A Multi-Agent Architecture Framework for Energy Systems Design

Lamia Ben Romdhane

Thèse de doctorat

NNT : 2020UPASS141

université PARIS-SACLAY

A MAS Architecture Framework Dedicated To Energy Systems

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud et le commissariat à l'énergie atomique et aux énergies alternatives

École doctorale n°580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Paris, le 21 Juillet 2020, par

LAMIA BEN ROMDHANE

Composition du Jury :

Chantal Reynaud Professeur, Université Paris Sud	Président
Zahia Guessoum Maître de conférences, HDR, , Université Reims Champagne Ardenne	Rapporteur
Brahim Hamid Maître de conférences, HDR, IRIT	Rapporteur
Olivier Boissier Professeur, Ecole des Mines de St Etienne	Examinateur
Reda Bendraou Professeur, Université Paris Ouest Nanterre la Défense	Examinateur
Chokri Mraidha Chercheur, HDR, CEA	Directeur de thèse
Saadia Dhouib Ingénieur chercheur, CEA	Co-Encadrant
Sleiman Hassan Ingénieur chercheur, Renault	Co-Encadrant

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Un Cadre Multi-Agents Pour la Conception de Systèmes Energétiques

Mots clés:Ingénierie Dirigée par les Modèles;Système multi-agents;Système d'énergie

Résumé: Au cours des dernières années, les systèmes multi-agents (SMA) sont devenus l'une des technologies les plus prometteuses pour la conception et le développement des systèmes énergétiques intelligents nommés aussi Smart-Grid. Cependant, l'utilisation de la technologie d'agent dans l'ingénierie des systèmes pour modéliser, contrôler et simuler les comportements des systèmes énergétiques, pose encore de nombreux défis : méthodologiques ; techniques (liés à l'ingénierie des SMA en général) ; de standardisation et d'exploration architecturale (liés au domaine de Smart Grid en particulier). Cette thèse propose un cadre architectural conforme à la norme ISO 42010, contenant l'ensemble des conventions, principes et pratiques pour la description des architectures multi-agents établies dans le domaine des Smart-Grids, comme une solution adéquate pour résoudre les problèmes mentionnés ci-dessus. Le cadre architectural s'appuie sur l'Ingénierie Dirigée par les Modèles (IDM) pour résoudre les problèmes techniques et méthodologiques. Ce cadre est supporté par une méthodologie qui adhère à l'utilisation des standards d'agents et des standards énergétiques dans les phases d'analyse et de conception des SMAs. Le cadre se base sur une approche d'évaluation des styles d'architecture multi-agents pour sélectionner le style le plus approprié pour répondre à des exigences non fonctionnelles liées à un domaine d'application spécifique. Par ailleurs, un langage de modélisation indépendant des plateformes d'agents a été proposé permettant la modélisation des SMAs et l'analyse des modèles développés pour vérifier leur conformité au style d'architecture SMA sélectionné. L'approche a été prototypée dans un environnement d'Ingénierie Dirigée par les Modèles et évalué sur un cas d'application représentatif du domaine des Smarts-Grids.

Title: A Multi-Agent Architecture Framework for Energy Systems Design

Keywords: Model-Driven Engineering; Multi-agent system; Energy System

Abstract: In recent years, multi-agent systems (MAS) have emerged as one of the most promising technologies for the design and development of intelligent energy systems, also known as Smart-Grid. However, the use of agent technology in systems engineering to model, control and simulate energy system' behavior still faces many challenges: methodological; technical (generally related to MAS engineering); standardization and architectural exploration (specifically related to the Smart Grid domain). This thesis proposes an architectural framework in accordance with ISO 42010, containing all the conventions, principles and practices for the description of multi-agent architectures established in the field of Smart-Grids, as an adequate solution to solve the problems mentioned above. The architectural framework relies on Model Driven Engineering (MDE) to solve technical and methodological problems. This framework is supported by a methodology that adheres the use of agent and energy standards in the MAS analysis and design phases. The framework is based on a multi-agent architecture style evaluation approach to select the most appropriate style to meet non-functional requirements related to a specific application domain. In addition, a platform-independent agent modeling language was proposed to model MASs and analyze the developed models to verify their compliance with the selected MAS architecture style. The approach was prototyped in a Model Driven Engineering environment and evaluated on a representative application case from the Smarts-Grids domain.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Contents

List of Figures

List of Tables

I am dedicating this thesis to my family and many friends. A special feeling of gratitude to my loving parents Othman and Essia who gave me life and love and who provided me all the conditions for the success,
To my brother and my sisters for their support and their precious trust, which strengthens me and renews my energy in moments of doubt,
To my whole family,
To all my Friends . . .

Chapter 1

Introduction

This Chapter presents an introduction to our thesis, in which we give an overview of the topics it deals with. First, we present the context and motivations of this thesis, which are followed by the research questions that we have identified. Then, we enumerate the contributions of the thesis, and finally, we present the structure of this thesis document.

1.1 General Context

Modern energy systems are becoming more complex (Gungor et al., 2011). They are composed of different interacting entities that allow an intelligent production, distribution and consumption of energy. Software technology is used to optimize the production, distribution, and consumption of electricity, and to regulate the flow of electricity between suppliers and consumers.

In the recent years, Multi-Agents Systems (MAS) have emerged as a promising technology for the design and development of energy systems (McArthur et al., 2007b; McArthur et al., 2007a). MAS technology enables the implementation of large and complex distributed applications, and the development of autonomous control agents that are able to coordinate in a cooperative and fault-tolerant manner (Wooldridge and Jennings, 1995). MAS is considered as an appropriate technology to develop software solutions to manage the future power system known as Smart Grids. It is proved in the literature that agent technology is suitable for many problem in the SG domain (McArthur et al., 2007b).

As modern systems are becoming more complex, so does the need for an approach to increase productivity, reduce rework, and make system integration and maintainability easier. Model Driven Development (MDD) is introduced as a mean to move the focus of developers from pure coding to analysis and to make system modeling independent of the platform that will be used for system deployment. Using the notion of transformations, MDD allows a system model to be transformed to the desired programming language on the desired specific platform. MDE techniques have proven to be a palliative solution that enhance reusability, portability and interoperability of designs and implementations (Schmidt, 2006).Model Driven Engineering (MDE) perspective is based on the classical Model-Driven Architecture approach (MDA) (Kleppe et al., 2003). The use of MDA techniques in the development of MAS shall allow the interoperability between heterogeneous agent systems. Therefore, using it for the development of multi-agent systems (MAS) emerges in a natural way.

1.2 Research Challenges

The engineering of a MAS system, as a complex and distributed system, requires a great effort and an expertise in the agent-based engineering area. The uptake of such technology in a specific application domain like energy system requires knowledge and expertise of agent-based software development environments. This is not the role of the energy systems engineers who shall focus on the issues in their field and not on the underlying complex software infrastructure. Hence there is a need to provide an agent-based development framework to manage this complexity.

The MAS technology shall ensure scalability, reliability and flexibility, whereas the use of the domain standards with the FIPA [1] standard can guarantee interoperability within the MAS itself and between heterogeneous MASs from different designers. Existing proposals in the literature focused on helping agent system designers to develop their software solutions by proposing several MAS methodologies (Cossentino and Potts, 2002; Wooldridge, Jennings, and Kinny, 2000; Pavón, Gómez-Sanz, and Fuentes, 2006), agent modeling languages (Hahn, Madrigal-Mora, and Fischer, 2009; Trencansky and Cervenka, 2005), development and verification and validation tools. Some of them use the model driven techniques in the development process to increase the reuse of model and different conceptual patterns and to reduce the cost of the whole process from analyses to implementation, test and maintenance (Hahn, Madrigal-Mora, and Fischer, 2009; Amor, Fuentes, and Vallecillo, 2004). Design methodologies provide a structured analytical approach to the design of multi-agent systems. However, MAS developers should be aware that current methodologies do not guarantee fully flexible and extensible solutions (McArthur et al., 2007a).

All these works are generally domain independent; i.e., they can be adapted for any application domain. However, the problem from our perspective, in the context of using the agent technology in a specific field, is not only about how to develop the agent-based solution, but if the developed solution meets the non functional requirements of the system such as scalability, reliability, flexibility and interoperability, in a real execution environment.

The challenges for developing agent-based solutions for energy systems can be categorized into the following four major aspects:

1. Methodological issues, which are due to the variety of existing MAS design methodologies and different agents anatomies.

2. Technical issues, which are related to the diversity of implementation approaches and agent platforms.

3. Issues of interoperability among heterogeneous MAS and among the agents themselves, which are related to the communication semantics.

4. Architectural evaluation issues, which consider the evaluation of MAS architectural styles for selecting the appropriate one for a given application domain that has to meet better the selected non-functional requirements of the target application domain.

These challenges justify the need for an architectural framework to help in the development of MAS solutions in the energy sector that meet the system functional

[1] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

and non-functional requirements. Two requirements can be distinguished for this to happen:

- Interoperability among heterogeneous MAS and among the agents themselves, which are related to the communication semantics;
- MAS architectures need to adapt to the increasing complexity of electric grids. Decentralized architectures are favored to support scalability; i.e., the computational load for the resource allocation is divided between a number of computers, and the risk for communication bottlenecks is smaller. However, to add or remove a provider or customer, it shall be slightly better in centralized architectures since changes may only be necessary in one part of the system. As these two objectives are in most cases contradictory, especially when optimal results are expected, a trade-off has to be found to define the appropriate MAS architecture that better improves the Quality of Service and answers the central design intention regarding the Non Functional Requirement (NFR) (such as Scalability, Resilience, Modifiability, and Load Balance).

This dissertation focuses on the design and development of agent based systems, and especially on their architecture, in order to meet the previous requirements. The underlying Research Questions (RQ) are:

1. (RQ1) How to easily and efficiently design and implement agent systems for smart grid application?
2. (RQ2) How to adhere to standards in the design phase to guarantee interoperability?
3. (RQ3) Which MAS architecture is more adequate to provide the most appropriate balance between required quality attributes?

These concerns are studied through an application, which, further than providing test cases for answering the previous issues, also aims at providing answers to the following smart-grids specific questions:

- How a residential demand response system should be designed in order to reduce demand during peaks?
- What is the appropriate MAS architecture for this problem according to our design intention?

The results of the research and development on these research topics, guided by RQ1, RQ2 and RQ3, lead to the major contributions of this Ph.D. dissertation.

1.3 Thesis contributions

This thesis provides solutions to the crucial problems that hinder the development of a framework for guiding the design of agent systems, aligned with the principles of model-driven engineering, to manage smart grids.

The contribution of our work consists in proposing a tool-based MAS development methodology for energy systems. Our methodology shall allow us to perform a non-functional analysis of energy systems in the early design phases and to select the appropriate MAS architectural style in order to meet the system requirements. Our approach provides a modeling language dedicated to platform-independent

MAS design. This language is based on the support provided by the UML language and extend the semantic of the existing PIM4Agents metamodel.

In order to make our methodology usable by engineers, we have implemented the MAS4ES framework that allows to design and implement MAS solutions, that helps the users to adhere to standards in the analysis and design phases, and to meet the system requirements.

Our framework has to deal with the three research questions mentioned before. The research question (1) *"How to design and implement agent systems for smart grid application?"* is addressed by the definition of a **methodology**, to support the use of the MAS4SG framework, based on a Platform Independent Meta-Model (PIMM) called ML4Agents meta-model, which abstracts from the target execution platforms, and includes the relevant concepts for modeling MAS solutions for managing ESs, facilitating their interoperability. Our methodology guides the power systems engineers in the use of the MAS4SG framework in order to develop a MAS solution for a specific problem within the energy system domain. Furthermore, we follow the principles of Model Driven Engineering (MDE) in order to simplify the design process by enabling the reuse and verification of the models.

The research question (2) *"How to adhere to standards in the design phase to guarantee interoperability?"* is addressed by enriching the framework with two libraries, namely: (1) a library to model an ontology based on the standardized Common Information Model (CIM) in order to standardize the ontology concepts within the energy system application allowing interoperability (semantic) between agents from different MAS application. And (2) another library including specification of FIPA protocols to be reused for modeling agent interactions, allowing agents from different MAS platforms to discover each other and to communicate with messages among each others.

The research question (3) *"Which MAS architecture shall be preferred to provide the most appropriate balance between required quality attributes?"* is addressed by an exploration approach of the MAS architecture for selecting the appropriate architectural style for a given application that has to satisfy a given set of non-functional requirements.

1.4 Thesis structure

This thesis is structured in three parts and several appendices, plus the bibliography references and acronyms. The contents of the rest of this thesis manuscript being as follows:

Part 1, made of chapters 2 and 3 presents the context of the thesis and gives an overview of the related work and studies that make use of MDE technology in the specification and development of MAS and the uptake of the agent technology in the engineering of energy system applications.

Part 2 is made of chapter 4 and 5 . Chapter 4 describes the architectural framework we have proposed to support the uptake of the agent technology in the energy system domain application, whereas Chapter 5 presents our methodology designed to make our framework usable by designers in the MAS engineering domain. The methodology shall define the architectural design rules and the set of activities needed to design a MAS architecture.

Part 3, made of the following chapters 6 and 7, presents the validation of the thesis contributions; i.e., Chapter 6 presents the UML profile that implements our

modeling language used by the proposed framework and describes the libraries we developed to be used during the analysis and design phases, Chapter 7 presents the microgrid-based demand response case study on which we have applied our methodology.

Chapter 8 concludes our thesis work by analyzing the attainment of objectives and the contributions of the work, presenting the scientific publications achieved, along with the research lines open for future work. Finally, the Appendices extend and clarify information to give a better understanding of some of the issues presented in previous chapters.

Part I

Context and State Of The Art

Chapter 2

Context

2.1 Introduction

This chapter details the context of our thesis by introducing the fundamental concepts that are strongly related to our work. We will present the application domain we target which is the Smart Grids; the agent technology for modeling, simulating and controlling complex systems; the uptake of this technology in the engineering of software solutions to manage specifically energy systems; and finally the benefit of the model driven development techniques in the engineering of Multi-Agent Systems (MASs).

2.2 Smart grids

The current section presents the ongoing transition towards the smart grid which is emerging as the main paradigm for the modernization of the electric grid. The benefits of smart grids are presented, such as the ability to improve resilience to disruption, being self-healing and increasing consumer choice. Multiple characteristics, requirements and standards have been listed, and are expected to be in our consideration in the engineering of software solution to model, control and simulate Smart Grids. Finally, we end this section by presenting the microgrid concept as a discrete energy system capable of operating in parallel with, or independently from, the main power grid.

2.2.1 Definition

Intelligent networks or "Smart Grids" are electricity networks that, thanks to computer technology, can adjust the flow of electricity between suppliers and consumers. Smart Grids is one of the denominations of intelligent electricity distribution network which uses computer technology to optimize the energy production, transmission, distribution and consumption.

According to the U.S. Department of Energy [1] (DoE):

Definition 2.1. "Smart grid" generally refers to a class of technologies that people are using to bring utility electricity delivery systems into the 21st century, using computer-based remote control and automation. These systems are made possible by two way digital communications technologies and computer processing that has been used for decades in other industries. They are beginning to be used on electricity networks, from the power plants and wind farms all the way to the consumers

[1] US Department of Energy, Available [Online]: https://www.energy.gov/

of electricity in homes and businesses. They offer many benefits to utilities and consumers mostly seen in big improvements in energy efficiency and reliability on the electricity grid and in energy users' homes and offices.

The U.S. Department of Energy (DOE) describes the Smart Grid also as

Definition 2.2. "an intelligent electricity grid—one that uses digital communications technology, information systems, and automation to detect and react to local changes in usage, improve system operating efficiency, and, in turn, reduce operating costs while maintaining high system reliability."

2.2.2 Smart Grids benefits

According to the National Institute of Standards and Technology (NIST) (Fang et al., 2011), the expected benefits of smart grids are:
Improving Power Reliability and Quality: Better monitoring using sensor networks and communications and faster balancing of supply and demand.

Minimizing the Need to Construct Back-up (Peak Load) Power Plants: Better demand side management and the use of advanced metering infrastructures

Enhancing the capacity and efficiency of existing electric grid: Better monitoring using sensor networks and communications and consequently, better control and resource management in real-time.

Improving Resilience to Disruption and Being Self-Healing: Better monitoring using sensor networks and communications and distributed grid management and control.

Expanding Deployment of Renewable and Distributed Energy Sources: Better monitoring using sensor networks and communications and consequently, better control and resource management in real-time, better demand side Management, better renewable energy forecasting models.

Automating maintenance and operation: Better monitoring using sensor networks and communications and distributed grid management and control.

Reducing greenhouse gas emissions: Supporting and encouraging the use of electric vehicles and renewable power generation with low carbon footprint.

Reducing oil consumption: Supporting and encouraging the use of electric vehicles, renewable power generation with low carbon footprint and better demand side Management.

Enabling transition to plug-in electric vehicles: Can provide new storage opportunities.

Increasing consumer choice: The use of advanced metering infrastructures, home automation, energy smart appliances and better demand side Management.

2.2.3 Smart Grids Characteristics

An electrical grid is an interconnected network for delivering electricity from suppliers to consumers. It consists of generating stations that produce electrical power, high-voltage transmission lines that carry power from distant sources to demand centers, and distribution lines that connect individual customers as shown in figure 2.1 that describes the traditional power grid. A smart grid provides power utilities with digital intelligence to the power system network. It comes with smart metering techniques, digital sensors, and intelligent control systems with analytical tools. It enables the two-way flow of energy from power to plug to be automated, monitored and controlled as shown in figure 2.2. A general consensus is that the

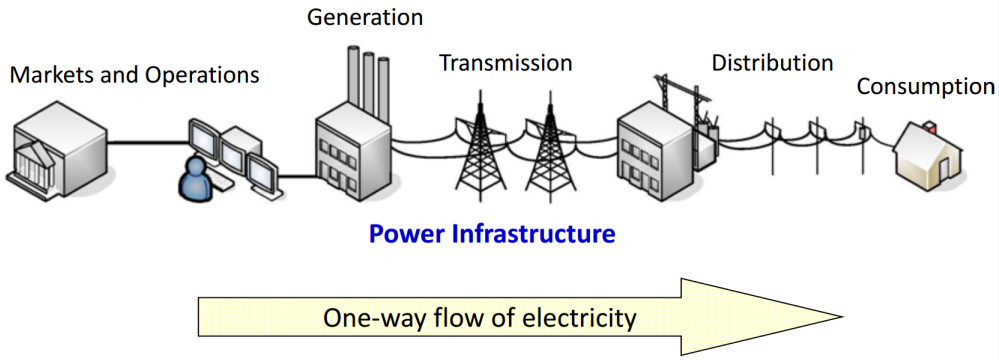

FIGURE 2.1: Traditional Power Grid

smart grid relies on the addition of a communication and control network to an updated electric grid. Hence, a Smart Grid is an electrical infrastructure (legacy power system) plus the communication and the control infrastructures.

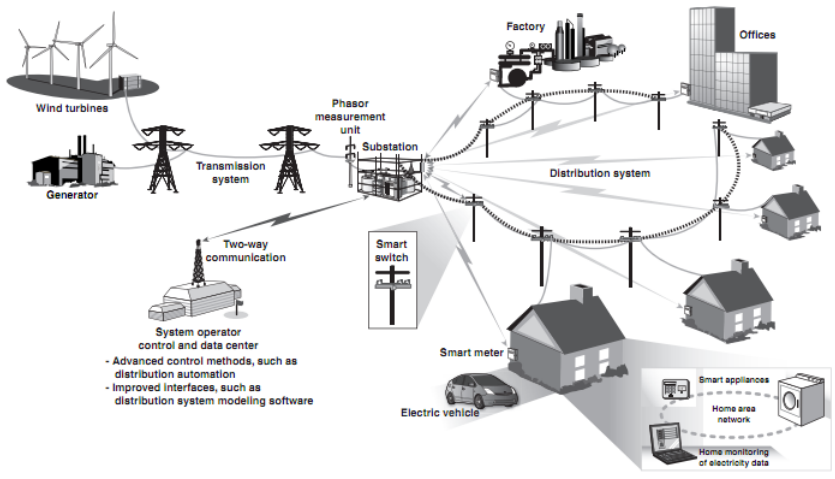

FIGURE 2.2: Smart Grid

In terms of overall vision and according to the US department of energy [2] the Smart Grid is:

Intelligent: capable of sensing system overloads and rerouting power to prevent or minimize a potential outage; of working autonomously when conditions require resolution faster than humans can respond, and cooperatively in aligning. the goals of utilities, consumers and regulators.

Efficient: capable of meeting increased consumer demand without adding infrastructure.

Accommodating: accepting energy from virtually any fuel source including solar and wind as easily and transparently as coal and natural gas; capable of integrating any and all better ideas and technologies - energy storage technologies, for example - as they are market-proven and ready to come online.

Motivating: enabling real-time communication between the consumer and utility so consumers can tailor their energy consumption based on individual preferences,

[2] US Department of Energy, "The Smart Grid: An Introduction", Available [Online]: https://www.smartgrid.gov/thesmartgrid/smartgrid.html

like price and/or environmental concerns.
Opportunistic: creating new opportunities and markets by means of its ability to capitalize on plug-and-play innovation wherever and whenever appropriate.
Quality-focused: capable of delivering the power quality necessary - free of sags, spikes, disturbances and interruptions - to power our increasingly digital economy and the data centers, computers and electronics necessary to make it run.
Resilient: increasingly resistant to attack and natural disasters as it becomes more decentralized and reinforced with Smart Grid security protocols.
Green: slowing the advance of global climate change and offering a genuine path toward significant environmental improvement.

2.2.4 Smart Grids Requirements

Some of the above-mentioned characteristics lead to the identification of several requirements of the smart grid, below we present the revelent requirements of smart grids: By studying the characteristics and definitions of Smart Grids, we have identified a number of requirements:

- Extensibility/Scalability: Smart Grids require scalability that refers to its capability of being expanded or upgraded easily to satisfy ever-increasing growing load demand.
- Flexibility: Smart Grids are intelligent and efficient and thus require flexibility in order to meet the energy demand through renewable sources.
- Fault tolerance: Smart grids are known as resilient systems consisting of diverse digital operations including smart meters, smart appliances, ... used for fault detection and outage prediction to avoid failures of the system.
- Interoperability: Smart grids are composed by heterogeneous communicative entities and thus interoperability should be addressed to allow communications between them.

2.2.5 Microgrids: the smarter distribution

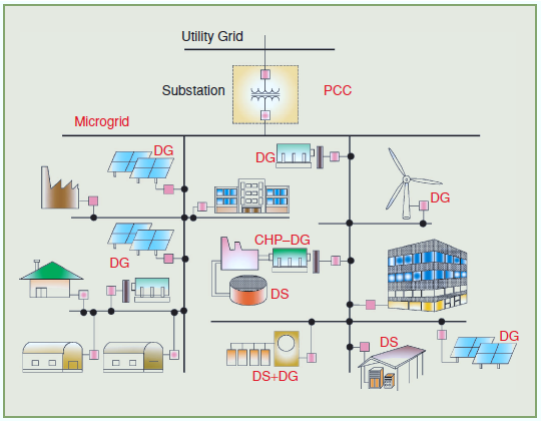

FIGURE 2.3: A typical microgrid structure including loads and DER units serviced by a distribution system.

The electric grid is currently undergoing important changes, it is evolving from an entirely centralized structure to a decentralized one due to the massive development of distributed renewable energy sources (Yoldaş et al., 2017). Microgrids have recently emerged as a new paradigm for future power systems because they can host multiple renewable energy resources in local distribution systems and also can supply reliable electric power to customers (c.f. Figure 2.3). A microgrid is a discrete energy system consisting of distributed energy sources and loads capable of operating in parallel with, or independently from, the main power grid.

Microgrids can be considered as controllable units such as distributed energy resources and controllable loads can effectively control the amount of power consumption or generation. Therefore, microgrids can be defined as autonomous power networks that can act as a controllable unit in power systems and can make various contracts with utility companies such as demand response program.

Compared to the typical loads, microgrids have more strengths as participants in the DR program because they have not only controllable loads but also energy resources. Therefore, microgrids have more flexibility to control the overall power consumption. For example, when the energy resources produce electric power, the loads in microgrids do not need to be cut out (Yoo et al., 2013).

This evolution requires new control approaches that tolerate the highly distributed nature of the grid and the intermittency of renewable energy sources. In (Lasseter, 2011), the author believes that the use of microgrids can simplify implementation of many Smart Grid functions, including reliability, self-healing, and load control. Thus the microgrid-based approach is a very promising method to manage energy system.

2.3 Multi-Agent Systems: Concepts and Approaches

MASs are useful for designing distributed systems requiring autonomy of their entities (e.g. energy systems). They offer an attractive alternative to implement such systems. A multi-agent system generally includes an environment with multiple objects (i.e. such as resources) and agents, where agents represent active and intelligent entities that manipulate (control, perceive, modify, ...) the MAS environment's objects. An agent is capable to perceive its environment and to act on it, whereas its behavior tends to meet its objectives according to its perception (Wooldridge and Jennings, 1995).

An agent can communicate with other agents, and it has to be able to understand the working environment and the communication language in order to work properly; i.e., an agent is only able to communicate about facts expressed in a predefined ontology, which describes the concepts of the domain and the relationship between these concepts. This allows agents to understand the messages received from the other agents. Interoperability between MASs is very important and the Foundation for Intelligent Physical Agents' (FIPA [3]) standards are used by MAS (McArthur et al., 2007a). This ensures that the agents from different platforms are able to discover each other and to communicate by messages using the FIPA ACL (Agent Communication Language).

[3]FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

2.4 Multi-agent System for power engineering applications

This section explains why MAS technology is a good candidate for power systems engineering.

2.4.1 Analogy of MAS and Smart Grids characteristics

As shown earlier, smart power grids may be considered as complex systems, whereas agent-based technology could be a practical way to achieve efficient and reliable modeling, control and simulation of such large and heterogeneous systems.

Many works prove the potential of MAS technology in power system engineering (McArthur et al., 2007b; McArthur et al., 2007a; Moradi, Razini, and Hosseinian, 2016) showing that MAS is adaptable and applicable with different power engineering categories. Furthermore IEEE recommends the use of agent technology to address the challenges in smart grid engineering (McArthur et al., 2007b; McArthur et al., 2007a). The MAS and agent properties can meet the smart grid requirements presented above in (see A.2.1). Advantages of MASs for tackling the requirement of smart grids include the following properties:

- **Autonomy/pro-activeness**: The autonomous intelligent agents should automatically be *flexible*, indeed they are able to schedule theirs own actions in order to achieve its goals. This can be done through selecting the most appropriate action from a number of possible actions. *Flexibility* relates to receiving many requests and cannot fulfill them all within a reasonable timescale, from which to decide whether to fulfill the request, the priority of the task, and if other actions should also be scheduled.

  The agents framework provides the functionality for messaging and service discovering, allowing new agent to integrate and communicate easily and without effort. Thus, extra functionality can be added simply by deploying new agents, which use service location to find others to communicate with. This allows systems to be *extensible*.

- **Open MAS Architectures**: An open agent architecture allows flexible communication between heterogeneous agents from different agent platform and implemented in different languages. This is achievable through adherence to messaging standards: the separation of an agent from its environment means that the messaging language an agent understands is important for inter-agent communication, rather than the programming language in which it was implemented (McArthur et al., 2007b). The Foundation for Intelligent Physical Agents (FIPA) [4] is one of the most known standards for an open architecture. Open MAS architecture places no restrictions on the programming language or origin of agents joining the system. Under the FIPA model, this is achieved through a separate agent called the Directory Facilitator (DF). The DF is often compared to the "Yellow Pages" phone book. Agents wishing to advertize their services register with the DF. Visiting agents can then ask (search) the DF looking for agents which provide the services they desire. Consequently, the FIPA standard supports *interoperability* between agent-based systems developed by the different companies and organizations. The FIPA Agent Management Reference model, provides an open architecture, to which agents can easily be

[4] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

added and removed. Thus, the open MAS architecture contribute to *extensible* systems.

