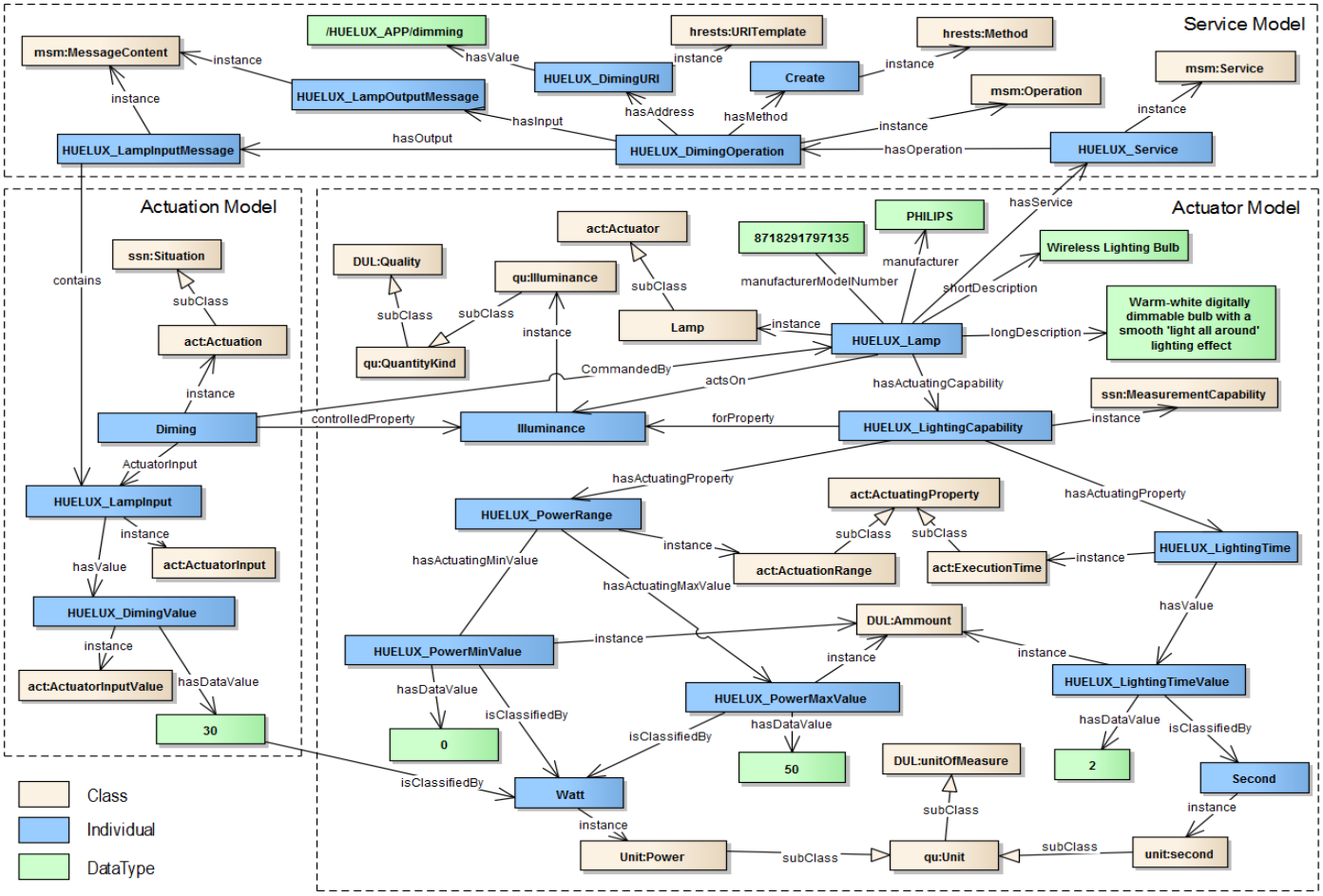

Figure 3 Actuator model instance according to IoT-O

E. Semantics extension to the oneM2M Standard

Through oneM2M working group WG5, semantics is already envisioned for the oneM2M standard. However, as of its Candidate release, semantics aspects are not tackled yet. In this subsection, we will discuss a possible extension to oneM2M standard in order to support semantic data. The main idea is to use specific resource's attributes in order to augment the resource with its semantics. Such attributes can be used to give some indication about the ontology associated with the content being transported. Two options are then available.

