MÁSTER EN INGENIERÍA DE SISTEMAS Y SERVICIOS

PARA LA SOCIEDAD DE LA INFORMACIÓN

Trabajo Fin de Máster		
Título	Distribution of microservices for hardware interoperability in the Smart Grid	
Autor	Jesús Rodríguez Molina	
Tutor	José Fernán Martinez-Ortega	VºBº.
Ponente		
Tribunal		
Presidente	Ph. D. Lourdes López Santidrián	
Secretario	Ph. D. Juana Sendra Pons	
Vocal	Ph. D. Martina Eckert	
Fecha de lectura	17-07-2015	
Calificación		

El secretario:

Universidad Politécnica de Madrid

Escuela Técnica Superior de

Ingeniería y Sistemas de

Telecomunicación

Distribution of microservices for hardware interoperability in the Smart Grid

Author

Jesús Rodríguez-Molina

Tutor

Ph. D. José Fernán Martínez Ortega

July 2015

I'm sick and tired of hearing things

From uptight, short-sighted, narrow-minded hypocrites

All I want is the truth

Just gimme some truth

John Lennon, Give me some truth

Table of contents

Table of figures

Table of charts

Acknowledgements

I would like to show my gratitude to all the people that have helped me with their support to successfully complete this Master Thesis. Without them, finishing it would have been much more difficult.

First of all, I would like to thank Pedro Castillejo, Alexandra Cuerva, David Gómez and José Antonio Sánchez for contributing to have a most enjoyable work environment. It has been very interesting to share opinions about the most varied topics with them and consider points of view that differ from mine. I feel very grateful for having met them and I am sure that life will give them all they want.

I would also like to thank Esther Moreno and Carlos Estévez for their hard work and all the times that, like with Pedro, Sandra and David, we have worked together during difficult times. They deserve the best, and given their motivation, capabilities and tenacity, they are sure to get it.

This Master Thesis is one of the many outputs resulting from the encouragement of Ph. D. José Fernán Martínez Ortega. The responsibilities and the work that he has trusted me with have made me a much better engineer from what I was when I started working in this research group. In the end, my contributions to the GRyS research group result from the chance that I was given by him to be here now.

My Chinese colleagues have also been a source of inspiration due to their relentless work and help. I am sure than Yuanjiang Huang, Xin Li, Ning Li, and Xin Yuan will become great researchers and scientists and will accomplish all their personal and professional goals.

I will like to thank my parents for their support when I have felt more frustrated and stalled (which has happened several times), as well as Corianne, for giving me the peace of mind that I have needed during my lowest ebbs and the positivity and happiness she has brought into my life.

Last but not least, I want to mention my friends, especially Javier López and all the people that I have met in this university school (César, Félix, Adriana, Álvaro, Carlos, Diego, Guille, Randa, Fran), for all the good times that we have spent together.

Resumen

Debido al incremento significativo de la población y su deseo natural de mejorar su nivel de vida, la utilización de la energía extraída de las materias primas mundiales, especialmente en forma de electricidad, ha aumentado de manera intensa durante las últimas décadas. Este hecho plantea un reto de solución complicada, el cual es cómo garantizar que se dispondrá de la energía suficiente como para satisfacer la demanda energética de la población mundial.

De entre todas las soluciones posibles que se pueden adoptar para mitigar este problema una de ellas es de casi obligatoria adopción, la cual consiste en racionalizar la utilización de la energía, de tal forma que se minimice su malgasto y pueda aprovecharse durante más tiempo. Una de las maneras de conseguirlo es mediante la mejora de la red de distribución de electricidad para que ésta pueda reaccionar de manera más eficaz contra problemas comunes, tales como los picos de demanda de energía o previsiones imprecisas acerca del consumo de electricidad.

Sin embargo, para poder implementar esta mejora es necesario utilizar tecnologías del ámbito de las TIC (Tecnologías de la Información y la Comunicación) que a menudo presentan problemas en algunas áreas clave: integración de infraestructura de medición avanzada, interoperabilidad e interconectividad de los dispositivos, interfaces que ofrecer a las aplicaciones, diseño de medidas de seguridad, etc. Todos estos retos pueden implicar una ralentización en la adopción de la red eléctrica inteligente como un sistema para alargar la vida y la utilización de la energía disponible.

En este Trabajo Fin de Máster se sugiere una propuesta para una arquitectura de intermediación que posibilite la resolución de estos retos. Además, una implementación y las pruebas que se han llevado a cabo para conocer el rendimiento de los conceptos presentados también han sido incluidas, de tal forma que se demuestre que los retos que plantea la red eléctrica inteligente pueden ser solventados.

Summary

Due to the significant increase of population and their natural desire of improving their standard of living, usage of energy extracted from world commodities, especially shaped as electricity, has increased in an intense manner during the last decades. This fact brings up a challenge with a complicated solution, which is how to guarantee that there will be enough energy so as to satisfy the energy demand of the world population.

Among all the possible solutions that can be adopted to mitigate this problem one of them is almost of mandatory adoption, which consists of rationalizing energy utilization, in a way that its wasteful usage is minimized and it can be leveraged during a longer period of time. One of the ways to achieve it is by means of the improvement of the power distribution grid, so that it will be able to react in a more efficient manner against common issues, such as energy demand peaks or inaccurate electricity consumption forecasts.

However, in order to be able to implement this improvement it is necessary to use technologies from the ICT (Information and Communication Technologies) sphere that often present challenges in some key areas: advanced metering infrastructure integration, interoperability and interconnectivity of the devices, interfaces to offer the applications, security measures design, etc. All these challenges may imply slowing down the adoption of the smart grid as a system to prolong the lifespan and utilization of the available energy.

A proposal for an intermediation architecture that will make possible solving these challenges is put forward in this Master Thesis. Besides, one implementation and the tests that have been carried out to know the performance of the presented concepts have been included as well, in a way that it can be proved that the challenges set out by the smart grid can be resolved.

1 Introduction and objectives

1.1. General statement

When the increase in population that has been experienced during the last century is taken into account, it becomes obvious that there is a need to satisfy the energy demand for this ever-growing amount of population living in the planet. However, it must be taken into account that the energy resources that are held by the Earth are abundant but finite; thus, they may become scarce in the future and therefore must be used in a more rational, sustainable manner. Otherwise, energy shortages affecting the whole humanity in an unpredictable way are prone to happen. Nevertheless, there are several contributions that can be made to further improve this task and using Information and Communication Technologies (ICT) would be something advantageous, as they are capable of adding some intelligence to a system that will make possible reducing its energy consumption, while keeping the customers satisfied.

This section offers an introduction to the proposal that has been developed as a way to prove that a system can be built to show how an intermediation architecture is used to interconnect several pieces of equipment and provide reliable results. To begin with, the importance of the smart grid, software intermediation architectures and their interweavement in the smart grid is mention in the following subsections.

1.1.1. The importance of the smart grid

Since the second industrial revolution, which happened by the latter half of the nineteenth century, electricity has become the typical output resulting from the consumption of energy resources used to power and activate many different kinds of facilities, equipment and devices. In order to make use of this energy, though, it must be transmitted wherever it is going to be transformed. As wireless, over-the-air transmission of electricity going beyond the concept of inductive power [1] [2] requires further research to be done, the most sensible solution to transfer that generated electricity is setting up a power grid installed and deployed in a wide area that usually ranges from the facility when the very electricity is produced (a task usually tackled by power plants) to the end user facilities. As shown in Figure 1, when energy –and specifically, electricity- is produced, it involves a power plant of differing nature consuming natural resources so that their energy will be used when mobile parts of the plant turn the extracted energy (mechanical in case of waterfalls, chemical from coal or

oil, nuclear in nuclear power plants) into mechanical energy that will activate the movement of a turbine that, either by itself or due to its physical connection to an electricity generator, will turn that mechanical movement into electricity.

Once electricity is generated, it will be transported throughout a power grid with some significant differences in the utilized infrastructure. Commonly, the entity responsible for the provisioning of this power grid, and specifically, the sections of the grid where high voltage power is transferred, is the Transmission System Operator (TSO). As addressed in [3], impact of external elements in the TSO may be of major importance. For instance, depending on the local legislation, regulations or other non-technical parameters, the TSO may also be managing the offer/demand balance of electricity in a more or less wide location. Furthermore, since the final objective of the power grid is delivering energy to the end users (regardless of their activities and overall features, for example, whether they are home dwellers, part of a department store staff, paramedics in a hospital, etc.) there is another entity that commands all the requirements to connect these end users to the power network, which is the Distributed System Operator (DSO). The role played by DSOs, regarding how final customers become plugged to the grid is likely to increase its complexity in the future, as they incorporate a growing number of renewable and distributed energy sources [4].

There are two more entities participating in electricity distribution: the aggregator or retailer and the end consumer/prosumer. The aggregator is involved in managing low voltage power transfer to the places where electrical energy will be turned again into a different kind of energy depending on the needs of the customer. The aggregator/retailer is in charge of purchasing electricity from the TSOs and DSOs, defining the procedure to measure it and how the cost of the consumed energy is charged to the end users. Plus, consumers can be regarded as the most important creators of value within the smart grid, for they will transform the received energy into other kinds that are of critical importance for businesses or public services.

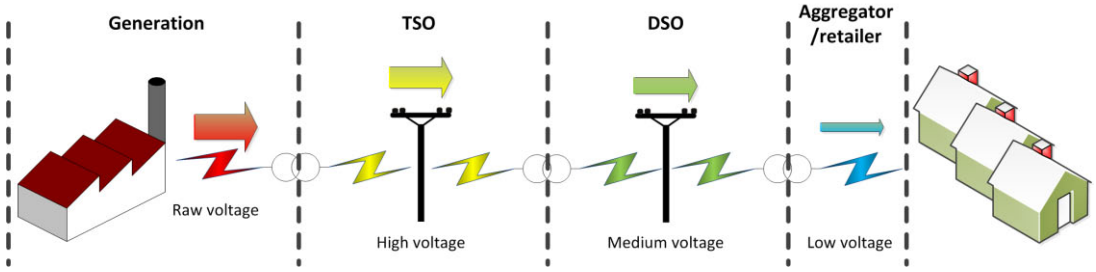

Figure 1. Electric power production and distribution, as shown in [5]

However, it has to be noted that this model of electricity production and consumption has remained essentially unchanged during the lifetime of the power grid installations; while huge breakthroughs have been made as far as energy generation and consumption are concerned, the way to transfer and use electricity have been established as a one-way, static interchanged with energy producers and consumers having roles of very difficult transmutation. In fact, it can be claimed that the basic structure of power grids dates from more than one hundred years ago [6]. Consequently, this model lacks the flexibility that is required to tackle several needs of the power grid, such as:

- Load balance among customers: the systems used to balance the energy demand among customers involved are too clunky and do not fix situations where energy remains unused by one client while another one is having power shortages because the latter is demanding more energy than the one that can be provided by the grid.

- Energy demand peaks: although energy demand peaks overload the power grid and are usually behind electrical accidents in the power grid facilities, currently there are limited ways to make energy usage more regular and shave those peaks in demand.

- Integration of infrastructure used by Renewable Energy Sources (RESs): there is an increasing global trend in usage of renewable energy resources (solar panels, windmills, etc.) that, when used in a local environment, become harder to be integrated.

- Self-production, self-storage and self-consumption of renewable energy: the new pieces of equipment and appliances used to generate energy have become so compact that it is possible to use them by individual or family-level consumers. The aggregation of the produced energy from this side of the power grid is highly challenging in a one-way system.

Consequently, many of the improvement works carried out are more focused on patching a flawed part of the grid rather than renewing the paradigm that is put into practice. Although this is fully understandable on the grounds that the more a system is used, the harder it is to replace it, regular power grid is increasingly becoming a

constraint for the usual development of businesses, public services and any other activity related with energy usage.

Fortunately, many of these issues can be solved by the implementation and deployment of the smart grid. Clarks W. Gelling has claimed that the smart grid is a way to provide intelligence to power systems by means of "*the use of sensors, communications, computational ability and control in some form to enhance the overall functionality of the electric power delivery system. A dumb system becomes smart by sensing, communicating, applying intelligence, exercising control and through feedback, continually adjusting*" [7]. The smart grid has the ability of receiving feedback from all the sensors, as well as data from the Information and Communication technologies that have become encased in it, so additional energy resources can be integrated (such as RESs) and energy is consumed in a more efficient way, as the whole system is more aware of current electricity demand and how to deal with it.

What is more, the rationalization of energy usage saves energy resources that can either be allocated for other tasks or not imported at all, depending on the availability of the commodities in each of the areas where the smart grid is deployed (European Union, for instance, must import the bulk of the oil and gas that are consumed in its territory).

Finally, the inclusion of RESs that are placed in small or medium-sized facilities owned by consumers (such as flats or residential country houses) should be understood as something of a major impact, as the energy users are enabled now to produce energy and pour it into the power grid (thus using it in a profitable manner), store it for their own use or employing it as a complementary energy source for their purposes. Therefore, consumers do not only "consume" energy but also "produce" it, and must be referred to as "prosumers" within the framework of the smart grid. The renewed appearance of the smart grid has been re-drawn in Figure 2. In this case, Distributed energy Generation (DG) plays a role that was non-existent before, as consumers are now capable of producing electricity and influence in energy trading markets as they had not done before [8].

Furthermore, Distributed Energy Resources (DERs) can also be included as a part of the energy that participates in the system; they are usually linked to RESs such as sunlight or wind energy.

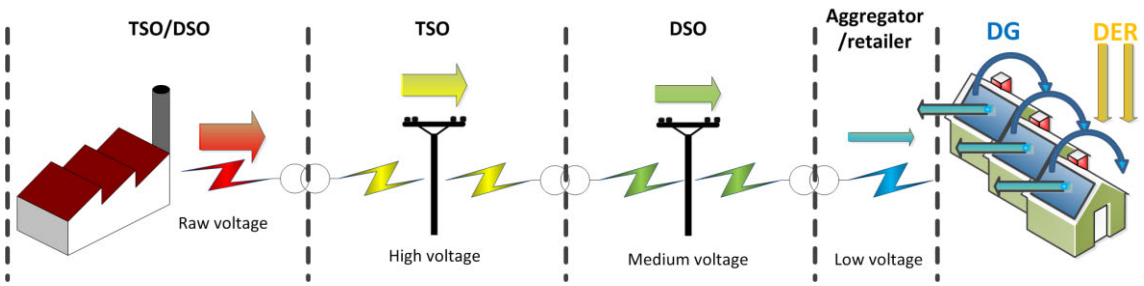

Figure 2. Electric power production and distribution under the smart grid, as shown in [5]

There are many advantages to be taken into account when using the smart grid in comparison with the regular power grid. As it is shown in Table 1, the most prominent benefits of a deployment of this nature are: availability of profuse amounts of data (by means of the sensing and ICT-based applications), participation of prosumers in the energy interchange process, expansion of businesses models to reflect the active participation of prosumers –as opposed to the passive role that consumer-only applications take-, more extensive usage of RESs, a distributed way to produce energy that will rely not just in a reduced set of power plants (with the technical, economical and even societal issues that it may present) and the usage of mechanisms bent on making decisions and taking pre-emptive actions that will ensure the satisfactory delivery of energy.

Although some of this advantages are not immediate (deploying all the required hardware equipment and ancillary software modules takes time, and testing and maintenance operations are required to deal with the new components), the advantages become evident, and even overwhelming in the long term when they are considered, as shown in Table 1.

Table 1. Comparison of power grid features with and without the smart grid

Without Smart Grid	With Smart Grid
Offline, scarce data	Online, abundant data (Big Data)
One-way stream	Two-ways interchange

Without Smart Grid	With Smart Grid
Producers and consumers	"Prosumers"
Static business models	Dynamic business models
Focus on fossil-based, non-renewable energies	Focus on renewable energies
Centralized energy production	Distributed energy production
Weak preventive mechanisms	Strong preventive mechanisms
Little use of Information and Communication Technologies	Widespread use of Information and Communication Technologies
Infrastructure with scarce intelligence	Information inference and decision making features
Reduced amount of participating agents	Potentially huge amount of participating agents

As it can be inferred, the usage of Information and Communication Technologies becomes pivotal for the deployment of the smart grid. However, the usage of this sort of technology brings about another challenge that can effectively disrupt the efforts done to include them, which is the heterogeneity of the devices and software technologies that are expected to be used. The challenges that have to be addressed are as follows:

a) Protocols and communication techniques lack any kind of standardization or a *de facto* predominant protocol that is used in a widespread manner. Thus, whenever data has to be transferred, network protocols from other areas have to be ported to this domain, with potentially suboptimal results in terms of performance.

b) Applications in the area of knowledge of the smart grid are quite defined as far as the front end is concerned (they usually involve Graphical User Interfaces that are interacted by end users), but offer not a standardized, or even easy to

understand manner to connect to all the other parts of the system that are withheld from the end user point of view.

c) Hardware infrastructure is not more homogeneous than the software used in the smart grid. Advanced Metering Infrastructure (AMIs) are the typical pieces of equipment that are installed in dwellings or facilities to measure the usage of energy, but what makes a meter to be an AMI and the functionalities that should assume are not clearly defined and it is often not clear where the intelligence of the smart meters is. In addition to that, open hardware solutions are an appealing solution to prosumers willing to invest time and knowledge to have a meter that can be regarded as AMI (Figure 3, [9] [10]), so they increase the heterogeneity of a possible AMI to a great extent.

Figure 3. Smart meters using proprietary devices combined with Arduino Uno [11] boards

d) The manufacturers selling equipment and devices related to the smart grid (Real–Time Units, Phasor Measurement Units, smart meters) are sold as proprietary equipment with non-optimized resources to interconnect them, thus resulting in interconnectivity and interoperability issues difficult to solve.

Overall, interoperability and interconnectivity of hardware devices by means of networking protocols and software modules is hard to obtain in a seamless manner, and it usually implies additional work for engineers to solve these issues until a reasonable solution is achieved. Clearly, an intermediation layer is required so that the disparity among hardware devices will be unnoticed by the end users, regardless of their consumer or prosumer nature, and services can be provided to applications.

1.2. Intermediation architectures and middleware

An intermediation architecture –or similarly, a middleware or a middleware architecture- can be defined as an intermediate software layer with several prominent functionalities. The first time ever that the concept of middleware appeared was in a NATO document dating back to October 1968 [12]. Later on, it became more widely used as a mechanism to interconnect new applications with legacy systems [13].

As far as those functionalities are concerned, they are basically divided in two groups: on the one hand, the most easily expectable utility that a middleware architecture can provide is abstracting the heterogeneity and complexity of the lower layers (particularly the one dealing with hardware and devices, but also the ones related with connection or connectionless transport layer communications, as shown in Figure 4) so that there will be operations that, while performed in very different manners and using different programming languages and resources, will be presented as homogeneous-looking for the end user or developer that is making use of them. This is how the impression that they are obtained from a homogeneous set of devices will be given to the end users.

On the other hand, there are other functionalities also used as a way to enrich the capabilities of the overall system where the middleware architecture is deployed. For example, device registration can be (albeit it is not a mandatory task) done within the middleware architecture by many different means (databases, registration encrypted files, etc.). In addition to that, almost any software module can be integrated as another component of the architecture, providing in this way a huge degree of flexibility and an ever-increasing adaptability and scalability to the deployment, by adding functionalities to this software layer.

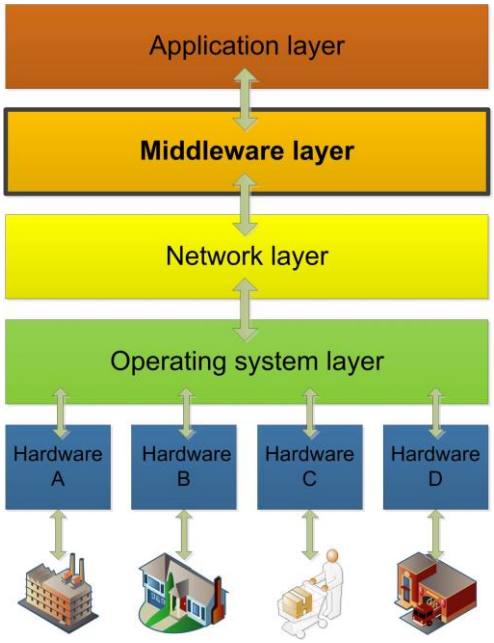

Figure 4. Middleware layer location, as in [14]

There are several features that are fulfilled by the most typical middleware architectures, which will have to be regarded as the minimum functionalities they must offer:

a) **Hardware agnosticism**. The middleware architecture must be usable regardless of the hardware platforms that are being installed in the system. By "hardware platforms" it is implied not only the location where the middleware has been deployed (or more likely, locations, as it can work in a distributed manner), but the devices and pieces of equipment where the information of an overall system is obtained from, since the middleware architecture must be operational regardless of the characteristics used in the devices that are collecting the information that will be afterwards processed by the deployment. This is extensive to the facilities or dwellings where the data collection devices have been located.

b) **Software agnosticism**. At the same time, the middleware architecture used in a deployment must remain functional regardless of the differences of the software that each of the involved devices has installed in themselves (operating systems, local applications, etc.). This is of major importance since the underlying software tier used for intermediation cannot be reconfigured

every time there is a minor change in the software installed in the pieces of equipment used in a system, or if a new application is added.

c) **Data agnosticism**. The data that are transmitted must be done so under any kind of format compliant with the high or low level messages implied in the information transmission processes. Furthermore, the format of the messages that is used for data interchange (eXtensible Markup Language or XML, JavaScript Object Notation or JSON, etc.) should not be a determinant factor that thwarts the performance of the middleware architecture, although it is likely that the different size and features of the interchanged messages end up adding some kind of differences in the performance of the system.

d) **Distribution degree**. The most pragmatic way to develop a middleware architecture is having it as a distributed system deployed in several pieces of equipment, so that the modules it is made of can be more easily adapted to the bottlenecks that some of the hardware devices may impose to the deployment of the middleware architecture. What is more, it can be used as a mechanism to guarantee that the middleware architecture has an acceptable degree of resilience: for example, if some of the pieces of equipment where the middleware is running suddenly becomes unusable (they take some kind of damage, deplete their battery, their safety becomes jeopardized, etc.) the other remaining parts of the system will be able to perform their usual functionalities to an extent. How their performance turns to be afterwards will depend on the level of interweavement that the middleware components have among them. Distribution is one of the most prominent features of the middleware architecture and it is taken for granted in a wide plethora of developments, such as relational databases in distributed data structures [15] or distributed measurement in spaceflight measure ships [16].

e) **Minimum functionalities**. There will be several functionalities that, as explained before, are the ones expected from the middleware point of view to be offered as a minimum. The usual ones will be low level heterogeneity abstraction and application access via intermediation interfaces. Plus, there are some other characteristics that must be taken into account. For instance, security functionalities must be offered, because if a system cannot guarantee some basic security services (such as privacy, data integrity, authentication,

etc.) then the system will be regarded as unsafe and its usage will become discouraged.