Flexibility is offered by an open architecture of agents with good social ability. Indeed, when an agent could not fulfill a task, another agent having the same capability can handle the task on his behalf. Consequently, this flexibility leads to the design of a *fault-tolerant* system.

- **Robustness**: It was explained by (Shehory, 1998) and (McArthur et al., 2007b), including the fault-tolerence.

  According to (Shehory, 1998), one of the advantages of MAS is the distribution of execution, which allows for increase in overall performance. In addition, failure of one agent does not necessarily imply a failure of the whole system. The robustness provided by MAS is further increased by replicated capabilities. This replication is enabled by having multiple agents with same or similar capabilities. In such cases, when an agent that has some capability becomes unavailable, another agent with a similar capability may be approached. Replicated capabilities are more natural (and useful) in open MAS, however can support robustness in close MAS as well. The disadvantage of this replication is in the resulting redundancy.

Table 2.1 summarizes the multi-agent system properties that meet the smart grids requirements. We can conclude that MAS can be seen as an approach for extensible, flexible and robust solutions and that MAS standard (i.e., FIPA) supports interoperability between different agent systems.

MAS and Agents Properties	**SG Requirements**
Autonomy/pro-activeness	Extensibility/Flexibility
Open Mas architecture	Extensibility/Flexibility/Interoperability
Robustness	Fault tolerance

TABLE 2.1: MAS and agent properties answer Smart Grid requirements

2.4.2 Application of MAS in power engineering

Generally, there are two ways to use the MAS technology: (1) as an approach for modeling and simulating complex systems; (2) as an approach to the construction of robust, flexible, and extensible systems. These ways are used for engineering MAS solutions dedicated to energy systems. The next section gives more details on the application of MAS in power engineering for the two approaches.

2.4.2.1 Modeling and Simulation

MAS can be used as a modeling approach; i.e., it can be used to model the characteristics of smart grids and to simulate its behaviors. The application of this approach is to simulate marketplace.

2.4.2.2 Distributed Control

MAS can be used as an autonomous system to control the smart grid on behalf of the human being. It is applied mostly in distributed control area (such as: protection application, energy management system, demand response...) In chapter 7, we

present a use case for the Demand Response (DR) problem and for that we introduce the DR problem in more detail in this section.

Demand Response: One of the most researched fields for electric systems flexibility is called Demand Side Management (DSM) (Palensky and Dietrich, 2011), which aims to improve flexibility on the consumer side. The implementation of DSM programs can range from improving energy efficiency with better insulation materials to fully autonomous energy systems that automatically respond to shifts in supply and demand. The Association of Edison Illuminating Companies (AEIC) Load Research Committee[5] explains the major component of the Demand Side Management (DSM) and highlights and discusses many of the more prominent Demand Response programs currently in use. DSM can be implemented in two ways: through energy efficiency or Demand Response (DR). Since we focuses on the latter, we will define and explain the concept of Demand Response in more detail. Demand response (DR) is a change in the way energy is consumed by the client of an electricity company, as energy demand is better adjusted to the offer [6]. DR is a way to respond to cases of maximum energy demand so that users can be restricted from accessing all or part of their network energy consumption when they are asked to do so.

Demand Response refers to programs that encourage participants to make short-term reductions in energy demand. DR aim at reducing demand peaks by shifting or shedding loads directly or indirectly, in response to supply conditions. DR activation can last from a couple of minutes to some hours depending on the DR program, and might include shutting down a non-critical manufacturing process or shifting critical load consumption. On-site generation and storage systems can also be used to adjust loads drawn from the grid.

Demand Response problem belongs to the energy management domain. In order to choose the most appropriate MAS architectural style for the DR problem, we can explore MAS architectural styles that are recommended to be appropriate for the energy management domain. Afterwards, we evaluate them according to a set of required quality attributes. Thus, we tried to classify all possible MAS architectural styles for energy management domain in the next section to show the variety of possible styles and thus put the attention of the problem in the selection of the appropriate style for a given application in a specific application domain.

2.4.3 MAS architectural style for Energy Management application domain in Intelligent Electrical Networks

Multi-agent systems (MASs) can distribute computational burden to local agents and can consider the characteristics of individual entities by using intelligent algorithms. The agents can obtain information by monitoring local systems and spontaneously communicating with other agents (Wooldridge, 2009). Each agent includes intelligent algorithms to make decisions on behalf of the corresponding microgrid entity such as distributed generators and local loads. .... MAS may consist of large numbers of agents operating in rapidly evolving dynamic environments (Kantamneni et al., 2015). Since data and environment are decentralized, the roles and responsibilities of intelligent agents need to be clearly defined to resolve potential conflicts that may arise through agents interactions. A basic structure of a MAS can be broadly classified into many architectures. There are many aspects for characterizing

[5] AEIC Load Research Committee 2009, "The Association of Edison Illuminating Companies (AEIC)" Available [Online]: https://aeic.org/

[6] AEIC Load Research Committee 2009, "Demand Response Measurement and Verification", Available [Online]: https://www.smartgrid.gov/files/demand_response.pdf

the space of possible MAS architectures; the *control* schema and the *communication* infrastructure aspect. We were inspired by some state of the art papers (Kantamneni et al., 2015; Jayasinghe and Hemapala, 2015; Katiraei et al., 2008b; Tsikalakis et al., 2006) to derive the two aspect and to summarize a set of architectural styles. These works explain the different architectural style by identifying theirs characteristics and by giving the advantages and disadvantages of each architectural style.

In the following, we will present the possible control strategies and communication infrastructures to manage autonomous entities in a microgrid for energy management problems. The set of possible architectural styles depends on the number of aspects fixed to characterize the MAS architecture and depends on the chosen application domain: below we present possible MAS architectural styles for energy management application domain where the architectural styles are classified in two aspects.

2.4.3.1 Communication structure for microgrid energy management

As per the definition of MAS, it is all about the coordination of a large number of agents to achieve a global objective where several architectures are being developed for software agents. But when it comes to the topic of energy system management limitations such as communications and the requirements of high performance devices shall create some constraints. For the microgrid based energy management applications, three main architectures can be found from literature (Kantamneni et al., 2015), which are Centralized architecture, Distributed architecture and Hierarchical architecture depending upon the communication and coordination strategy. In Jayasinghe and Hemapala, 2015, authors describe the Hierarchical and Centralized architectures and give their pros and cons.

- **Centralized (Horizontal) architecture**

  This architectural style (c.f. Figure 2.4) is characterised by a collection of homogeneous, non-communicative agents. Such a strategy usually contains a single central coordinator to control the entire system for system management. According to (Jayasinghe and Hemapala, 2015), the speed of the process is higher than the hierarchical method but the need for a high performance central computer and data jams are disadvantages of this method. This centralized coordination strategy was more popular than decentralized coordination where a decentralized coordination strategy was suggested to overcome such limitations.

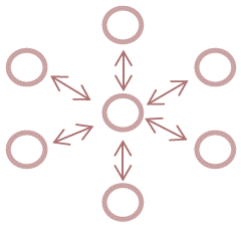

FIGURE 2.4: Centralized architecture

- **Distributed architecture**

  According to (Kantamneni et al., 2015), a distributed architecture (c.f. Figure 2.5) is characterized by a collection of communicative agents managed by

a single layer control structure. Each local agent has knowledge about its own part of the network and for which it is responsible, indeed no single agent has a complete knowledge of the whole domain. Instead, individual agents are allowed to discover the global information through communication and coordination with their neighbors. A single common communication framework facilitates interaction among all agents.

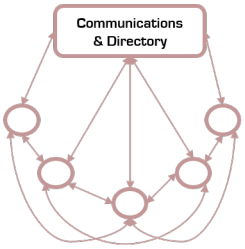

FIGURE 2.5: Distributed architecture

- **Hierarchical (Vertical) architecture**

  This is the most conventional and used architecture (Figure 2.6). In most of state of the art papers, this model is being used where the information gathered by a lower level agent is fed in to the upper level agent (Jayasinghe and Hemapala, 2015). The objectives are achieved through either a centralized or a decentralized supervisory control that includes three hierarchical levels:

  - Grid Level: Distribution System Operator (DSO) and market operator (MO)
  - Management Level: Microgrid Central Controller (MCC)
  - Field Level: Local Controllers (LCs) associated with each DER unit and/or load.

  The DSO is intended for an area in which more than one microgrid exists. In addition, the MO (can be more then one) is responsible for the market functions of each specific area. The main interface between the DSO and the microgrid is the MCC. The MCC assumes different roles ranging from the maximization of the microgrid value to the coordination of LCs. The LC controls the DER units and the controllable loads within a microgrid. Depending on the control approach, each LC may have certain level of intelligence. In a centralized operation, each LC receives settings from the corresponding MCC. In a decentralized operation, each LC makes decisions locally. Of course, in any approach, some decisions are only locally made; e.g., an LC does not require a command from the MCC for voltage control.

  The main advantage of this method is that the complexity of communication is very low because sending of messages to the same agent by several agents will be avoided. The main disadvantage is the time consumption for the communications increases and the overall performance will be slower.The other main disadvantage is due to the lack of visibility of all the agents in the network (since the agents only communicate with their neighbours), which makes it difficult to find the optimal path (Jayasinghe and Hemapala, 2015).

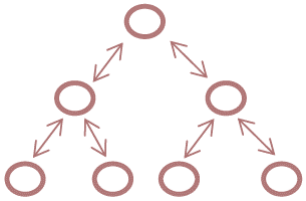

FIGURE 2.6: Hierarchical architecture

2.4.3.2 Intelligent Control Strategies for microgrid energy management

Generally, the control structure of such systems can be classified into three categories: centralized, fully distributed, and hybrid control (Dehghanpour, Colson, and Nehrir, 2017). If multiple (and at times conflicting) objectives must be met, then each energy source may not operate optimally, and a compromised operating decision may be achieved.

Centralized and decentralized approaches have their pros and cons, this is discussed in Tsikalakis et al., 2006; Katiraei et al., 2008a .

A brief description of each control category follows.

- **Centralized Control Architecture** In a centralized control scheme, the mea-

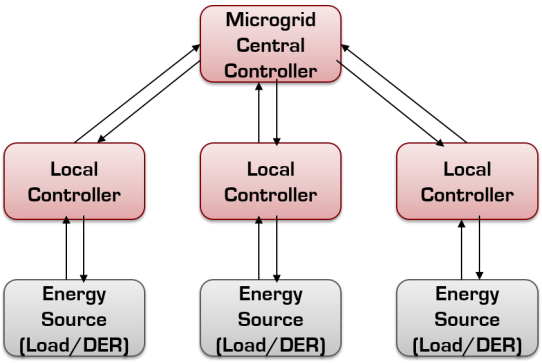

FIGURE 2.7: Central control structure

surement signals of each MG energy source (generation, storage, and load) are sent to a Microgird Central Controller MCC through their own Local Controller LC, as shown in Figure 2.7.

The centralized controller acts as an energy supervisor and makes control action decisions based upon measured signals and objective functions, which are communicated to each local controller.

Objective functions may be conflicting; for example, to minimize system operation and maintenance costs and environmental impact (carbon footprint), while maximizing system efficiency may be competing objectives, complicating the achievement of a solution.

Often, multiple-objective (MO) problems do not have a single solution but rather a set of non-dominated solutions, called a Pareto set, which include alternatives representing potential compromises among the objectives. This creates a range of choices available to decision-makers and provides them with trade-offs between multiple objectives. Control signals are sent to corresponding energy sources for the purpose to develop the appropriate amount

of aggregate output power. This MO energy management system can achieve trade-off optimal solution based on all available information, e.g., objectives and constraints. However, this scheme suffers from a heavy computation burden and is subject to single-point failures. Also, data privacy cannot be assured since energy sources have to share sensitive information with the centralized controller.

- **Distributed Control Architecture** In a fully-distributed control scheme, mea-

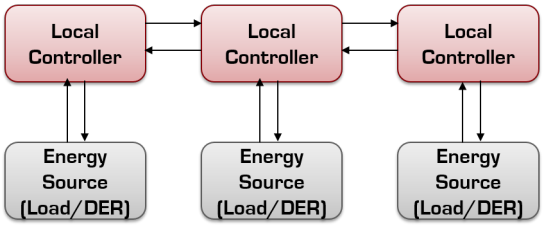

FIGURE 2.8: Local control structure

surement signals of each MG energy source are sent to corresponding local controllers, as shown in Figure 2.8. According to (Dehghanpour, Colson, and Nehrir, 2017), these controllers communicate with one another for the purpose of collaborating. An advantage of this scheme is the ease of "plug-and-play" operation. Also, different parties do not need to share sensitive data with their peers. In this architectural style, the computation burden of each controller is greatly reduced, and there are no single-point of failure. However, its disadvantage is the communication system complexity.

- **Hybrid Control Architecture** Hybrid or Mixed control, a more practical scheme

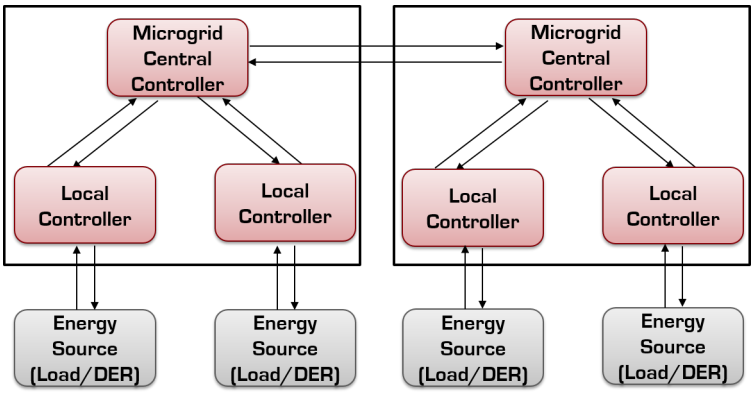

FIGURE 2.9: Illustration of a hybrid centralized and distributed control scheme.

that combines centralized and distributed control, as shown in Figure 2.9. A centralized control is used within each group of distributed energy sources within a microgrid, while distributed control is applied to a set of groups. With a hybrid energy management scheme, local optimization is achieved via centralized control within each group, while the distributed control is used to make coordination among the microgrid central controllers of the different groups. Thus, the computational burden of each controller is reduced, and single-point of failure problems are mitigated (Dehghanpour, Colson, and Nehrir, 2017).

2.5 Model Driven Engineering

Model driven engineering (MDE), which is based on meta-model principles, is gaining more and more attention in software systems due to its inherent benefits (Gascueña, Navarro, and Fernández-Caballero, 2012) . Indeed, MDE techniques have proven to provide solutions that enhance reusability, portability and interoperability of designs and implementations (Schmidt, 2006). Therefore, the application of MDE for developing MASs emerges in a natural way. Currently, the use of the model-driven engineering (MDE) approach throughout the software development process is becoming more popular (Gaševic, Djuric, and Devedžic, 2009). MDE concerns the exploitation of models as the cornerstone of the software development process. It allows both developers and stakeholders to use abstractions closer to the business domain of interest than to computing concepts. Thus, it reduces the complexity and improves the communication. This can lead to better portability and interoperability (Gascueña, Navarro, and Fernández-Caballero, 2012).

MAS development framework and methodologies follow an MDE (Schmidt, 2006) perspective based on the classical MDA (Kleppe et al., 2003). MDA techniques are applied to abstract developers from existing specific agent-oriented methodologies and platforms.

Authors in (Gascueña, Navarro, and Fernández-Caballero, 2012) have addressed the topic of using MDE techniques to solve the MAS engineering problems that we resumed in:

- Complex, expensive and time-consuming process, i.e. the MAS engineering process require more time to analyzing and designing models and to perform coding.
- There is a gap between the design models and the existing implementation languages that requires the use of MDE techniques that introduce refined design models directly implementable in a programming language.
- Difficult system integration and maintenance.

In table 2.2 we give for each MDE property the MAS engineering problem it deals with.

MDE Properties	**MAS Engineering Problems**
Reduce development cycle time	Complex and time-consuming process
Derive agent implementations from designs	Gap between design and implementation
Make system integration and maintenance easier	System integration and maintenance is difficult

TABLE 2.2: MDD properties solve MAS engineering problems

2.6 Conclusion

We have presented the general context around which the research work of our thesis is carried out. This is a key chapter for understanding the rest of the thesis manuscript. Smart grids are presented by defining their characteristics and consequently their relevant requirements. We then justify the uptake of the MAS technology in the engineering of energy systems by identifying the MAS and agent properties that meet the obtained energy system requirements. Agent standards and

energetic standards are presented to highlight their importance in solving interoperability within MAS solutions dedicated to energy systems. To validate our proposal, a state-of-the art problem was chosen, allowing us to check the issues that would face a MAS solution designer for electrical networks, including the architectural style and the functional and non-functional requirements. The last section describes the benefits of the use of model-driven engineering techniques in the MAS development.

Chapter 3

State of the art

3.1 Introduction

The complexity and intelligence of energy systems has increased in the recent years, whereas using Multi-agent Systems (MAS) has been recommended by IEEE (McArthur et al., 2007b) for developing software solutions for modeling, controlling, and simulating their behaviors. This chapter presents the related work that has been developed in the literature around the engineering of MASs dedicated to Smart Grids. Engineering of MASs for a specific application domain, i.e., energy system, requires exploration of existing standards. In order to choose the most appropriate standards, we first review the smart grids core standards proposed for the engineering of energy systems. Afterward, we review the generic MAS engineering methodologies and frameworks that are independent from the application domain but can be adapted to be used in the energy sector, and finally we survey the proposed approaches for engineering MAS solutions specific to the energy system domain where design choices are made in order to meet the Smart Grids's requirements.

3.2 Smart Grids Core standards

This section presents the main core standards for the engineering of smart grid applications.

3.2.1 The Smart Grid Architecture Model (SGAM)

The Smart Grid Architecture Model (SGAM) was introduced by the Smart Grid Coordination Group in 2012 (CEN-CENELEC-ETSI, 2015). It focuses on a structured description of a distributed Smart Grid System to identify standardization gaps. The SGAM framework allows also the validation of Smart Grid use cases. Indeed, it is intended to present the design of smart grid use cases in an architectural viewpoint and allows the validation of smart grid use cases and their support by standards.

The framework provides a set of concepts, viewpoints, as well as a method to map use case information and thus a structured approach for Smart Grid architecture development. With its coordinated set of viewpoints, it allows to depict various interrelated aspects of Smart Grid architectures (information, communication, ...) and supports the identification and coordination of elements on different levels of abstraction (CEN-CENELEC-ETSI, 2015).

The **interoperability** is seen as the key enabler of smart grids (CEN-CENELEC-ETSI, 2015). Interoperability connotes the capability of two or more networks, systems, devices, applications, or components to exchange and readily use information-securely, effectively, and with little or no inconvenience to the user.

The SGAM framework is established by merging the concept of the interoperability layers. This merges the results in a model which spans three dimensions **Domains**, **Interoperability (Layers)** and **Zones**. A central element of the SGAM is its five-layered, cubelike visualization, that basically combines the interrelated viewpoints (c.f. Figure 3.1).

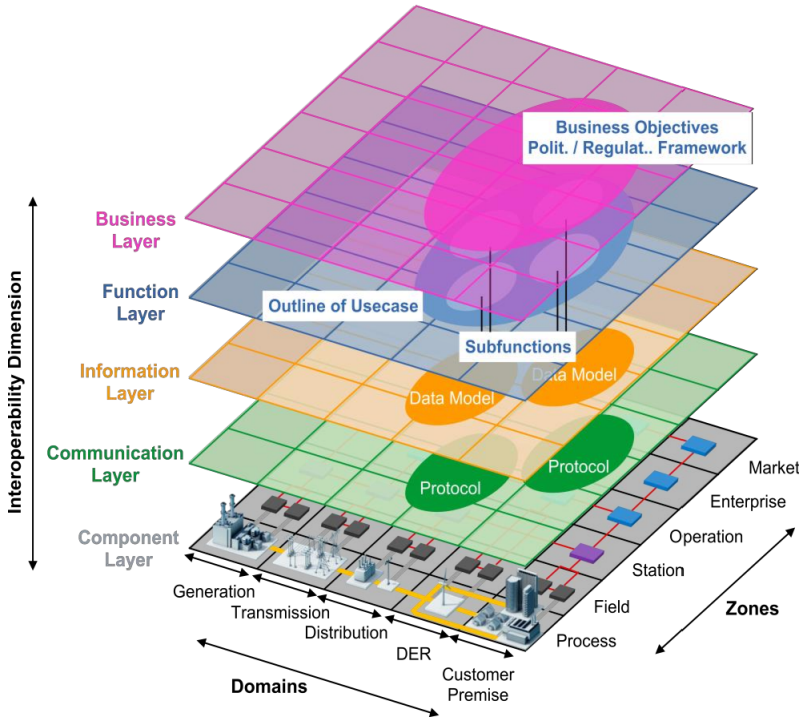

FIGURE 3.1: Entire Smart Grid architecture model visualization

Consisting of five interoperability layers, the SGAM framework allows the representation of entities and their relationships in the context of smart grid domains, information management hierarchies and interoperability aspects.

The interoperablilty layers represent business objectives and processes, functions, information exchange and models, communication protocols and components. Each layer covers the smart grid plane, which is spanned by electrical domains and information management zones. The intent of this model is to represent on which zones of information management interactions between domains take place.

The SGAM interoperability layers described in (CEN-CENELEC-ETSI, 2015) are listed below:

Business Layer:The business layer represents the business view on the information exchange related to smart grids.

Function Layer: The function layer describes functions and services including their relationships from an architectural viewpoint. The functions are represented independently from actors and physical implementations in applications, systems and components. The functions are derived by extracting the use case functionality.

Information Layer: The information layer describes the information that is being used and exchanged between functions, services and components. It contains information objects and the underlying canonical data models. These information objects and canonical data models represent the common semantics for functions and services in order to allow an interoperable information exchange via communication means.

Communication Layer: The emphasis of the communication layer is to describe the protocols and mechanisms for the interoperable exchange of information between components in the context of the underlying use case, function or service and related information objects or data models.

Component Layer: The emphasis of the component layer is the physical distribution of all participating components in the smart grid context. This includes system actors, applications, power system equipment, protection and tele-control devices, network infrastructure and any kind of computers.

The Smart Grid Architecture Model (SGAM) provides a structured basis for the design, development, and validation of new solutions and technologies. And thus, our thesis contribute in the engineering of MASs dedicated to Smart Grids by considering the SGAM from a MAS engineering perspective. Indeed, our solution tends to be compliant with the SGAM's viewpoints and standardization.

The Information layer describes the information that is being used and exchanged between functions, services and components. It contains information objects and the underlying canonical data models (e.g., the IEC CIM and other standards like 61850 for system automation). These information objects and canonical data models represent the common semantics allowing an interoperable information exchange. A description of the International Electrotechnical Commission (IEC) CIM standard follows.

3.2.2 The Common Information Model (CIM)

Interoperability is a real challenge which has to be solved by future Smart Grids (Uslar et al., 2012). Due to the multiplicity of manufacturers and operators, standards have to be created to enable interoperability between hardware and software from different providers, and reduce development costs. NIST recently published a roadmap for smart grid interoperability standards (Arnold et al., 2010). The Institute of Electrical and Electronics Engineers (IEEE) has also been working on numerous smart grid-related standards; some of them are named in NIST's roadmap.

IEEE is uniquely positioned to guide smart grid **interoperability standardization**, there are more than 100 IEEE standards available or under development relating to the Smart Grid in diverse fields, including the over 20 IEEE standards named in the NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0. The NIST report describes a high-level reference model for the Smart Grid, identifies nearly 80 existing standards that can be used now to support Smart Grid development and identifies high priority gaps for which new or revised standards are needed. Below we can find a list of the core International Electrotechnical Commission (IEC) standards identified as relevant to the Smart Grid (see Table 3.1).

The IEC CIM standard is used in the electricity domain, covering transmission, distribution, markets, generation, and related business processes. The key idea of the CIM is to define a common language in order to allow both: exchanging data between different companies, and exchanging data between company applications. The core packages of CIM are defined in the IEC standard 61970-301, which defines the components in the power system using the Unified Modeling Language (UML) (Uslar et al., 2012).

Generally, only a part of the CIM (so-called CIM profile) is used for modeling a given energy system solution. CIM profiles allow to define a subset of the CIM, including only the classes and associations required for modeling a specific solution. The U.S. DoE [1] affirms the the Smart Grid will be a system of interoperable systems.

[1] US Department of Energy, Available [Online]: https://www.energy.gov/

Core Standard	Topic
IEC 61970	Common Information Model (CIM) / Energy Management
IEC 61968	Common Information Model (CIM) / Distribution Management
IEC 62325	Common Information Model (CIM) / Energy Market
IEC 61850	Communication networks and systems for power utility automation
IEC 62351	Security
IEC 62056	Data exchange for meter reading, tariff and load control
IEC 61508	Functional safety of electrical/electronic/programmable electronic safety-related systems

TABLE 3.1: IEC Standards for Smart Grid

The systems must share a common meaning of the exchanged information. This reminds us with the required interoperability among the agents from different agent systems, where the agents should represent the information with a common representation (common vocabulary). The SGAM proposes some standards to ensure interoperability within the exchanged data between different energy systems.

The SGAM's information layer defines the information objects to be exchanged within the Smart Grid systems along with the associated canonical data models (e.g., the IEC CIM). Thus, the IEEE PES MAS working group [2] has announced that a CIM based ontology can be proposed to be considered as an upper ontology for energy management MAS solutions to ensure interoperability between agents from different agent system (McArthur et al., 2007b; McArthur et al., 2007a).

3.3 Review of MAS engineering methodologies and frameworks

This section surveys the MAS approaches and methodologies that are domain independent and can be used for the engineering of MAS solutions to solve the problems in the energy sector.

The MDE approach for MAS development has been the focus of several approaches (Amor, Fuentes, and Vallecillo, 2004; Pavón, Gómez-Sanz, and Fuentes, 2006). The Malaca Agent Model (Amor, Fuentes, and Vallecillo, 2004) proposed a mapping from the design models produced by existing agent oriented methodology to the Malaca model, which is a common and neutral agent model (i.e., a Platform Independent Model (PIM)) that implements all the concepts required by FIPA-compliant agent platforms, and from Malaca to the different platform specific agent models (i.e., a Platform Specific Model (PSMs)).

(Pavón, Gómez-Sanz, and Fuentes, 2006) introduced the INGENIAS Development Kit (IDK), which is a set of tools for modeling, verifying, and transforming agent models. It provides Model Driven Development (MDD) tools for MAS development based on the INGENIAS metamodel.

Some authors have tried to provide a standardized metamodel for MAS (Beydoun et al., 2009; Bernon et al., 2004): (Bernon et al., 2004) proposed a metamodel based on three existing metamodels for MAS, namely: Gaia (Wooldridge, Jennings, and Kinny, 2000), PASSI (Cossentino and Potts, 2002), and ADELFE (Bernon et al.,

[2] IEEE PES Multi-Agent Systems Working Group, Available [Online]: https://site.ieee.org/pes-mas/

2002); (Beydoun et al., 2009) proposed the so-called FAML model, which is intended to resolve the interoperability problems among the agent-oriented methodologies, and based on five existing metamodels, namely: ADELFE (Bernon et al., 2002), PASSI (Cossentino and Potts, 2002), Gaia (Wooldridge, Jennings, and Kinny, 2000), INGENIAS (Pavón, Gómez-Sanz, and Fuentes, 2006) and Tropos (Bresciani et al., 2004).