The first option, as we dubbed *inline integration*, relies on using a resource attribute in order to carry the full definition of the ontology in an appropriate format. Using this option, any application (or process) can understand the received content after the interpretation of the ontology definition. Such attribute is not yet defined in oneM2M standard, but can be pushed in future oneM2M releases.

The second option, as we dubbed *reference-based integration*, uses the semantic attribute *ontologyRef* of type *xs:anyURI*, already defined in the Candidate release of oneM2M standard, to point the application to a remote

location where it can find the full definition of the ontology. It can also be used to serve as just a reference to an ontology supposedly known by the application (i.e. through the use of predetermined ontology catalog).

It is clear that these two options have their pros and cons. Indeed, while the inline integration option gives great flexibility since every data element will carry its own ontology, this option will introduce an important overhead in all communications by increasing significantly the payload size. Moreover, this can be very inefficient especially for communications to/from very constrained M2M devices. The reference-based integration introduces less overhead when compared to the first option. However, the need for an external repository (or a hard-coded/known catalog) may introduce new issues such as the optimal repository location, definition coherence across multiple repositories, access rights, repository availability, etc. Moreover, such option may introduce new delays related to the acquisition of the necessary ontology definition prior to data processing. This may be problematic in case of delay sensitive applications.

Although both options present advantages and disadvantages, we believe that at this stage of the oneM2M standard definition, the use of the dedicated semantic attributes remains the best solution in order to introduce interoperability at the data level in the oneM2M standard without questioning the foundations of this new standard and in order to comply with its first published release.

III. SEMANTIC DATA INTEROPERABILITY

In this section, and through comprehensive use cases, we present the usability of our new generic data model. Also, we will introduce OM2M [5], our developed horizontal platform which is compliant with both SmartM2M and oneM2M standards. The use case will also feature our living lab aka ADREAM smart building and put the addressed challenge in a real scenario.

A. *LAAS smart building: ADREAM*

ADREAM is the LAAS-CNRS smart experimental building. The main originality of this instrumented building compared to already existing ones is that it is a « living lab » of 1700m² since it is both a research tool and a building with offices for the researchers.

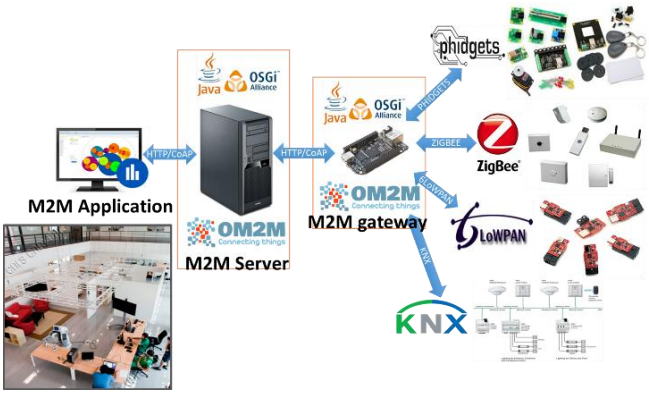

Figure 4: M2M testbed within ADREAM building

The building includes 500m² of technical platforms (IoT, robotics, ambient intelligence, energy) and 700m² of offices. It hosts our smart apartment equipped with various sensors and actuators connected using different networking technologies. The device set includes different sensors (temperature, humidity, luminescence, presence, etc.) but also actuators such as electric plugs attached to different elements: lamps, fans, humidifier, etc. all these devices are gathered around different gateways. Each gateway is specialized in one or two networking technologies (Zigbee, 6lowpan, KNX, Phidgets). Finally, these gateways are connected to one central server. **Figure 4** illustrates the testbed composition.

B. *Seamless device discovery and interaction*

In order to demonstrate the interoperability aspect achieved by the OM2M platform through its compliance with the oneM2M standards and its support of a generic data model IoT-O, we propose a simple scenario where the software platform is able to discover newly plugged devices such as sensors and actuators, browse the exposed attributes and methods, and finally interact with these devices by retrieving sensed data or triggering actions.