1.3. Middleware integration in the smart grid

Integrating a middleware architecture as part of a smart grid (or, to be more precise, as part of a microgrid integrated in a larger smart grid) can be a challenging task due to the multiple works that have to be done to guarantee that it ends up working seamlessly. All in all, the difficulties that may be found mirror the ones that are faced when integration Information and Communication Technologies into the power grid is done (that is to say, when turning the regular power grid into the smart grid). The most prominent challenges that may be found during the inclusion of an intermediation architecture in a system are as follows:

a) There are some elements capable of handling software modules that must be included in the deployment. Depending on the age of the infrastructure where this middleware architecture is going to be included, pieces of equipment of greater or lower complexity may have to be used, thus resulting in an increase of the expenditures that have to be done in the system. Commonly, though, if the power grid where the architecture is going to be installed has already been migrated to the smart grid, changes that will have to be tackled are less significant, as middleware is basically software with low system requirements and devices do not need to be more powerful than regular PCs or laptops are usually required. It must be noted, though, that most of the embedded systems that are used in a microgrid may not be capable of handling a complete middleware architecture by themselves, as they may be either be sold as a product unable to have new software installed due to brand policies, or they might be simply not powerful enough to handle a wider set of services than the ones they have originally been conceived for.

b) The development of a middleware architecture may trigger issues that are usually found while developing software, as there will be several activities that will have to be made (requirement analysis, use cases, design of a proposal, implementation of the proposal, etc.) before any tangible results are available.

c) The interfaces that are used for the applications are likely to be changed, as rather than being connected to the communication network that is used for data

transmission (or even to the very end devices that are offering the information) there will be an intermediation layer that will be transmitting the information to the network and the pieces of equipment.

However, improvements are critical for this kind of development activities because they will guarantee that the benefits of the smart grid become enabled by means of the middleware architecture. The most remarkable advantages regarding the inclusion of it are:

a) Interoperability can be augmented to a significant level. Since the applications and devices will connect to a single layer rather than to a collection of different applications, there will be more pieces of equipment that can be incorporated to the system, regardless of the manufacturer, its services and the procedures that have been used to make the equipment. In the end, middleware architectures deal with the services that are offered by them, rather than the hardware they are made of.

b) Services can be extended. The services that are used in the system can be expanded –as long as the pieces of equipment where the middleware architecture is running do not run out of computational resources- at the middleware layer so that the new functionalities will be stored as a part of it and accessed by providing new interfaces, both to the higher levels and the lower ones. Those services can be aimed in two directions, being one directed into the system (offering functionalities that will offer support to the system performance but not necessarily providing something new for the end users, like access security, service composition, event registration, etc.) and the other one out of it (offering facilities that, in the end, will be regarded as new services for the end users, such as sensor measurement, energy consumption during a specific period of time, etc.).

c) Services can be composed. There are several services that can be offered as obtained from simple ones but by collecting information from them and processing it to achieve a higher of knowledge about a certain condition in the surroundings of the system. For example, if information is obtained about the temperature of one location and the electrical current that travels throughout a set of wires on that location, an assessment on the risk of that location to get fire can be made by combining the temperature-related information and the

current-related data. This way of providing services will be further explored in the middleware architecture that has been put forward in this manuscript.

d) Scalability can be enhanced at every level. Since the consequences of adding a new set of devices can be solved at the middleware level (the tier where most of the changes are done, as it is the place with most of the information located in the system), rather than outsourcing the challenges to the communications layer or the applications that use the system. Flexibility is also improved at the application layer, as applications only need a high-level point to connect them to the middleware architecture and are oblivious to any other edition done to the architecture. Therefore, rather than having to re-program the application again every single time a change is done in the middleware architecture, they will be used the same even if the way to obtain the services from the architecture changes.

e) As it can be inferred from the previous point, applications can be implemented with a different degree of complexity: either they will be using just a single access point to the middleware or they will have some implemented messages or pieces of data that come in handy for the information transfer between the different layers of the system.

All in all, the addition of the middleware architecture layer will guarantee that a) it insulates the upper layer applications from the complications that are coping with devices and manufacturers of different backgrounds and characteristics and b) middleware offers end users or developers a collection of operations that can be accessed with ease.

What is more, adding this kind of development to a system as the smart grid will make possible the implementation of new services of critical importance, as devices at the user´s end (Advanced Metering Infrastructure, homebrew controllers, etc.) will be able to interchange their information and be integrated in the smart grid without any limitation regarding the data format they are capable of providing or their own capabilities. The appearance of the whole system will be as shown in Figure 5.

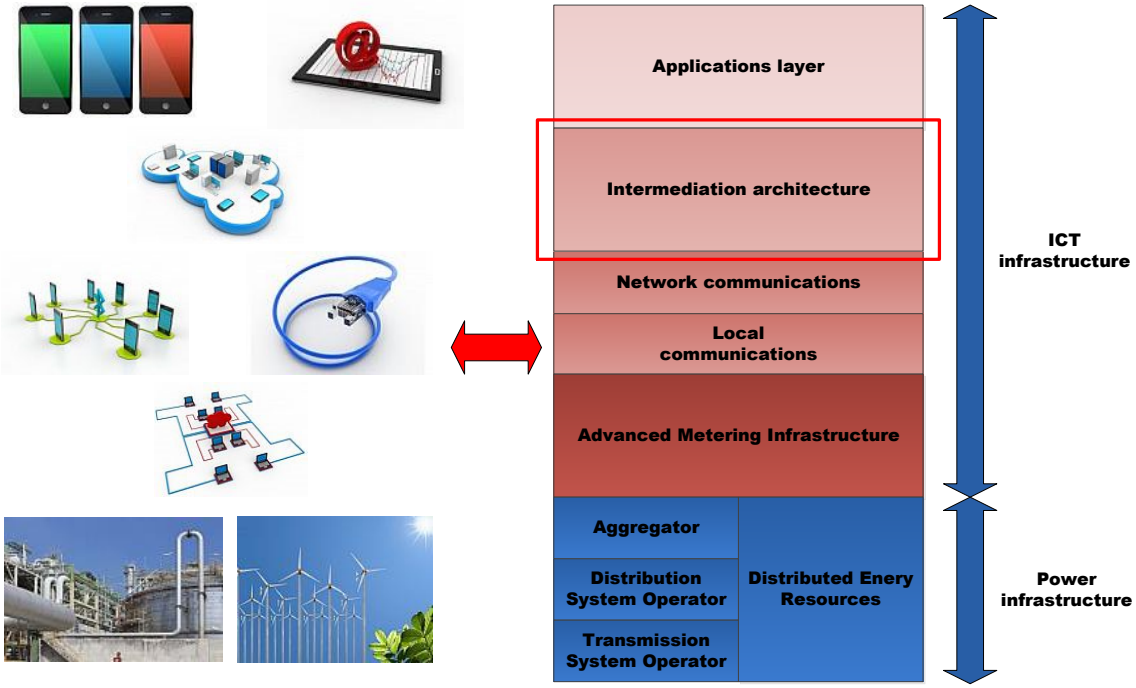

Figure 5. Layered overview of the smart grid, with the intermediation architecture highlighted, as in [17]

As displayed, the intermediation architecture will hold its position in the ICT-based infrastructure that is used to turn the power distribution grid into the smart grid. As long as the information can be transferred from the lower layers with no issues that location will be suitable enough because the middleware architecture requires the information from those devices, as well as the networked communications, to gather the data that is received from the different hardware components of the deployment. In this way, it becomes clear that the inclusion of a middleware architecture is a development that offers major advantages to have it included in a microgrid system, once it has been enhanced with ICT technologies to optimize the distribution and utilization of the electricity produced in a geographical area.

1.4. Motivations, objectives and contributions

The major motivation that has guided the elaboration of this manuscript has been expanding the state of the art and knowledge regarding the area involving intermediation architectures for the smart grid with a proposal aimed to solve the open issues that have been highlighted. In this way, one of the purposes of this Master

Thesis has been gathering the scattered knowledge found while doing the research activities that have been undertaken during the period of time culminating in the fulfilment of this document, so that they would be provided as an ordered work with a tangible result to show after a high level of information has been gathered.

The main objective of this Master Thesis has been the design and implementation works done for an intermediation architecture which has as main focus its utilization as a part of a development for the smart grid. In order to achieve this objective in a satisfactory manner there are some actions that have been taken: to begin with, a study on the state of the art regarding middleware architectures for the smart grid has been done in order to evaluate the strong points and weaknesses of the most prominent proposals done in this area of knowledge. That study has resulted in the acknowledgement of the features of the existing middleware architectures for the smart grid, their evaluation and the extraction of several open issues that hinder the full development of intermediation architectures for this kind of systems. Taking into account these obtained results, a proposal for a middleware architecture to be used for a smart grid has been designed. This design considers the most important use cases that have been analysed, and how they would be satisfied, by means of several Unified Modelling Language (UML) diagrams. These analysis and design stages have been completed and the proposal has been implemented to an extent to guarantee that the ideas that are put forward are done so in a realistic manner. Furthermore, the implementations works that have been carried out have been tested to evaluate the performance of a system and assess whether a deployment as the one that has been made is realistic enough or there are some other practical aspects (such as the software container of the services that have been implemented) that have to be taken into account.

Finally, the contributions of this Master Thesis are several: on the one hand, a proposal regarding a set of services that has not been conceived in any of the ones that have been assessed before has been included here. It is notorious that while some of these services have been implemented by several proposals with success, none of them provide the whole set of them as a unity in their area of knowledge. On the other hand, the implementation and testing works that have been done are a novelty by themselves, as there are several pieces of work that have not been conceived as able to be included within a distributed middleware architecture for the smart grid, nor have many of the implemented services evaluated by using different software tools that allegedly perform the same functionalities. Additionally, it must be considered that a

significant portion of the ideas and contributions that have been written here are part of what is described on scientific papers that have been published as the output of profuse research activities.

1.5. Structure of the document

This document has been structured in a way that makes it easier to read and understand the displayed concepts, following an information flow where each of the chapters can be regarded as stages that must be completed in order to have a good grasp of the proposal that is going to be presented in it.

i) An introduction to the document has already been offered. The purposes of the smart grid, how intermediation architectures are used and why integrating them as a holistic system is a good idea has been discussed in the previous sections. In addition to that, the motivations, objectives and contributions that have been chosen have also been described.

ii) A study on the state of the art of the solutions within the scope of this manuscript has been added in section 2. What is done in this case is describing the most important characteristics of each of the proposals that are studied here, along with an assessment of them that takes into account their degree of fulfilment when compared to the objectives that a middleware architecture is expected to have for the smart grid. This evaluation process has been done by means of a rubric to guarantee the objectivity of the ideas and impressions included in this document. Open issues found as a result of the previous study done on the state of the art are shown too. Basically, the proposals have several flaws that are common in this kind of systems, so obtaining information on how to tackle them is of major importance, since when the proposal done in this manuscript is presented, it is able to suggest solutions to solve this open issues.

iii) Section 3 offers a description of the proposal considering how it has been conceived to deal with the open issues that were presented in the previous section. In order to provide a complete description of it, several detailed UML diagrams have been included to explain how the proposal has been designed, the use cases that it is capable of solving and what components the intermediation architecture is made of.

iv) Section 4 includes information about the implementation works that have been done as a way to offer a framework that can be used to test the proposal and check its performance as far as their main functionalities are concerned.

v) Section 5 provides the conclusions and future works that have been extracted from this Master Thesis (in the case of the former), some manners to expand and improve the work that has been presented here so far (in case of the latter) and the research papers and projects that have been used to create this document as part of the research activities done.

vi) Finally, bibliographic references are offered in the last section so that the readers will be able to check the original sources of data used in this manuscript.

2 State of the art in intermediation architectures

2.1. Problem statement

In order to offer a proposal that solves, or at least attempts to solve the common issues that are found in intermediation architectures for the smart grid, a study must be done previously to find out about the features of the presented works. This section of the document contains the study that has been carried out to fulfil this purpose, the features that have been deemed as important and how they have been assessed. There are two main research papers that have been used as references ([17], [18]). Their resulting work and the references that were used to create them have been the cornerstone of the study on the state of the art.

2.2. Study of intermediation architectures

There are several considerations to be taken into account when facing a study of the state of the art in middleware and intermediation architectures for the smart grid. First of all, features from the different proposals must be evaluated to get a grasp of how well is a presented work fitting the objectives mentioned in this manuscript. The characteristics that have been regarded as critical enough to be evaluated are:

i) **Architecture features.** This characteristic assesses the complexity of the studied proposals, that is to say, the quantity and quality of components that have been included in them. There may be an intermediation architecture that has security or interface-dedicated modules while the other architecture is lacking them; in that case, the former proposal will be considered as superior.

ii) **Semantic capabilities.** This is a feature of special prominence, for it will evaluate the capability of the whole system where the intermediation architecture is included to incorporate information of semantic nature. Semantics can be defined as the capability of offering knowledge, or even wisdom, by means of software languages used to organize the knowledge that has been acquired from a system. As mentioned in [17] "*Semantics allows entities to become aware of the transferred data and consequently, knowledge can be inferred from the transmitted information*".

   Commonly, ontologies are heavily involved in the acquisition of information from raw data that is typically done by using semantics. An ontology can be

described as a container of concepts used within the context of a deployment that encases the information of the system, as well as how they are related one to the other, and how knowledge is inferred from those relationships. In this way, the knowledge that information is trying to provide can be offered to be used either by a device linked to the system or to an application. As shown in Figure 6, a piece of equipment can request the data format that is used for semantic queries.

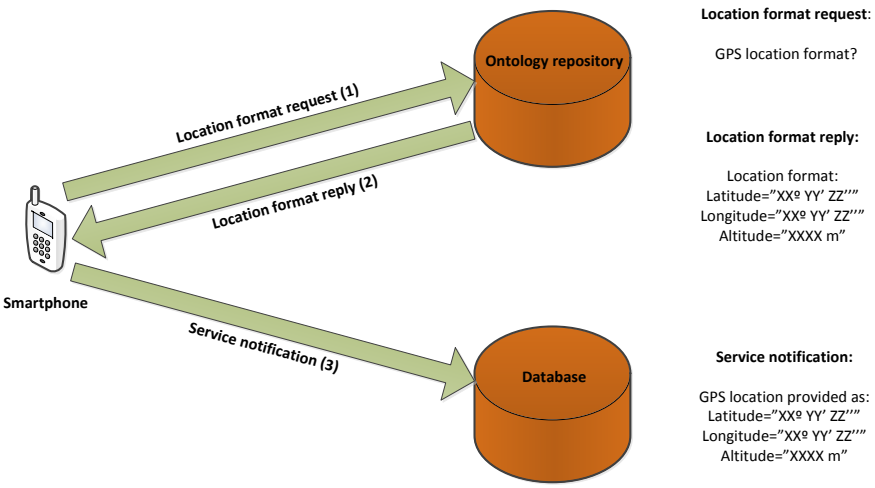

Figure 6. An example of ontology usage, as depicted in [19]

iii) **Information management.** It is also important to take into account how different pieces of information are handled. For example, a proposal that is capable of using different formats of data or metadata as information sources to become integrated in a system is regarded as more interesting than another one that cannot perform the same kind of functionalities. In addition to that, should the proposal be able to integrate different kinds of semantically enabled information, it would be assessed as a major breakthrough.

iv) **Distribution level.** The degree of distribution that the proposals are enabled with (that is to say, the degree of consistency to the procedures of distributed systems, such as data message interchange among several sources of data) is also a matter important enough to have it evaluated. If, for instance, the proposal works locally or in a few machines with plenty of computational resources, it cannot be regarded as a very distributed one. On the contrary, the more devices are implied in the proposal, and with

fewer capabilities, the more interesting it is from an engineering point of view.

In this way, proposals will be evaluated taking into account to what extent they have completed this software implementation model. Four different tables have been created for this purpose; they can be regarded as summarized rubric grading systems for the developments that have been assessed. Table 2 shows the first table that has been used; it shows how the general features of an intermediation architecture are evaluated.

Table 2. Rubric for architecture features

Architecture features	Grade Description
Grade: 5	Relevant design and implementation details are provided about the architecture. Performance testing in a real scenario is provided. Maintenance information is hinted or openly offered.
Grade: 4	Design and implementation details are provided about the architecture. Performance testing has been done by using simulation tools.
Grade: 3	Implementation information about the architecture is provided to an extent. Performance testing has been done by using simulation tools.
Grade: 2	Generic, loose data is provided by the authors of the proposal, such as figures or charts Important information is missing.
Grade: 1	Information about the architecture is non-existent or irrelevant.

A similar table has been created to assess the semantic capabilities of the proposal; its content can be seen in Table 3.

Table 3. Rubric for semantic capabilities

Semantic capabilities	Grade Description
Grade: 5	Semantic capabilities are fully displayed a far as design, implementation and testing are concerned. A way to update the

Semantic capabilities	Grade Description
	ontology used is fully offered too.
Grade: 4	Semantic capabilities are described in terms of design, and implementation. Some minor testing details are provided.
Grade: 3	Information regarding semantics and the ontology used is provided from the implementation point of view.
Grade: 2	Semantic capabilities are somewhat mentioned.
Grade: 1	No semantic capabilities available or mentioned as a future work.

The same idea has been put into practice in order to evaluate the information management procedures of the proposals, as shown in Table 4.

Table 4. Rubric for information management

Information Management	Grade Description
Grade: 5	A full-fledged description involving the information management platform used is provided. Maintenance procedures are also offered
Grade: 4	A very detailed description about the platform regarding its features is provided. Testing is offered as a simulation and maintenance is hinted.
Grade: 3	Descriptive details are offered involving at least its implementation.
Grade: 2	Data of the information management platform is provided as a generic approach.
Grade: 1	Descriptions and valuable data of the information management platform are scarce or missing.

Finally, a last table has been created to have an accurate view of the degree of the distribution level of the proposals. It is shown in Table 5.

Table 5. Rubric for distribution level

Distribution Level	Grade Description
Grade: 5	The system has been conceived as something of a distributed nature from the very beginning, as reflect in the works that have been carried out.
Grade: 4	The system is a distributed one, as from the analysis and design stages was conceived as something of that sort. Tests are done by using a simulated environment
Grade: 3	The system can be regarded as distributed, although some essential information is missing or underdeveloped. Tests have been done on the proposal.
Grade: 2	The system barely classifies as distributed, some important pieces of information (analysis, design, etc.) are missing.
Grade: 1	The system cannot be considered as distributed or no information is provided about that matter at all.

Each of the following subsections contains a description of the proposals that have been found to be relevant enough within the scope of the one that is described in this manuscript.

2.2.1. GridStat.

Gridstat can be regarded as one of the most ambitious proposals about including a middleware/intermediation architecture in a smart grid-based system. Its authors [20] put forward the idea of having an intermediation architecture as the element that will collect all the data from the hardware devices that are deployed within the context of a power distribution grid. According to the design that has been carried out, there are two separated tiers or planes in the proposal, namely a) the one used for information or data plane, responsible for forwarding data from the information sources to its destination according to the orders that are received from b) the management plane, used for resource allocation and data adaptation under changing circumstances.

Gridstat uses two kinds of interactions to govern the system: command interactions and forwarding interactions. They are transmitted via an event channel,

expected to be utilized as an element capable of providing intercommunication among status routers that are linked to publishers and subscribers under an event-oriented model. Furthermore, several software modules have also been described with the idea of offering an actual implementation of the depicted theoretical concepts, such as 1) a module devoted to electronic product code information, 2) a directory services relying on a repository to perform its functionalities, 3) a data dump module based on the Common Information Model (CIM), an open standard of common usage to specify infrastructures related to the power grid (with a conceptual schema that is periodically update to include elements of an IT environment [21]) and a reader interface module, used for the tag readers that have been employed to test the middleware architecture. Figure 8 depicts the overall appearance of the architecture.

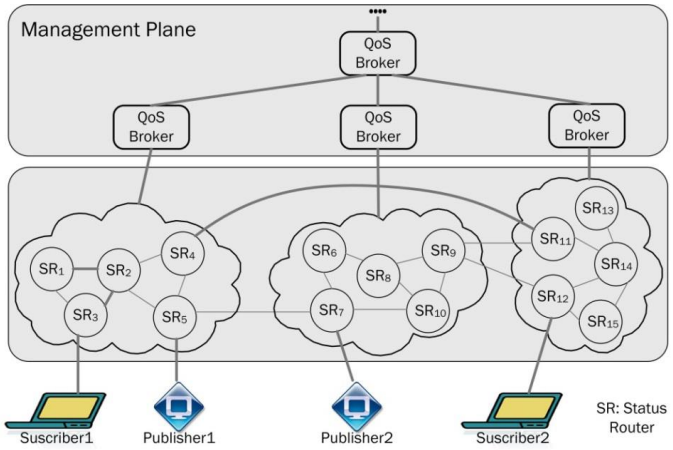

Figure 7. Example of a Gridstat deployment, as represented in [18]

As previously mentioned, this intermediation architecture has been designed and implemented under a publish-subscribe pattern, so when a publisher advertises the availability of data streams to the management plane, a broker used both for service publishing and Quality of Service evaluation will broadcast the existence of that service to all the participants involved in the system, as well as other details, such as the publication rate of the system. One of the main worries of the authors is using this intermediation architecture to deal with any kind of reliability and latency issues that may hinder the performance of the overall system, so the status routers used in the deployment will forward incoming pieces of data throughout the event channels, as stated by the rules commanded by the management plane.