(Hahn, Madrigal-Mora, and Fischer, 2009) proposed PIM4Agents, which is a Platform Independent MetaModel (PIMM), that is intended to contribute to the interoperability between domain-specific architectures and agent platforms. It aims to abstract the developers from existing agent-oriented methodologies and platforms. The proposed metamodel provides the core language to be used in an agent-oriented software development process, which conforms to the principles of MDD. It describes the MAS aspects, namely: agent, organization, interaction, behaviors and environment. Furthermore, they provided two model transformations that allow to transform the created models into textual code that can be executed with JACK (Agents, 2006) and JADE (Bellifemine, Poggi, and Rimassa, 1999).

We notice a large use of interaction protocols by the proposed MAS methodologies. For example, PASSI (Cossentino and Potts, 2002) uses the interaction protocols of FIPA [3] for the specification of agent interaction. It is very important to use FIPA standard to address the interoperability issues. Indeed, FIPA guarantees the communication between heterogeneous agents from different agent platforms and it is implemented in different languages as it is one of the most important standard for an open MAS architecture. The use of standardized interaction protocol like FIPA enables diverse intelligent systems to collaborate and allows flexibility and scalability since new agents can enter the MAS and use the standardized communications and protocols, further improving the provided functionality and information.

Some other work propose a conceptual framework for agent-based software engineering to be used for any domain. (Warwas et al., 2012) presents a model-driven framework for engineering mulit-agent systems named Bochica. The framework's task is to capture the design decisions for a system under consideration on a platform-independent level of abstraction and to project this design to a target platform. Bochica combines the benefits of a platform-independent approach with the possibility to address concepts of custom application domains and execution environments. In (Fischer and Warwas, 2012), an extension of the framework is proposed in order to make the MAS design methodologies for MAS that are already available from literature can be connected to Bochica.

(Silva and Lucena, 2007) introduce a multi-agent system conceptual framework called TAO to understand distinct abstractions, their relationships and interactions in order to support the development of large-scale MASs. The proposed framework elicits an ontology that connects consolidated abstractions, such as objects and classes, and frequently used MASs abstractions (agents, roles, organizations and environments). In order to define a MAS modeling language that contemplates all the concepts described in TAO, they propose the MAS-ML modeling language. MAS-ML extends UML by preserving all object-related concepts that constitute the UML meta-model while including the agent-related concepts described in TAO. Unfortunately, the MAS-ML is not an abstract modeling language. Indeed, the MAS-ML models defined on PIM stage will be transformed in UML models at PSM stage,

[3] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

based on only an object-oriented framework for implementing MAS. The application code (i.e., java code) is generated from UML models.

Some of the solutions presented above solve the methodological issues in the MAS development process, whereas others tackled the methodological and technical issues by following the model driven development perspectives. Most of them target all application domains and none of them is specific to the engineering of the energy systems where energy standards should be adhered to in the analysis to constraint design choices. Indeed, the standards like Common Information Model CIM data models support the design of a common representation of agent's knowledge, i.e. the agent's ontology, which will determine the agent's communicative abilities (i.e., the communicative acts and interactions it supports). This point was taken into consideration and discussed in the report of the IEEE PES MAS working group (McArthur et al., 2007a) on the topic of the subject of the possibility of proposing an alternative methodology that can be used in the area of smart grids.

In the following, we review the MAS engineering approaches that target the energy domain in particular.

3.4 Review of MAS engineering approaches for SG

Several surveys have been done on the application of MAS for modeling energy systems (McArthur et al., 2007b; Kremers, 2013). Existing MAS solutions for ESs generally propose ad-hoc solutions; i.e., they were designed for resolving specific problems that were not intended to be reused (Kremers, 2013). Furthermore, these solutions were designed to run on a specific agent platform, without considering the interoperability among different agent systems, neither with other technologies. (Herbst et al., 2012) presented an overview of energy modeling with the multi-agent approach. In the following, we review the proposed agent solutions for Smart Grids.

(Koritarov, 2004) proposed an agent model and a simulation approach by modeling the participants and their reactions to the changing economic, financial, and regulatory environments in ESs. (Hernandez et al., 2013) presented a MAS model for Virtual Power Plants based on two aspects: demand forecasting and the coordination of producers and consumers in order to balance the energy production. In the distributed control of Smart Grids field. (Pipattanasomporn, Feroze, and Rahman, 2009) proposed a MAS model to detect upstream outages, by proposing four types of agents with their own roles and responsibilities, namely: a control agent, a distributed energy resource (DER) agent, a user agent, and a database agent. The paper discuss the design and implementation of their MAS solution that includes four steps: agent specification, application analysis, application design and application realization. They use FIPA protocols for agent interaction. This proposal is specific to the Intelligent Distributed Autonomous Power System (IDAPS) which is a distributed smart grid (i.e., IDAPS can be perceived as an intelligent microgrid).

(Logenthiran, Srinivasan, and Shun, 2012) presented a MAS for a specific problem within the smart grids domain. They propose a generic architecture of an intelligent agent, and defining the different types of agents described by their roles, where authors focus on validating the proposed architecture and gives no details on the methodology and the implementation techniques they used to develop the proposed agent solution.

A MAS approach was proposed to power system disturbance diagnosis (Hossack et al., 2002) where authors followed their own MAS methodology. This methodology is breifly described in (McArthur, McDonald, and Hossack, 2003) and was recommended by the IEEE MAS working group for developing MAS solutions in the energy sector (McArthur et al., 2007a): "it contains all the stages required to design an agent system for any specific task, and is therefore a suitable process to follow".

Recently, (Constantin et al., 2017) presented a MAS methodology for MAS for nonresidential buildings, by providing an ontology for MAS in buildings and a data model for the application to nonresidential buildings, both of which were developed with a focus on existing standards and are reusable and expandable. It is a critical work as it is based on the use of energetic standards (such as the IEC 61850 standard) to build the data model and it reuses the existing ontology that are already used as standards, to build their ontology. Furthermore, they proposed a complete methodology for the design and implementation, but it supports particularly the application to nonresidential buildings.

Our review of the literature reveals that existing MAS solutions for energy systems are generally based on ad-hoc models for resolving specific problems, which run on specific agent platforms, without considering the interoperability among different agent platforms (Hahn, Madrigal-Mora, and Fischer, 2009) and that were not intended to be reused (Kremers, 2013). Most of them, to the best of our knowledge, do not consider a complete process of system development. Furthermore, several works have addressed design and implementation issues of MASs but few works have focused on the quality of service for the developed MASs.

3.5 Summary

In this chapter, we have presented the related work for the development of MASs. To improve the development of MAS solutions to solve ES problems, we propose a MAS architectural framework dedicated to Smart Grids.

Many methodologies have proven their efficiency in the development of MAS solutions, however those which follow the model driven engineering techniques are more efficient to enhance reusability, portability and interoperability of designs and implementations (Schmidt, 2006). Furthermore, none of the existing frameworks was intended for developing MASs for ESs where MAS standards (such as FIPA standard) should be followed to meet the Smart Grids requirements presented in A.2.1 and energetic standards to constraint design choices. Thus, a **MAS architectural framework dedicated to Smart Grids** that follows the MDE techniques can be proposed to solve technical and methodological problems.

The presented approaches use the MAS technology in the engineering of software solutions to solve some engineering issues within the energy system domain. Some of them adhere to energetic standards in order to support interoperability in the inter-agent communication, whereas others use agent standards (i.e., FIPA) to support interoperability between agent systems. Thus, a **MAS architectural framework dedicated to Smart Grids** that *follows the MDE techniques* and adheres to energetic and MAS standards can be proposed to solve technical, methodological and interoperability problems.

Proposing a methodological MAS approach that even adheres to energetic standards and agent standards could solve only the methodological and standardization issues. However, there is still a need for an approach that takes in consideration the non-functional requirements of the target application domain in the design and

analysis phase. Thus, a **MAS architectural framework dedicated to Smart Grids** that supports the MAS architecture evaluation can be proposed to solve MAS architectural exploration problems.

In the best of our knowledge, there is no proposal for an MDD MAS methodological approach dedicated to energy system that meets the Smart Grids requirements (i.e. interoperability between agent systems and between agents) and that considers the non-functional requirements of the target application domain to improve the Quality of Services (QoS) of the developed MAS solution.

In this thesis, we address the four challenges presented in 1.2 by proposing an architectural framework, compliant with ISO 42010 (ISO, 2011), to develop MAS solutions dedicated to Smart Grids. This work contributes in the following manner: it defines a **methodology**, to support the use of the MAS4SG architectural framework to address the methodological issues(i); based on a Platform Independent Meta-Model (PIMM) called ML4Agents meta-model to solve technical issues (ii); the proposed framework adheres to the FIPA standard and CIM standard to solve interoperability issues (iii) in the communication and application level, i.e. it provides a set of reusable libraries to be used in the analysis and design phases to model the ontology, the domain's entities and the agent's interaction; to solve the MAS architecture style evaluation issues (iv) it enables, with the ML4Agents concepts, modeling MAS architecture styles which will be evaluated to select the most appropriate architecture style that better answers the modeled non-functional requirements. This is done in the analysis phase and may be considered as a step towards improving the Quality of Services (QoS) at a very early step in the process of developing MASs. A requirement specification phase is supported also by the proposed methodology to enable the specification of non functional and functional requirement.

Table 3.2 compares some of the presented works that specifically target the energy domain or not and that we have decided that they are very significant and suitable for the engineering of MAS solutions dedicated to energy systems. We have defined six fundamental criteria on capabilities and possessions of the proposed approaches and we have evaluated the presented approaches according to those criteria. These six criteria are listed as follows:

- Smart Grids-Specific: solution dedicated to develop MASs in the energy system domain
- Documented: Documentations and guidelines
- Interoperability in the communication level: using standardized interaction protocols for communication and connection
- Interoperability in the application level: using standardised ontology or standard based ontology for the specification of the exchanged knowledge
- QoS: propose mechanism to meet non functional (i.e., QoS) to improve the quality of service
- Portability: provision of a metamodel for platform independent metamodeling of MASs.

The last line in the table presents our contributions and compares them to the state of the art works with respect to the aforementioned criteria.

	MAS Approaches & Methodologies	Methodology	Design Ontology	Portability	Interoperability		QoS
					Communication Level	Application Level	
Smart Grids-Independent	Ingenias Pavón, Gómez-Sanz, and Fuentes, 2006	Y	N	N (JADE)	N		N
	DSML4MAS Hahn, Madrigal-Mora, and Fischer, 2009	Y	Y	Y (JACK&JADE)	Y		N
	PASSI Cossentino and Potts, 2002	Y	Y	Y (FIPA platforms)	Y (FIPA AIP)		N
	ADELFE Bernon et al., 2002	Y	N	N	N		N
Smart Grids-Dependent	PEDA Hossack et al., 2002	Y	Y	N	N	N	N
	Pipattanasomporn, Feroze, and Rahman, 2009	N	Y	N	Y	N	N
	Constantin et al., 2017	N	Y	N	Y	Y (Standard ontology)	N
	MAS4SG	Y	Y	Y (FIPA platforms)	Y (FIPA AIP)	Y (CIM Ontology)	Y

TABLE 3.2: Evaluation of the state of the art's MAS Design methodologies and approaches

3.6 Conclusion

The thesis goal is to provide the principals and processes needed in engineering multi-agent systems for applications in the power and energy sector by offering guidance and recommendations on how MAS can be developed. This thesis aims to help the uptake of the MAS technology within the power industry. By giving the set of conventions, principles and practices for the description of architectures established in the field of Smart-Grids by means of an architectural framework. The proposed framework shall support multiple approaches and processes for the design of MAS dedicated to Smart Grids. Those processes will define the architectural design rules and the set of activities needed to design a MAS architecture.

To validate our approach and to show its feasibility, we do the following: we implement the architectural framework, and we perform experiments on a well known use case from the literature (Logenthiran, Srinivasan, and Shun, 2012). The experimentation results show the effectiveness of our approach and validate its feasibility.

In the following chapter, we describe in detail the proposed framework.

Part II

Thesis Contributions

Chapter 4

A MAS Architecture Framework Dedicated To Smart Grids

This chapter presents an architectural framework as a solution for the challenges encountered in the engineering of Multi-agent solutions in the power sector. In this chapter, we describe in detail an architectural framework named Multi-Agents System For Smart Grids (MAS4SG). This framework supports the MAS development in the domain of Smart Grids regarding the MAS design requirements **(Silva, Castro, and Tedesco, 2003)** in order to establish the key properties of a Multi-Agent Systems.

4.1 Introduction

The previous chapter showed that using MAS solutions for SG software, in a process that complies with the principles of the MDA, requires a consideration of certain requirements of smart grids systems such as interoperability and system performance. This chapter addresses the description of MAS architectures established within a specific domain of application and/or community of stakeholders by means of an architectural framework that supports MAS engineering for energy systems. By support, we mean to give a guidance and recommendations for the development of such solutions in order to meet the smart grids requirements.

This chapter is organized as follows: Section 4.2 explains how the framework conforms to existing standards to solve the interoperability requirement; Section 4.3 introduces the MAS4SG framework. It presents the involved stakeholders, and the different viewpoints that frame the stakeholder concerns and that are important for modeling MAS in a precise and adequate manner. Then, we propose a modeling language for the energy system application domain, and finally, Section 4.4 concludes this chapter.

4.2 Standard conformance

If the application of MAS technology is to be widespread within power engineering, then the adoption of standards that promote interoperability between different systems is essential. The proposed framework is compliant with the existing standards to overcome the problem of interoperability between different multi-agent systems that comes in two different levels. First, the interoperability between agents from different designers, i.e. their ability to discover each other to maintain communications, regardless of the host or platform on where they are located. Second, at the semantic level of communication, interoperability must be ensured among agents, whether belonging to the same platform or not, by having the same data dictionary

in order to understand each other. The considered standards include the MAS standards (e.g. FIPA standard [1]) and their relation to existing energetic standards (e.g., common information model (Uslar et al., 2012)).

The Foundation for Intelligent Physical Agents' (FIPA) was formally accepted as a standard by the committee of the IEEE Computer Society in 2005. FIPA aims to define specifications and standards that can be used to support interoperability between heterogeneous interacting agents and agent-based systems developed by the different companies and organizations.

To ensure interoperability among agents on different platforms at the semantic level of communication, we refer to the SGAM[2] standard to adopt the CIM standard in building the ontology used by agents in communication. This is aligned with IEEE's recommendation (McArthur et al., 2007a) to use an upper CIM ontology (i.e., based on the IEC 61968/61970 Common Information Model (CIM) standard (Uslar et al., 2012)).

In the following we introduce in detail the MAS4SG framework.

4.3 The MAS4SG specification

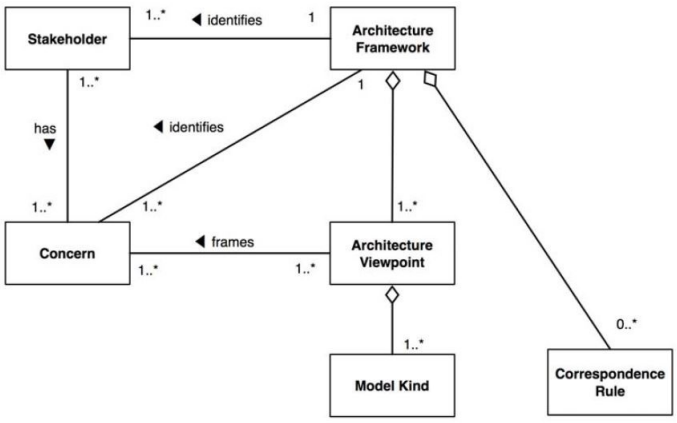

FIGURE 4.1: Conceptual model of an architecture framework

The ISO/IEC/IEEE 42010 Conceptual Model of Architecture Description (ISO, 2011) defines the term Architecture Framework (AF) as:

Definition 4.1. "An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community".

As shown in Figure 4.1, the framework consists of entities such as stakeholders, concerns, architecture viewpoints, model kinds and correspondence rules, as well as the relationships between these entities.

The ISO/IEC/IEEE 42010 defines the Architecture Framework (AF) elements in this way: "Stakeholders of a system have concerns. A concern could be held by one or more stakeholders. Concerns arise throughout the life cycle from system needs

[1] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

[2] SGAM 2012, "Smart grid reference architecture" CEN-CENELEC-ETSI, Tech. Rep.,

and requirements, from design choices and from implementation and operating considerations. A concern could be manifest in many forms, such as in relation to one or more stakeholder needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues pertaining to the system).

An architectural framework shall include one or more architecture viewpoints that frame those concerns. There are two aspects for a viewpoint: the concerns it frames for stakeholders, and the conventions it establishes on views. An architecture viewpoint frames one or more concerns. A concern can be framed by more than one viewpoint. The viewpoint establishes the conventions for constructing, interpreting and analyzing the view to address concerns it frames. Viewpoint conventions can include languages, notations, model kinds, design rules, and/or modelling methods, analysis techniques and other operations on views. An architecture viewpoint shall specify one or more model kinds used in this viewpoint" (ISO, 2011). In the sequel, the MAS4SG architecture framework stakeholders, viewpoints, model kinds, and modeling language are described.

4.3.1 MAS4SG Stakeholders

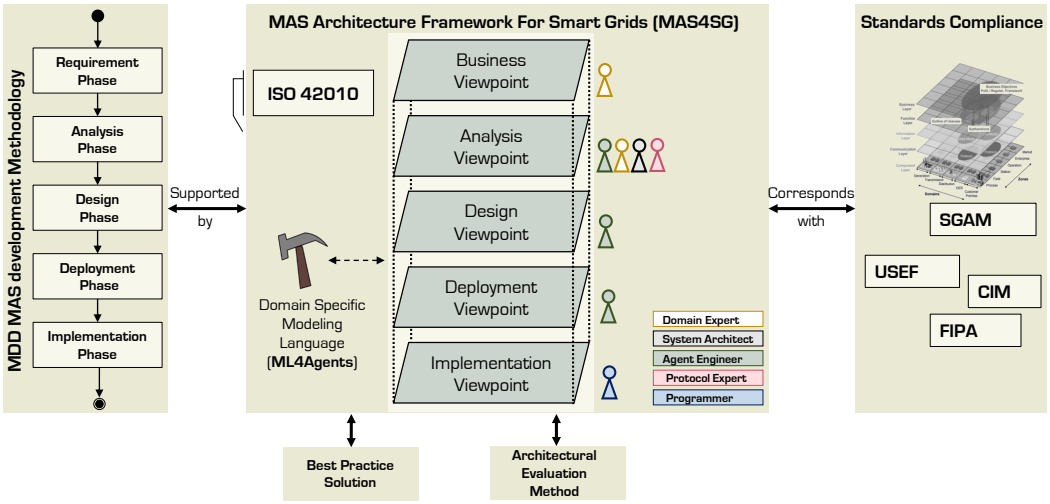

FIGURE 4.2: MAS4SG Architecture Framework's Stakeholders

We consider the following stakeholders for MAS development for applications in power engineering domain (c.f. Figure 4.2), which are derived from MAS approaches (e.g., (Warwas et al., 2012)):

- *Energy Domain Expert*: Domain experts have the knowledge regarding the application domain for the energy system. An energy domain expert is responsible for: (i) the specification of the functional and non functional requirements, i.e. the problem decomposition and the selection of the quality attributes that constitute the key drivers for designing software systems in the process of architectural decision making; (ii) the design of the agent data model, and (iii) the design of the ontology for the application domain.

- *Agent System Architect*: The agent system architects are responsible for creating a library of broadly applied MAS architectural style to be used for the evaluation of MAS architecture styles. The agent system architect selects an appropriate architectural style derived during system engineering and software requirements analysis.

- *Protocol Experts*: Protocol experts are responsible for specifying communication and negotiation FIPA protocols. They offer a library of FIPA interaction protocols to facilitate the interaction modeling within a MAS application. Indeed, a set of reusable model artifacts are given to be used to help engineers in constructing MAS mainly for interaction specifications.

- *Agent Engineer*: The agent engineer is the end user of the development environment. He uses an agent methodology to develop the MAS application. Model repositories are used to cooperate with the other stakeholders and for reusing existing model artifacts.

- *Programmer:* The programmer is responsible for applying the transformation techniques (e.g., model to model and model to text transformation techniques) to generate an executable code on a specific agent platform. He is responsible for refining the generated code where necessary. He is involved in the modeling, code generation, and evaluation tasks.

4.3.2 The MAS4SG Viewpoints

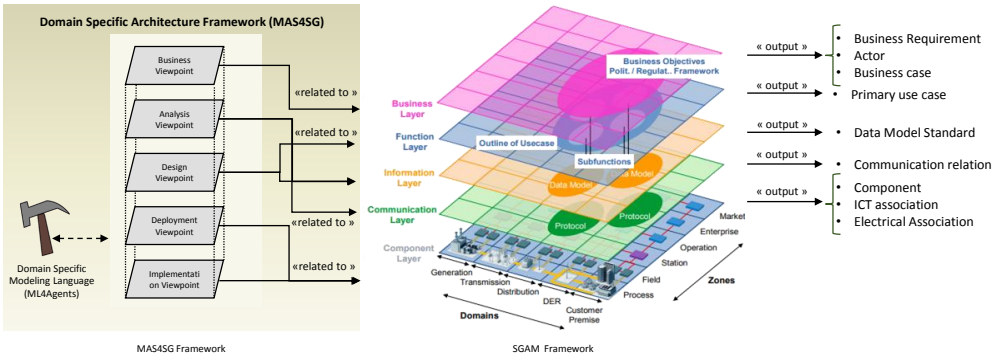

FIGURE 4.3: MAS4SG's viewpoints relate the SGAM's viewpoint output

The framework's viewpoints frame the stakeholder concerns and give the framework the ability to be connected to a model driven methodology (c.f. Figure 4.3). The deliverable of the methodology's phases are a set of system specifications that provide a set of guidelines for structuring the specifications expressed as models corresponding to different abstraction layers. Multi-agent systems can be seen through several angles and therefore many possible viewpoints can be proposed. Similar to the SGAM viewpoints, we consider the following viewpoints:

- *Business viewpoint*: connotes the domain requirement description. It specifies the system requirements (i.e., functional and non-functional requirements), which could be derived from the existing requirements of the application development. In alignment with the SGAM standard, the business viewpoint

is related to the SGAM business layer (c.f. Figure 4.3). This viewpoint focuses on the context and the requirements of the system without considering its structure or processing. The viewpoint's outputs is a set of functional and non-functional requirements.

- *Analysis viewpoint*: deals with the detailed analysis of the requirements and identifies the roles and interactions required to achieve the global goal. The analysis viewpoint considers also two main concepts: the ontology design and the knowledge needed to fulfill the requirements defined by the business viewpoint. The viewpoint outputs are the roles, abstract goals, interactions and organizations models that design a part of the abstract specification focusing on the operational capabilities of a system, and without considering the technologies related to a specific agent platform. The analysis viewpoint corresponds to the function, information and communication SGAM's layer (c.f. Figure 4.3).

- *Design viewpoint*: focuses on identifying the types of agents and the functions required to perform their role, which is identified by the inter-agent communications. The viewpoint's outputs are given by a set of specific goals, agent types, behaviors, collaboration and capabilities models that complete designing the abstract specification from a specific agent platform.

- *Deployment viewpoint*: identifies the concrete instances of agents and organizations defined by the analysis and the design views. The viewpoint's outputs are given by a set of instances, configuration and resources. The deployment viewpoint corresponds to the SGAM's component layer (c.f. Figure 4.3).

- *Implementation viewpoint*: the model transformations are used to generate a concrete multi-agent platform-dependent design model (i.e., from a Platform Independent Model to a Platform Specific Model that reflect the lowest abstract modeling layer) and manual refinements are applied where necessary. Another transformation is considered by this viewpoint to generate the code from the generated platform specific model (PSM to Text).

In order to be able to describe these viewpoints, an agent-platform independent modeling language (abstract syntax) has to be proposed to support the agent system modeling. The architectural framework will be completed by a concrete syntax, defining the notation of that language.

4.3.3 MAS4SG Model Kinds

The proposed ML4Agents language is able to support the modeling of all the properties and relationships of the MAS entities. The ML4Agents concrete syntax is formulated to allow the use of graphical notation (diagrams) for each MAS aspect. In the following, we define the needed diagrams for each viewpoint of the MAS4SG architectural framework:

- *Business viewpoint*: Requirement diagram

- *Analysis viewpoint*: Ontology diagram, Environment Diagram, Interaction diagram and Service Diagram

- *Design viewpoint*: Agent diagram, Role diagram, Organization diagram, Collaboration diagram and Behavior diagram

- *Deployment viewpoint*: Deployment diagram
- *Implementation viewpoint*: Platform Description Model diagram

The description of these diagrams is detailed in section 6.2.3.

4.3.4 The Modeling Language ML4Agents

4.3.4.1 ML4Agents Rational

Modeling languages can be used at a high level of abstraction to abstract specifications and designs from implementation technologies and platform specificity. They are needed for creating MAS views matching the framework's viewpoints.

According to (Sturm and Shehory, 2014), agent-based software engineering, is to: (i) design the agent systems based on an agent-based methodology, and to (ii) take the resulting design artifacts as an entry to manually implement the agent system. This manual transformation from an abstract specification into a concrete implementation, introduces a gap between design and implementation. Thus, a platform independent modeling language is needed to allow abstract modeling (i.e.from agent platform) of an agent system, which can be transformed later to a platform specific model and then to a code.

The DSML4MAS (Hahn, 2008) modeling language, as discussed in 3.3, allows modeling platform independent agent systems and consists of an abstract syntax (PIM4Agents meta-model (Hahn, Madrigal-Mora, and Fischer, 2009)) providing the vocabulary of the language.

The proposed ML4Agents modeling language is based on the PIM4Agents metamodel. ML4Agents's core includes: (i) the concepts from the PIM4Agent metamodel that manipulate many concepts common to agents; (ii) concepts from FIPA constraints to be adapted to FIPA-compliant agent platform, and (iii) concepts for the description of MAS architecture style. The intended modeling language supports the PIM design level. We have significantly extended the PIM4Agents metamodel with specific modeling elements that allow the design of FIPA specifications. Accordingly, the proposed ML4Agents supports many common concepts to the agents and adds concepts from FIPA constraints in order to be adapted to the FIPA-Compliant agent Platforms.

The PIM4Agents's concepts, as discussed in 3.3, are independent from the agent platforms. By adding concepts for FIPA specifications we restrict the execution on only FIPA compliant platforms in order to guarantee the interoperability between the different agent systems.

Since our aim is not to compete with the existing modeling languages by proposing a new one, we have chosen the most suitable modeling language and we have adapted it according to the energy systems development needs. The abstract syntax of ML4Agents is presented in the sequel.

4.3.4.2 The ML4Agents Metamodel: Core Elements

This section presents the abstract syntax for the proposed modeling language. We have defined a metamodel for agents called ML4Agents that includes specific concepts for supporting the ontology and the requirement specifications. In particular, we consider the FIPA specifications to describe the Agent Interaction Protocol (AIP) and the Agent Communication Language (ACL). The foundation of our metamodel is the PIM4Agents metamodel.

The ML4Agents meta-model defines a set of analysis, design, and implementation concepts and a set of constraints between them. It defines the main concepts and relationships used to define MASs to be executed on FIPA-compliant platform. Our choices to adapt and extend the concept of PIM4Agents are motivated by the need to incorporate the specific properties of the energetic systems and meet its requirements, like interoperability and attribute-driven design (see 5.5). A design in the ML4Agents metamodel describes the overview of the metamodel core packages. It is structured in ten packages focusing on a specific aspects of a MAS.