The scenario setup, as showed in **Figure 4**, includes different devices attached to an M2M gateway through local network technologies such as wireless ones: ZigBee, 6lowpan, or wired technologies such as Phidgets or KNX. The M2M gateway is connected to a M2M server. The M2M gateway entity is equipped with mapping modules that translate every communication with a specific networking technology into a generic communication protocol that is completely independent from the transport protocol or the network access. Thus, the support of new technologies or protocols is simply achieved through the implementation of the translation module (i.e. Interworking Proxy Unit). When the IPU discovers a new device through the specific technology discovery mechanism, it will expose this device along its attributes and methods to other entities in the M2M system. From the M2M system perspective, any data or action request is routed to this IPU in order to be translated to the specific technology operations. In this way, any application present in the M2M system can access the new discovered resources (device, device's attributes, device's actions, etc.) using standardized restful operations, and this can be achieved without any knowledge of the underlying network technology or its low level mechanisms.

Furthermore, since all exchanged messages are augmented with semantics as discussed in section II.B. One application cannot only have access to the data being generated by the device or the actions it exposes, but also the it can understand the meaning of these data. Indeed, since the application has access to the ontology that "defines" the data, it can browse the ontology and map the received data elements into this ontology and perform the appropriate processing.

IV. TOWARDS AUTONOMIC M2M SYSTEMS

In this section, we demonstrate how the IoT-O ontology can be used to develop autonomic M2M systems [6,7] capable of self-management to hide intrinsic complexity to administrators and users. The main goal here is to dynamically reconfigure the CSE resource architecture based on semantic matching between registered applications.

A. *Autonomic service for resource architecture dynamic reconfiguration*

In the normal case, an application must perform manually several complex search request to discover relevant resources and perform several subscriptions to monitor the evolution of interesting resources. In addition, an application has a partial view of its M2M environment, then it becomes very complex to find the right resources especially in huge and highly distributed M2M system.

Introducing a new autonomic service with a global view of the M2M system capable of configuring CSE resource architecture when needed is a challenge. The objective here is to enable any application to dynamically discover interesting devices and to exchange data with the right communication mode according to its description, role, and relationships. To meet this goal a representative model of M2M system knowledge is required to assist the execution of the management process. Since the IoT ontology covers all required M2M concepts needed for this use case, it will be used as a knowledge model by an autonomic service.

B. *Data model for automated reasoning*

Figure 5 shows a partial view of the IoT-O ontology highlighting the main concepts needed for the self-configuration process. In one hand, the IoT-O enables to represent physical things like sensor and actuator devices that respectively observes and acts on a quantity kind. In the other hand, it enables to represent nonphysical things like monitor and controller managers that respectively monitors and controls a quantity kind. Each thing is registered to a node and is accessible via a set of web services. Additional concepts of IoT-O can be considered to further refine the semantic matching process for a more advanced configuration.