Overall, Gridstat is an accurate proposal in terms of a middleware architecture for the smart grid that relies on communication capabilities and a hierarchized structure

to perform its duties. However, it must be noted that, while the testing activities that were carried out are welcomed, the implied pieces of equipment may be way more powerful than the ones that can be found as end user appliances, either in some kind of facility or in a regular dwell. Besides, semantic capabilities are not mentioned in the paper, so semantic annotation features and knowledge inference cannot be expected from this piece of work. The assessment of this piece of work is offered in Table 6.

Table 6. Gridstat assessment

Characteristic	Mark	Explanation
Architecture features	5	Complete description of the architecture. Testing in a real scenario. A Java-coded demo is offered in the project website [22]
Semantic capabilities	1	No semantic capabilities are mentioned
Information Management	5	A separated middleware layer has been conceived from scratch. Code updates are provided as maintenance.
Distribution level	3	The proposal has been tested located in several pieces of equipment, although they may not be as expected in a smart grid.

2.2.2. Service-oriented Middleware for Smart Grid

In this piece of work, the authors suggest an intermediation architecture for the smart grid that will be guided under the principles of service-oriented middleware (which, by proxy, imply a Service Oriented Architecture). Zhou and Rodrigues mention in this proposal [23] that they are able to deal with problems associated to heterogeneous services by conceiving their architecture as a service-guided, user-centric one. What is more, it is claimed that several design principles have been considered, like independence from any kind of hardware, portability and interoperability. These features are indeed of major importance when doing a requirement analysis and design for a smart grid. The main interest of the proposal is managing a heterogeneous service infrastructure able to operate with devices with varying purposes.

As shown in Figure 8, the infrastructure proposed by the authors is made up by three levels or parts: a transmission part, a control part and a user part (as shown in Figure 8). The transmission part can be further divided into three smaller modules (generation, communication and distribution) and has been conceived for the adaptive data transfer of smart meter infrastructures. On the other hand, the control part is used as a go-between located between the other two layers. This part uses a mechanism that, among other functionalities, is focused on managing the communications of all the implied devices in a microgrid, as well as providing a degree of Quality of Service. Finally, the user part provides a mechanisms to offer what the authors refer to as "experience improvement at the user part", which involves jitter performance evaluation, delay or reliability, which are mandatory for the usage that is provided by the application layer.

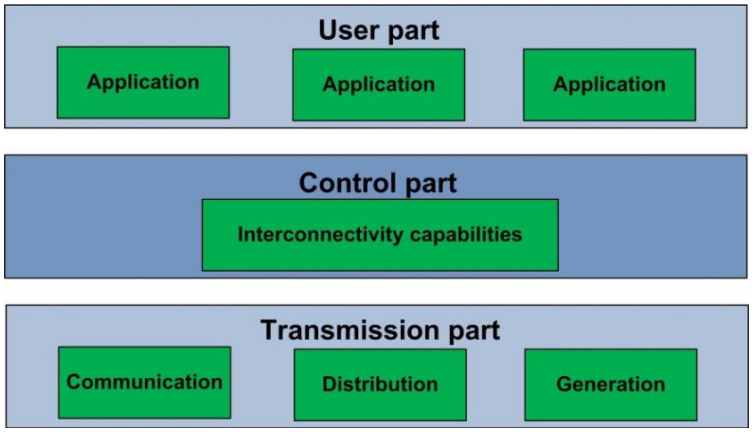

Figure 8. Service-oriented middleware archtiecture, as in [18]

Among the aforementioned applications, the authors stress the importance of taking into account a variety of them, ranging from spectrum efficiency to cognitive radio-based applications. This proposal has been tested by measuring different stages involved in communication operations –access control, message transmission, power allocation and service quality- by means of a network simulation and statistical data have been obtained as a result.

In a nutshell, the proposal is adequate for the regular middleware functionalities that can be expected, and the authors mention the underdevelopment of intermediation architectures for the area of the smart grid. Unfortunately, there is not a semantic description procedure that is used in the proposal, nor there are other functionalities that can be regarded as of major importance, such as context awareness, let alone using semantic features to offer information related to the context. Plus, it must be born

in mind that the obtained results have been done so by using a simulator rather than an actual deployment. Last but not least, the wireless standard that has been utilized as part of the work done (802.11b) has overwhelming requirements for low capability devices that are likely to be present when performing data gathering or information harvest; these latter devices are more likely to use standards as 802.15.4. The evaluation of this proposal is displayed in Table 7.

Table 7. Service-oriented middleware assessment

Characteristic	Mark	Explanation
Architecture features	4	Detailed explanation of the architecture. Tests run in a simulator.
Semantic capabilities	1	No semantic capabilities are mentioned
Information Management	4	Separated middleware layer with differentiated functionalities
Distribution level	3	Distribution is plausible, but AMI may present challenges

2.2.3. Ubiquitous Sensor Network Middleware

The authors of this proposal [24] mention sensor networks as agents of choice able to provide information. From the authors´ point of view, there are several functionalities that any system that is using a middleware architecture should be able to take for granted –Quality of Service, data filtering, security- that can be successfully handled by using their Ubiquitous Sensor Network Middleware (hereinafter it will be referred to as USN). According to the approach made in this proposal, middleware should be divided in three different sub-levels that, while having different software components, will use a common security manager. The lowest level is used for hardware device interactions: a sensor network common interface and a sensor network monitor are used are used here. The second level has some semantic capabilities, namely, a sensing data mining processor, a context-aware rules engine (so that different behaviours will be inferred by the system while, at the same time, the context they are involved in will be considered), as well as an event processor and the security manager present in every level of the proposal. The third an uppermost layer is the one that is in direct contact with the applications that will access the middleware, that is to say, requests and responses done to/from Advanced Metering Infrastructure,

distributed energy generation, supervision, demand response control procedures. The appearance of the architecture is as depicted in Figure 9.

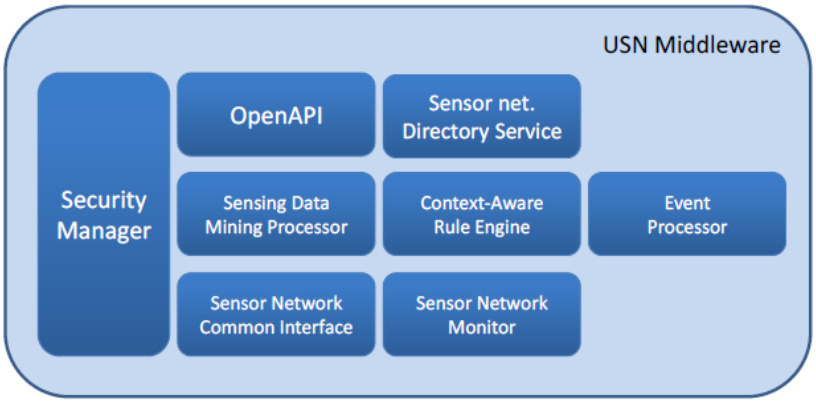

Figure 9. USN architecture as depicted in [18]

As far as the permormance of this architecture is concerned, it interacts as a messenger between the network layer and the applications that are installed at the application level. In order to implement the different functions and business intelligence that are required, Open Service Environment (OSE) is used. Power Line Communications (PLC) can also be used in this proposal to provide a communications network. In the end, USN is a proposal with a more hollistic approach than the others, as it attempts to integrate low capbility devices and semantic characteristics in the middleware architecture that describes. However, there are some drawbacks that must be taken into account. To begin with, the system strongly relies in another middleware proposal named COSMOS (Common Systems for Middleware of Sensor Networks). While this is not an unwanted feature by itself, COSMOS was not conceived for its usage in smart grids, so some of the activities done to develop it can result in a port rather than a design done for the smart grid from the beginning. Moreover, details on the middleware components that have been developed and what specific functionalitites are performed can be rconsidered as scarce. The evaluation of this proposal is done in Table 8.

Table 8. USN assessment

Characteristic	Mark	Explanation
Architecture features	4	A thorough description of the architecture is provided. Testing and maintenance details could be improved.

Characteristic	Mark	Explanation
Semantic capabilities	3	Semantic capabilities are described as a component in the second level of the architecture.
Information Management	4	The architecture seems capable of handling devices of different capabilities. It uses a non-smart grid-based solution as a source of data
Distribution level	3	Due to the modular nature of the proposal. Components could be used to communicate several different machines. Little information is provided about it, though.

2.2.4. OHNet (Object-Based Middleware for Home Network)

The authors of OHNet (Object-based Middleware for Home Network) [25] put forward an intermediation architecture that interconnects objects of different nature so that they are capable of establishing communications among themselves. Usually, these pieces of equipment imply home devices at one end and hardware components at the other one. The authors of this development claim that it can be employed to schedule home consumption during off-peak periods of time, resulting in optimized ways to consume energy (one of the main objectives of the smart grid). As it happens with other proposals, OHNet is structured in three different layers: Network, Library and Application layers; each of those layers is further divided in other software modules.

Firstly, the Network layer employs a Virtual Network Adapter (VNA) responsible for abstracting the different features of each of the protocols used at lower levels. In order to do so, VNA uses a Device Routing Table (DRT) for the kind of protocol that is used or a device identifier. Secondly, the Library layer offers information about the hardware installed with regards to three different modules (Object Discovery, Object Management and Connection Management modules) and four different objects (State, Function, Control and Streaming Objects). Lastly, the Application layer is located at the top of the architecture and is expected to provide an API to the users and developers

interested in including this intermediation architecture solution in a microgrid, as well as the services it is capable of providing (Initialization, Discovery and Description). The appearance of the elements included in the architecture can be seen in Figure 10.

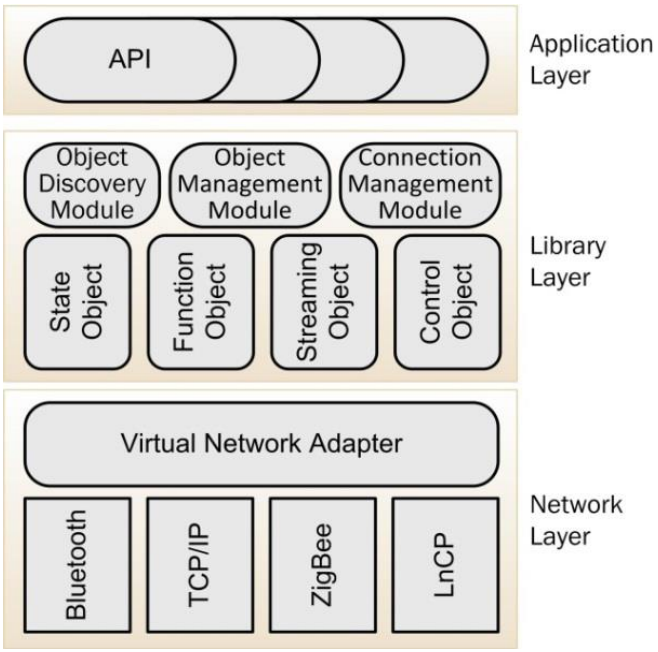

Figure 10. OHNet architecture appearance, as shown in [18]

As for the testing activities that have been carried out for this architecture, physical devices have been used for this task. Specifically, embedded boards depicting a heater, a clock, a laptop and smart phones have been considered here. Plus, connectivity was guaranteed at different levels, either by Bluetooth protocol or TCP/IP architecture. Since smart meters are equipped with Device Routing Tables, they are able to send request messages (for instance, they can ask a laptop for electricity data) that will be forwarded to the suitable piece of equipment to provide the answer.

When all is said and done, OHNet proves that an intermediation architecture can be used to provide interconnectivity for different elements of a smart grid, especially if the testing done with actual devices is born in mind. Unfortunately, there is no information about semantic capabilities (it is assumed there are none) and the proposal seems to have been conceived for a residential environment, and it is yet to be known if it would be possible to have it porter to a different environment (factories, department stores, etc.). This proposal has been evaluated in Table 9.

Table 9. OHNet assessment

Characteristic	Mark	Explanation
Architecture features	5	The design has many details. Tests have been carried out in actual devices.
Semantic capabilities	1	No semantic capabilities are mentioned.
Information Management	4	Complete description of how to use the solution. No maintenance plants are described.
Distribution level	2	Limited to home environments.

2.2.5. MDI (Meter Data Integration)

The authors of this intermediation architecture [26] refer to it as a development capable of integrating the functionalities of Advanced Metering Infrastructure in a microgrid; in fact, they speak about a "Unified Solution for Advanced Metering infrastructure" that uses this proposal as the layer to guarantee it, in what is named as Meter Data Integration (MDI). As it can be inferred from its name, its most prominent function is unifying the information that is transmitted from Advanced Metering Infrastructures and Distribution Management Systems (DMS). Fulling this purpose, though, requires facing issues in information models, communication protocols and the final location of the MDI layer.

Given that the MDI layer is expected to interconnect AMIs and DMSs, it must be capable of intercommunicating differing data models and communication protocols. One interesting idea of this proposal is changing its implementation depending on the software tools that are used in a deployment (note that changes are introduced in the implementation of the proposal, rather than the analysis and design phases), which may range from Enterprise Service Bus (ESB) to a Supervisory Control and Data Acquisition (SCADA). Three are several components working in this proposal: AMI and DMS adaptors (which are used to acknowledge the information obtained from the end devices they are connected to), an Information Translation and Verification Structure. The Information Translation and Verification Structure is provided along with some structures or facilitates, such as a messaging infrastructure called Loosely Coupled Event Infrastructure. Last but not least, an MDI Monitor is also enabled to track the

real-time status of the components belonging to this MDI layer. Figure 11 shows the appearance of the middleware layer conceived by the authors.

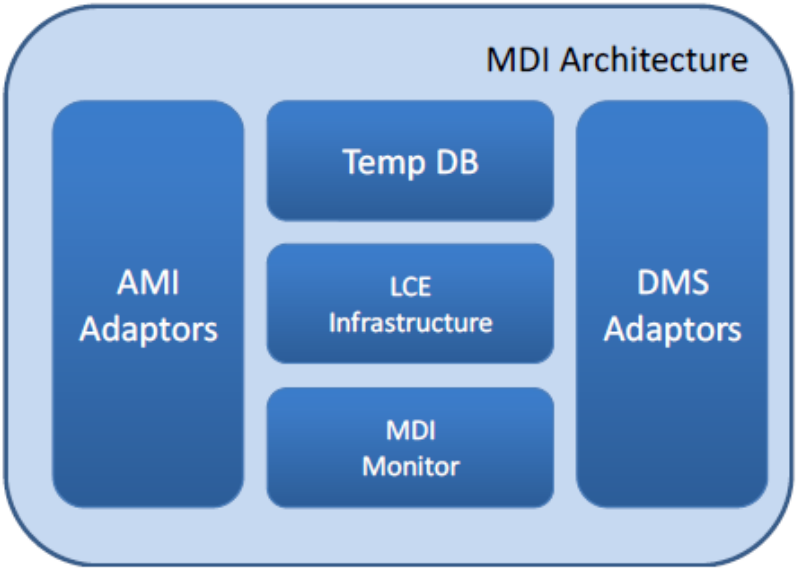

Figure 11. MDI architecture appearance, as depicted in [18]

The tests performed to evaluate the performance of the proposal were done so taking into account several features that it was able to provide: performance, scalability, extensibility and adaptability. The functions of the MDI layer are expected to cover three different kinds of actions: DMSs polling information from AMIs, AMIs publishing meter data to the DMSs and DMSs pushing control commands. Among other sources, the control information provided about the operations at the MDI layer has been taken into account for the tests, and SCADAs, Web Services and MDI have been used as the tools to perform queries and responses.

There are several ideas of this proposal that should receive consideration, like using a software tools as an ESB, which is widely employed for interoperable, distributed solutions. However, judging from the content of the proposal, it seems that has purely been used as a way to integrate data of different nature. While it is a remarkable achievement, the capacity that ESB systems have to send messages from one device to another one is not put into practice here, so it is not a proposal with a strong distributed content. Alas, semantic capabilities are not mentioned as part of the proposal, so knowledge can be hard to be inferred according to the specifications of this system. Table 10 shows the assessment that has been done to the proposal.

Table 10. MDI Assessment

Characteristic	Mark	Explanation
Architecture features	5	A significant amount of information is provided by the authors of the proposal. Tests have been carried out with actual equipment.
Semantic capabilities	1	No semantic capabilities are mentioned.
Information Management	3	It has been conceived as a separated layer, but the description is confusing sometimes.
Distribution level	3	Distributed tools are used, little mention is done on the message interchange between same level middleware entities.

2.2.6. IEC 61850 and DPWS Integration

Sucic *et al*. [27] put forward an intermediation architecture that is characterized by the integration of two different standards, being one of them used for the design of devices enabled with Ethernet network characteristics found in industrial environments and power systems (IEC 61850 [28]) and the other one involved in pieces of equipment equipped with web services (Devices Profile for Web Services, DPWS [29]). The authors of this proposal consider the latter as the cornerstone to build middleware architecture, so by using DPWS the proposal offers a standard-compliant, event-driven Service Oriented Architecture for semantic-enabled smart grid automation. As far as the very proposal is concerned, usage of IEC 61850 makes possible the definition of an automation architecture that can be used during a prolonged period of time. The implementation has been done so with three functionalities in mind: semantic data modelling (with an application scope built with certain data sets and functional constraints), data-exchange services (which use several data models for vertical communications) and some more characteristics for engineering and managing IEC 61850 systems, such as XML-formatted files that describe systems according to the System Configuration Description Language). Semantic additions of IEC 61850 are also praised by the authors of the proposal, as Abstract Communication Service Interface (ASCI) in order to link abstract services of both application level implementations and IEC 61850. However, usage of DPWS becomes as something advisable to do, as the components of the former standard are becoming obsolete due

to fast-paced smart grid development. Specifically, DPWS is used for event-driven and service-oriented architectural purposes. In the end, the authors of the proposal support the concept of ASCI integration into an intermediation architecture that, at the same time, makes use of IEC 61850 so as to provide Service-Oriented Architecture. The way that this proposal is specifically implemented can be seen in Figure 12.

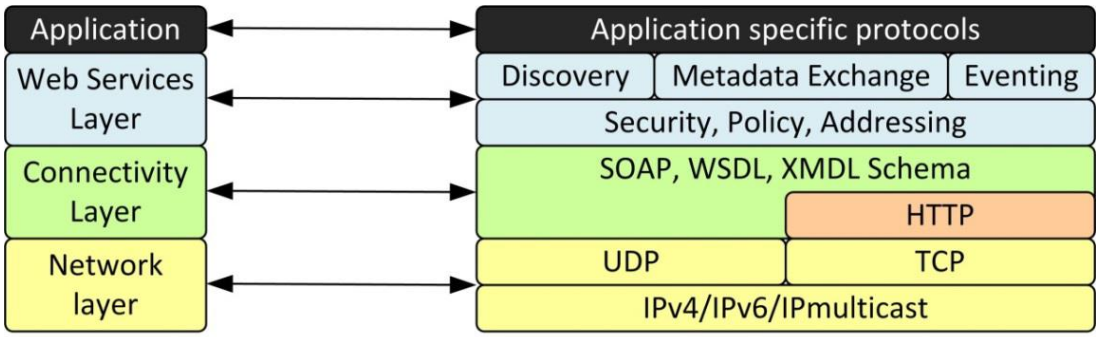

Figure 12. DPWS + IEC 61850 specific implementation, as in [18]

The services that have been involved in testing activities are basically three: a) one has been employed to manage settings or Report Control Block (RCB) objects at the server end, b) a second one is used to retrieve those settings and c) one third service that supports a data delivery mechanism. DPWS device functionality description makes use of WS-Metadata Exchange as the procedure to retrieve metadata.

Overall, this proposal makes a significant leap forward in terms of access (providing interfaces for the application layer) and semantic capabilities. In spite of it, there are some weaknesses that must be taken into account: the model of Web services, although useful by its own right, is quite demanding in terms of computational resources (DPWS presents the same issue), so integrating them in low capability devices that may be present in a microgrid deployment may pose a difficult challenge. The assessment of this proposal has been summarized in Table 11.

Table 11. DPWS + IEC 61850 assessment

Characteristic	Mark	Explanation
Architecture features	4	Detailed explanation of the functionalities offered. No plans for the proposal maintenance.

Characteristic	Mark	Explanation
Semantic capabilities	3	They are added as part of the development works done in the architecture. Ontologies or semantic annotations are not mentioned as part of them, though.
Information Management	5	A complete description about the proposal levels of work is provided. Obsolescence is attempted to be tackled.
Distribution level	2	Distributed capabilities are hinted from the technologies used but not fully demonstrated.

2.2.7. IAP-INMS

The name of the proposal comes from the company where it has been developed (IAP) and the acronym used for Integrated Network Management System (INMS). García *et al.* [30] offer their own example about a distributed intermediation architecture used to govern the hardware components that are found in a microgrid deployment. An Integrated Network Management System is offered with the implementation of software agents capable of providing their functionalities in an event-based, real-time middleware architecture. This latter architecture is described as a requirement with the purpose of interchanging data among the implemented agents, as well as tasks related with control functionalities.

There are several functional blocks that have been conceived for the proposal that match the architectures that have been reviewed for their own development activities: fault handling, performance management, events and alarm management and integration capabilities. These functional blocks result in several functionalities (statistics generation, authentication, status monitoring, authorization, smart services provisioning, interface activity recording and audit, time alignment, etc.) that will have to be satisfied for the end user. As mentioned before, agent paradigm is utilized here by enabling what are considered as the main expected functionalities of software agents (social capabilities, autonomy, proactive intelligence, temporal continuity, mobility, rationality in global goals and learning ability) conceived for service implementation.