For each package, the semantic descriptions corresponding to the contained concepts and their properties are provided in the following section. The metamodel of each aspect is depicted by a figure where the highlighted concepts present the extension of the existing PIM4Agents metamodel.

- **The ML4Agents Package:** The ML4Agents package describes the internal subpackages of the ML4Agents meta model and their relationships. The relationships are UML «import» dependency, which is a directed relationship between an importing namespace and imported package, that allows the use of unqualified names to refer to the package members from the other namespace(s).

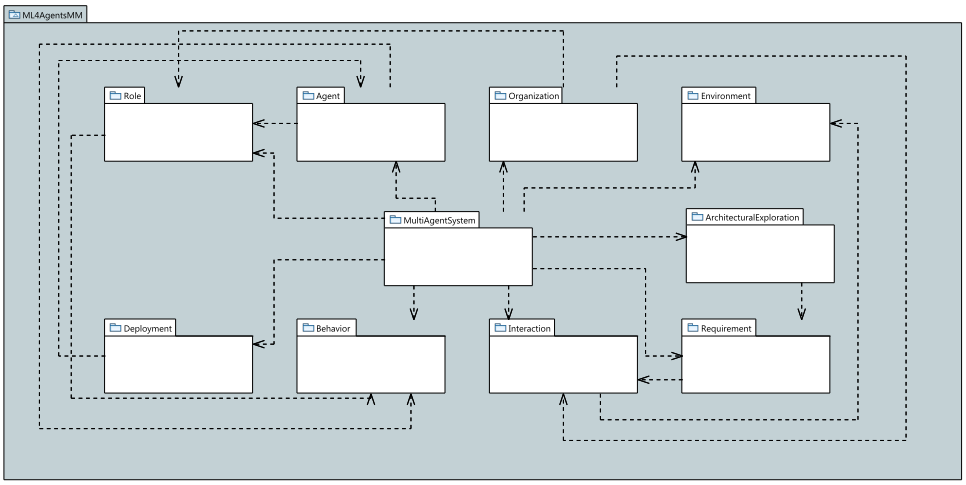

FIGURE 4.4: The ML4Agents model diagram

The purpose and contents of each package denoted in the Figure 4.4 are described in subsequent sections.

The ML4Agents Model elements description:

- *Multiagent system package*: it allows defining MASs by introducing the main blocks for describing a MAS, such as: roles, agents, organizations and interactions;
- *Requirement package*: it contains the concepts to describe the system functional and non-functional requirements;
- *Agent package*: it contains the concepts to describe the agents, the capabilities they have to solve tasks and the roles they play within the MAS. Additionally, the agent package defines to which resources an agent has access to and which behaviors it can use to solve tasks;

- *Organization package*: it contains the concepts to describe how single autonomous entities cooperate within the MAS and how complex organizational structures can be defined;
- *Role package*: it covers the abstract representations of functional positions of autonomous entities within an organization or other social relationships. Moreover, the role package describes the provided and the required services by roles;
- *Behavioral package*: contains the concepts to describe the internal behavior of agent or plans used for achieving predefined objectives or goals, it can be defined in terms of combining simple actions to more complex control plans;
- *Interaction package*: it contains the concepts to describe how the flexible interaction and interaction protocols takes place between autonomous entities or organizations;
- *Environment package*: it contains the concepts to describe any kind of resource that is dynamically created, shared, or used by the agents or organizations;
- *Deployment package*: it contains the concepts to describe the MAS application at runtime, including instances of agents and organizations.
- *Architectural Exploration package*: it contains the concepts to describe the MAS architectural style classification, including the dependency between the architectural styles and the non functional requirements, i.e., the architectural style may influence or support the non functional requirements (i.e., quality attributes).

- **The MultiAgentSystem Package:** The multiagent system package, which is depicted in Fig. 4.5, comprises the concept of *MultiAgentSystem* and *Message*.
  The MultiAgentSystem Package concepts description:

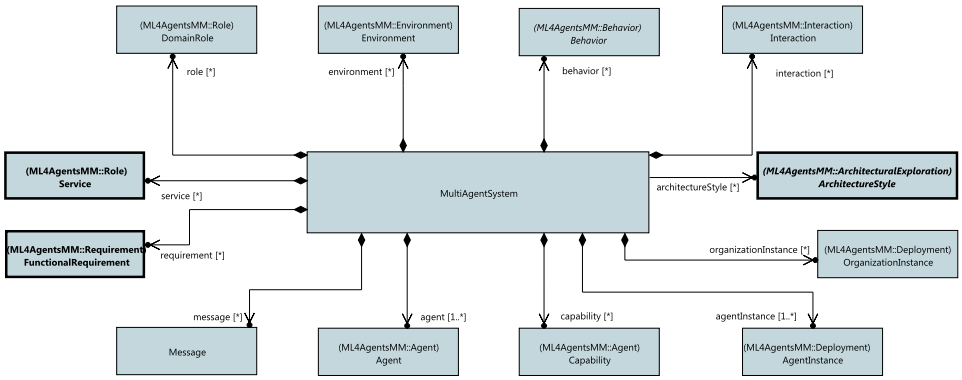

FIGURE 4.5: The MultiAgentSystem package

- ***The MultiAgentSystem concept:***
  - **Definition** The multiagent system concept is the system root element that is composed of the core concepts of MASs (i.e. Agent, Behavior, DomainRole, . . .).
  - **Generalizations**
    - None

* **Associations**
  - agent: Agent::Agent[1] represents all different kinds of Agent types composing the MAS
  - agentInstance: Deployment::AgentInstances[*] represents all run-time AgentInstances available in the MAS
  - organizationInstance: Deployment::OrganizationInstances[*] represents all run-time OrganizationInstances available in the running system
  - role: Role::DomainRoles[*] all different kinds of DomainRoles available to be played by Agents
  - behavior: Behavior::Behaviors[*] internal Behaviors that are used by Agents for achieving goals
  - interaction:Interaction::Interaction[*] a communication pattern as an allowed sequence of ACL messages between agents and the constraints on the content of those messages
  - capability: Behavior::Capability[*] defines all sorts of Capabilities that can be owned by any entity within the MAS
  - environment: Environment::Environment[*] constitutes the collection of Environments and contained Resources that can be accessed by any kind of entity (i.e. Agent or Organization) in the MAS
  - message: Message[*] represents the kind of Messages that are sent between Agents, possibly in accordance with the Interactions referred by the interaction attribute
  - service: Role::Service[*] illustrates all different kinds of Service available provided and required within a MAS
  - requirement: Requirement::Requirement [*] specifies all Requirements required by the MAS
* **Attributes**
  - Name: String[1]
    Represents the unique identifier of the Multi-agent System concept
* **Constraints**
  - None

- **The Requirement Package:** The Requirements package, which is depicted in Fig. 4.6, focuses on the concepts of *FunctionalRequirement* and *NonFunctionalRequirement* specializing the concept *Requirement*. All kind of functional requirement should be fulfilled by at least one *Plan*.
  The Requirement Package concepts description:
  - ***The FunctionalRequirement concept:***
    * **Definition** The FunctionalRequirement concept specifies a function that a system or system component must be able to perform. It is a function that the MAS has to exhibit or the behaviour of the system in terms of interactions perceived by the user
    * **Generalizations**
      - Requirement::Requirement
    * **Associations**

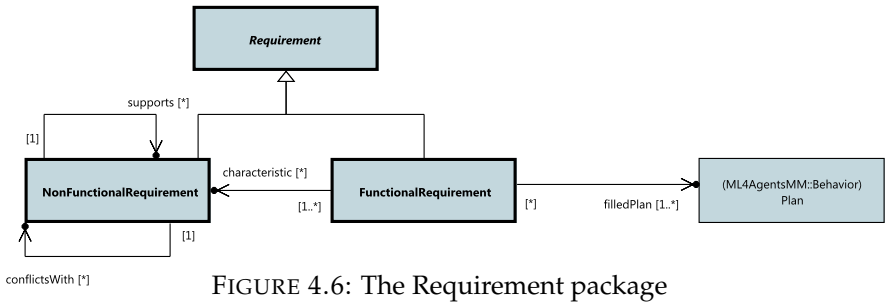

FIGURE 4.6: The Requirement package

- filledPlan: Behaviour::Plan[1..*] specifies the plans that satisfy this functional requirement
- characteristic: Requirement::NonFunctionalRequirement[*] defines the characteristics of this functional requirement

* **Attributes**
  - None
* **Constraints**
  - None

– ***The NonFunctionalRequirement concept:***

* **Definition** The NonFunctionalRequirement concept provides a constraint on the developed MAS. It represent the quality attributes that are related to the performance of the MAS and constraint the functionality of the MAS.
* **Generalizations**
  - Requirement::Requirement
* **Associations**
  - conflictsWith: NonFunctionalRequirement[0..*] declares that the source NonFunctionalRequirement is in conflict with the target NonFunctionalRequirement, i.e., to obtain a better understanding for the potential conflicts and the trade-offs needed between software quality attributes. No need to trade off the quality attribute if there is no conflict between the non functional requirements (Quality Attribute).
  - supports: Requirement::NonFunctionalRequirement[0..*] declares that the source NonFunctionalRequirement supports the target NonFunctionalRequirement.
* **Attributes**
  - Name: String[1]
* **Constraints**
  - None

- **The Agent Package:** The Agent Package which is depicted in 4.7, focuses on the concepts of *Agent* and *Capability*.
  The Agent Package concepts description:

– ***The Agent concept:***

* **Definition** The ***Agent*** concept is used to represents an autonomous entity capable of acting in the environment and interacts with other agents in its MAS. The agent has to fulfill its goals for which it uses its capabilities.

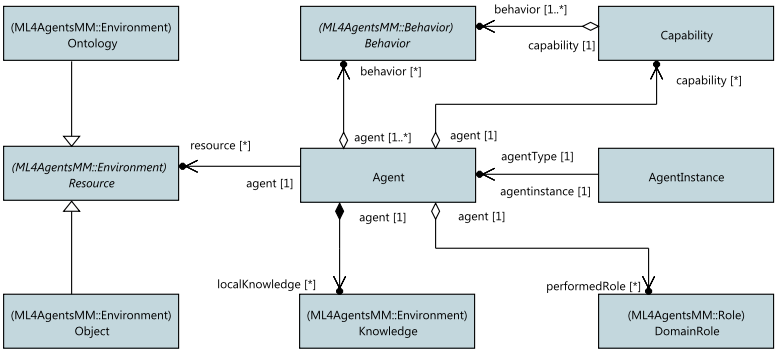

FIGURE 4.7: The Agent package

* **Generalizations**
  - None
* **Associations**
  - capability: Capability[*] The agent may have certain *Capability* that represent the set of *Behavior* the *Agent* owns.
  - behavior: Behavior::Behavior[*] *Behaviour*s that Agent can implement. Behaviors are shared by many agents and they are are not related specifically to one Agent.
  - performedRole: Role::DomainRole[*] Agent plays one or more *DomainRole*
  - localKnowledge: Environment::Knowledge [*] Private Knowledge represents the information (i.e. beliefs) an Agent could have about the world used for, among others, making decisions.
  - resource: Environment::Resource[*] An Agent has access to a set of Resources from its surrounding *Environment*.
* **Attributes**
  - Name: String[1]
* **Constraints**
  - self.behavior->union (self.capability.behavior)-> union (self.performedRole.providedBehavior) ->size()>=1
    Agent has at least one associated behavior, which is either used directly by the Agent, through the Capabilities or performed DomainRoles.
  - self.behavior->union (self.capability.behavior)-> union (self.performedRole.providedBehavior) -> includesAll (self.performedRole.requiredBehavior)
    Any Behavior required by a performed DomainRole must either be provided by the Agent's Behavior or Capabilities.

- **The Role Package:** The Role Package, which is depicted in 4.8, focuses on the concepts of ***Role***, ***DomainRole*** and ***ActorRole***. Additionally, we introduce the concept ***Service*** provided or required by *DomainRole*s within a Collaboration. *DomainRole* can provides or require a Service, i.e. the DomainRole can provide a service when it participate to a *Protocol* and can require a *Service* by initiating a protocol, thus the *DomainRole* can initiate or participate in a protocol. The

concept of *ActorRole* has two specializations, i.e. *Initiator* and *Participant*.
The Role Package concepts description:

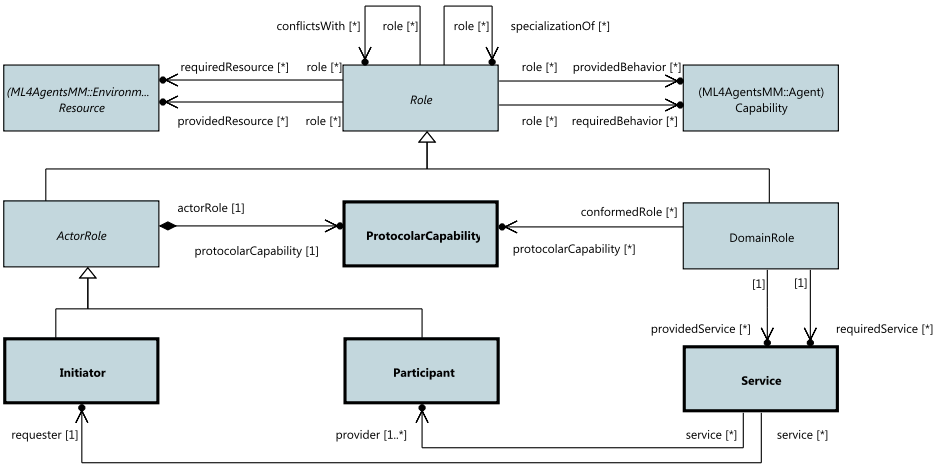

FIGURE 4.8: The Role package

- ***The DomainRole concept:***
  - **Definition** DomainRoles are portions of the agent's social behavior characterized by some specificity such as a goal, or providing a functionality/service, and doing so, it can also access to some resources.
  - **Generalizations**
    - Role
  - **Associations**
    - requiredService: Service[*] represents the services required by this domain role
    - providedService: Service[*] represents the services provided by this domain role
    - protocularCapability: protocularCapability[*] the provided capability for initiating and/or participating an interaction protocol.
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - self.providedService->forAll(svc | self.providedBehavior ->includes(svc.provider.protocolarCapability))
      If the DomainRole provides a service within a collaboration, then the DomainRole should conform to the protocolarCapibility specified by the ActorRole who provides that service.
- ***The Service concept:***
  - **Definition** In each FIPA protocol, there is an initiator that requests a service from a participant. Thus, we propose the Service concept. The domain roles collaborate to request and provide services, where each service agreed a protocol that specifies the choreography of the

interaction, i.e., the interaction process for entities that request and provide the service. In FIPA [3], the service is described by: a name, a type, a set of protocols interaction supported by the service, a list of ontology supported by the service, languages description supported by the service and the name of the agent who owns the service and the list of service properties. In ML4Agent the service concept is related to the domain role, and thus the agent owns the service provided by the domain role he plays.

* **Generalizations**
  - None
* **Associations**
  - requester: Initiator[1] represents the actor role initiator that initiates a protocol in order to request a service (i.e., the interaction object of that protocol).
  - provider: Participant[1..*] represents the actor role participant that responds to a protocol in order to provide a service (i.e., the interaction object of that protocol).
  - supportedProtocol: Interaction::Protocol[1] a protocol interaction supported by the service. A service can be requested and provided within an interaction that agreed to one protocol.
* **Attributes**
  - Name: String[1]
* **Constraints**
  - self.supportedProtocol->size()=1
    each service should be supported by one protocol

– ***The ActorRole concept:***
  * **Definition** The ActorRole concept defines the position of Agents (or their run-time representatives, i.e., AgentInstances) within an Interaction.
  * **Generalizations**
    - Role
  * **Associations**
    - protocolarCapability: ProtocolarCapability[1] Any actor role should provide a particular capability that specifies the interaction process associated to that role. The protocolarCapability gives an order for the sending and receiving messages within a specific protocol under temporal constraints.
  * **Attributes**
    - Name: String[1]
  * **Constraints**
    - self.protocolarCapability->size()=1
      each actorRole should provide one protocolar capability

- **The Organization Package:** The Organization Package which is depicted in Fig. 4.9, focuses on the concepts of *Organization*, *OrganizationalAgent*, *Collaboration*, *DomainRoleBinding* and *ActorRoleBinding* .

[3]FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

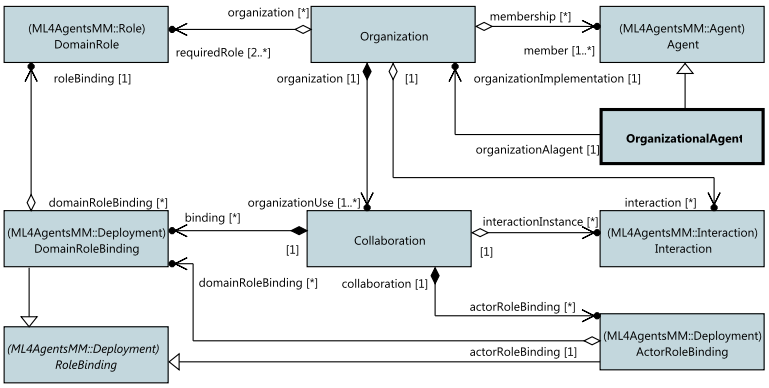

FIGURE 4.9: The Organization package

The Organization Package concepts description:

- ***The Organization concept:***
  - **Definition** The Organization concept, as it is defined in (Hahn, 2013), can be used to provide a social structure to foster the interaction of its members, Organization lays the foundation for social ability, it only manifests the structure of the Organization by characterizing which DomainRoles are part and which Interactions are used by the DomainRoles addressed.
  - **Generalizations**
    - None
  - **Associations**
    - requiredRole: Role::DomainRole[2..*] defines the DomainRoles the Organization needs to assign in order to achieve the intended goals coordinate its DomainRoles
    - organizationUse: Collaboration[1..*] refers to the different kinds of collaborations inside an organization.
    - member: Agent::Agent[1..*] refers to the agents member that plays domain roles required by the organization
  - **Attributes**
    - Name: String[1] defines the name of the Organization
  - **Constraints**
    - self.requiredRole->includes (self.organizationUse.binding.roleBinding) restricts the set of DomainRoles a Collaboration uses to the set of required Organization's DomainRoles.
    - self.interaction->includes(self.organizationUse.interactionInstance) restricts the Interactions referred to by a Collaboration to the Interactions that can be applied by its Organization
    - self.organizationUse -> forAll (c1, c2 : Collaboration | (c1.binding.roleBinding) = (c2.binding.roleBinding) implies ((c1.binding.membership) -> union(c1.binding.membership)) <> ((c1.binding.membership) -> intersection(c1.binding.membership))) states that any two different Collaborations of an Organization, which require the same set of DomainRoles, must at least refer to one different AgentInstance through the Membership concept.

- self.requiredRole->includes(self.organizationUse.actorRoleBinding.domainRoleBinding) the Collaboration's ActorBindings must only refer to DomainRoles (through the DomainRoleBindings), which are addressed by the Organization as required

– ***The OrganizationalAgent concept:***

* **Definition** The OrganizationalAgent is a special kind of Agent, that can be considered as an autonomous and intelligent entity able to interact with other Agents or Organizations. Therefore , the OrganizationalAgent can perform Roles and have Capabilities which can be performed by its members, be it Agents or Organizations.
* **Generalizations**
  - Agent::Agent
* **Associations**
  - organizationImplementation: Organization[1] defines an organization that can be considered as its implementation
* **Attributes**
  - Name: String[1] defines the name of the Organizational Agent
* **Constraints**
  - self.organizationImplementation->size()=1

– ***The Collaboration concept:***

* **Definition** The concept of Collaboration defines the relationship between DomainRoles required by an Organization and ActorRoles of an Interaction. It make assumptions about which of its organization's DomainRoles interact in which manner in its organization's social context.
* **Generalizations**
  - None
* **Associations**
  - interactionInstance: Interaction::Interaction[*] depicts the different types of Interactions the Collaboration instantiates to define how organizational members represented by the required *DomainRole* are coordinated.
  - binding: Deployment::DomainRoleBinding[*] defines the Collaboration's bindings between AgentInstances and DomainRoles
  - actorBinding: Deployment::ActorRoleBinding[*] describes the Collaboration's bindings between ActorRoles of its Interactions and DomainRoles of its Organization.
* **Attributes**
  - Name: String[1] defines the name of the Collaboration
* **Constraints**
  - self.actorBinding->forAll(ab1,ab2 | ab1.name<>ab2.name) all ActorBindings the Collaboration makes use of are unique
  - self.binding->forAll(b1,b2 | b1.name<>b2.name) all DomainRoleBindings the Collaboration makes use of are unique
  - self.binding->size()>=2

- (self.interactionInstance.actor.representedRole) =
  (self.actorRoleBinding.actorRole)
  the set of ActorRoless part of any Interaction the Collaboration makes use of (through the variable interactionInstance) is equal to the ActorRole referred to by the Collaboration's ActorRoleBindings.
- self.binding->forAll(b1, b2 | b1.roleBinding.conflictsWith = b2.roleBinding or b2.roleBinding.conflictsWith = b1.roleBinding implies
  ((b1.membership.agentInstance)-> intersection
  (b2.membership.agentInstance))->size()=0)
  if two DomainRoles have a conflict with each other the same AgentInstance must not be bound to both DomainRoles at the same time.
- self.actorRoleBinding->forAll(ab1, ab2 | ab1.actorRole.conflictsWith = ab2.actorRole or ab2.actorRole.conflictsWith = ab1.actorRole implies
  ((ab1.domainRoleBinding.membership.agentInstance) ->intersection
  (ab2.domainRoleBinding.membership.agentInstance)) ->size()=0
  if two ActorRoles have a conflict with each other the same AgentInstance must not be bound to both ActorRoles at the same time.
- self.binding->forAll(b1,b2 | b1.roleBinding <> b2.roleBinding)
  A DomainRole must not be addressed by two or more DomainRoleBindings within the same Collaboration.
- self.actorRoleBinding->forAll(ab1,ab2 | ab1.domainRoleBinding <> ab2.domainRoleBinding)
  A DomainRoleBindings must not be addressed by two or more ActorRoleBindings within the same Collaboration.
- self.binding->includes(self.actorRoleBinding.domainRoleBinding)
  ensures that any DomainRoleBinding addressed by an ActorRoleBinding is part of the Collaboration's DomainRoleBindings.

- **The Behavioral Package:** The Behavioral Package, which is depicted in 4.10, includes the concept to describes how behaviors and plans are structured. It focuses on the concept *Behavior*, *Plan*, *Task* and *AgentAction*.

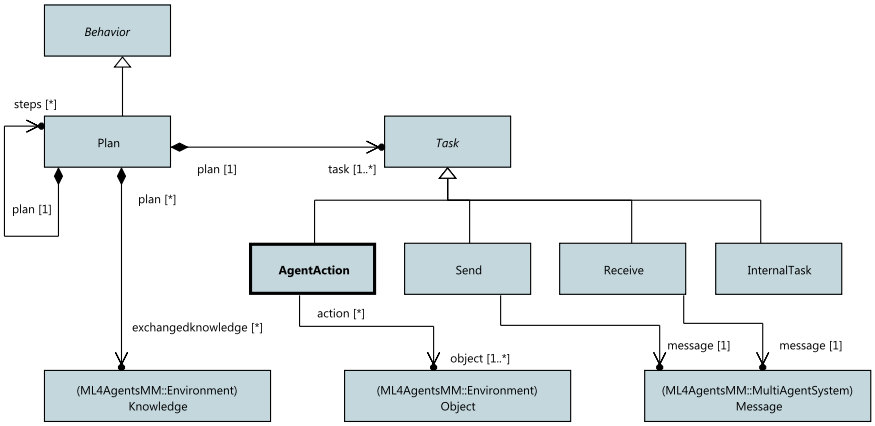

FIGURE 4.10: The Behavior package

The Behavior Package concepts description:

- ***The Plan concept:***
  - **Definition** The *Plan* is a mechanism to specify an Agent's internal processes. Plans could either be composed by more complex control structures (i.e. plan composed by other plans) or by simple atomic activities (i.e. Task).
  - **Generalizations**
    - Behavior
  - **Associations**
    - steps: Plan[*] defines all associated complex Plan used for achieving goals
    - exchangedknowledge: Environment::Knowledge[*] depicts all Knowledge that are globally accessible in the Plan
    - task: Task: [1..*] defines all associated basic tasks used for achieving goals
    - satisfiedRequirement: Requirement::FunctionalRequirement[*] represents the FunctionalRequirements fulfilled by the Plan
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - ((self.steps.task)->union(self.task))->size()>=1
      each plan must own at least one task
- ***The Task concept:***
  - **Definition** The *Task* concept specifies a Behavior like sending and receiving a Message.
  - **Generalizations**
    - None
  - **Associations**
    - None
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - None
- ***The AgentAction concept:***
  - **Definition** The concept *Action* is a specialization of the *Task* concept. It is considered as a simple behavior performed by agent to interact with the *Object* of its *Environment* such as sensing or actuating behavior.
  - **Generalizations**
    - Task
  - **Associations**
    - object: Object[1..*] refers to the objects which the action interact with (i.e. action like act on object or perceive object)
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - None

- **The Interaction Package:** Fig. 4.11 depicts the interaction aspect of the ML4Agents Metamodel. This interaction aspect includes the concepts *Interaction*, *Protocol*, *ACLMessage*, and *Performative*, as well as, *Actor*. The ability to communicate is one of the core characteristics of the agents and the organizations in MAS. **The**

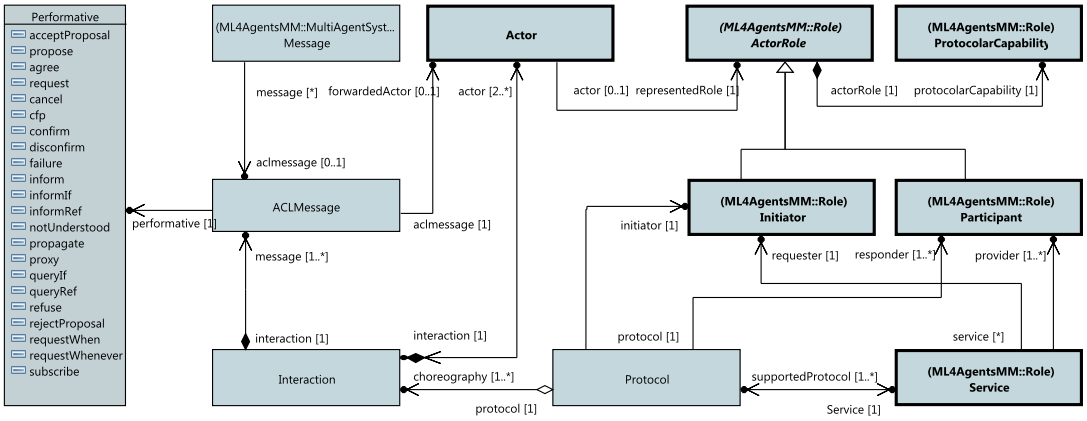

FIGURE 4.11: The Interaction package

Interaction Package concepts description:

- ***The Protocol concept:***
  - **Definition** The *Protocol* concept defines (i) a communication pattern between several parties as an allowed sequence of messages and (ii) the constraints of the content of these messages to form a conversation of a particular type.
  - **Generalizations**
    - None
  - **Associations**
    - choreography: Interaction[1..*] The choreography describes the details of message exchanges between two Actors that represent the protocol's ActorRoles (i.e., the Initiator and Participant actorRoles))
    - initiator: Role::Initiator[1] defines the Initiator role that initiates the protocol
    - responder: Role::Participant[1..*] defines all Participant role that respond the protocol
    - service: Role::Service [1] refer to the service associated to the protocol (i.e., the service is the interaction object for which the protocol interaction is made)
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - self.choreography.actor.representedRole-> includes(self.initiator-> union(self.responder))
      the initiator and the participant associated to the protocol should be represented by its choreography's actors.