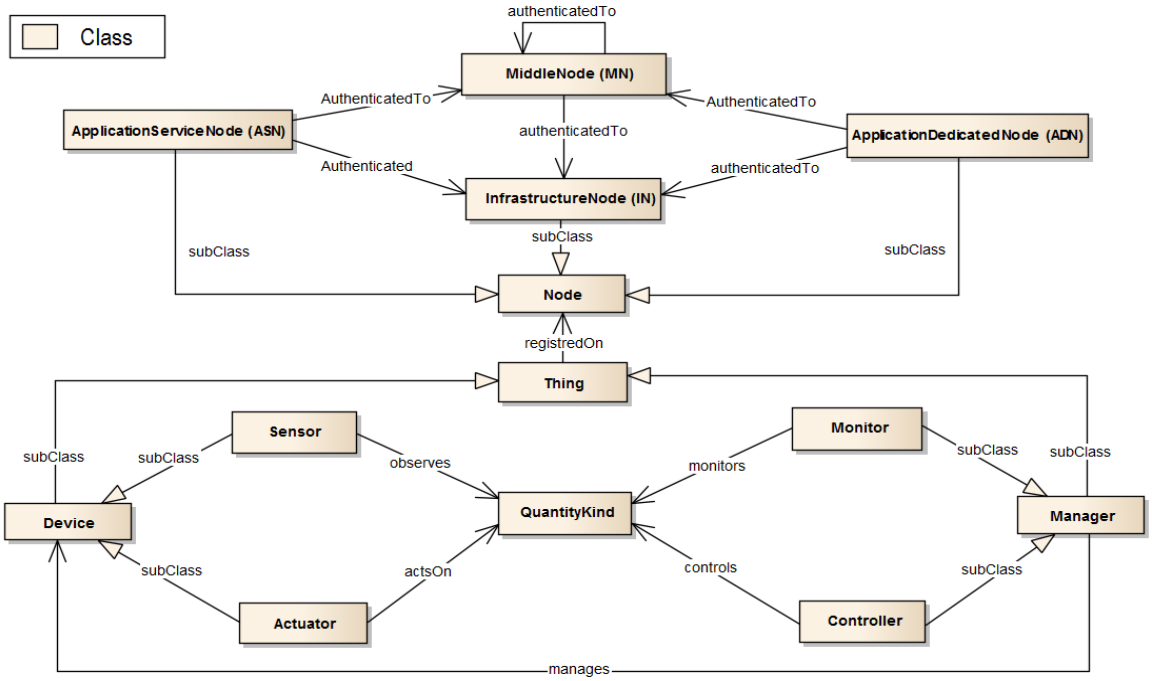

Figure 5: IoT-O main concepts and relationships used for reasoning

C. *Smart building use case*

Let's consider a building equipped with a gateway connecting locally some devices like an electricity meter, a luminosity sensor, and a lamp that created on the gateway an "ASN-AE_ElectricityMeter", "ASN-AE_LuminositySensor", and "ASN-AE_Lamp". An "ASN-AE_LampController" and an "ASN-AE_LuminosityMonitor" applications are registered on the gateway to dynamically update the lamp state according to the luminosity level. Let's suppose that a lamp controlling application "ADN-AE_LampController" and a power consumption monitoring application "ADN-AE_ElectricityMonitor" are deployed on the user Smartphone to enable the user to manually monitor and control his devices. In addition, A smart metering application "IN-AE_ElectricityMonitor" is registered on the smart metering server to track the building consumption.

The autonomic service should discover and monitor the description of all registered applications, reasons on the IoT-O ontology model using inference rules to find relevant

matching, and finally reconfigure the resource architecture accordingly to set up the required connections.

Initially, the autonomic service discovers the IN-CSE and so adds the "IN_SmartMeteringServer" individual as instance of the "IN" class in the IoT-O instance. Then, it retrieves the registered applications and looks mainly for the "DESCRIPTOR" container to find the application descriptions. The "IN-AE_ElectricityMonitor" individual is added as instance of the Monitor class and is linked to the "Electricity" QuantityKind. Then, the autonomic service searches for the authenticated nodes and so adds the "ASN_HomeGateway" individual as instance of the ASN class, and the "ADN_Smarthone" individual as instance of the ADN class. It creates also a subscription on the IN to be notified of new authenticated nodes. **Figure 6**: IoT-O smart building use case instance example corresponding M2M ontology instance as generated by the Autonomic service.

For each discovered node, the autonomic service retrieves the registered applications and adds them to the IoT-O instance. For the gateway, The "ASN-AE_ElectricityMeter" is added as an individual of the Sensor class and linked to the "Electriciy" QuantityKind. The "ASN-AE_LightSensor", "ASN-AE_Lamp", "ASN-AE_LampController", and "ASN-AE_LuminosityMonitor" individuals are added respectively as instances of the Sensor, Actuator, Monitor, Controller classes, and are linked to the "Luminosity" QuantityKind.

For the Smartphone, the "ADN-AE_ElectricityMeter" individual is added as instance of the Sensor class and linked to the "Electriciy" QuantityKind. The "ADN-AE_LampController" individual is added as instance of the Monitor class and linked to the "Luminosity" QuantityKind.