As far as the architecture at large is concerned, it has been designed in a typical fashion involving three different levels or layers: a) there is a Network Mediation layer (processes data transfers from/to the smart grid devices while it establishes connections with them), a Management Application layer (composed by the applications that run in backend systems) and a Middleware Communication Services layer (located in between the other two and intercommunicating them). The authors claim that these layers have been tested on an access network running on top of an IP service using BPL devices. According to these tests, there is a deployment consisting of several AMI concentrators that have 10 different variables monitored, which are polled every 15 minutes to collect data. In addition to that, the tests have been carried out by means of a low bandwidth access network between the intermediation architecture and the AMI concentrators. Other tests that have been performed involve the application management layer and some other deployed applications at the backend side (performance monitoring, usage data collection, alarm management, provisioning, etc.).

The intention of adding characteristics as software agents and semantics in this proposal and using an actual deployment of hardware devices to test the reliability and viability of the system must be praised. Unfortunately, there are little to no mentions done about how ontologies and the inference engine is used, nor there are other services of major importance present, such as the ones related with information security. Besides, low capability devices are not considered under this proposal, except for the usage of low bandwidth (128 Kbps) that is not that specified. The assessment of this proposal has been summarized in Table 12.

Table 12. IAP-INMS assessment

Characteristic	Mark	Explanation
Architecture features	5	A detailed description of the features of the architecture is provided. Tests have been carried out in a deployment with actual hardware devices.
Semantic capabilities	3	Semantic capabilities are mentioned as part of the proposal, although there are no data about the inference engine
Information Management	3	It has been conceived as a separated

Characteristic	Mark	Explanation
		intermediation layer from the beginning. Some features of the architecture could be improved.
Distribution level	2	Taking into account the technologies used during the tests, distribution can be hinted, but there are no explicit mentions.

2.2.8. Self-Organizing Smart Grid Services

In this proposal, the authors [31] put forward a proposal that they claim is capable of organizing itself in an autonomous manner, hence the name of Self-Organizing Smart Grid Services. Instead of offering information about a middleware architecture, though, they provide an algorithm that may be placed at the core of a smart grid-based intermediation architecture. In this proposal, the authors stress the importance of having self-organizing services for an environment like the one in the smart grid: a) autonomous behaviour from the participating nodes is guaranteed, b) adaptive adjustment can be done, c) service reliability can be improved in unreliable environments, d) maintenance requirements can be minimized, e) work can be done under conditions where interaction patterns are not possible and f) scalability can be provided.

When considering design and implementation stages, this proposal has been subdivided into two layers with different purposes: infrastructure level and decision level. A middleware architecture is expected here to be used at the infrastructure level to display services to the upper, external layers, such as routing, data aggregation, data filtering and data replication. In a standard deployment, the infrastructure level will receive data from the decision level with the idea of making a decision involving infrastructure components; cloud resources can be used if they are required to finish the duties that have been assigned to the middleware layer. At the same time, Decision level uses a meta-model bent on providing semantic capabilities for the processes enabled at the infrastructure level. Decision level can be subdivided in four modules: required information (so as to define the kind of data that are required), design process (where actions to solve any kind of issue with decision-making mechanisms are taken into account), distributed database (in order to retrieve the information required) and service controller (in case a service has to be triggered to obtain the information). In the

end, the proposal will have an infrastructure level that receives directive metadata from the decision level, and the latter will be used to request information to the infrastructure level to take decisions regarding needed information, design processes or a distributed database deployed in the system.

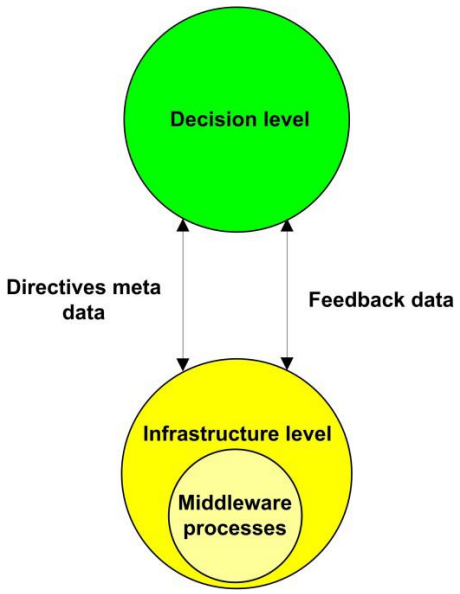

Figure 13. Overall structure of the self-organizing smart grid services, as depicted in [18]

A use case is put forward with the aim of testing the proposal (although it is not clear whether it is a simulated or an actual one with physical pieces of equipment). In this use case, a smart grid is deployed and electricity is supplied by means of different energy sources. Should there be a switch involved in the power transfer that results damaged, the power grid will reconfigure itself to carry on providing power supply, rather than having end users requesting the power company about the incidence. There are several parameters that the authors consider to evaluate the quality of a self-organizing service: degree of scalability, degree of robustness (assesses adaptability and resilience), target orientation, degree of emergence, flexibility, reliability and degree of parallelism (how nodes join or leave a system from different sides at the same time).

Overall, using a dynamic meta-model to be leveraged by a smart grid is a compelling idea, but the authors only offer the description of their algorithm and how it could be encased in an intermediation architecture. Furthermore, the concept of using a meta-model is enabled at the decision level, not at the infrastructure one where the

middleware would be located. Finally, as described for other proposals, only scarce information can be inferred from the manuscript about semantic capabilities and semantic annotations. Clearly, the intentions of the authors were including their proposal as part of a middleware architecture, rather than designing a middleware architecture containing their algorithm as one of its prominent components. The assessment of this proposal can be seen in Table 13.

Table 13. Self-Organizing Smart Grid Services assessment

Characteristic	Mark	Explanation
Architecture features	2	An algorithm to be used in a system is described, instead of a whole middleware architecture with more components or services. It is not clear how tests have been run.
Semantic capabilities	2	They are mentioned, but not in a clear way (partly involved in the decision-making mechanisms).
Information Management	4	The algorithm has been described thoroughly, but as a component to be included in an intermediation architecture.
Distribution level	2	A distributed database is described as one technology that can be used; little else is said.

2.2.9. Secure Decentralized Data-Centric Information Infrastructure for Smart Grid

Kim et al. [32] offer a description of the middleware architecture that put forward as a result of their own work, which is claimed to be a secure, decentralized data-centric information infrastructure applied for the purposes of the smart grid. The strongest point in the proposal is that it has been designed to offer features such as security and decentralized data information in the context of a smart grid. This proposal takes into account the infrastructure that has to be used when enhancing the power grid with Information and Communication Technologies: on the one hand, Internet Protocol (IP) is used as a way to establish communications at the network layer. On the

other hand, the authors mention to have dealt with the most usual challenges that can be found about applications using electricity and power as their common ground: distributed data sources, latency-aware data transactions, real-time event updates and security. Like many of the former middleware architectures that have been studied, this one contains three different modules: a non-time critical data event module (used in case that data can be transferred with some latency), a control commands module (so as to harvest control information) and a critical data event module (involved in a distributed network storage system). In this way, as depicted in Figure 14, the middleware layer is effectively located between the network-related infrastructure of the microgrid and the power applications that are used in the environment where the intermediation architecture is deployed.

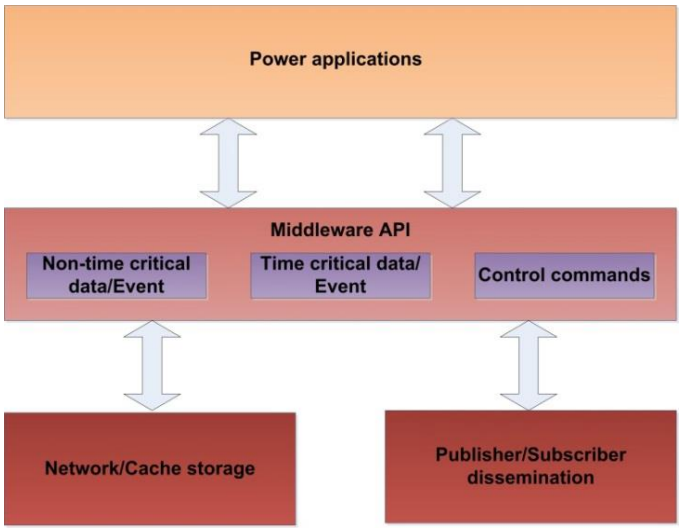

Figure 14. Components and location of the proposal, as in [18]

In addition to the described modules, the authors mention that their middleware architecture has been designed in a way that can also incorporate self-healing and self-configuring capabilities. The idea on decentralization is also used as a tool to solve issues related with information transfer (hardware behaving as a bottleneck for the system, scalability). Some other challenges that had to be faced during the implementation stages involve routing, naming and information forwarding. Additionally, Common Information Model is employed as a way to provide a standardized data format. Finally, security is treated as a feature of major importance, since each of the communications channels uses a key securely derived by the users that are part of the communication; this process requires a key exchange to be completed. As it happened with other middleware implementations, a publish/subscribe model is used as a way of

support the development activities of this middleware architecture. As a way to focus on the security capabilities, a lower security grid overlay network enables security functionalities used to prevent distributed Denial of Service (DoS) attacks.

This proposal has some strong points that are unnoticed in others: having a distributed architecture is a matter of major importance for the authors of this piece of work, and security has been fully enabled as an effective service that goes beyond many others suggested in other proposals. In contrast with the interest that security has in this proposal, though, semantic capabilities are not mentioned as featured in the proposal, and how this proposal has been tested or the equipment that is supposed to be used is almost no mentioned all, which may difficult to assess the realism and performance of the system, as reflected in Table 14.

Table 14. Assessment of the proposal

Characteristic	Mark	Explanation
Architecture features	4	A description profuse in data about the proposal is provided. Important features as security are stressed. Scarce information is offer in terms of testing and maintenance.
Semantic capabilities	1	Semantic capabilities are not mentioned in the proposal.
Information Management	5	Descriptive details about the information treatment are provided, many software modules have been implemented.
Distribution level	4	Distribution is explicitly mentioned as an objective of the proposal and the proposal has been designed with that feature in mind. No description of how hardware devices are added those distributed components is provided, though.

2.2.10. Middleware Services for P2P Computing in Wireless Grid Networks (Signal)

Hwang and Aravamudham offer their own contribution in intermediation architectures for the smart grid [33]. The authors share the concept of having a scalable middleware architecture as the basis to provide grid-based services, hence the name of their proposal (Scalable Inter-Grid Network Adaptation Layers or Signal). With the purpose of having this middleware architecture used by devices with some degree of portability or low capacities (as a mobile phone that, while not-so-lacking in capabilities, is not as powerful as a Personal Computer), a peer-to-peer architecture is suggested as a model to be deployed throughout the power grid. One of the main influences for this proposal has been Globus [34], an earlier project focused on computational grids rather than the smart grid explicitly.

The proposal itself makes use of data prefetching and caching procedures at the middleware level to enhance the performance of the deployment. Quality of Service facilities are also provided (support for resource and service discovery, etc.) and overall, several software modules have been conceived with the target of offering scalable and intelligent resource management. Those modules are as follows: a) a registry/discovery service module (used to communicate devices of different capabilities for service registration), b) a proxy-based layer (with the aim of communicating devices and computational resources by using interconnected proxies) and c) a job computational layer (meant to be used for storage devices, computational resources or memory issues).

This proposal performs its functionalities by sending requests to the remote locations able to answer the requests that have been sent if there are resources that can be allocated to offer that answer at that very moment. Plus, the web extension provided by the Open Grid Services Architecture (OGSA´s) will be used to access the services that can potentially be provided by using web services stored in a UDDI registry enriched with XML descriptions. Once the mobile device that has the piece of the distributed middleware architecture installed has to establish a connection with the proxy, it will be done so via Simple Object Access Protocol (SOAP) and will use additional secure authentication with the implementation of the Generic Security Service (GSS).

As the overall evaluation of this proposal, it must be mentioned how several features of major importance in a middleware architecture for the smart grid are used: the objective of using it in mobile phones gives an idea of the degree of distribution that it has been designed with. However, it must also be noted that it does take into account the addition of semantic value to the information that becomes included beyond web services. Finally, it is not clear how the performance of the architecture would be if there devices less powerful than a mobile phone (as a smart meter built with Arduino or Raspberry Pi [35]). The assessment of this architecture can be seen in Table 15.

Table 15. Middleware P2P services assessment

Characteristic	Mark	Explanation
Architecture features	4	A thorough description is provided in the script made by the authors. There are little details regarding maintenance of the proposal.
Semantic capabilities	1	No semantic capabilities are mentioned by the authors.
Information Management	4	Communication infrastructure is taken into account. Security is enable to an extent.
Distribution level	5	The proposal is expected to be installed and used in mobile phones.

2.2.11. A cloud optimization perspective

Xi Fang *et al.* offer their own ideas for intermediation architectures [36]. This time though, a cloud computing perspective is used to describe the proposal. It consists of four different domains to be taken into account, namely a) smart grid domain b) cloud domain, c) broker domain, and d) network domain. The smart grid domain is governed by three ideas: 1) a data item as the information object unit created by some information sources, 2) computational project as the module that utilizes data items and the output information that has been previously completed tasks and 3) a User able to access the information provided by the Data Items or the output results that are created by the Computational Projects. Additionally, the cloud domain offers one or several clouds containing a collection of different services (for example, different clouds may have different pricing tariffs for their users, as the characteristics of their

energy consumption may vary from one profile to another). Thirdly, the broker domain is the one in charge of mediation between the smart grid domain and the cloud domain whenever there are services that require a data interchange. Finally, the network domain is focused on the network infrastructure and data transmission among the former domains that existed. A depiction of the appearance of this proposal is displayed in Figure 15.

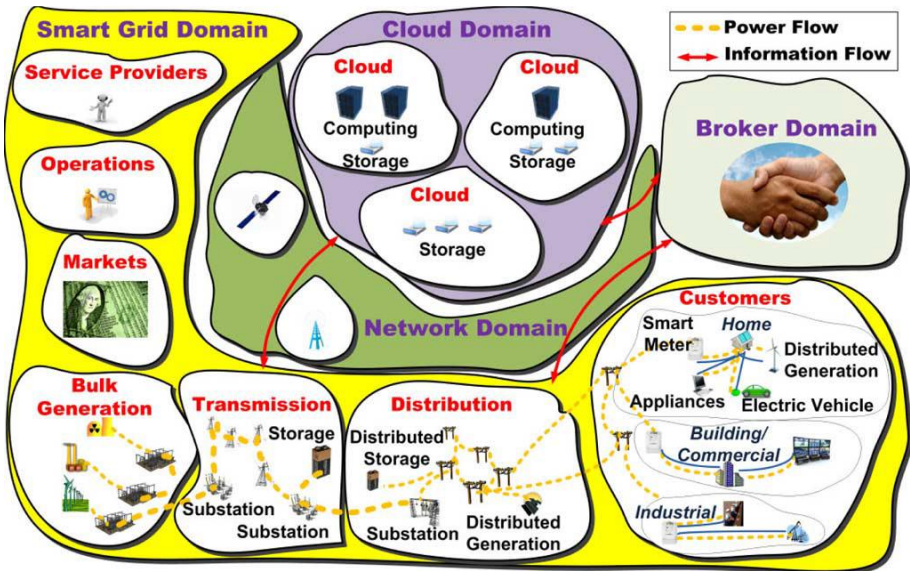

Figure 15. Cloud optimization perspective depiction, as in [36]

This proposal, as far as cloud computing is concerned, proves to be a way to have a distributed middleware architecture able to store significant amounts of data and perform operations that are more complex and heavier than the ones that could be done in an individual machine. Unfortunately, there are some drawbacks inherent to cloud computing systems that have to be addressed. For instance, cloud computing systems tend to be less secure than others that, while being distributed, use other kind of technologies. Due to this fact, they must have installed additional mechanisms as Intrusion Detection Systems (IDSs) [37]. Moreover, there are very few data on how hardware (and particularly, Advanced Metering Infrastructure) is connected to the system or information regarding high level applications. Last but not least, no information regarding semantic features has been included. The assessment of the proposal has been done in Table 16.

Table 16. Cloud optimization perspective assessment

Characteristic	Mark	Explanation
Architecture features	4	A complete description of the components of the architecture is provided
Semantic capabilities	1	No semantic capabilities are mentioned in the manuscript.
Information Management	4	A hierarchy on how to gather different services and applications is provided. Detailed information about the implement features is provided.
Distribution level	4	Cloud computing makes sure that the services can be accessed from devices of very different nature. AMIs are not specifically targeted.

2.2.12. KT's Smart Grid Architecture and Open Platform

The last middleware architecture that is going to be studied is the one that has been implemented by Jisun Lee *et al.* to manage information according to the commercial interests of KT Corporation [38]. From their perspective, the needs and requirements that the end customers imply new functionalities (as mentioned, Distributed Energy Resources, Electric Vehicle Integration, Demand Response Capabilities, Grid Performance Optimization, etc.) that must be satisfied. In order to do so, a platform for the management of energy must be created. This platform must have several facilities regarded as mandatory by the authors: interoperability for networked communications (regardless of whether they are wireless or not), interoperability and scalability for customer services (thus making desirable the usage of web services or a distributed architecture reliant on Service Oriented Architecture principles) and lastly, a business ecosystem should be presented as well, in case the customers turn themselves into prosumers and can play a different, more dynamic role in energy generation and distribution. All in all, the author´s intention is centralizing their proposal in the Advanced Metering Infrastructure that can be used by the end users, ranging from home dwellers to blue collar workers. In this way, they can keep the customers

aware of the energy that has been consumed. The openness of the solutions provided should also be studied (in fact, this feature has been added to the title of the proposal) as it allows a greater variety of customer-oriented services and communications, as well as third party energy services and applications. The overall appearance of the architecture is depicted in Figure 16.

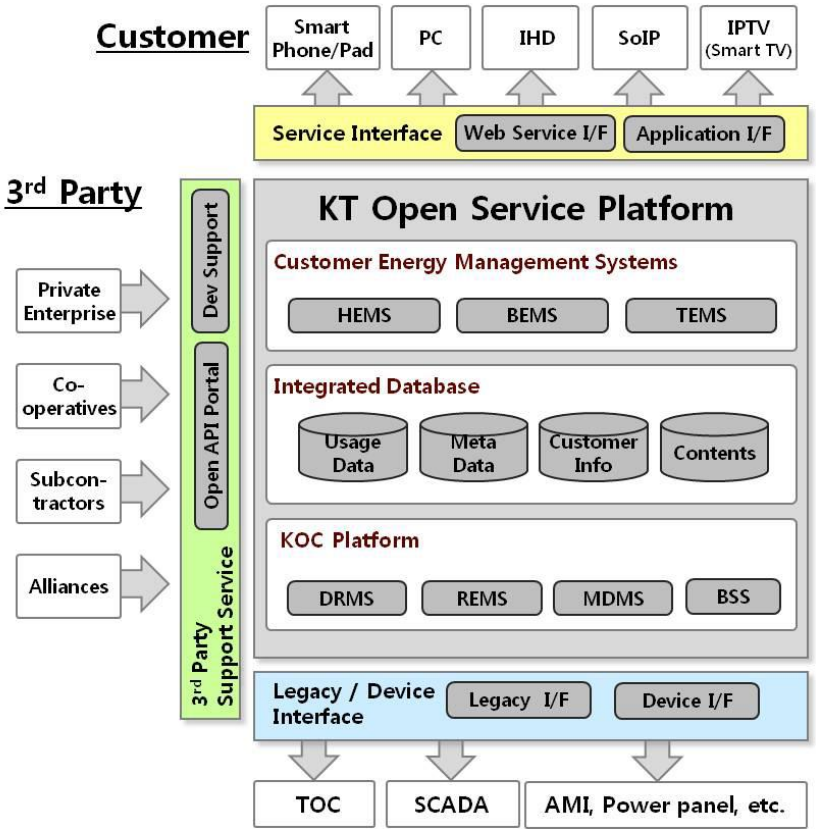

Figure 16. KT´s smart grid architecture depiction, as shown in [38]

As for the evaluation of the proposal, it offers the advantages of an eminent practical perspective focused on an economical exploitation that can be used to obtain a profit, thus guaranteeing its usability and feasibility. On the other hand, the scientific and research merits of the proposal fall somewhat short: a broader point of view that went beyond the Advanced Metering Infrastructure and user applications would be welcomed. What is more, the proposal does not take into account any separated entity that is used for information management and data heterogeneity (that is, a middleware architecture), nor semantics are foreseen to be integrated in any circumstance. The assessment of the proposal is offered in Table 17.

Table 17. KT´s smart grid architecture assessment

Characteristic	Mark	Explanation
Architecture features	5	Detailed architecture that has been conceived for its exploitation (designed, implemented, tested and maintained).
Semantic capabilities	1	Semantic capabilities are not mentioned in the proposal.
Information Management	4	Information treatment is provided, but it is mostly focused on the end-user applications.
Distribution level	2	The technologies used and their nature point at their possible distribution, but it is only hinted rather than explicitly mentioned in the proposal.

2.3. Open issues in intermediation architectures.

As it can be seen from the study that has just been presented, the approaches and viewpoints in each of the intermediation architectures are varied. As a common basis, there are some features that appear repeatedly in each of the proposals analyzed. For example, the intention of dividing the architectures into three levels is often the result of conceiving the middleware as something that must cope, at the same time, with lower, hardware-based equipment and high-level, application-based developments that are offered to the end users. While neither the applications nor the hardware devices are part of the intermediation architecture, it is crystal clear that it must offer some kind of access points for the end-to-end effective performance of the system. In a nutshell, the studied middleware architectures offer different mechanisms to provide the services and functionalities that are expected from a software deployment located in the middle of a system. A chart where all the grades of the studied proposals are summarized has been placed in this document (Table 18).