- self.choreography.actor->size()=2
  the associated interaction for any protocol should own only two actors

– *The Interaction concept:*
  * **Definition** The Interaction concept is a communication pattern that describes the details of message exchanges between two Actor that represent the protocol's ActorRoles (i.e., the Initiator and Participant actorRoles))
  * **Generalizations**
    - None
  * **Associations**
    - actor: Actor[2..*] defines all actors involved in the interaction
    - messages: ACLMessage[1..*] defines all exchanged ACL messages within the interaction
  * **Attributes**
    - Name: String[1]
  * **Constraints**
    - None

– *The Actor concept:*
  * **Definition** The Actor concept represents a lifeline in an interaction pattern, it can receive and send asynchronous messages. It represents an actorRole that initiates or responds to the protocol to which its interaction is assotiated.
  * **Generalizations**
    - None
  * **Associations**
    - representedRole: Role::ActorRole[1] refers to the ActorRole it represent in the interaction
    - interaction: Interaction[1] refers to the interaction in which the actor is involved
  * **Attributes**
    - Name: String[1]
  * **Constraints**
    - None

– *The ACLMessage concept:* the ACLMessage concept specifies the message exchange between Actors in an Interaction.
  * **Definition** The ACLMessage concept specifies the message exchange between Actors in an Interactions.
  * **Generalizations**
    - None
  * **Associations**
    - performative: Performative[1] defines the FIPA Performative of the ACLMessage (e.g., Request, Inform, ...).
    - forwardedActor: Actor[0..1] represents the Actor to whom the ACLMessage is forwarded
  * **Attributes**

· Name: String[1]

* **Constraints**
  · None

- **The Environment Package:** The Environment Package, which is depicted in Fig. 4.12, focuses on the concept *Environment*, *Object*, *Ontology*, *OntologyElement* and *Knowledge*.
  The Environment Package concepts description:

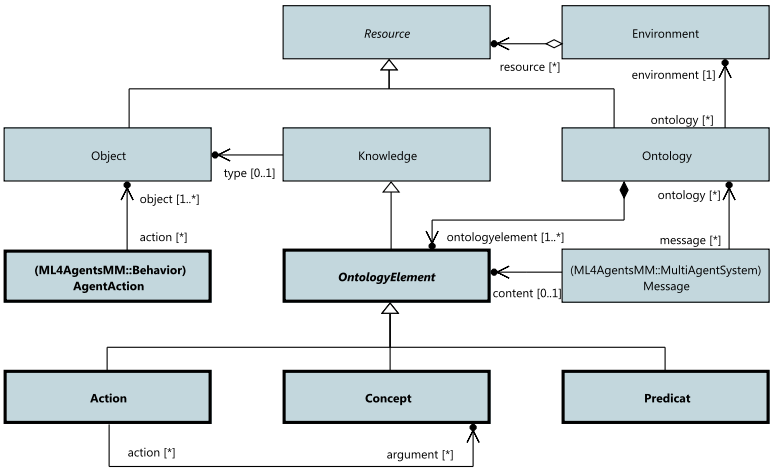

FIGURE 4.12: The Environment package

– *The Environment concept:*
  * **Definition** The *Environment* concept illustrates any kind of information or resources that can be accessed and used by agents. It may comprehend objects that can be either used by the agents, or information that can be perceived. According to PIM4Agents the environment is treated in terms of resources like objects available to the agent society. These resources can be accessed and changed by any entity having access to them.
  * **Generalizations**
    · None
  * **Associations**
    · resource: Resource[*] specifies the set of Resources available to the agents of the Environment.
  * **Attributes**
    · Name: String[1] defines the name of the Environment
  * **Constraints**
    · None
– *The Object concept:*
  * **Definition** The object concept defines the object that the agents can interact with (i.e. whether to perceive it or act on it) on its environment. Agents do not exist in pure isolation. Instead, agents normally interact with objects or other agents to solve particular tasks and goals. Objects are part of the environment.

    - **Generalizations**
        - None
    - **Associations**
        - None
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - None

- ***The Ontology concept:***
    - **Definition** The concept ***Ontology*** defines a representation of the environment. An ontology is used to represent the knowledge that is shared between different entities. It provides the terms and vocabulary used to represent knowledge so that both sender and receiver can understand.
      It is described in terms of *Concept*, *Action* and *Predicate*. The proposed representation is inspired by the representation of ontology in the PASSI metamodel (Chella et al., 2006).
      This ontology provides the following three elements (Ontology Element):
      - *Concept*: to designate an entity of the domain,
      - *Action*: to designate a transformation of a concept,
      - *Predicate*: to designate a predicate related to a set of concepts.
    - **Generalizations**
        - None
    - **Associations**
        - ontologyElement: OntologyElement[1..*] refers to all the included OntologyElement
        - environment: Environment [1] refers to the environment it represents.
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - None

  The concept of *OntologyElement* and its specializations: *Concept*, *Action* and *Predicate* are presented in Appendix **??**.

- ***The Knowledge concept:***
    - **Definition** The concept of *Knowledge* represents the information (i.e. beliefs) an Agent could have about the world used for, among others, making decisions. Moreover, the ontology represents the shared information (exchanged knowledge) where the concept (an ontology element) that describe the domain entity can be a specialization of the Knowledge concept. Thus the agent can have knowledge as instances of the Concept ontology element.
    - **Generalizations**
        - None
    - **Associations**

- type: Object[0..1] depicts the Object type represented by the Knowledge

* **Attributes**
  - Name: String[1]
* **Constraints**
  - None

- **The Deployment Package:** The Deployment Metamodel which is depicted in 4.13, includes the concepts *AgentInstance*, *RoleBinding*, *DomainRoleBinding*, *ActorBinding* and *Membership* The deployment aspect that defines (i) the runtime entities and (ii) how they are grouped into social structures defined in the organization aspect.
  The Deployment Package concepts description:

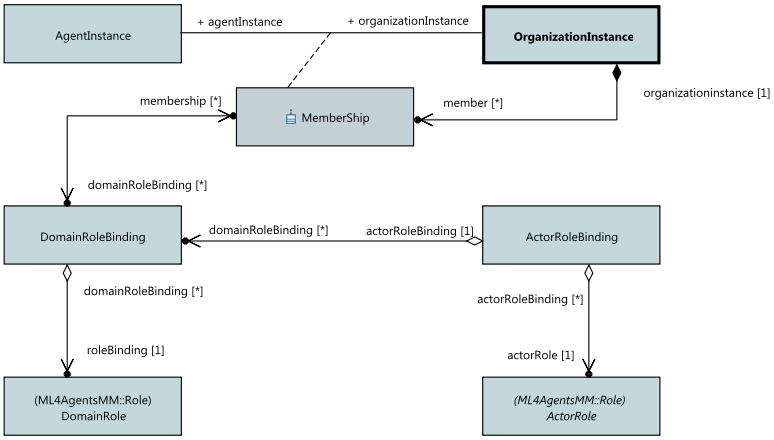

FIGURE 4.13: The Deployment package

– ***The AgentInstance concept:***

* **Definition** Similar to PIM4Agents, the concept of AgentInstance is introduced to specify the autonomous interactive entity in the running system. AgentInstances are directly assigned to either DomainRoles or Actors as role fillers. This assignment is done through the concept of a Membership that directly refers to a certain DomainRoleBinding. It indicates that the particular AgentInstance currently plays the DomainRole referred by the DomainRoleBinding. For specifying the binding between Actor and AgentInstance, the concept of ActorBinding is utilized.
* **Generalizations**
  - None
* **Associations**
  - agentType: Agent::Agent[1] the Agent type represented by the AgentInstance in the running system
* **Attributes**
  - Name: String[1]
* **Constraints**
  - None

- *The OrganizationInstance concept:*
  - **Definition** the concept of OrganizationInstance is introduced to specify the MAS's organization at run-time.
  - **Generalizations**
    - None
  - **Associations**
    - organizationType: Organization::Organization[1] represents the Organization type represented by the OrganizationInstance in the running system
    - members: Deployment::Membership[*] refers to the AgentInstances part of this OrganizationInstance of Type Organization.
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - None
- *The Membership concept:*
  - **Definition** The Membership concept of PIM4AGENTS defines the AgentInstances being member in other AgentInstances. However, only AgentInstances of type OrganizationAgent contain such Memberships. Each Membership encapsulates exactly one AgentInstance and additionally defines to which AgentInstance (of type OrganizationalAgent) it actually belongs to.
  - **Generalizations**
    - None
  - **Associations**
    - domainRoleBinding: DomainRoleBinding[*] relates to the kinds of DomainRoleBindings that establish the role bindings of this member
    - agentInstance: AgentInstance[1] denotes the AgentInstance, which is represented by this Membership
    - organizationInstance: OrganizationInstance[1] denotes the particular OrganizationInstance, which is owned the Membership.
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - None

  The concept of DomainRoleBinding and ActorBinding (similar to actorRoleBinding in our metamodel) are presented in details in Hahn, 2013.

- **The ArchitetcureStyleDescriptionPackage:** The ArchitetcureStyleDescriptionMetamodel, which is depicted in Fig 4.14, focuses on the concepts of *ArchitectureStyle*, *Layer*, *Node*, *Edge* and other specializations of the concept *ArchitectureStyle*. It is used for the representation and classification of MAS architecture Style.

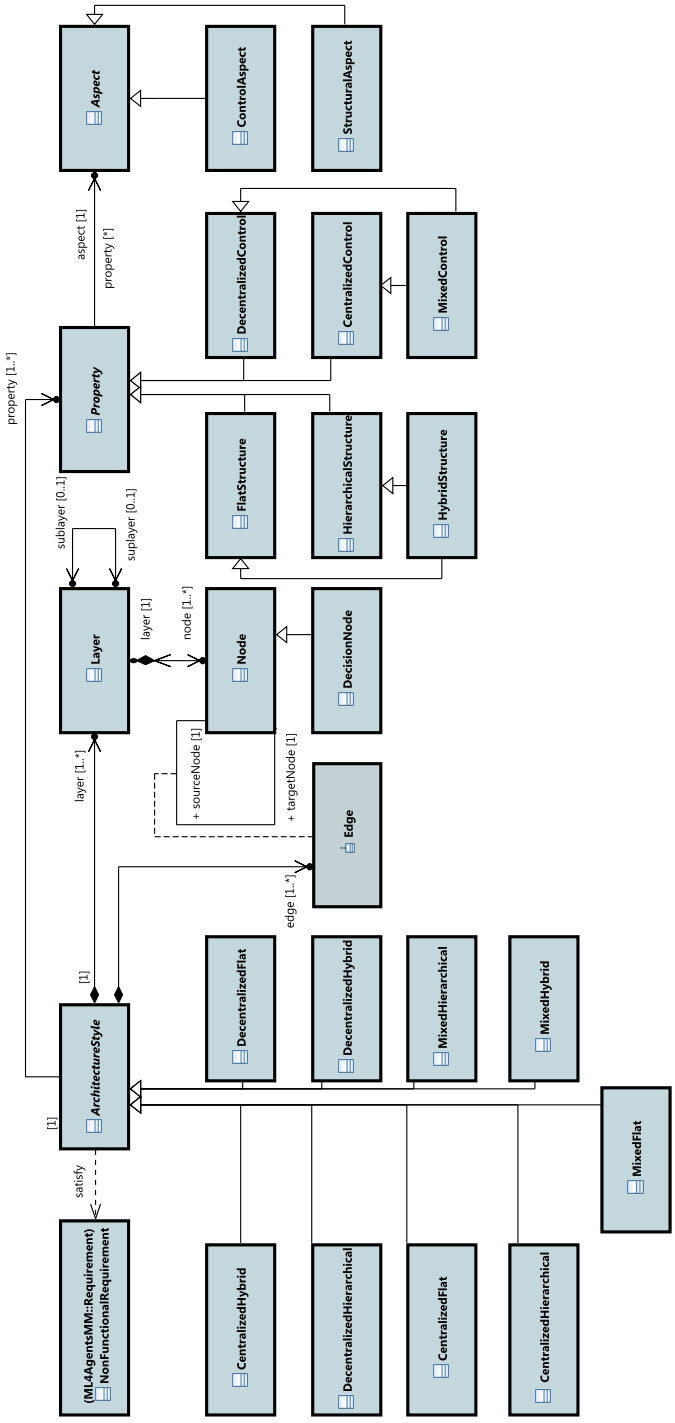

FIGURE 4.14: The ArchitetcureStyleDescription package

The ArchitetcureStyleDescriptionPackage concepts description:

- ***The ArchitectureStyle concept:***
  - **Definition** The architectural style concept consists of components, connectors, and constraints, and defines a family of systems with a specific pattern of structural composition.
  - **Generalizations**
    - None

    - **Associations**
        - property: Property[1..*] it is possible to characterize MAS architectural styles according to one or more properties
        - edge: Edge[1..*] defines all edges within an architecture (i.e., edges represent interaction links between the architecture's nodes)
        - layer: Layer[1..*] defines all layers within an architecture (i.e., layers represent a container of homogeneous nodes)
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - self.layer-> size >=1 the architecture style should include at least one layer
        - self.property->forall(p1,p2 : Property | p1 <> p2 implies p1.aspect <> p2.aspect) In case that the architectural style is characterized by more than one property, then all properties should have different aspect, i.e., an architecture cannot be flat and hierarchical at the same time where flat and hierarchical are properties related to the structure aspect.

- ***The Layer concept:***
    - **Definition** The Layer concept in a MAS architecture style includes homogeneous nodes, i.e., entities that behave similarly (e.g. interaction behavior) and have the same decision capabilities.
    - **Generalizations**
        - None
    - **Associations**
        - node:Node[1..*]
          each layer in a MAS architecture is composed by nodes that represent an entity in the system (i.e., it can be an agent or an organization);
        - sublayer: Layer[0..1]
        - suplayer: Layer[0..1]
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - self.suplayer->size()->size()>0 implies self<>self.suplayer
        - self.sublayer->size()->size()>0 implies self<>self.sublayer
        - self.suplayer<>self.sublayer
- ***The Node concept:***
    - **Definition** The Node concept in a MAS architecture style represents an interactive/autonomous entity in the MAS, i.e., the node can represent an agent or an organization, a specialization of node is presented (DecisionNode concept) to represent an entity that has a capability to take a control decision (i.e., the DecisionNode is a special node in the MAS architecture that represents an entity that take a control decision to solve problem within a MAS)
    - **Generalizations**
        - None

    - **Associations**
        - layer: Layer[1]
          each node belongs to only one layer
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - None

- ***The Edge concept:*** An edge in a MAS architecture represents an interaction between two entities (i.e., node), it is an association class that links two nodes
    - **Definition** The Edge in a MAS architecture style represents an interaction between two entities (i.e., node), it is an association class that links two nodes
    - **Generalizations**
        - None
    - **Associations**
        - targetNode: Node[1]
        - sourceNode: Node[1]
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - self.targetNode.layer.suplayer->includes(self.sourceNode.layer) or self.targetNode.layer.sublayer->includes(self.sourceNode.layer)
          Edge can link two nodes from the same layer or from different layers. However, the layers should be successive (i.e., the target node's layer should be a sup-layer or a sub-layer of the source node's layer)
        - self.targetNode<>self.sourceNode
          an edge cannot links a node with itself

- ***The CentralizedFlat concept:***
    - **Generalizations**
        - ArchitectureStyle
    - **Associations**
        - controlKind : CentralizedControl [1]
          this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
        - structureKind : FlatStructure[1]
          this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - **Attributes**
        - Name: String[1]
    - **Constraints**
        - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()=1
          only one decision node
        - self.layer->size()=1
          only one layer

- self.edge->forAll(e | e.sourceNode.oclIsTypeOf(DecisionNode) or e.targetNode.oclIsTypeOf(DecisionNode))
  all non-decision nodes communicate only with the unique decision node
-

– ***The CentralizedHierarchical concept:***

* **Generalizations**
  - ArchitectureStyle
* **Associations**
  - controlKind : CentralizedControl [1]
    this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
  - structureKind : HierarchicalStructure [1]
    this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
* **Attributes**
  - Name: String[1]
* **Constraints**
  - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()=1
    one decision node and exist on the top of the architecture
  - self.layer->select(l|l.suplayer=null and l.node->exists(dn | dn.oclIsTypeOf(DecisionNode)))->size()=1
    the decision node exists on the top of the architecture
  - self.layer->size()>=2
    architecture has at least two layer
  - self.layer.node->forAll (n | n.targetNode.mylayer->excludes(n.mylayer)) self.edge->forAll(e | e.sourceNode.layer<> e.targetNode.layer)
    in a hierarchical communication, each node communicate with a node from different layer (sublayer or suplayer)

– ***The CentralizedHybrid concept:***

* **Generalizations**
  - ArchitectureStyle
* **Associations**
  - controlKind : CentralizedControl[1]
    this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
  - structureKind : HybridStructure [1]
    this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
* **Attributes**
  - Name: String[1]
* **Constraints**
  - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()=1
    only one decision node that exists on the top of the architecture
  - self.layer->select(l|l.suplayer=null and l.node->exists (dn | dn.oclIsTypeOf(DecisionNode)))->size()=1
    the decision node exists on the top of the architecture

        - self.layer->size()>=2
          architecture has 2 or many layer
          and
        - self.edge->exists(e | e.sourceNode.layer<> e.targetNode.layer)
          exist inter-layer communication
        - self.edge->exists(e | e.sourceNode.layer=e.targetNode.layer)
          exist intra-layer communication

- ***The DecentralizedFlat concept:***
  - **Generalizations**
    - ArchitectureStyle
  - **Associations**
    - controlKind : DecentralizedControl [1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - structureKind : FlatStructure [1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
      there is two or more decision node in the architecture
    - self.layer->size()=1
      there is only one layer in the architecture no decision node (centralized) that communicate with all the rest of the nodes
- ***The DecentralizedHierarchical concept:***
  - **Generalizations**
    - ArchitectureStyle
  - **Associations**
    - controlKind : DecentralizedControl[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - structureKind : HierarchicalStructure[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
      Two or more decision node and exist on the top of the architecture
    - self.layer->select(l|l.suplayer=null and l.node-> exists
      (dn | dn.oclIsTypeOf(DecisionNode)))->size()=1
      the decision node exists on the top of the architecture
    - self.layer->size()>=2
      the architecture has two or many layer

        - self.edge->forAll(e | e.sourceNode.layer<> e.targetNode.layer)
          in a hierarchical communication, each node communicate with a node from different layer (sublayer or suplayer)

  - ***The DecentralizedHybrid concept:***
    - **Generalizations**
      - ArchitectureStyle
    - **Associations**
      - controlKind : DecentralizedControl[1]
        this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
      - structureKind : HybridStructure[1]
        this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - **Attributes**
      - Name: String[1]
    - **Constraints**
      - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
        Two or more decision node and exist on the top of the architecture
      - self.layer->select(l|l.suplayer=null and l.node->exists
        (dn | dn.oclIsTypeOf (DecisionNode)))->size()=1
        the decision node exists on the top of the architecture
      - self.layer->size()>=2
        architecture has 2 or many layer
      - self.edge->exists(e | e.sourceNode.layer<> e.targetNode.layer)
        exist communication inter-layer
      - self.edge->exists(e | e.sourceNode.layer= e.targetNode.layer)
        exist communication intra-layer
  - ***The MixedFlat concept:***
    - **Generalizations**
      - ArchitectureStyle
    - **Associations**
      - controlKind : MixedControl[1]
        this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
      - structureKind : FlatStructure[1]
        this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - **Attributes**
      - Name: String[1]
    - **Constraints**
      - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
        There are two or more decision node in the architecture
      - self.layer->size()=1
        There is only one layer in the architecture
      - self.edge->exists(e | e.sourceNode.oclIsTypeOf(DecisionNode) and
        e.targetNode.oclIsTypeOf(DecisionNode))
        exist at least one communication between two decision nodes

        - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->forAll(e | self.edge->select(ee | (ee.targetNode=e and not ee.sourceNode.oclIsTypeOf(DecisionNode))or (ee.sourceNode=e and not ee.targetNode.oclIsTypeOf(DecisionNode)))->size()=1)
          each decison node in the architecture communicate at least with a non decision node

- ***The MixedHierarchical concept:***
  - **Generalizations**
    - ArchitectureStyle
  - **Associations**
    - controlKind : MixedControl[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - structureKind : HierarchicalStructure[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
  - **Attributes**
    - Name: String[1]
  - **Constraints**
    - self.layer->size()>=2
      architecture has 2 or many layer
    - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
      two or more decision node exists on the top and on another layer
    - self.layer->exists(l|l.suplayer=null and l.node->exists(dn | dn.oclIsTypeOf(DecisionNode)))
      decision nodes exist on the top and on another layer
    - self.edge->forAll(e | e.sourceNode.layer<> e.targetNode.layer)
      hierarchical communication each node communicate with a node from different layer (sub or sup layer)
    - self.edge->exists(e | e.sourceNode.oclIsTypeOf(DecisionNode) and e.targetNode.oclIsTypeOf(DecisionNode))
      exist at least one communication between two decision nodes
    - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->forAll(e | self.edge->select(ee | (ee.targetNode=e and not ee.sourceNode.oclIsTypeOf(DecisionNode))or (ee.sourceNode=e and not ee.targetNode.oclIsTypeOf(DecisionNode)))->size()=1)
      each decision node in the architecture communicates at least with a non decision node

- ***The MixedHybrid concept:***
  - **Generalizations**
    - ArchitectureStyle
  - **Associations**
    - controlKind : MixedControl[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*
    - structureKind : HybridStructure[1]
      this attribute subsets the inherited attribute *property* of the super class *ArchitecturalStyle*

* **Attributes**
  - Name: String[1]
* **Constraints**
  - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
    Two or more decision node and exist on the top of the architecture
  - self.layer->select(l | l.suplayer=null and l.node->exists(dn | dn.oclIsTypeOf(DecisionNode)))->size()=1
    the decision node exists on the top of the architecture
  - self.layer->size()>=2
    architecture has 2 or many layer
  - self.edge->exists(e | e.sourceNode.layer<> e.targetNode.layer)
    exist communication inter-layer
  - self.edge->exists(e | e.sourceNode.layer= e.targetNode.layer)
    exist communication intra-layer
  - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->size()>=2
    two or more decision node exists on the top and on another layer
  - self.layer->exists(l | l.suplayer=null and l.node->exists(dn | dn.oclIsTypeOf(DecisionNode))) and self.layer->exists(l | l.suplayer<>null and l.node->exists(dn | dn.oclIsTypeOf(DecisionNode)))
    decision nodes exist on the top and on another layer
  - self.edge->exists(e | e.sourceNode.oclIsTypeOf(DecisionNode) and e.targetNode.oclIsTypeOf(DecisionNode))
    exist at least one communication between two decision nodes
  - self.layer.node->select(dn | dn.oclIsTypeOf(DecisionNode))->forAll(e | self.edge->select(ee | (ee.targetNode=e and not ee.sourceNode.oclIsTypeOf(DecisionNode))or (ee.sourceNode=e and not ee.targetNode.oclIsTypeOf(DecisionNode)))->size()=1)
    each decision node in the architecture communicate at least with a non decision node

Table 4.1 lists the ML4Agents packages used for the description of each viewpoint of the architecture framework. Their associated diagrams are detailed in section 6.2.3.

MAS4SG's Viewpoint	**ML4Agents packages**
Business viewpoint	ML4Agent::Requirement
Analysis viewpoint	ML4Agent::ArchitetcureStyleDescription ML4Agent::Interaction ML4Agent::Environment
Design viewpoint	ML4Agent::Role ML4Agent::Organization ML4Agent::Agent ML4Agent::Behavior
Deployment viewpoint	ML4Agent::Deployment
Implementation viewpoint	

TABLE 4.1: The used ML4Agents packages for each MAS4SG's viewpoint.

4.4 Conclusion

In this Chapter we introduced a MAS architectural framework to help power engineers to use the MAS technology in the power engineering field in order to develop agent-based software to solve problems of Smart Grids. We proposed the ML4Agents metamodel as a mean of description of the architectural framework viewpoint. This language conforms to power engineering requirements such as:

- Interoperability between agents: this is achieved by the integration of new concepts to model the ontology elements
- Interoperability between agent systems: this is achieved by the integration of FIPA specifications in the metamodel (FIPA interaction protocols, ACLMessages, Services, ...)
- Requirements specification: this is achieved by proposing a Requirement package including new concepts to model system requirement in the earlier phase.
- Architecture Style Evaluation: this is achieved by proposing an architecture style description package including concepts to describe MAS architecture styles and help in their evaluation.

This chapter focuses on the ML4Agents metamodel's concepts that define the vocabulary to be used by a model driven development process. The latter is presented in the next chapter.

Chapter 5

MAS4SG methodology

5.1 Introduction

This Chapter presents a model driven methodology to support the use of the proposed architecture Framework that is presented in the previous chapter 4.

A methodology is a collection of methods covering and connecting different phases in a process. The purpose of a methodology is to define a certain coherent approach to solve a problem in the context of a software process by preselecting and putting in relation a number of methods.

In this chapter we suggest a clean and disciplined approach to support the use of the MAS4SG framework to analyze, design and develop MASs for energy systems, using specific methods and techniques. In this dissertation, the term methodology denotes a set or collection of methods and related artifacts needed to support the use of the MAS4SG framework in the model driven engineering of MASs. The methodology has two important parts, (i) one part that describes the process steps of the approach, and (ii) one part that focuses on the work products and their documentation.

5.2 Methodology's prerequisites

A holistic methodology for smart energy systems engineering involves many disciplines. First, the methodology needs to be able to address the particular needs of the involved stake- holders such as Business Analysts, Domain Experts and Agent Engineers. Thus, a process is required since it gives guidelines and covers the Requirement, Analysis, Design, Deployment and Implementation phase.
Best practice solutions such as existing use cases or a certain Reference Architecture can be described. Such methodology allows reusing existing work (or part of them) in the Smart Grid domain and modeling MAS solutions for ES problems rather than starting from scratch. For example, the system requirements could be derived from the existing functional requirements of the application development and broadly agreed use cases can be reused. Furthermore, key roles, interactions and agent's behaviors could be designed following the Universal Smart Energy Framework [1] (USEF). In addition, the ontology design is based on the Common Information Model (CIM) standard; and the knowledge design is based on the existing energetic data standards identified within the SGAM's information layer.

Moreover, the development process should include the requirement modeling task. The availability of a requirement model should serve as a basis for the architectural style evaluation. Thus, it is recommended to include also the capability of

[1] "Universal Smart Energy Framework", Available [Online]: https://www.usef.energy

the architecture style evaluation activity. The development process should include the ontology modeling task.

And finally, for an interoperability need between the developed agent systems, (1) the design of ontology should be based on the existing energy standard IEC CIM, and (2) the design of interaction should reuse the specification of FIPA protocols.

5.3 Methodology's overview

A model driven methodology has to be proposed to support the use of the presented architectural framework (c.f. Figure 4.3). The MAS4SG framework relates to such methodology since the deliverables of its viewpoints are expressed as models corresponding to several levels of abstraction. This section presents a methodology

FIGURE 5.1: MAS4SG methodology overview

(c.f Figure 5.1) to support the use of the proposed MAS4SG architectural Framework including the five phases: (1) the requirement specification phase, (2) the analysis phase (3) the design phase, (4) the deployment phase and (5) the implementation phase. To formally define the MAS4SG methodology, the Software Systems Process Engineering Metamodel (SPEM) (OMG and Notation, 2008) has been used. SPEM 2.0 (OMG specification 2008) defines concepts of a process (process, phase, iteration, activity, task descriptor, ...) that can be used to construct models to describe software engineering process.