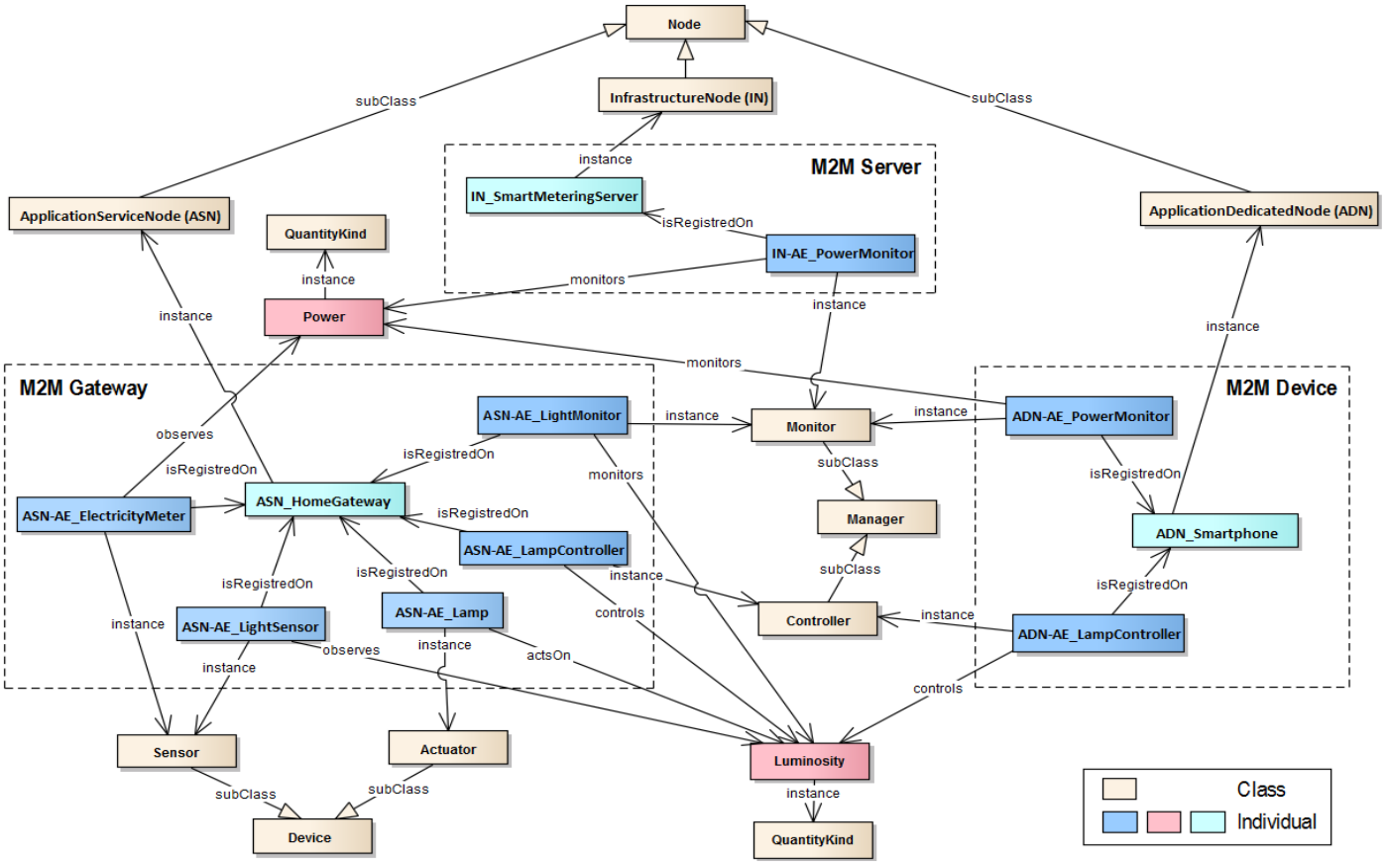

Figure 6: IoT-O smart building use case instance example

D. Semantic matching inference rules

Inference rules can be applied to infer new knowledge and so enrich the current IoT-O instance with new individuals and relationships. This new knowledge is necessary to understand the role of each application in the M2M architecture. It allows each application to take maximum advantage of the services offered by other applications.