Table 18. Middleware proposals evaluation chart

Name of the proposal	Architecture features	Semantic capabilities	Information Management	Distribution level	Total score
Gridstat	5/5	1/5	5/5	3/5	14/20
Service-oriented Middleware	4/5	1/5	4/5	3/5	12/20
USN	4/5	3/5	4/5	3/5	14/20
OHNet	5/5	1/5	4/5	2/5	12/20
MDI	5/5	1/5	3/5	3/5	12/20
DPWS + IEC 61850	4/5	3/5	5/5	2/5	14/20
IAP-INMS	5/5	3/5	3/5	2/5	13/20
Self-Organizing Smart Grid Services	2/5	2/5	4/5	2/5	10/20
Secure Decent.Data-Centric Information Infrastructure	4/5	1/5	5/5	4/5	14/20
Middleware services for P2P	4/5	1/5	4/5	5/5	14/20
Cloud optimization	4/5	1/5	4/5	4/5	13/20
KT´s smart grid architecture	5/5	1/5	4/5	2/5	12/20

As it can be watched in the chart, while almost all of the intermediation architectures offer an acceptable result, none of them is astoundingly better than the others, being the lowest score 10/20 and the highest 14/20 (this score is even shared by several proposals). This is due to the fact that there are common flaws that have been learnt after understanding each of the proposals presented. Those weaknesses can be regarded as the open issues present in the developments done so far in this area of knowledge. Even in the proposals that have been regarded as the best ones, these flaws are prone to happen (it must be noted, though, that rather than evaluating each of the proposals according to how good or bad they are, the main interest of the

study has been assessing how close the middleware architectures were to the concept that is held of what a complete intermediation architecture for the smart grid should be). The most common features to be addressed in this kind of software architectures are:

- Semantic capabilities are neglected by almost all the proposals. They are usually not present at all, and when they happen to appear, they are usually done so in vague ways, without any explanation about the inference engine is used, or how events are triggered once the inference engine detects one. Mechanisms on how the system is capable of acquiring knowledge from the collected data are usually not available either.

- The concept of “middleware” or “intermediation architecture” is sometimes not entirely understood and results in developments with blurry boundaries separating the layers devoted to the intermediation architecture and all the other functionalities. It is frequent that the middleware is regarded as a black box where functionalities can be added, regardless of their suitability. In other cases, there is not even a separated layer for the intermediation functionalities that are expected from a distributed system.

- Low capability devices are often not taken into account at all. The issue about this topic is due to the fact that many of the devices and technologies implied at the end mile of the smart grid (the ones that interact with end users) are usually underpowered when compared to a laptop, a Personal Computer or even a tablet or a mobile phone. Therefore, if the capabilities required by a system are overwhelmingly greater than the ones a low capability device is able to offer (like a node from a Wireless Sensor Network or an Arduino board) then the intermediation architecture is likely to fail when it is deployed to a real system.

- Architectures are sometimes regarded as mere ports from a non-based smart grid system. This is often a source of issues, as there are many services, especially those related to energy consumption and forecasting, that cannot be properly codified again, and in the end, porting the service from one architecture to another one results in using more implementation resources than the ones that would have been used if the service had been developed from scratch.

The proposal that is going to be described in the following section attempts to solve all these issues to a certain degree, and provides a good framework to implement distributed services with several capabilities that any middleware or intermediation architecture must have.

3 Proposal for an intermediation architecture

3.1. Introduction statement.

The proposal that has been defined here takes into account the main weaknesses that were detected during the study previously done on the state of the art. Consequently, it tries to address the capabilities and features that are expected of intermediation architectures in a deployment of this nature, that is to say, an architecture enabled with several relevant features (ontologies, security, etc.), semantic capabilities, a differentiated information management and a high degree of distribution. These aspects have been addressed by having a design with the following characteristics:

i) Low capabilities are taken into account. Devices of this nature have been incorporated in the proposal that has been developed to implement the analysis and design of the middleware architecture that is depicted in this section of the manuscript.

ii) The proposal is encased under the paradigm of distributed services. It uses the data that is collected from devices that have been scattered in a deployment done on a certain area.

iii) Security features are included. Privacy is taken into account when retrieving the information that has been previously harvested

iv) Semantic features are enabled. A method to represent data information under a certain format, as well as a light ontology, has been designed for data transmission and storage.

v) An inference engine has been conceived. With the idea of working closely with semantics, a semantic engine is brought into the proposal to obtain knowledge from the information that is gathered from the environment or the pieces of hardware present in the deployment.

A depiction of the architecture the way it has been designed has been added as Figure 17. It must be noted that it has been allocated between the application layer –which is the layer expected to be the place where applications run and use the services provided by the intermediation architecture layer- and the network layer –used to provide network communications at a lower level-.

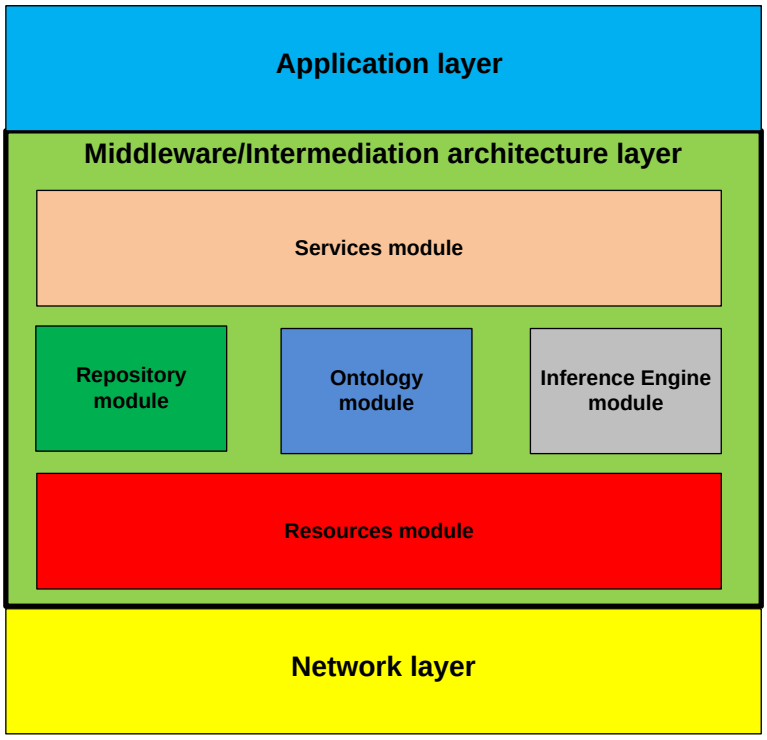

Figure 17. Software modules of the middleware architecture put forward

As it can be seen in Figure 17, there are five different software modules that compose the semantic middleware architecture. Each of them has been conceived as the ones that have to be used in order to solve the pointed out issues:

i) Resources module. This module is responsible for retrieving all the information from lower layers. Despite the immediate level that it encounters is the network layer, it has been designed taking into account the hardware devices that can be found in a deployment and the data format that must be used when information interchanges are done in the context of the deployment of a prototype.

ii) Repository module. The main task of this module is the storage of semantically annotated data required for different duties during the timespan that the middleware architecture is operational.

iii) Ontology Module. It is capable of integrating any new vocabulary resulting from the services that appear as a result of adding new end-user devices to the microgrid.

iv) Inference Engine module. It is able to provide the semantics needed to treat the information that is sent throughout the architecture. This module performs its functionalities in close cooperation with the ontology module.

v) Services module. This model is the closest one to the application layer, as it will behave as the front end of the middleware architecture. In addition to that, it is the software component that will add new services to the middleware architecture.

In order to better understand the capabilities and characteristics of the software modules that the proposal is made of, there are two software procedures that have been carried out. The first of them is a computational analysis, where the different subsystems the proposal is made of, the components each of the subsystems is composed by, and how they interact with each other, are displayed with the idea of expressing the relationships among the different software entities that are part of the proposal. The second one is a functional analysis where the behaviour of the involved parts of the proposal is described. Finally, an example of how the classes required on an Object Oriented programming (OOP) language can be codified is shown as the last piece of information offered in this section of the manuscript.

3.2. Computational analysis

When dealing with the computational analysis of the proposal, the subsystems that are going to be involved in the proposal must be considered, as well as the relationships that are present among them. While almost all of the subsystems will be tightly interweaved one to the other, there are some of them that, in order to establish a connection with another subsystem, will use an intermediary one, mostly due to the components that each of the subsystems is made of.

The subsystem diagram that has been obtained is depicted in Figure 18. As it can be noted, the service subsystem diagram is the one that has more interactions with the other subsystems. This is due to the fact that it is the one in touch the systems that are more related to the semantic capabilities of the proposal. On the other hand, the Repository subsystem has little contact with the other subsystems, except for the one used by the ontology. This does not come as a surprise since the repository subsystem is expected to be used just for the storage of semantic information that has been previously treated by the ontology.

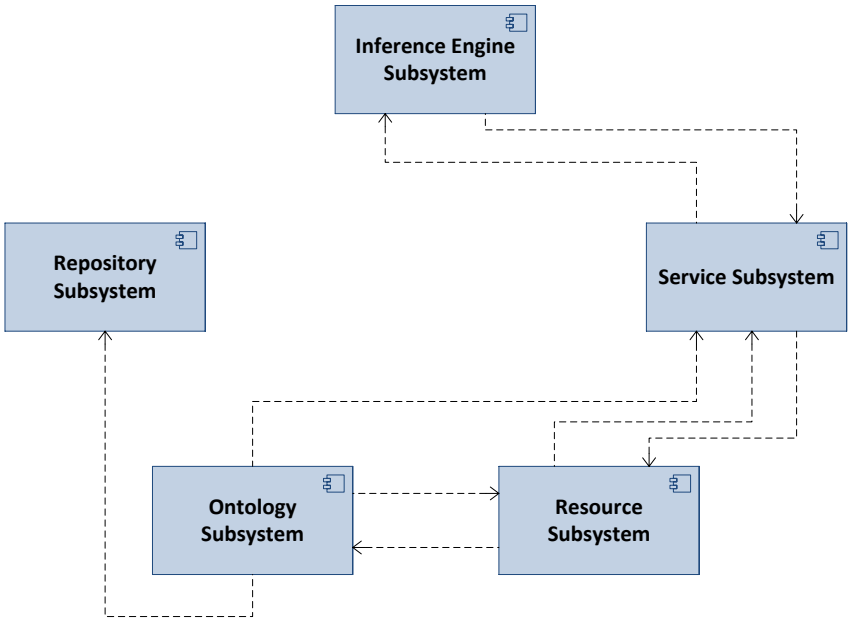

Figure 18. Subsystem diagram of the proposal

Once the subsystems that are used by the proposal have been established, each of them can be studied according to the features that are expected from them. For example, if the Resource subsystem is taken into account, it will be noticed that, since its functionalities involve the interaction with hardware, sensors and actuators from the deployment, there are two components of the system that are used for this purpose. In addition to that, there is one more component used for the devices inputs and outputs that will establish links with the Service subsystem. Figure 19 offers the component diagram of the subsystem.

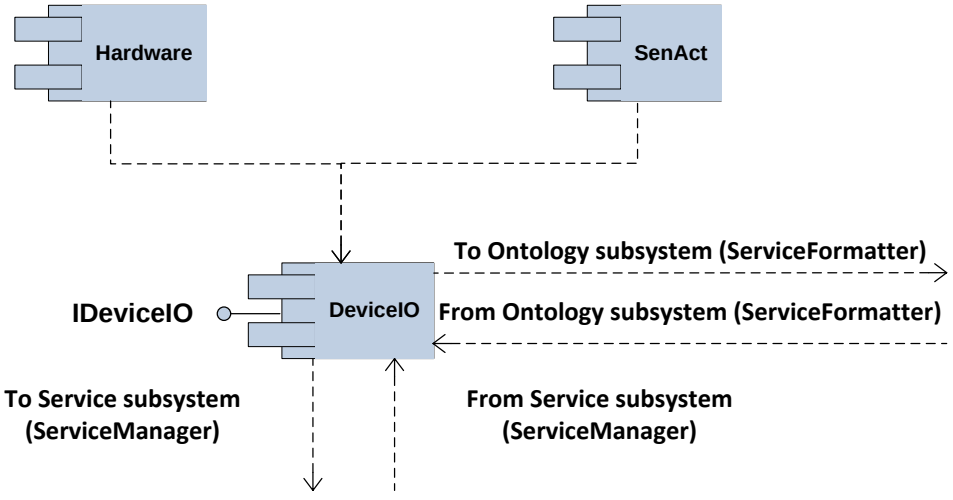

Figure 19. Resource subsystem components

The inference engine subsystem can also be designed from the same perspective. An inference engine collects information from the environment regarding the actions that are taking place, stores information about them in a facts repository and infers actions to be taken according to the rules stored in the Rules repository. If an action has to be triggered, it will be done so by means of an action trigger component. The component diagram of the subsystem is displayed in Figure 20.

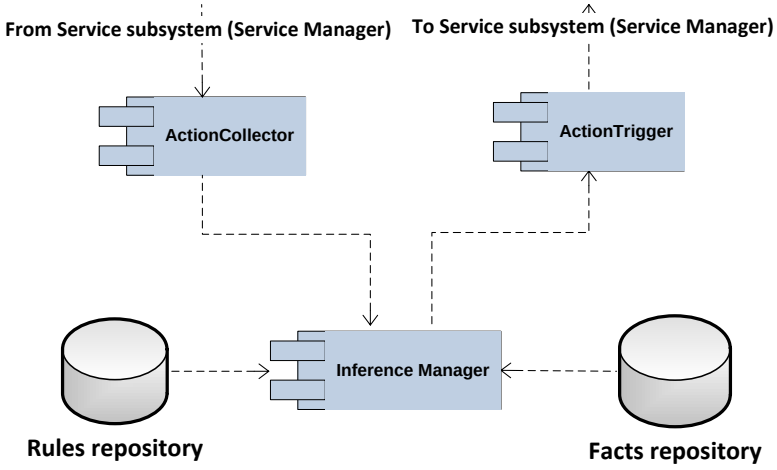

Figure 20. Inference engine subsystem components

The Ontology subsystem has been analysed too; in this case, since its two main functionalities involve formatting the information that is transferred throughout the system and updating the content that is present in the middleware ontology, there are two components that have been created with this purpose, as it seen in Figure 21.

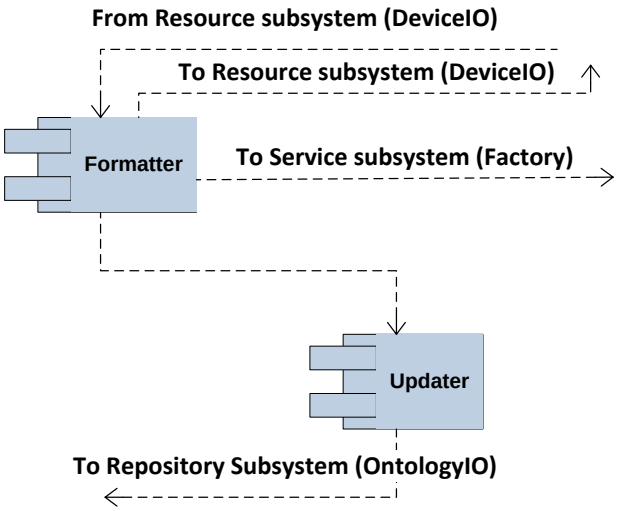

Figure 21. Ontology subsystem components

The service subsystem has to take into account some other parts of the overall proposal. To begin with, there should be a Service Factory that takes into account the information present in the ontology so that new services will be able to be created. In addition to that, a Request Manager will attend the requests done to the system. In order to perform this functionality in a suitable manner, this latter component of the subsystem will keep in touch with the Resource subsystem (to send the requests that have to be attended by the devices that are able to satisfy them as far as resource availability is concerned). The component diagram of this subsystem has been included in Figure 22.

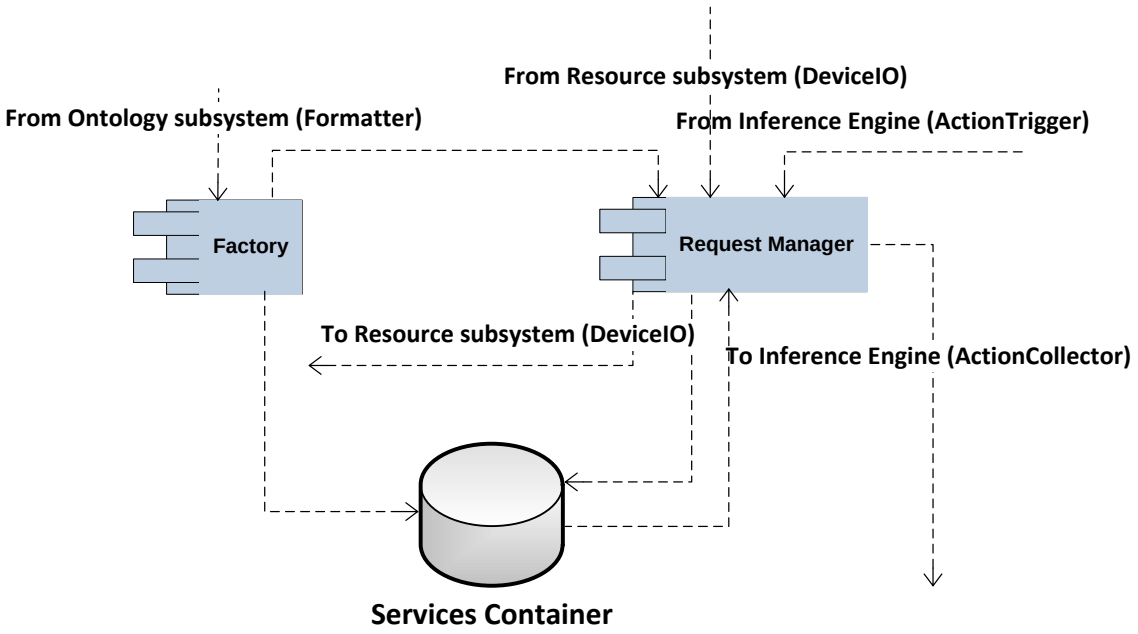

Figure 22. Services subsystem components

Finally, the Repository subsystem is the simplest one because it is expected just to store the formatted data that the Ontology subsystem has created for the whole system. Its main components have been located in Figure 23.

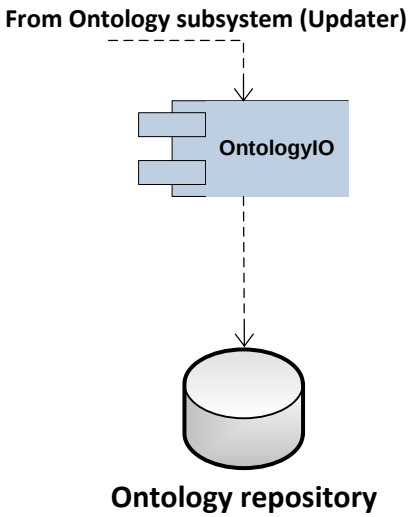

Figure 23. Repository subsystem components

3.3. Functional analysis

The process that is done here is different than the other one. It does, indeed, use UML diagrams to describe the proposal form a software engineering point of view, but the purpose of the diagrams differs from the ones that were present before, as use cases that are going to be dealt with is the main motivation for this stage.

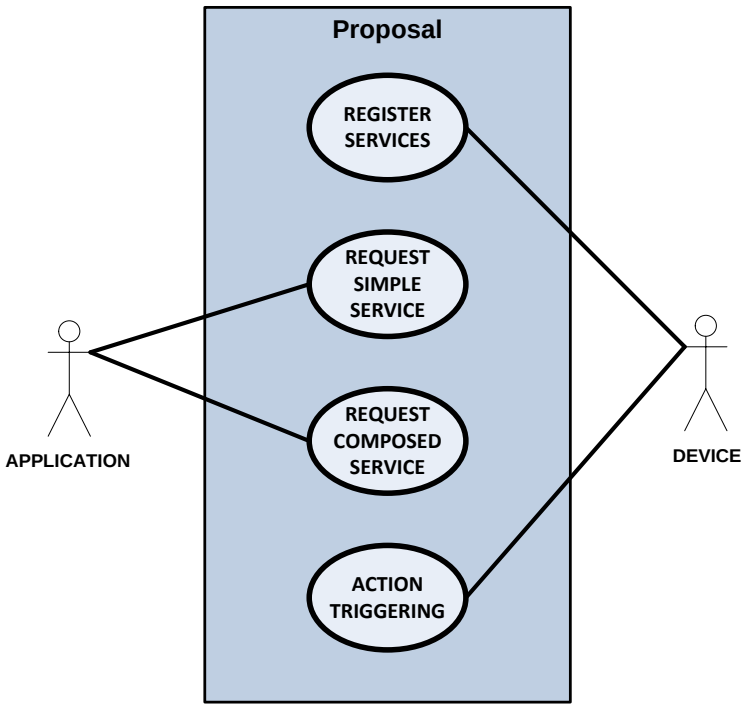

Figure 24. Diagram with the use cases of the proposal

As it has been depicted in Figure 24, there are four use cases that have been considered when designing the proposal. Those are the ones that are to be most frequently used when interacting with a middleware architecture in the context of a smart grid (and in several other scenarios too, as device registration and data retrieval are common procedures in architectures for distributed systems), which can be summarized as follows:

i) Device Registration: it is used to store information regarding the devices that are present in the system that has been deployed. Commonly, they will be located outside the middleware architecture and below the network communications, so they will establish connections with the system by means of the network communications.

ii) Simple data request: requests about the information that is measured by the devices are done as a way to obtain information from the system the middleware architecture is locate upon.

iii) Composed data request: these requests are basically the same as the ones that have been previously presented, with the difference that in this case they will be using the information from more than one service to offer another one with data inferred from the information that has been previously collected from separated, simple services.

iv) Action trigger: this service makes use of the capabilities of the Ontology subsystem to trigger actions that have to be activated in case there is an event that is taking place in the system.

In order to better define the behaviour and how this use cases are implemented in the system, sequence diagrams have been created that explain how each of the use cases behaves. For example, Figure 25 describes in a sequence diagram how device registration is done.