The proposed process starts with the requirement specification phase, which results in the artifact requirement model. The requirement specification phase typically involves Domain Experts.

The *analysis phase* typically involves agent system architects and system analysts (i.e., interaction, environment and domain modelers) and focuses on the interactions involved in the system and environment's resources (such as objects and ontology).

In the *design phase*, the agent engineers (such as role modelers, organization modelers, ...) continue with the architectural specification that is composed by two activities: structural MAS design and behavioral MAS design. The role, organization and agents are designed in the MAS structural design activity and the behaviors and collaborations are designed and refined in the MAS behavioral design activity. In the *deployment phase*, if instances of the running agents are known, then the deployment pattern should be designed before starting the final phase of the process which is the *implementation phase*, or we can go directly to the implementation phase and do the deployment on one of the execution platforms. In the implementation phase the programmer may execute the model transformations to an agent platform specific model and the model to text transformation to generate the code. The generated code can finally be refined within the underlying agent programming language.

SPEM uses phases to organise the various activities of a development method. We define activities and tasks included in each phase (see overview in Table 5.1).

In the following we will present each phase of the MAS4SG's methodology.

Phases	*Activities*	*Tasks*	*Input Work products created/modified*	*Output Work products created/modified*	*Responsible method-roles*
Requirement Phase		Requirement Specification	-Existing Set of Domain's Quality Attribute (0..1) -Existing Use Case Diagram (0..1)	Requirement Model	Domain Expert
Analysis Phase	***Architecture Style Exploration Activity***	Evaluating Architecture Style	-Architecture Style Library (0..1) -Existing Set of Domain's Quality Attribute(0..1) -Requirement Model	-Qualitative and Quantitative Evaluation Results	Domain Expert
		Selecting Architecture Style	Qualitative and Quantitative Evaluation Results	Architecture Style Model	Agent System Architect
		Selecting Interaction	-Architecture Style Model (0..1) -FIPA Protocol Library	Interaction Model(*)	Interaction modeler
		Modeling Domain	IEC CIM Profiles Library	Domain Model	Domain Expert
		Modeling Environment	IEC CIM Profiles Library (0..1)	Environment Model	Environment modeler
Design phase	***MAS Structural Design Activity***	Modeling Role		Role Model	Role modeler
		Modeling MAS	Environment Model (0..1) Organization Model (0..1) Agent Model Behavior Model Role Model (0..1)	MAS Model	MAS modeler
		Modeling Agent	Role Model (0..1) Environment Model (0..1)	Agent Model	Agent modeler
		Modeling Organization	Role Model	Organization Model	Organization modeler
	MAS Behavioral Design Activity	Modeling Collaboration	-Deployment Model -Interaction Model -Organization Model	Organization Model (Refined)	Organization modeler
		Modeling Behavior	Agent Model	Behavior Model	Plan modeler
Deployment phase		Modeling Instances	-Agent Model -Organization Model (0..1)	Deployment Model	Deployment modeler
Implementation phase		Generating Code	-Independent Model (PIM) -Platform Model	-Platform Specific Implementation (PSI) -Platform Specific Model (PSM)	Programmer

TABLE 5.1: MAS4SG's methodology fragments

5.4 The Requirement Specification Phase

The process of the Requirement Specification phase is depicted in Fig. 7.4

FIGURE 5.2: The process of the Requirement Specification phase

The Requirement Specification Phase aims at defining the system functional requirements for a particular system. Thus, it yields a Requirement Model which is designed with concepts from the ML4AgentsMM::Requirement package 4.3.4.2. To derive the desired functional requirement, broadly agreed use cases stories can be investigated to clarify the motivation for an application. In addition, during this phase, eventually non-functional requirements can also be considered. The requirement specification phase typically involves the domain expert and the requirement engineer.

The detailed process of the Requirement Specification phase is depicted in Fig. 5.3. It shows the mandatory input work products and the output work products. Additionally, it shows the performer role "Domain Expert" which is responsible for the requirement specification task.

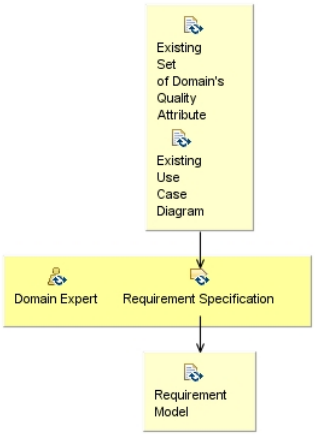

FIGURE 5.3: The detailed process of the Requirement Specification phase

5.5 The Analysis Phase

Subsequent to the Requirement Specification Phase, the Analysis phase includes one activity and three tasks. It starts with the architectural style exploration activity for the selection of the most appropriate MAS architectural style, then its process continues with the environment and domain modeling and the selecting interaction task. The process of the analysis phase is depicted in Fig. 7.6

Many MAS architectures were proposed in the literature for energy management. We noticed that some of them focus on control strategies (centralized, decentralized, hybrid, fully distributed, ...), whereas others focus on communication

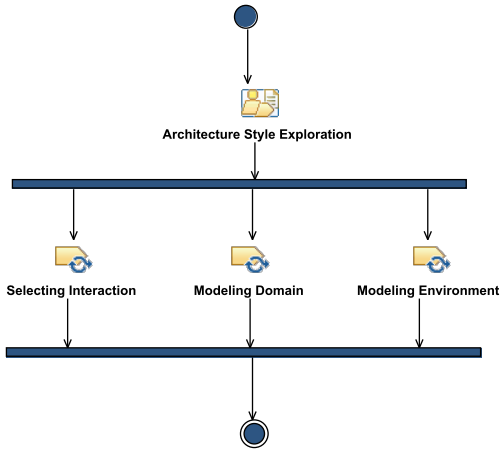

FIGURE 5.4: The process of the Analysis phase

infrastructures (centralized, hierarchical, hybrid, ...). In order to select the appropriate MAS architecture for energy management, two big questions came into our mind that are the following:

1- While building a MAS architecture, on which concept shall we focus, on agent communication or on control strategy?

2- According to many contradictory objectives, how to find a trade-off to define the appropriate MAS architecture?

Architecture style selection is an essential step in designing distributed software systems because finding an architecture that is able to satisfy non-functional requirements is an important issue. In order to cope with these problems, we propose to add the architectural exploration stage on the top of the methodology. We were inspired by an existing generic evaluation method (Wang and Yang, 2012) to help in the selection of the appropriate architecture style. This method is proposed to be used for any application domain. Software architecture selection is a multi-criteria decision-making problem in which different goals and objectives must be taken into consideration (Moaven et al., 2008).

Architecture is critical to the realization of many qualities of interest in a system, and these qualities should be evaluated at the architectural level. However an architecture, by itself, is unable to achieve qualities. It provides the foundation for achieving quality, but this foundation will be of no use if attention is not paid to the details (Felix Bachmann, 2003). Achieving quality attributes must be considered throughout design, implementation, and deployment. In this dissertation we take into account quality attributes starting from the analysis phase in order to guide design decisions and eliminate as early as possible design solutions that will be unable to fulfill the requirements. The objective is to shorten the development life-cycle and avoid late detection of design errors that can come after the implementation phase in the traditional engineering processes.

The process of the Architecture Style Exploration Activity, which is depicted in Fig. 5.5, contains two tasks: the *Evaluating Architecture Style* and the *Selecting Architecture Style* tasks. The process of the architectural exploration activity starts with the evaluation of the set of possible architecture styles that can be used to design the MAS architecture for a given MAS application domain. Once the analyst gets the evaluation result, he can select the most appropriate architecture style.

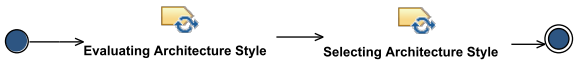

FIGURE 5.5: The process of the Architecture Style Exploration Activity

The *Evaluating Architecture Style* task takes as input the possible MAS architectural styles and evaluates them according to the non functional requirement specified in the requirement specification phase. We have built a library with many architectural styles broadly used in the energy system domain. In the sequel, we explain how we built this library and we give a guideline to extend the library with further MAS architectural styles.

How to characterize an architecture ?

There are many ways for characterizing the space of possible MAS architectures. We can classify the MAS architectural styles according to one or more aspect such as the control schema and the communication infrastructure aspect. The **structural aspect** that includes the communication and organization (hierarchical organization) characteristics of MAS such as the topology of the system, the degree of mobility and dynamics of the communications, and the degree of synchronization of interaction. The **behavioral aspect** or autonomy which considers the decision making and the degree of distribution of control. A number of different functional architectures have been proposed in literature to build multi-agent systems, but there is not yet a consensus on which one of those approaches is the most suitable for smart grid applications. The architectural model most commonly adopted in power systems is the multi-layered architecture. A three-layer architecture is proposed for managing the distributed energy resources. In this model, the bottom layer consists of the agents managing the physical resources (e.g., energy generators and power storage systems). The middle layer includes agents that provide high-level management services (e.g., fault diagnosis, protection and restoration, optimization of power parameters, ...) to the agents connected to the physical resources. Finally, the top layer contains the agents handling the user interfaces.

The combination of the MAS architectural style properties (properties related to different aspects) build the MAS architectural style library that we can use to select which architecture style could be a candidate for the specific domain application. Beside the MAS architectural style library the process reuse the non-functional requirements related to the target application domain defined in the output work product of the requirement specification phase.

Once this is done, then the first task in the process that is the evaluation of architecture style can be held by evaluating all the candidates architectural styles according to one or more quality attribute.

Which style is preferred for the MAS solution? To answer the second question, attention must be payed in understanding the domain requirements in order to help the system designer in making good decisions on which MAS architectural style should be preferred. Thus, guarantee that the resulting solution is adequate to system's requirements and deployable in the execution environment. Careful considerations should be given to the types of communication each agent can engage in. When designing to meet any requirements, it is important to consider the impact on other attributes and find trade-off between requirements. It is important to note that our aim is to help the system analyst in the selection of a MAS architectural style for which it exists an implementation that will be able to fulfil the system's

requirements. The refinement towards such an implementation will require additional analyses at the lower levels for the assessment of requirements fulfillment. When developing a software system, the potential of the chosen architecture is one important influence of the resulting system, but there are others. For example, familiarity with a particular architectural style, development organization, and coding standards may also influence the result. The output work product of this task is the qualitative and quantitative architecture styles evaluation results. This is the mandatory input for the next task which is the Selecting Architecture Style task that aims to select the appropriate architecture style between the possible candidates architectures based on the evaluation results. To select the appropriate architectural style, two evaluations should be handled.

- **Qualitative Evaluation:** For each criteria, evaluate the candidate architecture style; this can be done through the Analytic Hierarchy Process (AHP) (**saaty1980analytic**) or on subjective judgments. The subjective judgments can be based on experiences and previous evaluation.

- **Quantitative evaluation:** Typically, an architecture constitutes a balance between different quality attributes, just as different applications may require a specific balance or trade-off between quality attributes.

  To do that we were inspired by the method proposed in (Davidsson, Johansson, and Svahnberg, 2005), which is based on the use of the Analytic Hierarchy Process (AHP) that is a multi-criteria decision support method from Management Science.

  We quantify the trade-off between quality attributes (criteria) to select the appropriate MAS architecture that constitutes a balance between different quality attributes; this can be done by the following steps:

  - Prioritize the criteria in accordance with how they are important for a specific application using the pair-wise comparison technique.
  - We quantify the subjective assessments with normalized value.
  - Evaluation of the candidate architectural styles for a given specific balance between quality attribute for the target application. And then select the most suitable architecture style.

Each architectural style is represented in the MAS architectural style library by an architectural style model. Concepts from the ML4AgentsMM::ArchitecturalExploration package 4.3.4.2 are used to design such model. The architectural style exploration activity is the design activity starting point .

The detailed process of the Architecture Style Exploration Activity is depicted in Fig. 5.6. It shows the mandatory input and output work products. Additionally, it shows the performer role "Agent system Architect", which is responsible for the evaluating and selecting architectural style tasks.

The detailed process of the analysis phase is depicted in Fig. 5.7. It shows the mandatory input and output work products. Additionally, it shows the performer roles: "Interaction modeler", "Domain expert" and "Environment modeler", which are responsible for the selecting interaction, modeling domain and modeling environment tasks respectively.

Besides the architecture style selection, the Analysis Phase includes the domain and environment modeling which is based on existing CIM profiles model. These profiles are generated by an energy domain expert that restricts the CIM data model

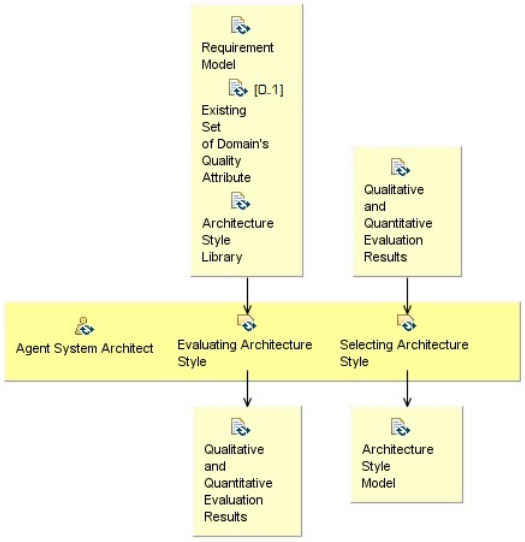

FIGURE 5.6: The detailed process of the Architecture Style Exploration Activity

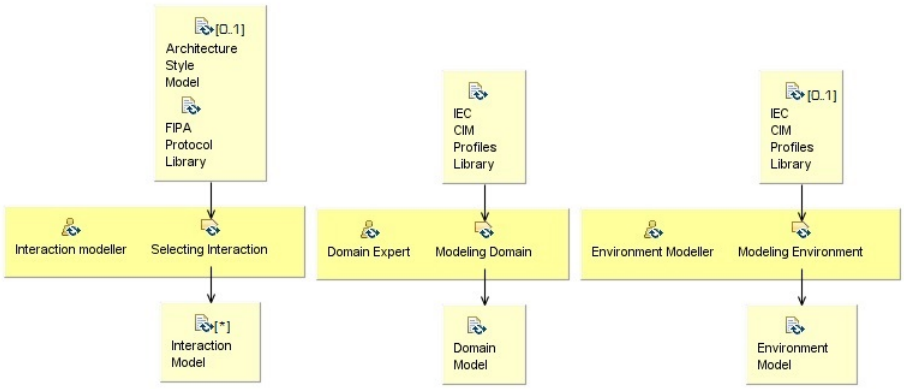

FIGURE 5.7: The detailed process of the Analysis phase

to produce a data model specific for the application domain problem. This data model includes the needed object to build the agents environment and to construct the ontology used by agents to communicate. The domain expert builds the IEC CIM Profiles Library that contains a CIM profile for each application domain of energy systems. The CIM profiles are technology-independent.

CIM Restriction: The International Electrotechnical Commission (IEC) CIM standard is used in the electricity domain, covering transmission, distribution, markets, generation, and related business processes. The key idea of the CIM is to define a common language in order to allow both: exchanging data between different companies, and exchanging data between company applications.

The core packages of CIM are defined in the IEC standard 61970-301, which defines the components in the power system using the Unified Modeling Language (UML). Generally, only a part of the CIM (so-called CIM profile) is used for modeling a given energy system solution. CIM profiles allow defining a subset of the CIM, including only the classes and associations required for modeling a specific domain. The CIM can be classified in different profiles known as domain profile.

This modularized structure will ease the CIM profiles reuse in different energy domain's scenarios and application.

The MAS environment in energy systems contains the agents, the objects they manage or interact with, and other resources. The controlled objects can be the electrical equipment that compose the ES. During this phase, the developer uses the IEC CIM standard to model the agent's knowledge (data model) on the ES components and how they are physically connected.

This step produces a CIM profile for a specific application in the energy system domain. It will then be used in the next phase, and can be reused whenever the components of the ES being modeled are already represented in the defined CIM profile. In addition, interoperability must be ensured among agents from different platforms at communication semantic level, thus an upper CIM ontology is recommended (i.e., based on the IEC 61968/61970 Common Information Model (CIM) standard (Uslar et al., 2012)). Thus, the second utility of the CIM restriction stage is to provide a basis to design the agent's ontology. However, the IEC CIM data model is still ongoing to be more complete to include concepts like prices. In the annex we explain in details the different steps to create a CIM profile and then how to use the CIM concept from that profile to model a real energy system example (the WSCC 9 bus system).

This aims to produce IEC CIM profiles models to be reused in the analysis phase to model the environment object and the domain entities building the agent ontology (i.e., Domain Model). The resulted CIM profile is imported into the application model in order to (1) model the relevant concepts of a given energy system, and the agent-object relation in order to model the agent's knowledge; (2) to model the ontology used by the agents to carry out a comprehensible conversation. This ontology shall allow the interoperability between the MAS agents in power engineering applications.

In the environment modeling task, the environment modeler will reuse the CIM profile library to model the environment with CIM objects. The environment model is designed with concepts from the ML4AgentsMM::Environment package 4.3.4.2. To model the agent ontology, the domain expert will adapt the CIM profiles model to model the ontology. Additionally, an ontology distinctly separates concepts from predicates, whereas the CIM data model has to contain all relationships within concept definitions. In CIM, the Equipment definition contains an attribute called EquipmentContainer, which indicates which container provide the root class for containing that equipment. However, in an agent messaging situation, this is more correctly modeled by a predicate called contain. The actions that agents could handle on a concept should be also designed. The domain expert should keep the concept names and definition as it is described by CIM but he can derive predicates from the concept definitions and add actions needed for a given MAS application. The idea is to build a CIM upper ontology including ontology for each specific applications (e.g., protection application, control application, ...). The ontology would ensure that different multi-agent systems would employ the same basic representation for common concepts. The ontology model is designed with concepts from the ML4AgentsMM::Environment package 4.3.4.2.

The analysis phase including also the selecting interaction task performed by an interaction modeler. In this task, the interaction modeler uses the FIPA protocol library, which is created by a protocol expert, to reuse the protocols specification model in order to select the interactions inside the MAS application. The specification of FIPA protocol uses the concept from the ML4AgentsMM::Interaction package 4.3.4.2. The section 6.3 shows how the FIPA protocol library is created and explains the way to reuse of the FIPA protocol models to model interactions in any

MAS application (i.e., this has been explained with the example of the Contrat Net Protocol).

5.6 The Design Phase

The process of the Design phase is depicted in Fig. 7.11. Subsequent to the Analysis

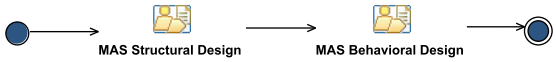

FIGURE 5.8: The process of the Design phase

Phase, the Design Phase aims at the design of the system architecture. We propose a Design Guidance Process of MAS architecture design, based on the Model-Driven Engineering (MDE) technique. The design approach uses as input the selected MAS architecture style produced by the architecture style exploration activity.

The produced model of the MAS architecture design phase is intended to be platform independent, which abstracts developers from existing agent-oriented platforms. To create the platform independent application model, the designer starts by designing functionalities of a role within a system. Generally the functionalities of system depicted by the functionalities of roles within the system should be specified and analyzed in the requirement specification phase in order to understand the behaviors of roles and guide the software design that implements the roles' functionalities. In contrast to the tasks, actions and plans of roles. The role model is designed with concepts from the ML4AgentsMM::Role package 4.3.4.2.

The designer continues with the MAS modeling task to produce the MAS model that gives an overview of the core element included in the agent system. The MAS model is designed with concepts from the ML4AgentsMM::MultiAgnetSystem package 4.3.4.2.

So after designing the agent's roles and the multiagent system, the types of the agents and the possible organizations can be designed. The organization structure depicts the selected architecture style structure. The agent model is designed with concepts from the ML4AgentsMM::Agent package 4.3.4.2 The Organization model is designed with concepts from the ML4AgentsMM::Organization package 4.3.4.2

The process of the MAS structural Design activity is depicted in Fig. 5.9

The detailed process of the MAS structural Design activity is depicted in Fig. 5.10. It shows the mandatory input and output work products. Additionally, it shows the performer roles: "MAS modeler", "Agent modeler", "Role modeler" and "Organization modeler", which are responsible for the modeling MAS, modeling agent, modeling role and modeling organization tasks respectively.

Then, the designer has to define the agent's behavior to describe how agents in the system coexist and in more details the internal architecture of the agent itself. The behavior model is designed with concepts from the ML4AgentsMM::Behavior package 4.3.4.2. The final task of the design phase, is the modeling collaboration task that specifies the collaborations between roles inside an organization. The collaboration model is designed with concepts from the ML4AgentsMM::Collaboration package 4.3.4.2.

The process of the behavioral Design activity is depicted in Fig. 5.11.

The detailed process of the behavioral Design activity is depicted in Fig. 5.12. It shows the mandatory input and output work products. Additionally, it shows the

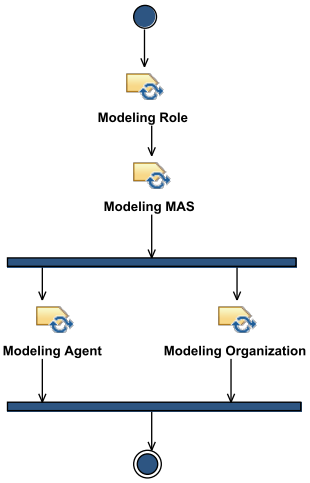

FIGURE 5.9: The process of the MAS structural Design activity

performer roles "Collaboration modeler" and "Plan modeler", which are responsible for the modeling collaboration and modeling behavior tasks respectively.

5.7 The Deployment Phase

This phase considers how agents are constructed in the runtime, this is done if the number of the agents instances is known in advance. It delivers the deployed application model which is designed with concepts from the ML4AgentsMM:: Deployment package 4.3.4.2. The process of the Deployment phase is depicted in Fig. 7.19 and the detailed process of the Deployment phase is depicted in Fig. 5.14. It shows the mandatory input and output work products. Additionally, it shows the performer role "Deployment modeler", which responsible for the modeling instances task.

5.8 The Implementation Phase

The process of the Implementation phase is depicted in Fig. 7.21 When the design is complete, as final phase, the programmer may execute the model transformations (model to model) to a platform specific model to be executed on a specific agent platform such as JACK or JADE. Then he can execute the second model transformation (model to code), this generates code, which can finally be refined within the underlying agent programming language.

5.8.1 Concrete Modeling

This stage allows the mapping of a PIM to one or more PSMs. The resulting application model from the previous phase is independent from the specific agent platform where the solution will be deployed. This stage aims to map the concepts of the application model to the concepts of the specific agent platform model such as JADE, JACK, ... In this phase, the designer can apply the mapping rules on the application model. The mapping rules define the transformations that shall be carried out to map the concepts of the ML4AGENTS metamodel to the concepts of the selected Agent Platform MetaModel (APMM). These rules are specified only one time and

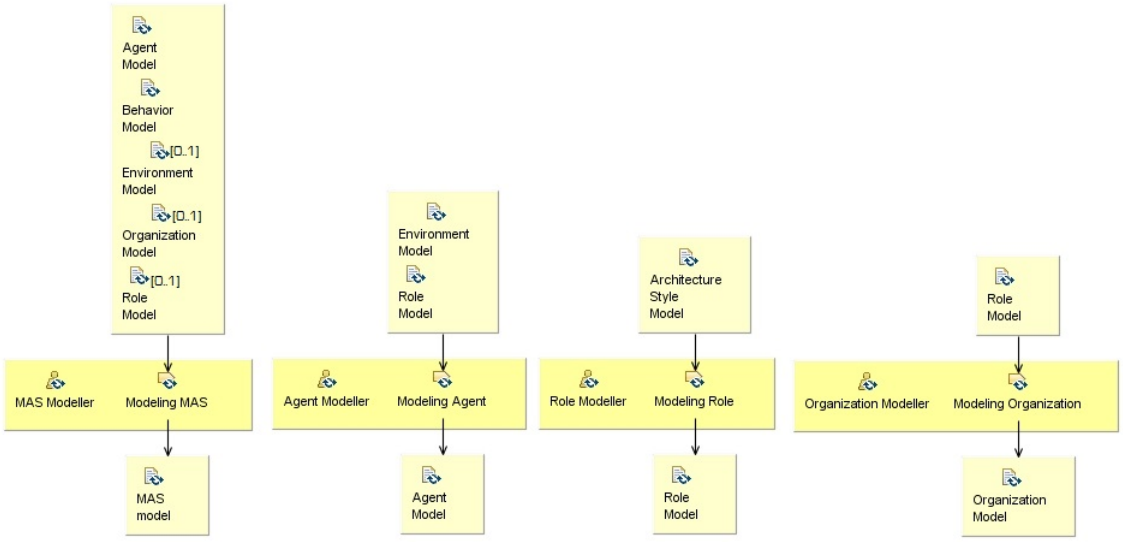

FIGURE 5.10: The detailed process of the MAS structural Design activity

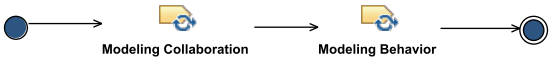

FIGURE 5.11: The process of the MAS behavioral Design activity

can be reused. The concepts derived from the CIM profile are instances of the object concept in ML4Agents, they will be implemented directly in a programming language (such as Java and C++). This phase is considered as a concretization since it transforms a given model into another one of a lower level of abstraction, i.e., starting from an application model that is platform independent model (PIM) to generate a MAS4SG (PSM) that is specific to the agent platform where the application will be executed. Below we present the most relevant mappings rules to transform the ML4Agents metamodel's concepts to the JADE metamodel's concepts:

- ML4Agents: Agent ! JadeMM: Agent
- ML4Agents: Organization ! JadeMM: Agent
- ML4Agents: ACLMessage ! JadeMM: ACLMessage
- ML4Agents: Behaviour ! JadeMM: Behaviour
- ML4Agents: resource ! JadeMM: ConceptSchema
- ML4Agents: Object ! JAVA.lang.object

5.8.2 Code generation

The second stage of the implementation phase is the generation of the code from the PSM model generated by the previous stage. For this step, we propose the use of the Java code generator within the papyrus UML tool, which generates the structural part of classes i.e., it generates the classes with their attributes and the function

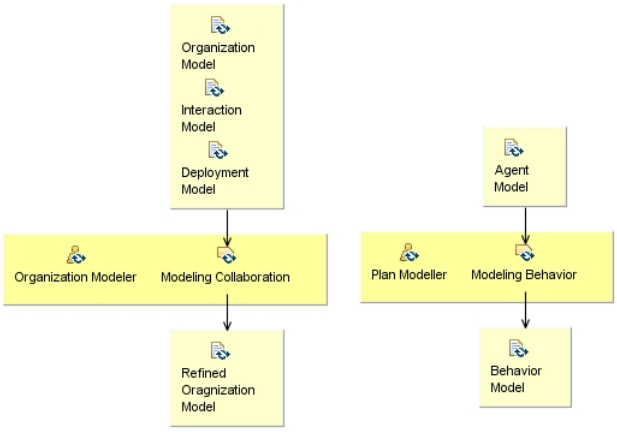

FIGURE 5.12: The detailed process of the MAS behavioral Design activity

FIGURE 5.13: The process of the Deployment phase

headers, and implements the connection between the classes. Furthermore, it generate the Java code from UML state machine. The next version of the Java code generator will take into account the behavioral part of the classes. Now, it is up to the user to implement the behavior code of the agents, depending on the MAS being developed.

The detailed process of the Implementation phase is depicted in Fig. 5.16. It shows the mandatory input and output work products. Additionally, it shows the performer role "Programmer", which is responsible for the generating code task.