In this example, two SPARQL [8] rules are applied by the autonomic service to find semantic matching between the registered applications. The first rule "Infer relationship between monitors and sensors" says that if there is a monitor that observes a particular quantity kind, and if it exists also a sensor that observes the same quantity kind, and if this monitor is not already managing this particular sensor, then as a result a new "manages" relationship is inferred to link this monitor to this sensor.

```
CONSTRUCT {
  ?monitor iot:manages ?sensor
}
WHERE {
  ?monitor rdf:type  iot:Monitor .
  ?sensor rdf:type  iot:Sensor .
  ?qKind rdf:type  iot:QuantityKind .
  ?monitor iot:monitors  ?qKind .
  ?sensor iot:observes  ?qKind .
  FILTER EXISTS { ?monitor iot:manages ?sensor }
}
```

The second rule "Infer relationship between controllers and actuators" says that if there is a controller that controls a particular quantity kind, and if it exists also an actuator that acts on the same quantity kind, and if this controller is not already managing this particular actuator, then a new "manages" relationship is inferred to link this controller to this actuator.

```
CONSTRUCT {
  ?controller iot:manages ?actuator
}
WHERE {
  ?controller rdf:type  iot:Controller .
  ?actuator rdf:type  iot:Actuator .
  ?qKind rdf:type  iot:QuantityKind .
  ?controller iot:controls  ?qKind .
  ?actuator iot:actsOn  ?qKind .
  FILTER EXISTS{?controller iot:manages ?actuator}
}
```

Using the same approach, more advanced rules can be applied including more constraints such as the device location, temporal requirements, or quality of services parameters.

E. Resource configuration execution

The Autonomic service is capable now to plan the list of required actions and to execute them by simply sending RESTful requests to the service platform. Indeed, to establish a relationship between a monitor and a sensor, a new subscription resource is created on the monitor sensor application containing the monitor contact. Respectively, to establish a relationship between a controller and an actuator, a new subscription resource is created on the controller application containing the actuator contact.

Concretely, the autonomic service automatically subscribes the "IN-AE_ElectricityMonitor" and the ADN_ElectricityMonitor" to the "ASN-AE_ElectricityMeter". It subscribes also the ASN-AE_LuminosityMonitor" to the "ASN-AE_LuminositySensor". Finally, the "ASN-AE_LampController" and the "ADN-AE_LampController" are subscribed to the "DA_LampContoller".

V. CONCLUSION

Current M2M standards aims to provide a horizontal service platform to enable communication interoperability between machines. However, the semantic data interoperability is not achieved which brings into question the horizontality of such platform. To overcome this challenge, a dedicated ontology for IoT called IoT-O has been defined. IoT-O merges together a set of popular ontologies and is enriched with new relevant concepts and relationships. Two possible integration with the oneM2M standard are discussed. The mean concepts and relationships of IoT-O are described using different use cases. An autonomic service making use of IoT-O and inference rules for resource architecture dynamic reconfiguration was explained as well.

As future work, we propose to calculate the overhead cost of the our solution and the resulting overload. We propose also to validate IoT-O in various vertical M2M domains such as e-health, transport, and smart grid. A set of contributions will be sent to OneM2M for integrating IoT-O concepts into the standard to move forward towards semantic data interoperability.

REFERENCES

[1] ETSI TR 101 584 - M2M – Study on Semantic support for M2M Data (v2.1.1), February 2013.

[2] Compton M, and al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. Journal of Web Semantics, 17. 25 - 32. 2012. ISSN 1873-7749.

[3] Swetina, J. and al. "Toward a standardized common M2M service layer platform: Introduction to oneM2M," Wireless Communications, IEEE , vol.21, no.3, pp.20,26, June 2014

[4] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific american, 284(5), 28-37.

[5] Ben Alaya, M. and al. (2014). OM2M: Extensible ETSI-compliant M2M Service Platform with Self-configuration Capability. Procedia Computer Science.

[6] Kephart, J.O.; and al. “The vision of autonomic computing,” Computer , vol.36, no.1, pp. 41- 50, Jan 2003doi: 10.1109/MC.2003.1160055

[7] M. Ben Alaya, T. Monteil. FRAMESELF: An ontology-based framework for the self-management of M2M systems. Concurrency and Computation: Practice and Experience, Wiley, 2013.

[8] Pérez J. and al. Semantics and Complexity of SPARQL. In The Semantic Web-ISWC 2006. Springer Berlin Heidelberg.