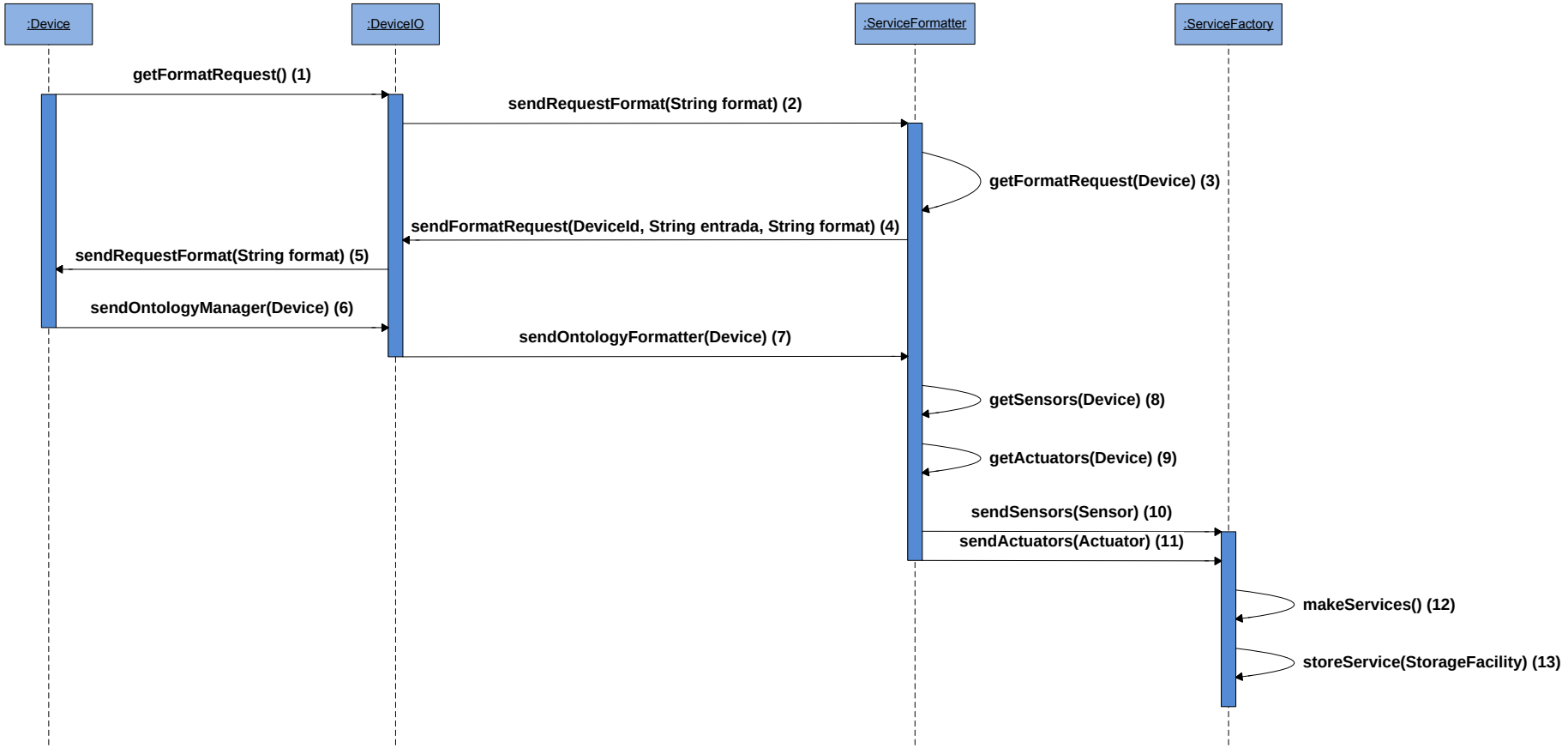

Figure 25. Registration sequence diagram

As it can be seen, there are several steps to be taken to complete the whole process. The steps that have to be taken to make the registration of a device are as follows: when a new device is willing to become connected to a microgrid that is installed as part of a smart grid, it will send a request about the storage format that is going to be used to keep the information in the system (1). This request is sent to the class responsible for format management (2) and once the request is retrieved over there, the format will be obtained (3). After that, the request will be sent back to the source device that originated it in the first place, being sent backwards from one class (4) to another (5).

In this response, the format that is going to be used within the system will already be contained, which is of critical importance so that the middleware ontology will know how to save the information about the device that has been provided. The correctly formatted data will be sent to the ontology (6) and afterwards to a class that is part of the ontology subsystem (7). Since the format that is used for the data request is the one that is matching what is used at the Ontology subsystem, sensing and actuating facilities of the device (8), (9) can be abstracted and sent (10), (11) to a software service factory that will define the services that can be invoked. Finally, those services are saved in case future updates change their nature or they have to be removed (13).

Once the device has already been registered, it can receive data requests and offer data responses. The services that can be requested, though, will be either simple ones (information will be retrieved from a single source of data) or composed ones (when information is retrieved from several sources of data). It is assumed that requests will be started from the application layer, since that is the most likely for them to be initialized. The sequence diagram used for simple data retrieval is located in Figure 26.

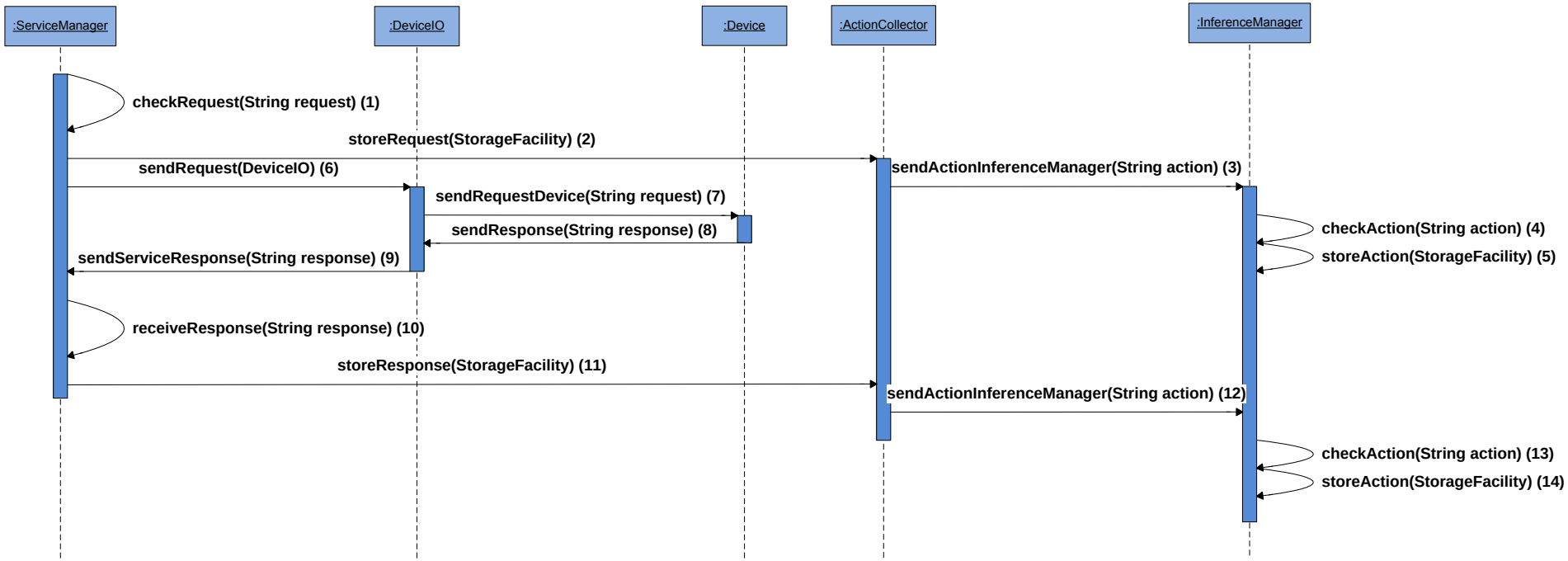

Figure 26. Simple data request sequence diagram

From what can be seen in this diagram, the steps to be taken for this use case are as follows: when a service request is made from the application layer to a device (which is already included as part of the system) is received at the semantic middleware architecture, a class that will assume the role of a service manager will check the request so that it will be determined whether the request has been done according to the requirements of the system (1). As a routine, the request will be stored (2) in case it has to be used afterwards by the Inference Engine subsystem; the new stored information will be notified to the component that is in charge of semantic inferences (3). As soon as it is checked that the request (or more accurately, the action) has no previous record (so that there will be no two similar actions stored in the system, 4) it will be definitely saved (5).

The request will be treated during this period of time too: its contents will be sent both to the class in charge of input and output data (6) and the one that is used for information retrieval from the device (7). Response messages (8, 9) will be sent back to the service manager (10) and will be processed once the response has been received. In this case, responses are sent for data storage as well (11) and will be notified for information inference (12). In the same fashion that was used for data requests, the content of the data response will be checked to know whether it already exist or not (13) and stored in case the response is new (14).

The other kind of service that can be provided is when a composed service request is made; in this case, though, the information retrieval procedure is slightly different, as it will be done so from several sources. The main motivation of composed services is offering a new kind of information based on the data provided by simple services. If, for example, there is a simple service capable of providing body temperature, another one is gathering information about heartbeat rate, and there is a third one that collects environmental temperature readings, a new service can be obtained by merging the information provided by those three individual services that is used to evaluate to possibility of injuries while doing some kind of exercise [39]. This new service will make use of the readings obtained from the simple ones to create thresholds that can evaluate the injury risk as if it was a simple service one, without the knowledge of the end user. An example of a composed service for this proposal is the one shown in Figure 27.

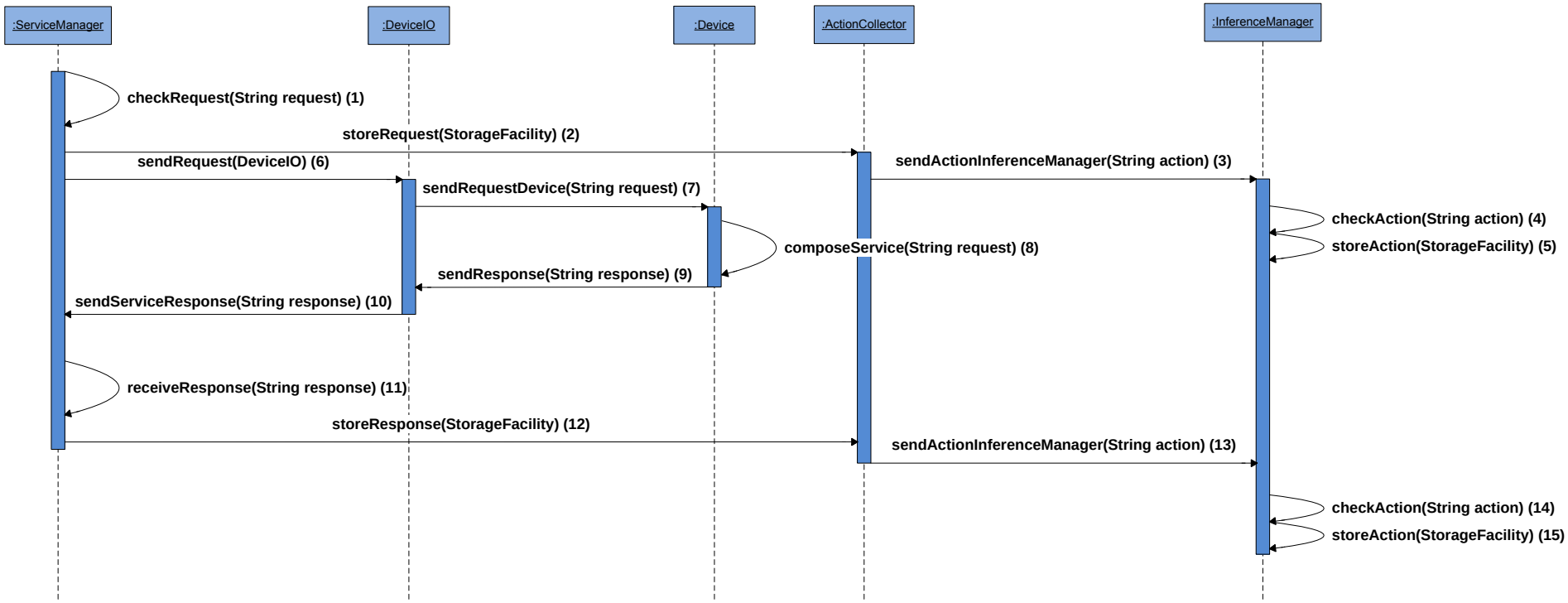

Figure 27. Composed data request sequence diagram

For this use case, the steps that have to be taken are very similar to the ones shown during the simple service case, albeit with a minor difference, as it will be explained. As it happened before, a request for a service (that is, a *composed* service) is done from the application layer and is resent (1) with the purpose of checking whether it is compliant with the format defined for the system. Once this has been done so, the request is sent to the Inference Engine subsystem (2) to be stored as a piece of information that was sent throughout the system. The request is forwarded to the Action Collector with the purpose of infer information of semantic nature. The request will be checked (4) and saved (5) in case it offers some new information. After the request is sent to the suitable class that is guiding the input and output operations (6) the requests are made (7). The main difference with the previous procedure regarding data requests is that, since it is a composed service, the data that are going to be transfer back to the application layer must be conformed in one method (8). As soon as the answer is composed it will be sent back to the location where it originated in the first place: the class using the input/output operations (9) will receive the answer and will be sent to the service manager (10). The answer that is received is also checked (11) and finally sent (12) back to the Inference Engine, which will make a decision in case actions have to be taken (13). Last but not least, the action that has been undertaken will be analysed so that its information is understood (14) and will be stored if regarded as suitable (15).

Finally, the last action that is going to be taken into account will be the triggering of events based on the data that is inferred by the system. This will be used whenever an action has to be executed from the system without previous request from an external party, such as a device or an end user. The main purpose with this functionality is giving the system a certain degree of autonomy and intelligence, so that it will be able to make decisions that will improve the overall performance of the system. The different stages that are followed in order to fully perform the actions expected from this service are in Figure 28.

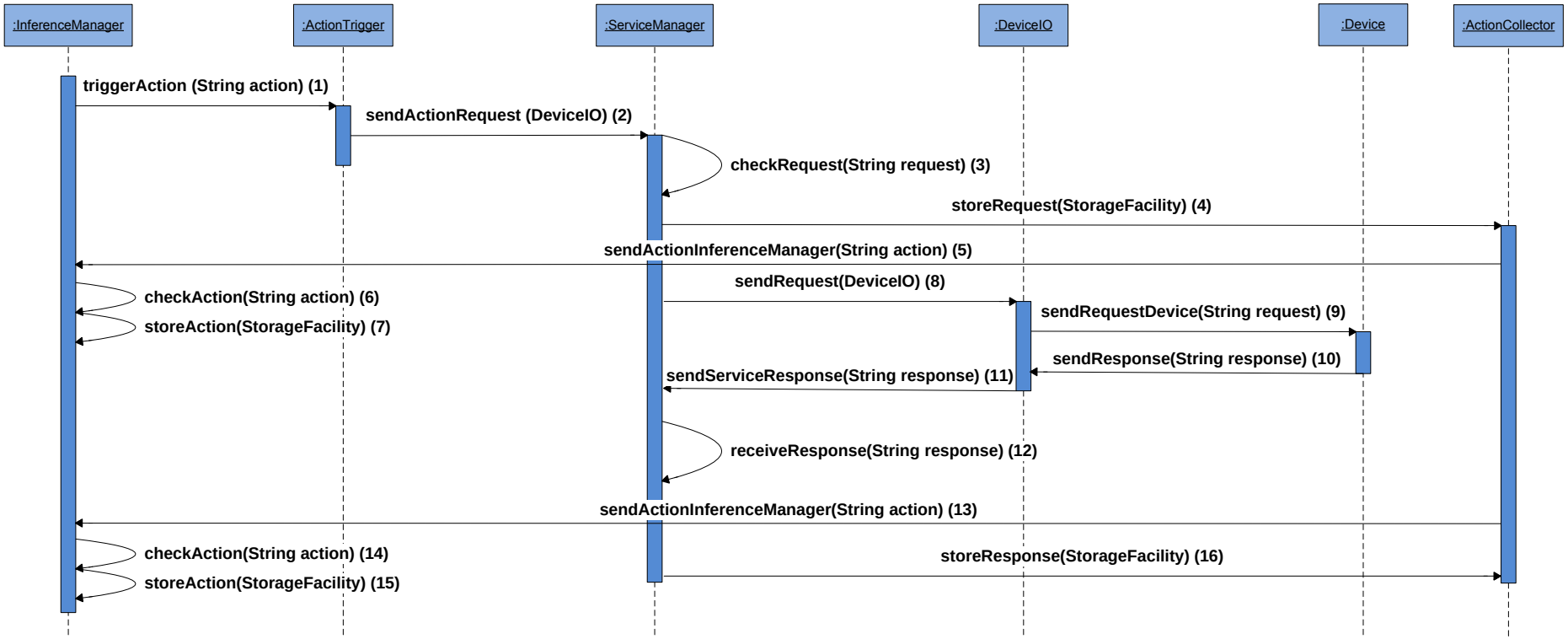

Figure 28. Action trigger sequence diagram

For this use case, the Inference Engine subsystem will play a more active role than the one it has done so far: rather than collecting information about the requests done, it will be the inference manager the part of the system that starts the whole process. Indeed, the Inference Manager class (1) will request the activation of an action to be taken by the Action Trigger entity (2). The final objective of this procedure is that there will be an action that will happen outside the boundaries of the middleware and will act on it, such as a piece of hardware or an application. As it has been done for all the other actions, the action request that is generated will be treated as any other message, so it will be checked whether there has been any kind of failure or information mismatch (3) and sent to be stored afterwards (4) for the sake of the historical data that is used by the system. This request for an action is sent back to the Inference Manager (5) that, as it has been done before, will be checked for inconsistencies (6) and stored (7) when it is triggered for the first time. At the same time, the request that has been executed to trigger an action will be treated as usual: the petition is sent to the class that is responsible for the management of input and output data (8) and the request is forwarded one more time to the class bounded to the device that is expected to provide the information (9). The device will send back the response (10), it will reach the service manager (11) and will be checked again (12).

Meanwhile, another action to be taken is sent from the Action Collector as the response is collected (13), investigated (14) and stored (15). The service response that is obtained will also be saved at the end of the request (16). In the end, an extra stage to merge the simple services into a composed one is the additional step required.

3.4. Functional analysis: class diagrams

As part of the developments that have been done for the project, class diagrams are provided as an accurate example of how to implement the system. The classes and the relationships offered here have been used for the implementation details of the solution that have been developed and tested in actual devices.

To begin with, the class diagram of the Resource subsystem is offered in Figure 29. Since this is the subsystem that most takes care of the hardware devices used in the deployment of one microgrid, its classes are basically dealing with the functionalities that are to be extracted from the devices used in the system, not only from sensor-and-actuator point of view, but also regarding the capabilities of each of the devices. It is interesting to have this information because it will be used to get a grasp of the performance of the services that can be provided.

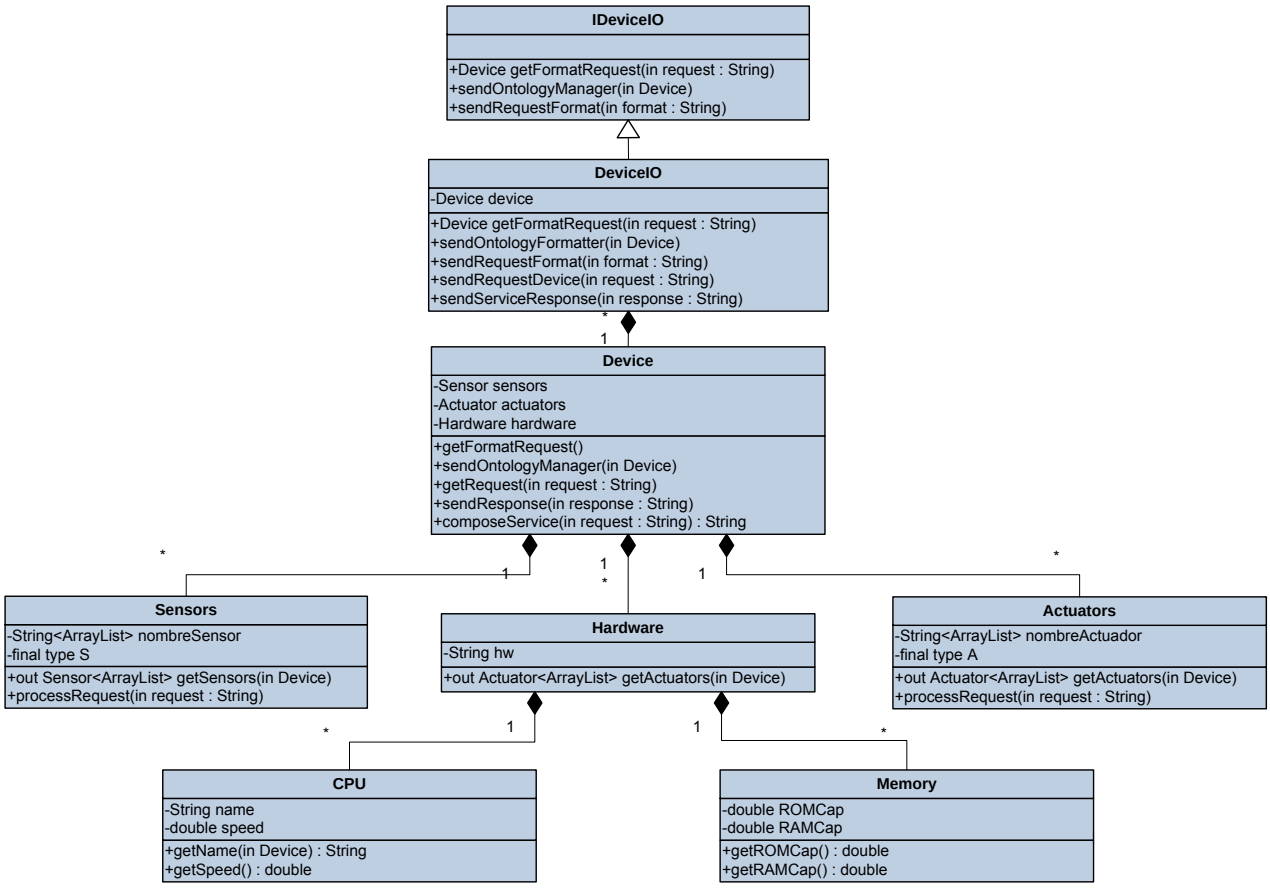

Figure 29. Resource subsystem class diagram

Additionally, the Ontology subsystem diagram uses two classes to perform its main functionalities: one is used to create (and format) the services that can be used in the system, so its main functionality will be extracting the information about the devices that are integrated into the system and composing new services with that information. What is more, there will be another class that uses the information obtained by the system to update it with the new services that can be obtained.