5.9 Conclusion

In this chapter, we presented a methodology to support the use of the MAS4SG framework to analyze, design and implement MASs for energy system. It follows the MDE process in order to guide the power engineers in the design and implementation of MAS applications, and to simplify the design task by reusing models. The methodology adheres to energy standards, and is divided into five phases ,namely: requirement, analysis, design, deployment and implementation phases. It gives a capability to evaluate and choose the most appropriate architectural style for the developed agent system in order to satisfy its non-functional requirement and contributes in the improvement of the system performance in terms of quality of service.

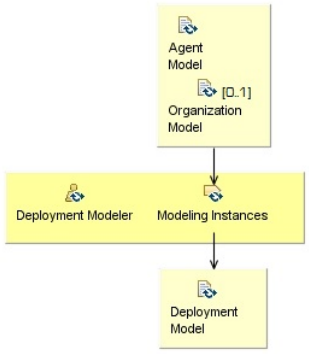

FIGURE 5.14: The detailed process of the Deployment phase

FIGURE 5.15: The process of the Implementation phase

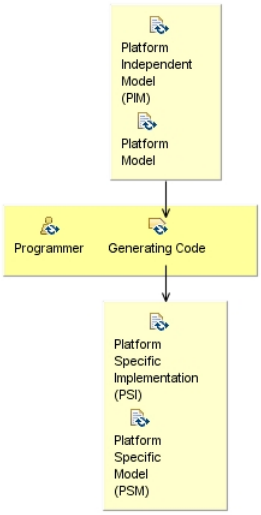

FIGURE 5.16: The EPF detailed process of the Implementation phase

Part III

Validation

Chapter 6

Implementation

6.1 Introduction

This chapter presents the implementation of the MAS4SG framework implemented in papyrus (Gérard et al., 2010). The previous chapter presented the abstract syntax defined by the ML4Agents metamodel and its semantics. The implementation of this modeling language in a UML tool requires the definition of a UML Profile. This chapter is organized as follows: Section 6.2 explains how the ML4Agents metamodel is implemented. Section 6.3 presents the implementation of the FIPA protocol library. Afterwards, the section 6.4 presents the implementation of the IEC CIM Profiles Library. Finally, Section 6.5 concludes this chapter.

6.2 DSL ML4Agents

This section offers a glimpse of "the main concepts of ML4Agents" language and gives few details about the syntax and semantics proposed for the ML4Agents specification. Furthermore, a specification of the ML4Agents constraints is presented in this section.

6.2.1 UML Profile based DSL

When we need to define a new domain-specific language that either restricts the number of UML concepts (metaclasses) or adds some constraints or syntactic sugar to them, while respecting the original semantics, we do not need to create a new language from scratch using the Meta-Object Facility (MOF) language. Instead, UML can easily be customized with the profile extension mechanism (Fuentes-Fernández and Vallecillo-Moreno, 2004). This mechanism is defined inside of the UML Infrastructure and used to adapt the existing metamodels to specific platforms, domains, business objects, or software process modeling. This extension mechanism is a part of the UML standard and thus it is supported (i.e., implemented) by UML tools. A UML profile is represented as a UML package that is tagged «profile». It has three main constructs for the definition of the required extensions: stereotypes, properties (formerly tagged values in previous versions of UML) and OCL constraints.

A UML profile that implements the ML4AgentMM::ArchitecturalExploration package, is presented in Figure 6.1. This UML profile has the Node stereotype that extends the UML Class metaclass with controlDecision property represented by the property controlDecision

In the following, we present an excerpt of the UML profile that implements the ML4Agents metamodel. The UML extension for agent modeling is presented in the Table A.1.

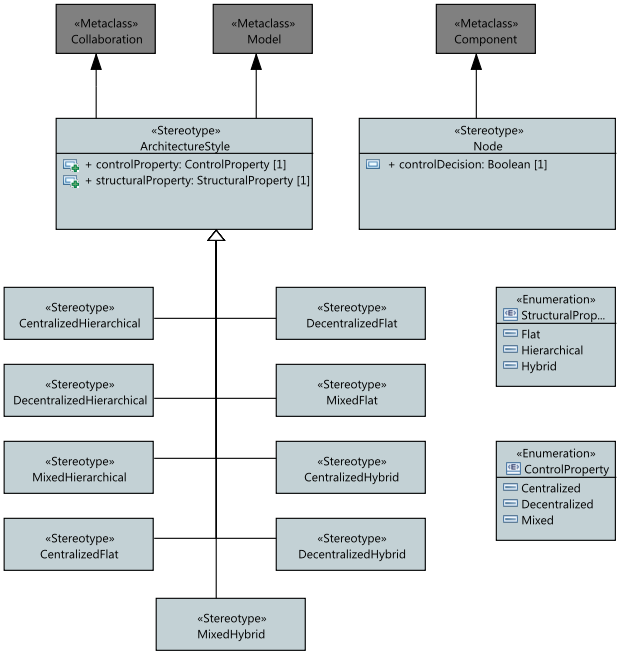

FIGURE 6.1: Architectural Exploration UML profile

6.2.2 Implementation of the ML4Agents constraints using JAVA

Like any other language, the ML4Agents language defines its own syntax and semantics. ML4Agents defines its new syntax and semantics using a profile, which extends a subset of the UML concepts. The profile defines new elements that extend some UML elements and constraints these elements with new constraints. To be consistent with the ML4Agents profile, a ML4Agents model must conform to the syntax and semantics defined by the ML4Agents specification. Figure 6.2 shows the java implementation of the one decision node constraint that check that there is only one decision node in any centralized control architecture style.

```
1  package org.eclipse.papyrus.architecturestyle.validation.CentralizedHierarchicalcategory;
2  import org.eclipse.core.runtime.IStatus;
12 public class OneDecisionNode extends AbstractModelConstraint {
13     public OneDecisionNode() {}
14     @Override
15     public IStatus validate(IValidationContext ctx) {
16         Model subsystem = (Model) ctx.getTarget();
17         System.err.println(subsystem.getName());
18         System.err.println(subsystem.getOwnedMembers());
19         int nbDecisionNode=0;
20          //TODO Auto-generated method stub
21         if (ConstraintsUtil.verifyArchitectureStyleProfileModelApplied(subsystem) )
22         {
23             for (NamedElement elt : subsystem.getOwnedMembers()) {
24
25                     Node node =UMLUtil.getStereotypeApplication(elt, Node.class);
26                     if ( node!=null && node.isControlDecision()){
27                         nbDecisionNode ++ ;
28                 }
29             }
30             for (PackageableElement elt : subsystem.getPackagedElements())
31                 { for (Element l : elt.getOwnedElements())
32                         { Node node =UMLUtil.getStereotypeApplication(l,Node.class);
33                                 if ( node!=null && node.isControlDecision()) { nbDecisionNode ++;  }
34                          }
35                 }
36         if(nbDecisionNode>1){
37             System.err.println("Only one decision Node should be existed in a centralized architechture");
38             return ctx.createFailureStatus("Only one decision Node should be existed in a centralized architechture");
39             }
40         }
41         System.err.println("validation rule");
42         return ctx.createSuccessStatus();
43     }}
```

FIGURE 6.2: The JAVA implementation of one decision node constraint.

ML4Agents Stereotype	UML Base Class
MultiAgentSystem	Model
Agent	Class
DomainRole	Component
ACLMessage	Message
Organization	Collaboration
AgentInstances	InstanceSpecification
OrganizationInstances	InstanceSpecification
MemberShip	Association
Interaction	Interaction
Actor	Lifeline
Service	Class
Capability	Interface
Knowledge	Property
Initiator	Interface
Participant	Interface
Collaboration	CollaborationUse
DomainRoleBinding	Dependency
Plan	StateMachine
Task	BehavioralFeature
AgentAction	BehavioralFeature
Environment	Package
Object	Class
Message	Signal
Ontology	Package
Protocol	Collaboration
Concept	Class
Predicat	Class
Action	Class
NonFunctionalRequirement	Class
FunctionalRequirement	Class
ArchitecturelStyle	Model, Collaboration
Node	Component
RequiredRole	Property
ProtocolarCapability	ProtocolStateMachine

TABLE 6.1: UML extensions for agent modeling

6.2.3 Models and Notation: ML4Agents Concrete Syntax

The proposed ML4Agents is able to support the modeling of all the properties and relationships of the MAS entities and whose the concrete syntax is formulated in a visual syntax allows using UML diagrams for each MAS aspect. ML4Agents defines structural diagrams: MAS diagram; Requirement diagram; Organization diagram, Environment diagram; Role diagram; Agent diagram; Service Diagram and Deployment diagram. In addition, ML4Agents defines the behavioral diagrams. These diagrams aim to represent the dynamic aspects of MAS: Interaction Diagram and Behavior Diagram. The visual syntax allows using diagrams to frame the concepts for its audience of stakeholders within the MAS4SG framework's views.

For each of these diagrams we will use UML diagrams as a model kind to create the MAS4SG framework's viewpoints (see Table 6.2).

Viewpoint	Model Kind	UML Diagram	Diagram Element's (Stereotypes)	Notation
Business viewpoint	Requirement Diagram	Class Diagram	FunctionalRequirement	see 7.5
			NonFunctionalRequirement	
			Plan	
			Satisfy	
Analysis viewpoint	Ontology Diagram	Class Diagram	Concept	see 7.10
			Predicat	
			Action	
			Ontology	
	Environment Diagram	Class Diagram	Environment	see 7.9
			Object	
	Interaction Diagram	Collaboration Diagram	Protocol	see 7.8
			ActorRoleBinding	
Design viewpoint	Role Diagram	Component Diagram	DomainRole	see 7.12
			ProtocolarCapability	
			Plan	
	Mas Diagram	Class Diagram	Agent	see 7.13
			Organization	
			Environment	
			Message	
			OrganizationalAgent	
			DomainRole	
	Agent Diagram	Class Diagram	Agent	see 7.15
			DomainRole	
			Knowledge	
			Capability	
			Plan	
			Play	
	Organization Diagram	Composite Structure Diagram	Organization	see 7.14
			RequiredRole	
			Protocol	
			OrganizationalAgent	
			Plan	
	Collaboration Diagram	Composite Structure Diagram	Collaboration	see 7.16
			DomainRoleBinding	
			ActorBinding	
	Behavior Diagram	StateMachine Diagram	Send	see 7.18
			Receive	
			InternalTask	
			AgentAction	
Deployment viewpoint	Deployment Diagram	Class Diagram	OrganizationInstance	see 7.20
			AgentInstance	
			MemberShip	

TABLE 6.2: The Concrete Syntax of ML4Agents

6.3 FIPA Protocol Library

The MAS4SG framework is compliant with the FIPA standard, thus we developed a library of FIPA protocols specification to be reused for the interaction specification task. FIPA is used to standardize communication between agents. The standard

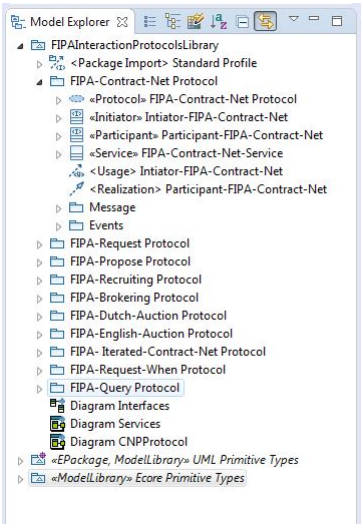

FIGURE 6.3: The protocol library

proposes for each protocol a sequence of sent and received messages and their associated communicative act (i.e., the communicative act used within a FIPA ACL message such as request, inform, ...). In MASs, an agent initiates a protocol to which one or more participating agents can respond. The agent initiator asks for a service for which the protocol is established and the participants provide that service. Thus, we notice that for each Agent Interaction Protocol (AIP) there will be two interaction roles played by the agents in order to ask for or provide a service.

The proposed library of protocols (c.f Figure 6.3) provides a UML specification for FIPA interaction protocols. The MAS designer can use it to design collaboration between domain roles within an organization and can use the specification of the interaction roles' behavior (c.f figure 6.6) to design the internal behavior of agents participating in an interaction (c.f figure 6.9). The idea of reusing the interaction behavior specification is inspired by the method proposed in (Jarraya and Guessoum, 2007), which define a reusable representation of interaction protocols.

We propose a three layered approach for the UML specification of any FIPA protocol, we illustrate this approach on the FIPA Contract Net Protocol (CNP).

- The interaction view: this level represents the overall interaction among agent defined by a protocol. We use the UML Sequence diagram to create this view (see Figure 6.4). An agent interaction protocol (AIP) describes a communication pattern as an allowed sequence of messages between agents and the constraints on the content of those messages. We choose to start modeling the AIP by using a sequence diagram with two lifelines refering to the two interaction roles (i.e., the initiator and participant interaction roles). The messages are asynchronous and stereotyped with «AclMessage» that has an attribute *performative* to indicate the performative of the communicated message.

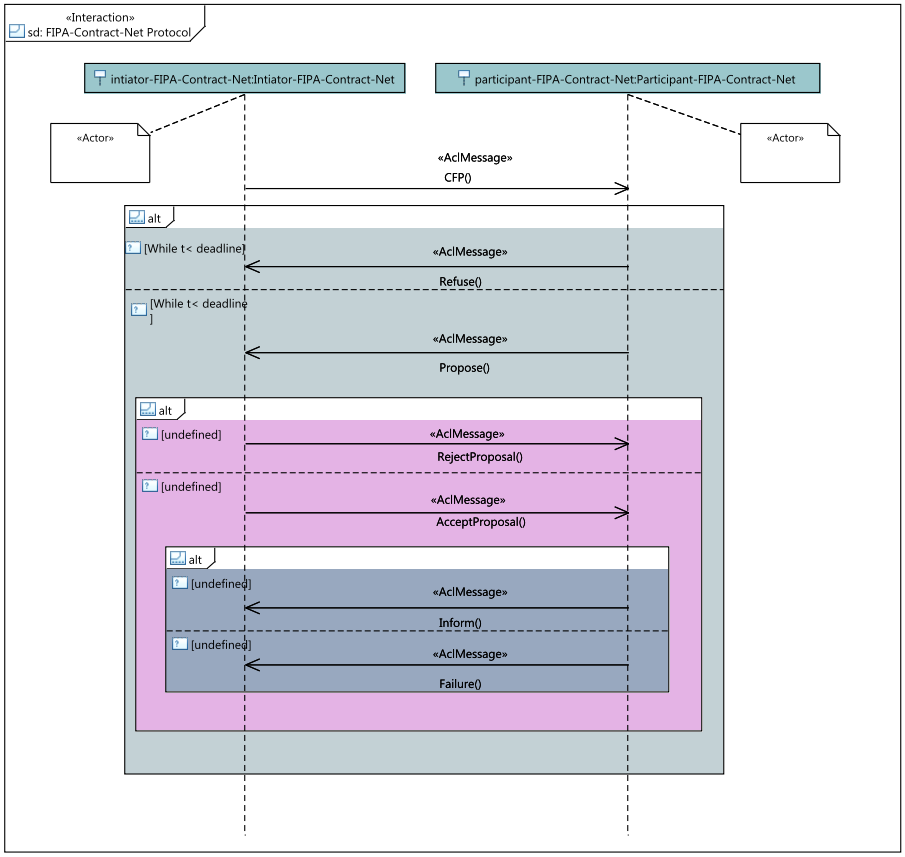

FIGURE 6.4: The CNP sequence diagram

- The service view: this level represents the service for which the protocol is established and the different interaction roles that provide and ask for that service. For this level we use the UML class diagram (see Figure 6.5). The service is a stereotyped class **«Service»** that uses a stereotyped interface **«Initiator»** that depicts the initiator interaction role and realize another that depict the participant interaction role. Each interaction role interface (i.e., the initiator and participant interface) has stereotyped abstract operations **«SendMessage»** for sending messages and others **«ReceiveMessage»** to handle received messages and attributes typed by a signal stereotyped **«Message»** for each message sent by that interaction role interface.
  The initiator and participant interfaces own a protocol state machine that models the interaction behavior of initiator and participant lifeline in the sequence diagram.

- The behavior view: this level represents the internal agent processing. It gives a specification for each interaction role's behavior. That behavior is specified with a protocol state machine which does not primary define behavior. Its base role is to define, when and on which conditions individual behavioral features (operation of sending and receiving messages) can own instances to be invoked. Thereby facilitating the design of agent's internal behavior. For this level we use the UML State Machine diagram (see Figure 6.6).

The FIPA protocol library can then be used for the modeling of any MAS application. The user can start with creating the protocol model that has two properties; an

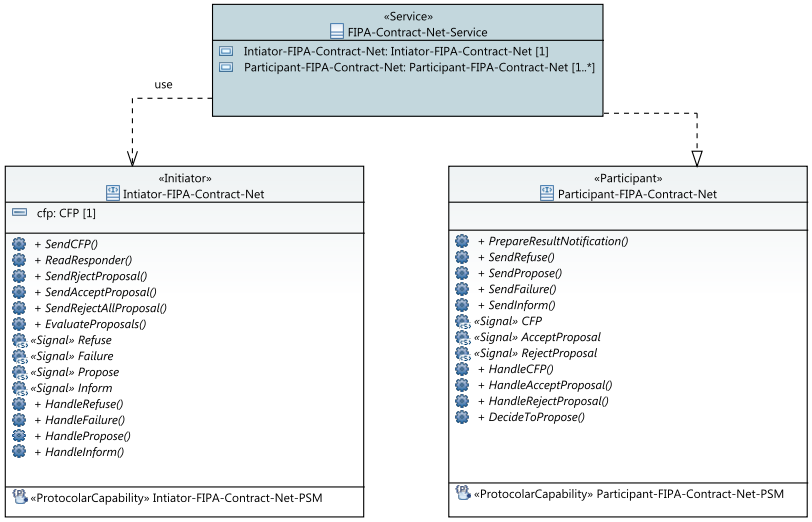

FIGURE 6.5: The CNP service contract

initiator and one or more participant of type interaction role designed in the service model within the second level (i.e. the service view). The protocol described by a stereotyped collaboration «Protocol», it has two stereotyped properties **«Initiator»** and **«Participant»** that refer to the interaction role interfaces used and implemented by its associated services. The protocol model for the CNP example is shown in Figure 6.7.

Furthermore, each agent that plays a role that offering a service should implement the participant interaction role. Each agent that plays a role requiring a service should implement the initiator interaction role. To do that, the agent should own a behavior that respects the protocolar capability of each interaction role it implements. We choose to model the interaction behavior of an agent with a state machine that respects the protocols defined in the third level (i.e., the protocol state machine). Figure 6.9 shows the protocolar behavior of the participant role in the CNP protocol (Protocol State Machine PSM). Figure 6.8 shows the agent behavior that plays the participant role, the latter shows how the agent's behavior (state machine) respects the order of the sending and receiving of messages prescribed by the protocolar behavior of the interaction role CNP participant.

6.4 IEC CIM Profiles Library

This library (Figure 6.10) provides the CIM data model (UML specification) of the different CIM Standards such as (IEC IEC61970, IEC IEC61968 and IEC62325) and provides restricted CIM data model (CIM profiles) classified by application domains such as Modeling and Simulation Application, Monotoring and Diagnostic Application, Distributed Control Application and Protection Application.

We explain in the appendice **??** how to restrict the CIM metamodel into CIM profiles from which we select the CIM classes, associations and attributes to be used to model the WSCC 9-bus test case. In the same way we can restrict the CIM data model to create domain application-specific profiles.

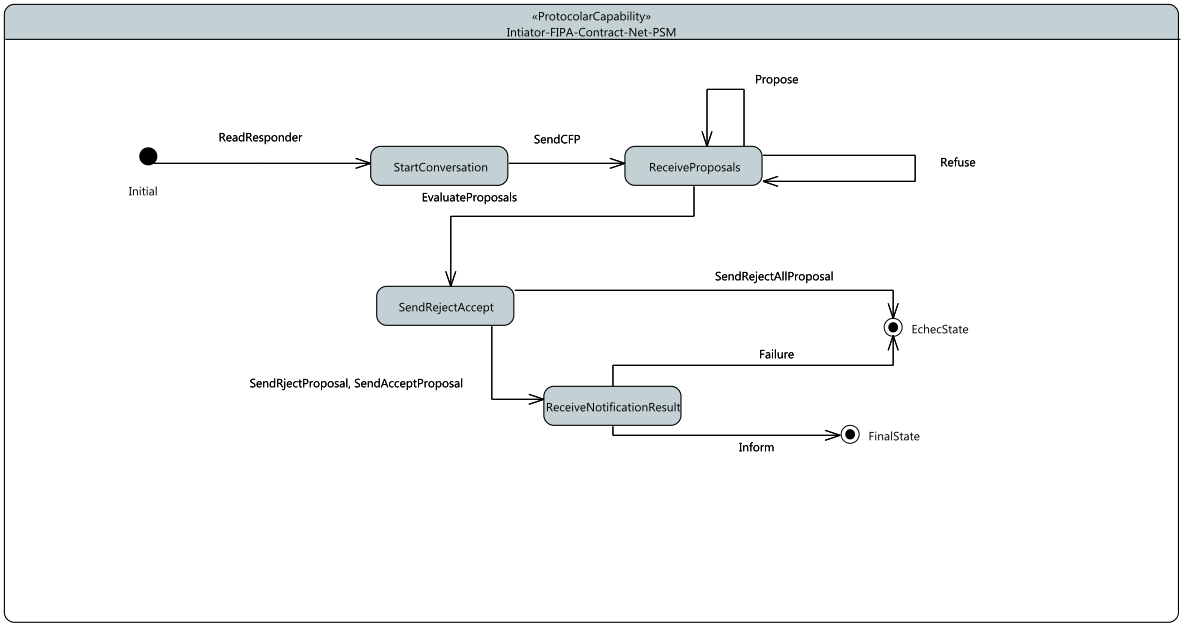

FIGURE 6.6: The Initiator-FIPA-Contract-Net protocol state machine

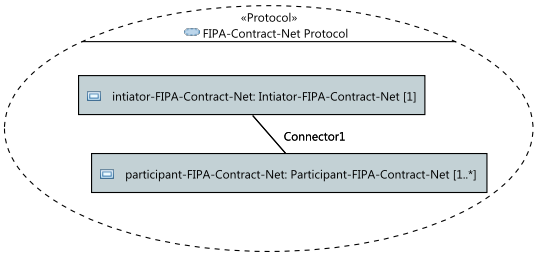

FIGURE 6.7: The CNP collaboration

6.5 Conclusion

In this chapter, we have presented the implementation of the MAS architecture framework dedicated to energy systems. These implementations have been set up to support our methodology for the engineering of MAS systems solutions to solve problems in the energy system sector. We have implemented the ML4Agents language. This is done through a UML profile that extends UML metaclasses. A FIPA protocol library was implemented to support and facilitate the interaction modeling task by reusing the generic UML specification of the FIPA protocols as well as the interaction role and their protocolar behavior. Furthermore, an IEC CIM Profiles Library was implemented to support the environment's objects and the ontology's elements for a specific application domain within the energy system sector. Finally, we propose a plugin for the validation that implements all the OCL constraints using the JAVA language to capture design errors of a MAS at the model level. The plugin includes also the java constraints for all the architectural styles specification proposed by the ML4Agents metamodel. Further ongoing implementations were not completed but could be finished to support the implementation phase by implementing the transformation techniques to automatically generate the code from the MAS application model.

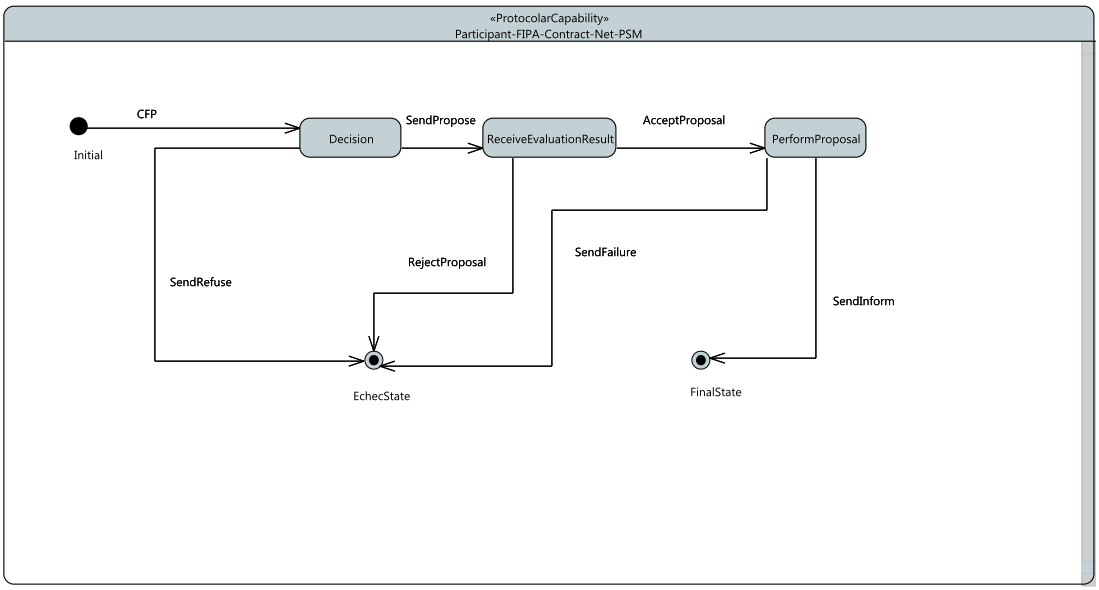

FIGURE 6.8: The Participant-FIPA-Contract-Net protocol state machine

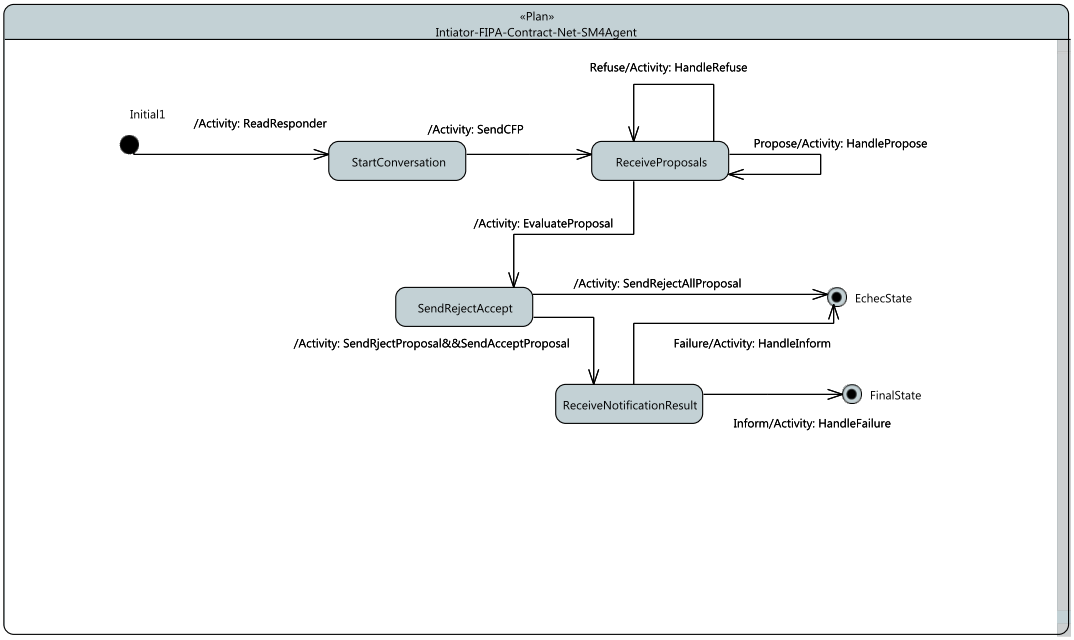

FIGURE 6.9: The agent behavior that depict the Initiator CNP interaction role

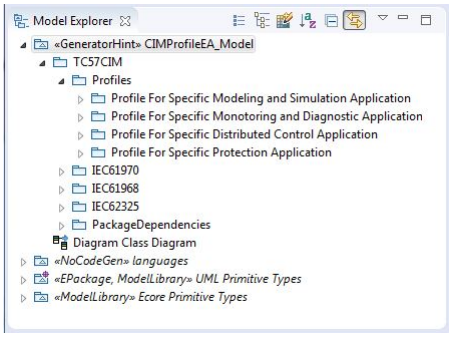

FIGURE 6.10: The CIM data Model library

Chapter 7

Experimentation and Evaluation

7.1 Introduction

In this chapter, we demonstrate how to apply the contributions of this dissertation in a real-world use case. For this purpose, we have chosen a microgrid energy management system for demand response, based on a Multi-Agent system. In the following, we firstly investigate a real-world scenario for the chosen use case and secondly we demonstrate the use of the MAS4SG framework in the engineering of an agent-based solution for the demand response problem for the presented use case. Finally, we present the obtained simulation results and we discuss the functionality validation of the proposed MAS4SG framework.