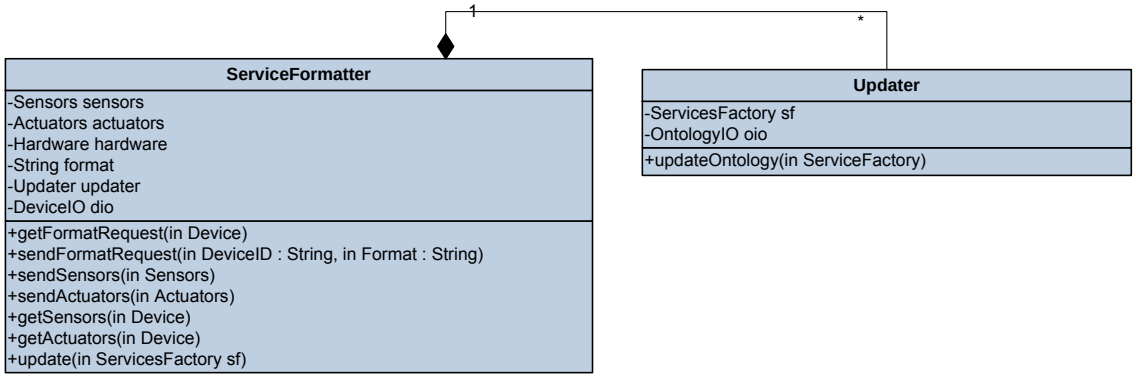

Figure 30. Ontology subsystem class diagram

The Service Factory subsystem has three different functionalities that are reflected in its class diagram: a) offering a template of the actual services that are going to be used in the system -by means of the information that provided with regards to the hardware components that are present- b) store them in a selected location of the system and c) control the service requests that are done upon the system in order to know their data format and where to send them after they have been deemed as matching the data format that is provided.

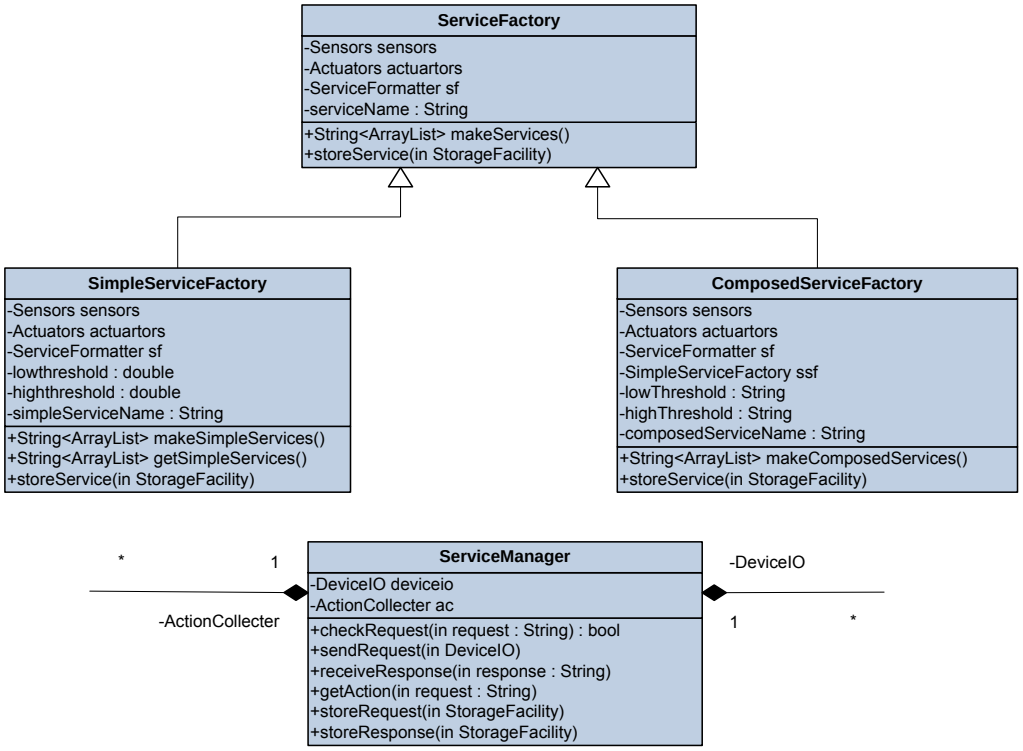

Figure 31. Service subsystem class diagram

Other subsystem of major importance is the Inference Engine, used to collect actions and trigger answers if considered necessary when assessing them with the procedure explained before. In order to perform this functionality, it appears as something clear that a class used to collect actions from the system will be required, as well as an inference manager that will check those actions, store them, and trigger some new in case it is necessary. Finally, an ActionTrigger class is used to launch the actions required in the system.

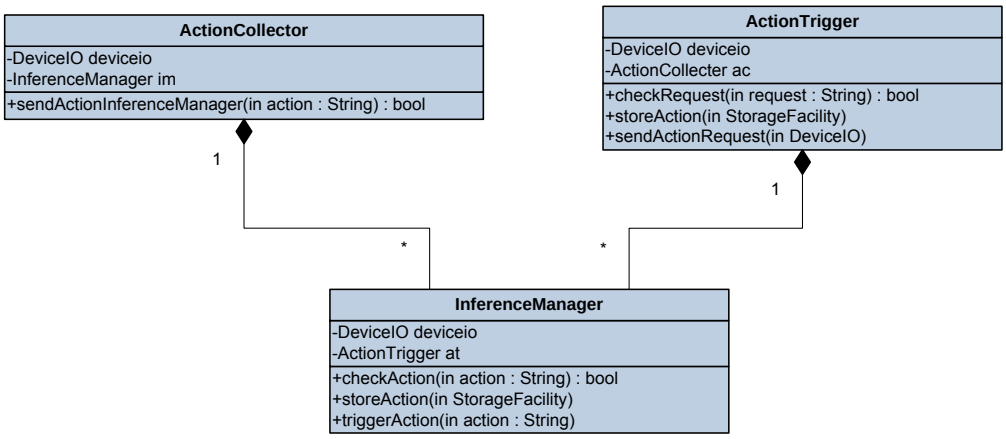

Figure 32. Inference Engine class diagram

Finally, the class that is present in the repository subsystem will be used to store all the information about the devices deployed in the system. Besides, the repository will be updated with new information about the services that have become available in the system as a result of the new pieces of equipment that are able to offer information.

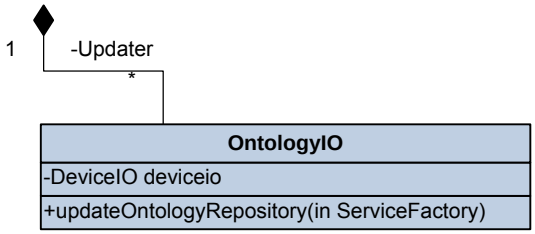

Figure 33. Repository class diagram

4 System validation and results exposition

4.1. Introduction to the prototype.

In this section, it is explained how most of the features of the proposal that presented before have been implemented and tested. Among the use cases that were presented in the previous section, the ones that involve the registration of the devices, as well as the request of simple and composed services on devices that have already been registered, have been treated here. A deployment of the nodes that have been utilized is presented, as well as statistical information on the tests carried out, along with data about the devices and protocols used during the experiments. The elements implied in the deployment have been portrayed in the deployment diagram that can be watched in Figure 34.

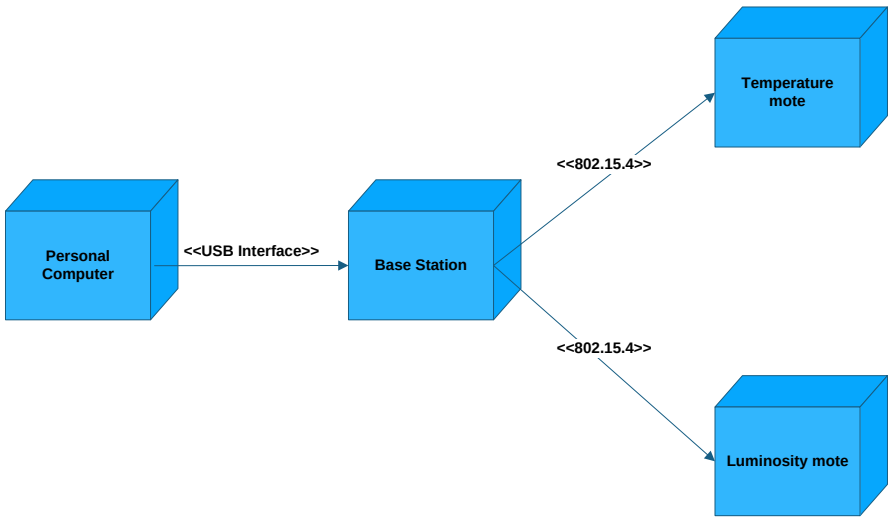

Figure 34. Deployment diagram of the prototype

4.2. Description of the prototype

In order to test the solution proposed, a prototype was deployed with several pieces of equipment containing the software that was implemented. This was done so because a) it was required to test and prove the reliability of the proposed solution for the system, b) having a distributed environment with several components playing important roles while, at the same time, c) having a distributed system with the bulk of the developed modules separated in different hardware components, as an overall appealing point of view to have the tests done. The prototype was basically made up by a Wireless Sensor Network and a PC use to collect data from the environment once the nodes used for this purpose became registered. Wireless Sensor Networks come in

handy to be used as distributed systems because they work as tiny pieces of equipment that make use of their computational resources with the cooperation of several alike hardware devices. Therefore, hardware devices that were used in the prototype are the usual elements that can be found in a Wireless Sensor Network deployed with actual components:

a) **Sensor nodes.** The nodes in charge of collecting data from the environment contained the developed software. After both of them became registered, one of the motes was used to collect information about temperature and the other one was used for luminosity data gathering, so that they would be used for simple and composed services. The specific mote model that was used as the node of the Wireless Sensor Network was Sun SPOT, which used to be manufactured by Oracle. Sun SPOT motes are compliant with most usual WSN protocols such as IEEE 802.15.4 and have installed a downsized version of the Java Virtual Machine (named Squawk) commonly used for Java developments. This simplified Java Virtual Machine is also used as the operating system of the mote [40]. At the same time, Java 2 Micro Edition is the programming language of choice for this platform, so it usually requires little familiarization with the programming environment. Its overall appearance can be watch in Figure 35.

Figure 35. Sun SPOT mote and base station

The features of this mote have been summarized in Table 19. While this kind of mote has been discontinued due to the manufacturer policies, they retain way above the average computational capabilities, with RAM and ROM resources that have yet to be matched by most of the other motes.

Table 19. Sun SPOT mote features, as in [41]

Central Processing Unit		Interfaces	
Microcontroller	AT91SAM9G20	Digital	GPIO
RAM	1Mbyte	Analogical	ADC
ROM	8Mbytes (Flash)	Serial	I²C, SPI, UART
Speed	400 MHz	Sensors	Light, Temperature, Accelerometer
Operating System	None, Squawk virtual machine performs that role.	Gateway Infrastructure	Mini USB for upload/download of code and communication.
Radiofrequency		**Others**	
Transceiver	CC1000	Battery	770mAhr Li-Ion Rechargeable Battery
Band	868/916MHz (MPR400CB), 433MHz (MPR410CB), 315 MHz (MPR420CB)	Consumption (idle)	30 uA
Data speed	38.4 Kbps	Consumption (Rx/Tx)	105.6 mW
Modulation	FSK	Price	329 € per kit (two motes and a base station). Now they are discontinued.
Range	Outdoors: 152.40m (MPR400), 304.8m (MPR410 y MPR420)	License	Open source.

b) **Base station.** As it can be expected from Wireless Sensor Networks, there is an element used as a gateway between the network itself and other, more regular environments, that involve user-oriented, common pieces of equipment (they may not involve an exclusively wired domain, though) for end clients. In this case, this functionality is done by the base station of the deployment. As base stations can be regarded as elements capable of interfacing between several domains, one of them will be the IEEE 802.15.4 protocol, and the other

one will be the architecture that is used in the Personal Computer running the virtual machine utilized for the tests.

c) **Personal Computer.** A PC has been used to as a way to visualize the information that is obtained from the wireless part of the system. In this way, it can be used as the application part of the system, since it allows an end user to employ the information provided by the system to his/her advantage. In addition to that, the PC is also used to store information about the services that are created by the service management part of the proposal, as it can be seen in Figure 36. The information that is provided in this case is the name of the service and the MAC address of the mote that is used in order to retrieve the information provided by the service. The application runs in a version of Ubuntu, a Linux-based operating system [42].

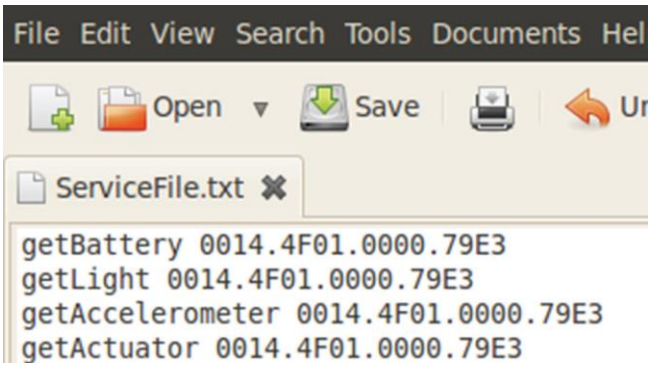

Figure 36. Service storage appearance

The operating system and the software applications that were being used in the PC were not installed directly in the hardware of the computer, but on a VMware-based virtual machine running on it. VMware player was chosen due its extreme popularity among virtualization platforms [43]. This is a common way to insulate an environment from another, when one of them has nothing to do with the other; also, failures in the virtual machine are likely to have little effect in the actual piece of hardware. The main characteristics of the Virtual Machine are depicted in Figure 37.

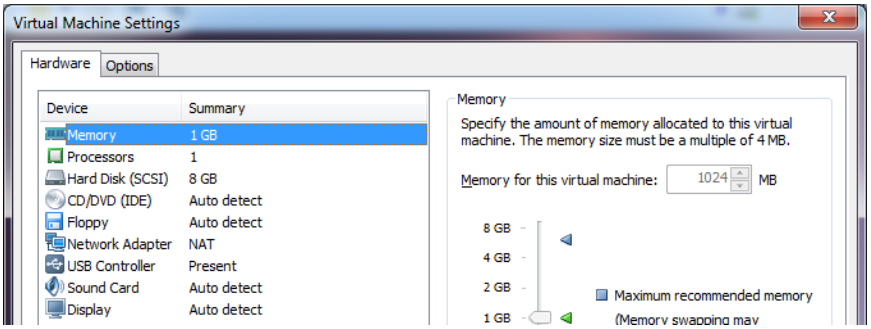

Figure 37. Virtual Machine main features

4.3. Performance of the prototype

The next section is used to present the results that have been obtained from the tests done on the deployment previously described. The idea was testing the services for several times to check not only the performance of the solution, but also its regularity and availability over time. There are three services that have been tested with the proposal: one is device registration in the system and the other two involve data collection from temperature and a composed service that takes into account information about temperature and luminosity.

4.3.1. Registration process

The first functionality that has to be tested is the registration of the services that are going to be offered. It is only natural that it has to be done this way because if the devices do not get registered then the services cannot be requested, as they remain unnoticed to the system. Overall, the registration procedure was made taking into account the steps described in section 3 of this manuscript. Registration was tested for 30 times to ensure that the data being collected was realistic and measured a typical registration procedure performance, rather than one single action that would be more dependent on casual irregularities that affect the system in a punctual manner. The results that were obtained are represented in Table 20.

Table 20. Service registration figures

Test (No.)	Time (milliseconds)
1	651
2	403
3	421

Test (No.)	Time (milliseconds)
4	384
5	553
6	447
7	418
8	442
9	410
10	504
11	446
12	618
13	349
14	406
15	529
16	544
17	314
18	455
19	508
20	429
21	521
22	466
23	346
24	441
25	467
26	426
27	555
28	348
29	527
30	274

With all the data that has been obtained, a graph has been elaborated to get a more immediate idea of the information that has been collected. It can be watched in Figure 38.

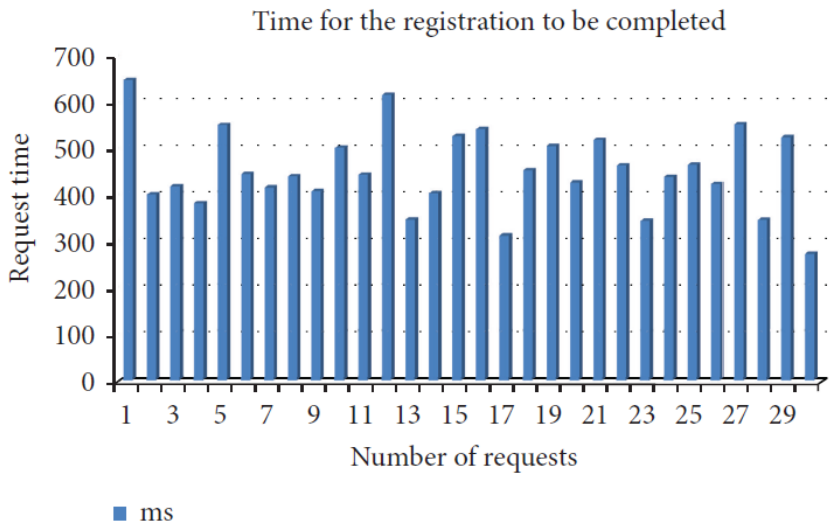

Figure 38. Service registration time values

As it can be seen, the registration process is done successfully in all cases (there was not even a registration failure during the tests that were performed) but there are some irregularities that have to be considered. A chart with statistical figures has been used here (Table 21) as a way to infer information from specific figures instead of just judging the appearance of the system results.

Table 21. Registration process prominent values

Datum Figure	Time (milliseconds)
Average value	453.4
Median value	444
Minimum value	274
Maximum value	651

There is some information that can be learnt from the data that have been placed here. Among the most important conclusions that have been reached the most important ones are as follows:

i) The average value (453.4 milliseconds) has a significant difference when compared to the minimum (274 milliseconds) and maximum ones (651 milliseconds): minimum is 45.5% lower than the average and maximum is 43.6% higher. This proportionally significant variations show that, despite

offering overall low amounts of time required to have the registration process completed, there is a certain degree of irregularity when it takes place. This can be due to how communications are made in the underlying layer of IEEE 802.15.4 protocol, as there are several connection attempts that can be done before finally establishing the connection due to the wireless media being saturated with data; it has to be remembered that the information required for the registration, although not very verbose, requires several interchanges between the involved elements (one for the format request, one for the format answer and a third one for the registration request with the format) until it is completed, so that the information interchange process is prone to have minor issues from time to time.

ii) Average and median values are relatively close one to the other. This implies that, although there are variations in the periods of time measured for this procedure, these variations happen in a somewhat similar proportion, so faster-than-usual registrations will make up for the consequences of having slower-than-usual registrations.

iii) However, the average value is slightly higher than the median value. This implies that slower-than-usual registration has a slightly higher impact than faster registrations, since the average value gets easily distorted by them. In the end, though, it can be assumed that outlier values have a small effect in the overall performance of the system.

iv) The obtained values are good enough in the sense that they are almost unnoticeable from a human point of view. Even at the worst obtained value, the registration is done in less than two thirds of a second, so registration can be managed with ease.

4.3.2. Simple service process (luminosity)

After the nodes, along with their capabilities, became registered, simple services have been tested in order to know what the performance looks like. As it was done before, the same process was repeated for 30 times to check the performance (and most importantly, the variations in performance) of the system. For this simple service request, the same procedure described in section 3 was used here. The data that were obtained are presented in Table 22.

Table 22. Luminosity request figures

Test (No.)	Time (milliseconds)
1	950
2	504
3	526
4	476
5	337
6	317
7	342
8	375
9	1153
10	885
11	324
12	295
13	392
14	1039
15	843
16	1178
17	467
18	318
19	300
20	257
21	344
22	986
23	370
24	283
25	501
26	616
27	1227
28	444
29	324
30	464

With the raw data that have been obtained after doing the information requests, a graph can be elaborated as the one that is shown in Figure 39.

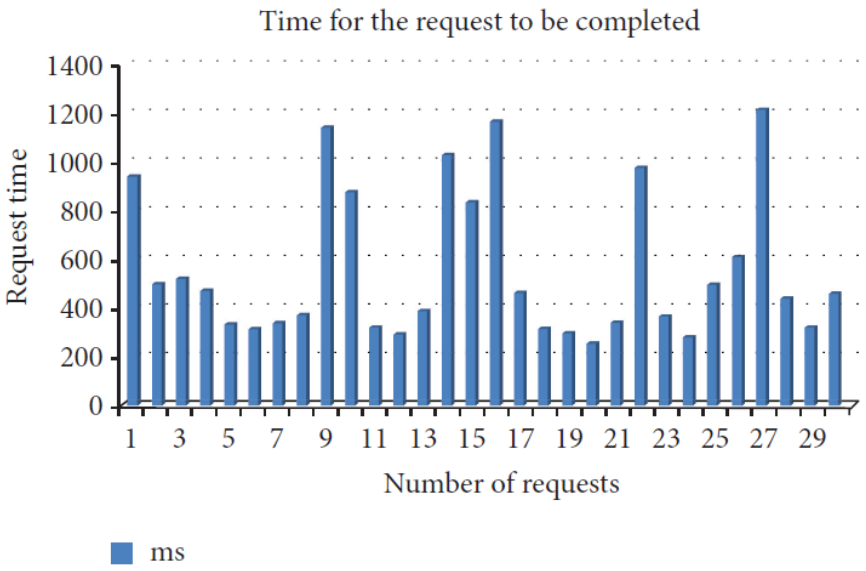

Figure 39. Simple service request time values

A simple glance will show that the results obtained here are more irregular (and overall, with higher time values) that the ones that were collected during service registration. Table 23 showing the most prominent statistical values reflects this impression as well.

Table 23. Simple service request prominent values

Datum Figure	Time (milliseconds)
Average value	561.2
Median value	454
Minimum value	257
Maximum value	1227

If this information is taken into account, there are several conclusions that can be made about the performance of the system as far as simple services are concerned, such as:

i) Disparity of the values obtained is far more significant in this case than in the previous ones: the minimum value is 54.2% of the average value than is obtained, and the maximum one is considerably higher than the average (maximum is the 218.6% -more than the double- of the average value). This

is clearly related to the irregularity that takes place when the system is used to retrieve data from single nodes.

ii) As a consequence of some of the high time values obtained from the system, major distortions on the figures for average and median values are added. It can be seen how the average value is higher than the median due to the fact that occasional high figures “push” the average value upwards, whereas median value is not as strongly affected by those differences. As a consequence, average and median values are more separated than what they used to be.

iii) The values obtained for simple data retrieval are still reasonable (in its worst possible case, it takes 1.2 seconds to answer a data request done) but it takes a longer amount of time to complete the requests.

4.3.3. Composed service process (comfort)

For the last batch of tests that were carried out, a composed service was utilized. As explained before, as “composed service” it must be understood a service that was obtained by retrieving information from two simple services, assessing it and composing a different service with the information retrieved. In this case, information about luminosity and temperature is used as a way to retrieve a new service that has been named as “comfort”. The figures that are collected regarding temperature and luminosity are evaluated jointly and a message is shown regarding the suitability of the comfort level based on some upper-level and lower-level thresholds used to determine comfort. The time figures that have been obtained for this set of tests can be seen in Table 24.

Table 24. Comfort request figures

Test (No.)	Time (milliseconds)
1	259
2	566
3	1131
4	551
5	1230
6	976
7	1184
8	308

Test (No.)	Time (milliseconds)
9	432
10	406
11	519
12	261
13	623
14	341
15	1203
16	945
17	1143
18	324
19	545
20	1197
21	1145
22	657
23	272
24	270
25	415
26	1161
27	281
28	609
29	259
30	620

As previously done, the figures obtained have been represented in a graph in order to get a more immediate and intuitive view of all the collected data. The graph has been placed as Figure 40.

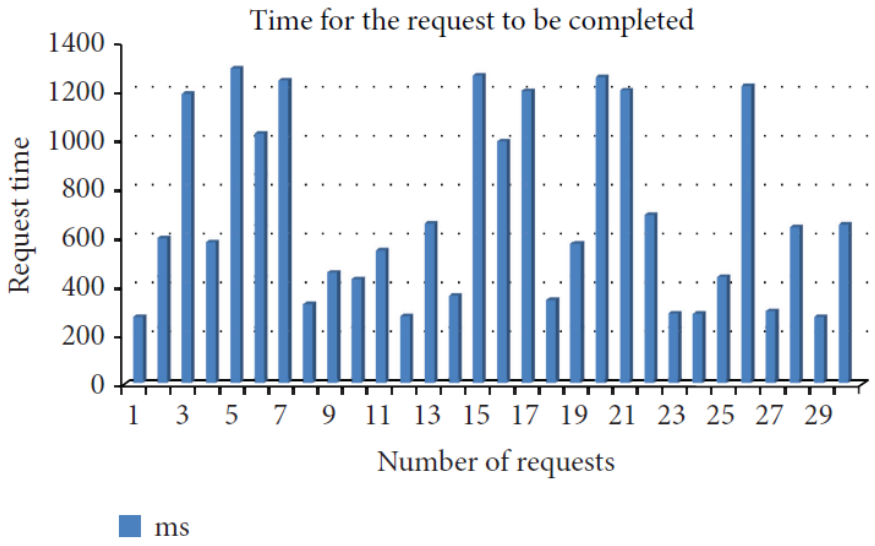

Figure 40. Composed service request time values

The obtained results obtained show that requesting a composed service demands only a slightly higher time to have the request completed than when a simple service is requested. This may look as surprising, as the composed service requires two simple services to be completed (and it would look as reasonable that an amount of time near the double of what is required for a simple one would be needed), but due to the implementation that was made to test this solution -which made possible requesting two different simple services simultaneously- the amount of time remains essentially the same. This happened like this because when the software was deployed into the Sun SPOT motes and PC, different ports were assigned for communications with the different motes. The extra milliseconds that were consumed to have composed services are related to the usage of the method used for service composition. The most significant statistical values of this data request have been placed in Table 25.

Table 25. Composed service request prominent values

Datum Figure	Time (milliseconds)
Average value	661.1
Median value	558
Minimum value	259
Maximum value	1230

As it was done before, there are several conclusions that can be inferred from the results obtained from the execution of this service:

i) There is a significant difference between the average, minimum and maximum values obtained, albeit the difference is more reduced than in the case that was evaluated before (the minimum value represents a 39.1% of the average, and the maximum value almost doubles the average one -186.1%-).

ii) The average and median values are still quite separated one from the other, and the average value is again higher than the median value (thus showing again that outlier, high values are distorting the average value). However, differences among them are less prominent (median value is 84.4% of the average value for this composed service, whereas it was 80.9% for the studied simple service).

iii) The amount of time consumed here is still reasonable, especially if it is considered that this time information is retrieved from two simple services, not a single one. It must be born in mind, though, that implementation details (rather than design ones) are of major importance to have an optimized performance for composed services; another different kind of implementation may have required more resources from the system, and the performance results that would have been obtained would greatly vary from the ones that have been depicted here.

4.4. Analysis of the prototype results

If the partial conclusions that have been extracted from the functionalities of the system that have been studied and considered in a holistic manner, an analysis of the prototype results can be done as the final result to evaluate the suitability of the system.

i) The performance figures obtained from the system are adequate for what can be expected by an end user: registration of the services that new devices can provide is done in less than a second (as an average value it is done in less than *half* a second) and it malfunctions very

rarely. Service request, regardless of their simple or composed nature, is also done in a negligible amount of time, even when it is done with low capability devices such as motes of a Wireless Sensor Network.

ii) However, irregular performance is observed in the system whenever there are service requests like the ones that have been done. Although they are most likely caused due to the information interchanges that are done via IEEE 802.15.4 protocol, it is an issue that must be further studied, as having an unpredictable system may be regarded as worse than a system with a lower, but more consistent, performance.

iii) Different approaches done during the implementation stage can lead to very different system results: the way that data from composed services are retrieved shows that implementation, while must be still decoupled from the analysis and design stages, has to be done in a self-conscious manner, rather than directly porting code from one system to another. Also, this detachment also guarantees that other system can use the same proposal design while varying their implementation details for each particular case.

5 Conclusions and Future Works

5.1. Conclusions

When all is said and done, several conclusions can be formulated from the study that has been described in this manuscript, as well as the proposal that has been presented and the implementation works that have been done, especially if the contributions that have been done here are taken into account. As the conclusions that have to be born in mind, the following ones can be formulated:

a) A study of the state of the art regarding the most prominent solutions in middleware and intermediation architectures for the smart grid has been done. These proposals have been assessed according to a set of criteria that was deemed as the most suitable for the evaluation of smart grid. In addition to that, and from that study, information is offered both about the open issues of the intermediation architectures in the smart grid, and how to tackle those issues for new proposals.

b) A proposal has been presented as an example that fulfils all the requirements expected from a semantic middleware architecture for the smart grid. Among the features that have been included in the architecture, modules regarding data format, semantic storage of information, device and application interfacing have been included.

c) Implementation details have been done regarding the suggested proposal in a way that they could match what is expected from a middleware architecture for the smart grid with the resources available during the implementation works. A Wireless Sensor Network has been used as a distributed system capable of retrieving information from the system (in a way not dissimilar to what a piece of Advanced Metering Infrastructure would do) with a PC where all the relevant pieces of information were presented.

d) The implementation has been evaluated on the basis of the performance results obtained. They show that, although a middleware architecture as the one that has been put forward here can be implemented in most of its features, there are some issues that have to be taken into account when deploying it into actual pieces of hardware, in the sense that some values are more unstable than what would be desired. Nevertheless, if the device constrains that have been used are taken into account, the system can be regarded as good enough to test the

majority of the use cases that have been described in the manuscript. Besides, as long as the design that has been presented is observed, it can be ported to different hardware and software tools (specifically, the ones that have more computation resources).

In addition to the conclusions that have been formulated, several future works have been conceived for future improvements and implementations of the solution.

5.2. Future works.

The list of works that would be useful to do in the future as an extension to the semantic middleware architecture presented here is as follows:

a) Different software tools could be used to install the classes and software packages developed for the proposal. For example, an Enterprise Service Bus could be used as a repository for the software modules that were programmed. Among the available ones, using JBoss Fuse, an open source ESB solution provided by Red Hat [44], seems like the most plausible option, due to the fact of its strong community support, its costless nature, and the capability to add interfaces to the software packages that are used as modules among the software packages themselves (Open Services Gateway Initiative, OSGi [45]), to the upper levels implementing web services (Representational State Transfer, REST [46]) and to the lower ones (Advanced Message Queuing Protocol, AMQP [47]) as reflected in [48].

b) Different devices can be used as the ones where the developed code is added. As Sun SPOT motes have been discontinued, different motes can be used to perform the same action. Furthermore, products from the open hardware community can be used to make Advanced Metering Infrastructure. Last but not least, open or proprietary hardware solutions could be used as well to test these solutions, although proprietary hardware may present issues related to the non-existent possibility of programming or modifying them.

c) Semantic queries using SPARQL (SPARQL Protocol and RDF Query Language, [49]) that would be done to retrieve stored information from an ontology are unconceivable for Wireless Sensor Networks, as they demand an

amount of resources that is unlikely to be available on a reliable basis for the nodes that are used here. Consequently, new solutions should be conceived during the implementation stage for semantic capabilities. For example, the ontology and the semantic capabilities could be retrieved from a more capable piece of equipment, such as a laptop or a computer, instead of relying on a Wireless Sensor network Node.

d) Since the information is retrieved from sensors, hardware devices have been included in the implementation of the proposal. However, an application with a Graphical User Interface could be developed to present the information in a more user-friendly manner.

5.3. Publications and projects

Since this Master Thesis is the result of some of the duties that have been completed in the GRyS research group (Grupo de Redes y Servicios de próxima generación, Next Generation Networks and Services Group [50]) of the Technical University of Madrid, there has been a flow of feedback from the work that has been done in the group and the results that are shown in this Master Thesis. Consequently, the developments that are shown here have been used as contributions for indexed scientific journals and research projects.

5.3.1. SCI-indexed journals

The following research articles have been used for the Master Thesis:

i) **Rodríguez-Molina, Jesús**; Martínez, José-Fernán, Castillejo, Pedro, de Diego, Rubén, "SMArc: A Proposal for a Smart, Semantic Middleware Architecture Focused on Smart City Energy Management," International Journal of Distributed Sensor Networks, vol. 2013, p. 17, 2013 [14].

ii) de Diego, Rubén; Martínez, José-Fernán; **Rodríguez-Molina, Jesús;** Cuerva, Alexandra, "A Semantic Middleware Architecture Focused on Data and Heterogeneity Management within the Smart Grid," *Energies,* vol. 7, p. 5953, 2014 [17].

iii) J.-F. Martínez; **Rodríguez-Molina, Jesús**; Castillejo, Pedro; de Diego, Rubén, "Middleware Architectures for the Smart Grid: Survey and Challenges in the Foreseeable Future," *Energies,* vol. 6, p. 3593, 2013 [18].

iv) **Rodríguez-Molina, Jesús**; Martínez, José-Fernán; Castillejo, Pedro; López, Lourdes, "Combining Wireless Sensor Networks and Semantic Middleware for an Internet of Things-Based Sportsman/Woman Monitoring Application," Sensors, vol. 13, p. 1787, 2013 [39].

5.3.2. Research projects

The presented Master Thesis has been involved in the development of the works that have been done as part of the following research projects:

i) LifeWear: Mobilized Lifestyle with Wearables (ITEA2) [51].

ii) e-GOTHAM: Sustainable-Smart Grid Open System for the Aggregated Control, Monitoring and Management of Energy (ARTEMIS) [52].

iii) I3RES: ICT-based intelligent integration of Renewable Energy Sources (FP7) [53].

Bibliographic references

[1] L. Hyung-Min and M. Ghovanloo, "A Power-Efficient Wireless Capacitor Charging System Through an Inductive Link,". *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 60, pp. 707-711, 2013.

[2] K. N. Mude*, et al.*, "Inductive characteristics of different coupling setups for wireless charging of an electric city-car," in *IEEE Internationa Electric Vehicle Conference (IEVC),* 2014, pp. 1-7.

[3] P. I. Domingues dos Santos e Abreu*, et al.*, "Identification and influence of the external elements on the transmission system operator's responsibility area," in *IEEE PowerTech (POWERTECH),Grenoble*, 2013, pp. 1-6.

[4] D. Metz*, et al.*, "Advantages of a dynamic Smart Grid training tool for DSO control centre staff," in *48th International Universities Power Engineering Conference (UPEC)'*, 2013, pp. 1-6.

[5] J. Rodríguez-Molina*, et al.*, "Business Models in the Smart Grid: Challenges, Opportunities and Proposals for Prosumer Profitability," *Energies,* vol. 7, p. 6142, 2014.

[6] V. C. Gungor*, et al.*, "Smart Grid Technologies: Communication Technologies and Standards," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 529-539, 2011.

[7] Z. Bradac*, et al.*, "Optimal Scheduling of Domestic Appliances via MILP," *Energies,* vol. 8, p. 217, 2014.

[8] Y. Wu*, et al.*, "Cooperative distributed energy generation and energy trading for future smart grid," in *2014 33rd Chinese Control Conference (CCC),* 2014, pp. 8150-8157.

[9] D. Berkeley. (2012). *An Arduino based project to read Watt-hour data from an Elster A100C electricity meter.* Available: http://www.rotwang.co.uk/projects/meter.html

[10] AvBrand. (2012). *Power monitor*. Available: http://www.avbrand.com/projects/powermonitor/

[11] Arduino. (2015). *Arduino Uno Web Site*. Available: https://www.arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardUno

[12] B. R. Peter Naur, "Report on a conference sponsored by the NATO SCIENCE COMMITTEE," 11th October 1968 1968.

[13] J. K. B. Nelson H. Weiderman, Dennis B. Smith, Scott R. Tilley. (1997). *Approaches to Legacy System Evolution*. Available: http://www.sei.cmu.edu/reports/97tr014.pdf

[14] Rodríguez-Molina, Jesús; Martínez, José-Fernán; Castillejo, Pedro; de Diego, Rubén; "SMArc: A Proposal for a Smart, Semantic Middleware Architecture

Focused on Smart City Energy Management," *International Journal of Distributed Sensor Networks,* vol. 2013, p. 17, 2013.

[15] R. de Paula Herrera and A. S. Felinto, "A distributed, multi-staged, high-throughput middleware for relational databases," in *IEEE Network Operations and Management Symposium (NOMS),* 2012, pp. 1370-1373.

[16] L. Guo*, et al.*, "Distributed measurement and control network of spaceflight measure ship based on ICE middleware," in *3rd International Conference on Computer Science and Network Technology (ICCSNT),* 2013, pp. 126-129.

[17] de Diego, Rubén; Martínez, José-Fernán; Rodríguez-Molina, Jesús; Cuerva, Alexandra, "A Semantic Middleware Architecture Focused on Data and Heterogeneity Management within the Smart Grid," *Energies,* vol. 7, p. 5953, 2014.

[18] Martínez, José-Fernán; Rodríguez-Molina, Jesús; Castillejo, Pedro; de Diego, Rubén, "Middleware Architectures for the Smart Grid: Survey and Challenges in the Foreseeable Future," *Energies,* vol. 6, p. 3593, 2013.

[19] S. Ovidiu Vermesan, Norway & Peter Friess, EU, Belgium, Ed., *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers, 2013,

.

[20] H. Gjermundrod*, et al.*, "GridStat: A Flexible QoS-Managed Data Dissemination Framework for the Power Grid," *Power Delivery, IEEE Transactions on,* vol. 24, pp. 136-143, 2009.

[21] DMTF. (2015). *CIM Schemas*. Available: http://dmtf.org/standards/cim/schemas

[22] WSU. (2010). *GridStat*. Available: http://www.gridstat.net/trac/wiki

[23] Z. Liang and J. J. P. C. Rodrigues, "Service-oriented middleware for smart grid: Principle, infrastructure, and application," *IEEE Communications Magazine,* vol. 51, pp. 84-89, 2013.

[24] A. Zaballos*, et al.*, "Heterogeneous communication architecture for the smart grid," *IEEE Network,* vol. 25, pp. 30-37, 2011.

[25] K. Jin Sung and K. Sung Jo, "An object-based middleware for home network supporting the interoperability among heterogeneous devices," in *IEEE International Conference on Consumer Electronics (ICCE)*, 2011, pp. 585-586.

[26] L. Zhao*, et al.*, "A Unified Solution for Advanced Metering Infrastructure Integration with a Distribution Management System," in *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, 2010, pp. 566-571.

[27] S. Sucic*, et al.*, "Standards-compliant event-driven SOA for semantic-enabled smart grid automation: Evaluating IEC 61850 and DPWS integration," in *IEEE International Conference on Industrial Technology (ICIT)*, 2012, pp. 403-408.

[28] P. Ferrari*, et al.*, "Mixing Real Time Ethernet traffic on the IEC 61850 Process bus," in *9th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2012, pp. 153-156.

[29] I. K. Samaras*, et al.*, "A Modified DPWS Protocol Stack for 6LoWPAN-Based Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics,* vol. 9, pp. 209-217, 2013.

[30] A. P. Garcia*, et al.*, "An intelligent agent-based distributed architecture for Smart-Grid integrated network management," in *IEEE 35th Conference on Local Computer Networks (LCN)*, 2010, pp. 1013-1018.

[31] A. Awad and R. German, "Self-Organizing Smart Grid Services," in *6th International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST),* 2012, pp. 205-210.

[32] K. Young-Jin*, et al.*, "A secure decentralized data-centric information infrastructure for smart grid," *IEEE Communications Magazine,* vol. 48, pp. 58-65, 2010.

[33] H. Junseok and P. Aravamudham, "Middleware services for P2P computing in wireless grid networks," *IEEE Internet Computing,* vol. 8, pp. 40-46, 2004.

[34] I. Foster and C. Kesselman, "The Globus project: a status report," in *Proceedings of the Seventh Heterogeneous Computing Workshop, 1998. (HCW)* pp. 4-18.

[35] R. P. Foundation. (2015). *Raspberry Pi Web Site*. Available: https://www.raspberrypi.org/

[36] F. Xi*, et al.*, "Evolving Smart Grid Information Management Cloudward: A Cloud Optimization Perspective," *IEEE Transactions on Smart Grid,* vol. 4, pp. 111-119, 2013.

[37] Y. Mehmood*, et al.*, "Intrusion Detection System in Cloud Computing: Challenges and opportunities," in *2013 2nd National Conference on Information Assurance (NCIA),* 2013, pp. 59-66.

[38] L. Jisun*, et al.*, "Customer energy management platform in the Smart Grid," in *14th Asia-Pacific Network Operations and Management Symposium (APNOMS),* 2012, pp. 1-4.

[39] Rodríguez-Molina, Jesús; Martínez, José-Fernán; Castillejo, Pedro; López, Lourdes, "Combining Wireless Sensor Networks and Semantic Middleware for an Internet of Things-Based Sportsman/Woman Monitoring Application," *Sensors,* vol. 13, p. 1787, 2013.

[40] Oracle, "Sun™ SPOT Programmer's Manual," *Oracle Labs,* 2011.

[41] Rodríguez-Molina, Jesús; "Semantic middleware development for the Internet of Things.," 2012.

[42] Ubuntu. (2015). *Ubuntu: The leading OS for PC, tablet, phone and cloud*. Available: http://www.ubuntu.com/

[43] VMware. (2015). *VMware download web site*. Available: https://www.my.vmware.com/web/vmware/free#desktop_end_user_computing/vmware_player/7_0

[44] RedHat. (2015). *Red Hat JBoss Fuse*. Available: http://www.jboss.org/products/fuse/download/

[45] (2015). *OSGi™ - The Dynamic Module System for Java™*. Available: http://www.osgi.org/Main/HomePage

[46] IBM. (2015). *RESTful Web services: The basics*. Available: http://www.ibm.com/developerworks/library/ws-restful/

[47] (2015). *AMQP - Advanced Message Queuing Protoco*. Available: https://www.amqp.org

[48] P. C. Parrilla, "Contribution towards intelligent service management in wearable and ubiquitous devices," DTE, Technical University of Madrid, Madrid, 2015.

[49] (2008). *SPARQL Query Language for RDF*. Available: http://www.w3.org/TR/rdf-sparql-query/

[50] CITSEM. (2015). *Next-Generation Networks and Services (GRyS)*. Available: https://www.citsem.upm.es/index.php/en/research-en/groups-en/group-of-next-generation-networks-and-services-grys-en

[51] GRyS. (2013). *Lifewear: Mobilized Lifestyle With Wearables*. Available: https://itea3.org/project/lifewear.html

[52] ARTEMIS. (2015). *e-GOTHAM Project (Sustainable-Smart Grid Open System for the Aggregated Control, Monitoring and Management of Energy)*. Available: http://www.e-gotham.eu

[53] I3RES, "I3RES: ICT-based intelligent integration of Renewable Energy Sources (FP7)," 2015. Available: http://www.i3res.es/v1