7.2 USE Case: MAS for Microgrid-Based Demand Response

The definition of Demand Response (DR) by the Federal Energy Regulatory Commission [1] (FERC) is stated as:

Definition 7.1. "Changes in electric usage by end-use customers from their normal consumption patterns in response to changes in the price of electricity over time, or to incentive payments designed to induce lower electricity use at time of high wholesale market prices or when system reliability is jeopardized."

DR aims at reducing demand peaks by shifting or shedding loads directly or indirectly, in response to supply conditions. DR activation might include shutting down a non-critical manufacturing process or shifting critical load consumption. On-site generation and storage systems can also be used to adjust loads drawn from the grid. DR programs can be defined as the changes in the electric power consumption of end-use loads during the critical period, normally the peak loading conditions. The concept of DR has been already explained in more detail in 2.4.2.2.

In the following, we present the test case we have considered for validation and later we present the real world scenario applied for the simulation test.

7.2.1 Details of the Smart Grid Model

To demonstrate and validate our proposed framework, we have first chosen a smart grid, proposed by (Logenthiran, Srinivasan, and Shun, 2012). The chosen SG is composed of three microgirds, including different types of customers in each, namely; residential, commercial, and indistrial customers. Our objective is to validate our MAS4SG framework by proposing a MAS solution for managing the DR problem of the chosen smart grid. Electrical network diagram of the smart grid is shown in

[1] Federal Energy Regulatory Commission, Available [Online]: https://www.ferc.gov/

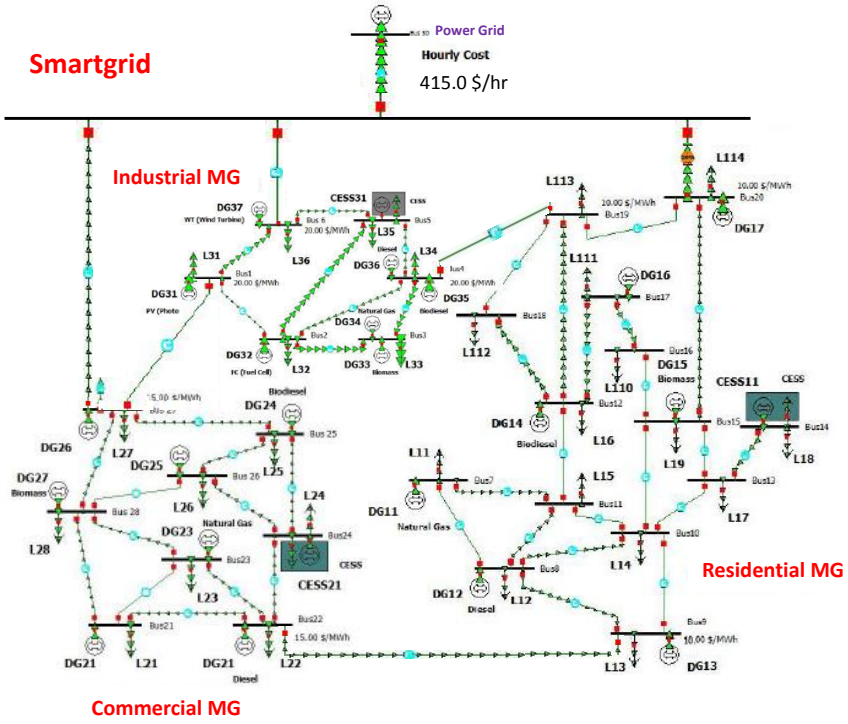

FIGURE 7.1: Electrical network of the smart grid.

Fig 7.1. The entire network operates at a voltage of 410 V. The types of the devices used in the SG and their consumption patters are given in (Logenthiran, Srinivasan, and Shun, 2012) for each microgrid. In order to adapt the SG for the given scenario (with DR), we adjust the given data by adding new information about priority, flexibility and the initial schedule for each device.

The grid contains three different microgrids:

- Residential Microgrid: The controllable loads in the residential microgrid have small power consumption ratings and short durations of operation. There are over 2600 controllable devices available in this microgrid from 14 different types of devices.
- Commercial Microgrid: The controllable loads in the commercial microgrid have consumption ratings which are slightly higher than those in the residential area. There are over 800 controllable devices available for control in this microgrid from 8 different types of devices. .
- Industrial Microgrid: The controllable loads in the industrial microgrid is the smallest among all three areas; however, the devices have largest consumption ratings and longest consumption periods. The reason for a small number of devices available for control can be attributed to the fact that most of the industrial loads are critical and cannot be subjected to load control. The control periods of the devices are similar to those in the other two areas. There are over 100 controllable devices belonging to 6 different types.

The aim of our validation scenario is to develop a MAS solution for a Demand Response (DR) system, where the agents schedule the controllable loads to reduce their consumption during peak hours in order to answer a special request to reduce the peak. The objective is to optimize the consumption of the controllable loads by

shifting their schedules; i.e., the controllable loads shift their electricity consumption to off-peak in order to shave the peak and to reduce the overall operational cost of the network.

Generally, the prices in price-based DR programs are high during peak periods and low during off-peak periods. Thus, the prices are calculated on the basis of the total forcasted load demands of the smart grid. The wholesale electricity prices are hourly-based are reported in Figure 7.2.

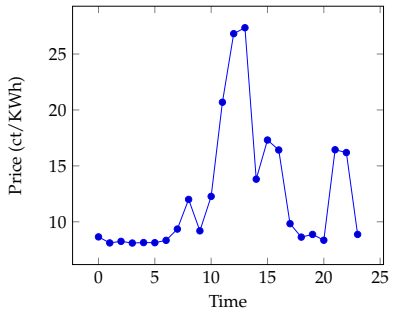

FIGURE 7.2: Wholesale energy prices

Since our aim is to validate our methodology and not to propose a new optimization technique, we have used a simple centralized optimization algorithm, inspired by Logenthiran, Srinivasan, and Shun, 2012, which runs on a daily basis. As input, this algorithm takes the forecasted load demands and the energy prices for a given day (Forcasted load demands for each microgrid is given in Logenthiran, Srinivasan, and Shun, 2012). It then calculates the objective load curves and tries to find the best load scheduling. The result is an optimized schedule for each shiftable device within the grid that brings the total load consumption curve as close as possible to the objective load consumption curve. The problem is mathematically formulated as follows:

$$\textbf{Minimize}: \quad \sum_{t=1}^{N=24} (P_{load}(t) - Obj(t))^2$$

$$\textbf{Where}: \quad P_{load} = P_{fixed}(t) + P_{shifted}(t) + P_{unshifted}(t)$$

$$\textbf{and} \quad Obj(t) = \sum_{t=1}^{N=24} (P_{load}(t)) * Pice_{Avg} * \frac{1}{Price(t)}$$

7.2.2 A real-world scenario

Generally, the independent system operator (ISO) must maintain a certain amount of power reserves all the time, to securely operate electric power systems. The ISO sends a special request to decrease the usage of electricity during peak hours. Under the DR Program, the customer has contract with a Load Aggregator (LA) to participate in the DR program. ISO pays the incentives to reward the DR participants. Customers that are not able to respect their contract are subject to penalties.

The overall procedure starts when the ISO requests for load reduction. The concerned Load Aggregators LAs request for DR participation (to reduce a certain amount of energy during the peak which start at 12:00 pm and lasts for 2 h.) to the MicroGrid Central Coordinator (MGCC). In this case study, only the residential microgrid central coordinator (see Figure 7.3) has a contract with a Load Aggregator to participate in the DR program, where the amount of load reduction (QDR) (400kW in this case study) and the monetary incentive are determined in the contracts. Consequently, the residential MGCC informs the load controllers of the DR event, which occurs at 12:00 pm and lasts for 2 hours. The Load Controller sends information

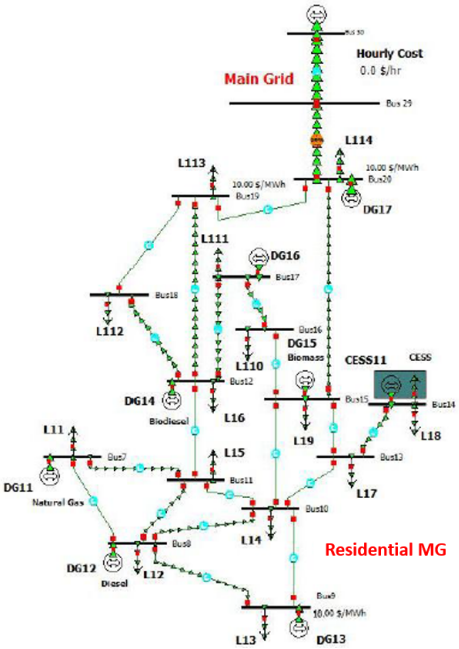

FIGURE 7.3: Network diagram of the residential microgrid

about the load schedule and the flexibility of its controlled Load that is a part of the residential microgrid. The controllable load may have no flexibility, which means that the consumption load cannot be shifted to another time, however it is still able to participate in the DR event by sending information about its consumption schedule to provide more efficiency in the optimization strategy of the whole microgrid. The MGCC runs the centralized optimization algorithm based on the collected data from all LCs, and requests in his turn all controllable loads to schedule their consumption in order to shave the peak load. To do this, the MGCC dispatches the new consumption schedules to all LCs.

The consumption profiles of the loads under the control in the residential microgrid are given in Table 7.1. Table 7.1 shows also the initial schedule, flexibility, priority and number of devices for each controllable load.

In the next section we will illustrate the use of our MAS4SG framework by going through the proposed case study to propose a MAS solution for microgrid energy management, while respecting the methodology steps proposed in the chapter 5.

7.3 Multi-Agent Based Microgrid Energy Management

This section provides the resulting specifications of each phase of our proposed methodology. Our challenge is to find the appropriate architecture and then design and implement it. In the following we will show how we used the proposed MAS4SG framework to solve this problems and how we can use it for modeling and implementing the chosen architecture. Thus, the resulting specification outputs of each phase of our proposed methodology (described in 5) are provided. For a better clarification we add the corresponding SPEM diagram for each phase of the methodology.

Device Type	Initial schedule	Profile			Flexibility	Priority	Number of devices
		1st Hr	2nd Hr	3rd Hr			
Dryer	17:00	1.2	-	-	6	3	189
Dish Washer	13:00	0.7	-	-	5	3	288
Washing Machine	15:00	0.5	0.4	-	6	3	268
Oven	12:00	1.3	-	-	3	2	279
Iron	18:00	1.0	-	-	5	3	340
Vacuum Cleaner	10:00	0.4	-	-	1	2	158
Fan	12:00	0.20	0.20	0.20	1	2	288
Kettle	21:00	2.0	-	-	2	2	406
Toaster	8:00	0.9	-	-	1	1	48
Rice-Cooker	12:00	0.85	-	-	1	1	59
Hair Dryer	8:00	1.5	-	-	1	2	58
Blender	9:00	0.3	-	-	1	1	66
Frying Pan	00:00	1.1	-	-	1	3	101
Coffee Maker	8:00	0.8	-	-	1	1	56

TABLE 7.1: Data of controllable devices in the residential area

7.3.1 The requirement phase

The process of the Requirement phase is depicted in Fig. 7.4.

FIGURE 7.4: The process of the Requirement phase

7.3.1.1 Modeling Requirement Task

The agent solution shall provide a solution for energy management within a smart grid to solve the demand response problem. The system shall be capable of answering the demand response request by shaving the peak. Thus, we can derive that the system requires only one functionality which is: Microgrid-based energy management during the Demand Response (DR) event. The system requires several non-functional requirements in order to improve the system performance regarding the service quality.

In The following, we list a set of the system's quality attributes. We have selected quality attributes that are related to the energy management domain but also depict our intention for this use case:

- Robustness: the control system should be capable of achieving all the tasks without fails: when any DR participant cannot fulfill their DR contracts for some reasons, microgrids in the neighborhood can help them not to break the contract by reduce more load consumption in peak hours.
- Responsiveness: To participate in DR programs, MGCC should be able to quickly answer the DR request, When MGCC requests load shifting, the load controllers send data about their consumption profile and flexibility instantly.

- Communication overhead: it is a critical factor that can affect throughput and real-time operations for any communication framework. Overhead should be low to accelerate throughput on low bandwidth links.
- Modifiability: It must be simple to change the system after it is implemented (and often deployed). In other words, easy adaptability without the need of significant support from technicians or engineers
- Scalability: The system should be good at handling large numbers of devices.

Interoperability: the MAS architecture should be open, i.e., anyone should be able to connect into the system his own load controller. This non functional requirement does no affect the choice of the MAS architecture style and could not consider as a quality attribute that affect the service quality. However, we have to mention it in this section.

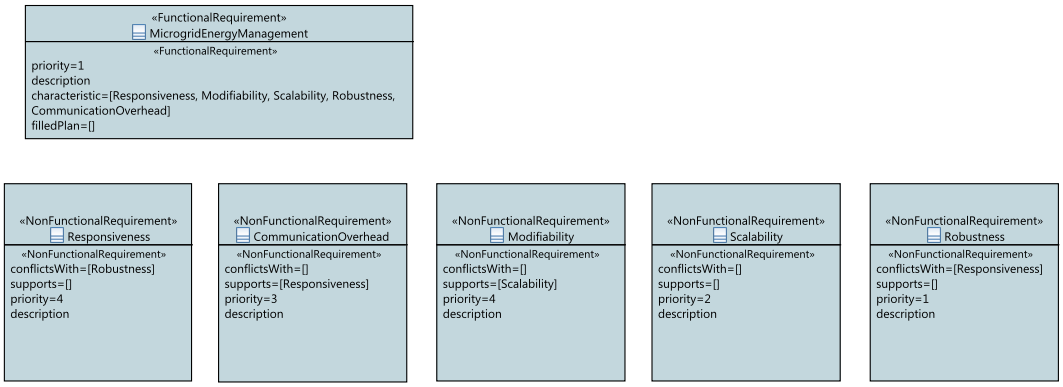

FIGURE 7.5: The Requirement diagram of the DR scenario.

7.3.2 The Analysis phase

The analysis phase aims to analyze the problem and the addressed solution. We start by choosing the appropriate MAS architecture for the proposed solution and then by building the data model, which depicts the agents knowledge and defines a base to model the agent ontology (i.e. the messages parameters). And finally, we select the interaction involved in the developed MAS. The process of the analysis phase is depicted in Fig. 7.6.

7.3.2.1 The Architecture Style Exploration Activity

Many MAS architectures were proposed in the literature for microgrid energy management. We noticed that some of them focus on the control strategies (centralized, decentralized, hybrid, fully distributed,...), whereas others focus on the communication infrastructures (centralized, hierarchical, hybrid,...). In order to select the appropriate MAS architecture, we need (1) to prepare the base of the MAS architecture candidates and (2) choose the appropriate one according to our intentions and objectives such as exchanged data, scalability and computation time. Once the appropriate architecture is chosen, it should be designed and implemented in a better way.

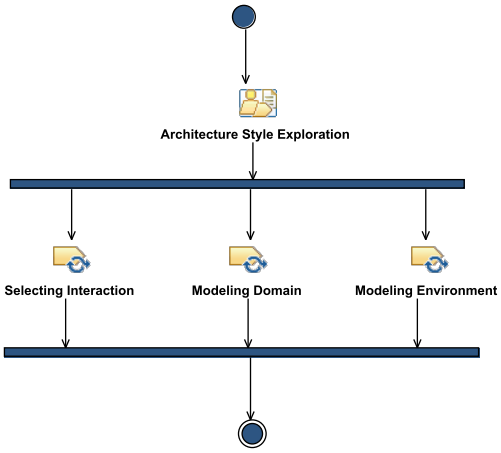

FIGURE 7.6: The process of the Analysis phase

In the following, we start with a description of the quality attributes and the chosen candidate MAS architectural styles. Then, we begin applying a quantitative and qualitative evaluation method on the candidate architecture styles against the chosen quality attributes (i.e., non functional requirements).

7.3.2.2 The Architecture Style Exploration Activity: Evaluating MAS architectural styles Task

In the section 2.4.3 we have classified the MAS architectures for energy management in microgrids according to the *control* schema and the *communication* infrastructure aspect and we discussed the potential and limitations of each architectural style. Two main architectural styles can be found in the literature such as Hierarchical Model and Centralized model depending upon the communication and coordination strategy. The control structure of such systems can be classified into three categories centralized, fully distributed, and Mixed control. Thus we have a base of six candidates MAS architectural styles:

- Hierarchical communication and Centralized control (HC)
- Hierarchical communication and Decentralized control (HD)
- Hierarchical communication and Mixed control (HM)
- Centralized communication and Centralized control(CC)
- Centralized communication and Decentralized control(CD)
- Centralized communication and Mixed control(CH)

MAS with a single level of equal peers is hard to make function well in a modern electric grid and shows weak scalability (Xiao et al., 2010). Thus, for the communication infrastructure, we adopt the hierarchical communication infrastructure. Such decision is subjective, and is based on our experience previous evaluation (Jayasinghe and Hemapala, 2015). In addition, a hierarchy is the earliest and the most widely used topology, in which agents are arranged in a tree-like structure. This topology is typically used in most current control systems.

For the control scheme, we will consider the hybrid or the centralized control because the fully distributed control measurement signals of each MG energy source

are sent to the corresponding local controllers. An advantage of this scheme is the ease of the "plug-and-play" operation and the computation burden of each controller is greatly reduced, and there are no single-point of failure. However, its disadvantage is the potential complexity of its communication system. Thus this strategy is not well adopted for the demand response problem within a microgrid and especially in the Emergency Demand Response (EDR) case where the participants should be able to quickly reduce a certain amount of loads. To the best of our knowledge, a MAS solution for the demand response problem in microgrid that adopts the fully decentralized control strategy has not been proposed. Thus, the base of Candidate MAS Architectural Styles is then reduced and includes only two candidates which are:

1- Hierarchical communication and Centralized control (HC)

2- Hierarchical communication and Mixed control(HM)

There are both advantages and disadvantages associated with each architecture. Objective functions may be conflicting and competing objectives, complicating the achievement of a solution. The two architectural styles are the alternatives representing potential compromises among many objectives.

The Evaluating Architectural Style Task It is possible to evaluate MAS architectural styles with respect to several different quality attributes. In the process of architectural decision making, quality attributes constitute the key drivers for designing software systems. Then, we have to quantify the trade-off between the quality attributes to select the appropriate MAS architecture that constitutes a balance between these attributes.

Qualitative Evaluation

The subjective judgments based on experiences and previous evaluation, thus we were inspired by this evaluation (Davidsson, Johansson, and Svahnberg, 2005). The pair wise method is used for the qualitative evaluation and the results are shown in table 7.2.

	Hierarchical Centralized (HC)	**Hierarchical Mixed (HM)**
Robustness	0,01	0,99
Responsiveness	0,5	0,5
Communication overhead	0,01	0,99
Modifiability	0,99	0,01
Scalability	0,01	0,99

TABLE 7.2: The score of each of the properties of the two architectural styles.

Quantitative Evaluation

An architecture may constitutes a balance between different quality attributes, just as different applications may require a specific balance or a trade-off between the quality attributes. We will now show how the trade-off between the quality attributes can be quantified. To do that we were inspired by the method proposed in (Davidsson, Johansson, and Svahnberg, 2005), which is based on the use of the Analytic Hierarchy Process (AHP) that is a multi-criteria decision support method from Management Science.

We quantify the trade-off between the quality attributes (criteria) to select the appropriate MAS architecture that constitutes a balance between different quality attribute; this can be done by the following steps:

1. Prioritize the criteria in accordance with how they are important for a specific application using the Pair-wise comparison (technique for prioritization) (c.f. Table 7.3)
2. We quantify the subjective assessments with normalized value (c.f. Table 7.4).
3. Evaluation of the candidate architecture style for a given specific balance between quality attribute for a particular application. And then select the most suitable architecture style.

		RO	RE	CO	M	S
Robustness	RO	-	RO	RO	RO	RO
Responsiveness	RE	-	-	CO	RE/M	S
Communication Overhead	CO	-	-	-	CO	S
Modifiability	M	-	-	-	-	S
Scalability	S	-	-	-	-	-

TABLE 7.3: AHP Pair-Wise Comparison Of Quality Attribute.

- Robustness =4
- Responsiveness =1
- Communication Overhead=2
- Modifiability =1
- Scalability =3

100= 4x+x+2x+x+3x; x=9,09 (approximate) where the coefficients in the equation are the number of occurrences of each criterion in the pairwise comparison matrix. This leads to:

- Robustness: 4x= 36%
- Responsivenes: x=9%
- Communication Overhead: 2x=18%
- Modifiability: x=9%
- Scalability: 3x=27%

Table 7.4 presents the priorities of the various properties in the case of a restricted communication and a scalable microgrid.

Robustness	Responsiveness	Communication	Modifiability	Scalability
0,36	0,09	0,18	0,09	0,27

TABLE 7.4: Priorities of the various properties in the case of a restricted communication and a scalable microgrid

Hierarchical Centralized (HC)	0,14
Hierarchical Mixed (HM)	0,58

TABLE 7.5: The score of each of the properties of the two architectural styles.

7.3.2.3 The Architecture Style Exploration Activity: Selecting MAS architectural style Task

Using the data described above, we are now able to instrument the AHP decision support hierarchy with the evaluations of the architectural styles for each of the criteria. We take the product of the priorities of the quality attributes, and multiply this with the corresponding value for each candidate architectural style.

- **Hierarchical Centralized (HC)**: 0.01*0,36 + 0,5*0,09 + 0,01*0,18 + 0,99*0,09 + 0,01*0.27
- **Hierarchical Mixed (HM)**: 0,99*0,36 + 0,5*0,09 + 0,99*0,18 + 0,01*0,09 + 0,99*0.27

The result of this is then summed for each candidate architectural style, and presented in Table 7.5. We can seen, the Hierarchical Mixed architectural style is the most suitable and seems to be the best choice, followed by the Hierarchical Centralized style.

Based on many observations and inspired by previous works (Xiao et al., 2010; Kantamneni et al., 2015), a hierarchical MAS energy management framework with three levels, including a grid level, a distribution microgrid level and a component level is presented in this section. Figure 7.7 shows the structure of the selected Hierarchical and Hybrid MAS architecture applied for this use case.

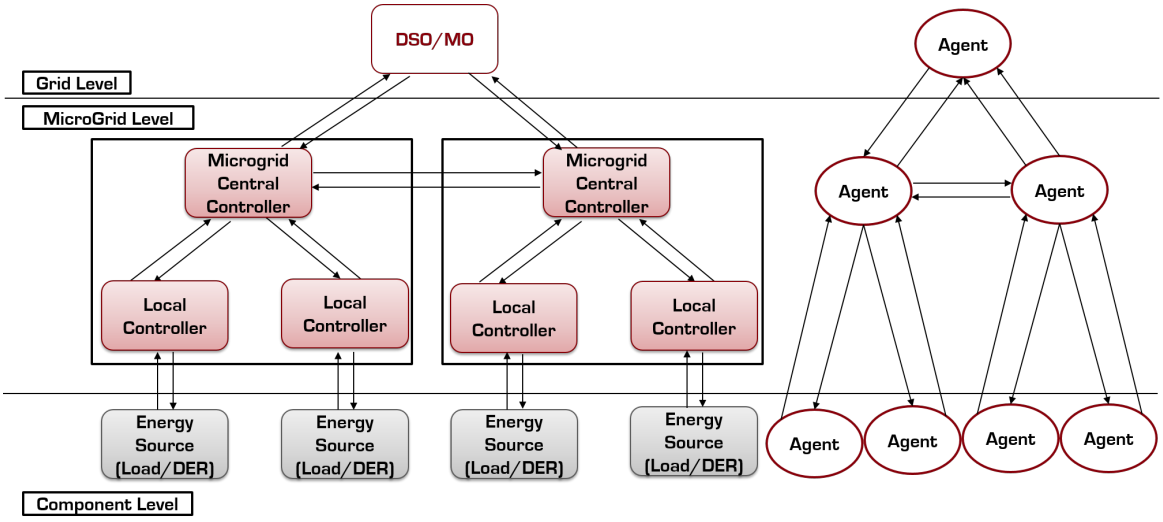

FIGURE 7.7: Schematic diagram of the Hierarchical MAS architecture for a Hybrid microgrid control.

Hierarchy is characterized by some agents having authority over the actions of other agents. Typically, the upper level agents are responsible for the critical decisions, handling large amounts of data and maintaining the overall policy, communication schedules and protocols. The proposed control scheme consists also of two

other layers of decision-making procedures. In the middle layer, the central microgrid coordinator (MGCC) coordinates multiple agents so that the overall microgrid can match the load reduction requested by the grid operator. In addition, when DR participants cannot fulfill their DR contracts for some reasons, microgrids in the neighborhood can help them not to break the contract by producing more power to the grids. Thus MGCCs interact with each other. The lower layer agents interact with actual sensors and devices that are connected to the microgrid. They sense and control components or devices of the microgrid, (i.e., controllable loads). In this layer, intelligent agents control the consumption schedule of individual microgrid loads.

The selected architecture corresponds to the architecture style *MixedHierarchical* which is already defined in the architecture style library. The term mixed refers to the control strategy, whereas the term hierarchical refers to the communication structure.

The chosen architectural style is applied on the first step to the whole model. later, once roles are designed, we can decide if nodes (i.e., roles) are decisional or not. This step is necessary for the verification and validation of the designed application, i.e., errors related to non conformance of the designed architecture with the selected architecture style can be detected. Figure 7.13, show the application of the selected architecture style on the application model.

7.3.2.4 Selecting Interaction Task

Two FIPA Protocols are chosen for the presented use case scenario; the FIPA-Contract-Net Protocol and the FIPA-Request Protocol (see Figure 7.8). The contract between the MGCC and the agents load can be reached by the process of decision-making and for which we choose the CNP for the interaction. Indeed, the MGCCAgent sends a call for proposal to LoadAgents to participate in the DR. LoadAgents sends information about the load schedule and the flexibility of its controlled Load if they accept the received proposal. The controllable load may have no flexibility and thus the LoadAgent refuses the proposal and sends anyway its consumption schedule to provide more efficiency in the optimization strategy of the whole microgrid. The MGCCAgent dispatches the new consumption schedules to all participating AgentLoads. We choose the FIPA request protocol for the interaction among MGCCAgents, i.e. if the MGCCAgent cannot answer the request for load reduction with a certain amount of load, it requests MGCCAgents that manage the microgrids in the neighborhood in order to achieve its goal.

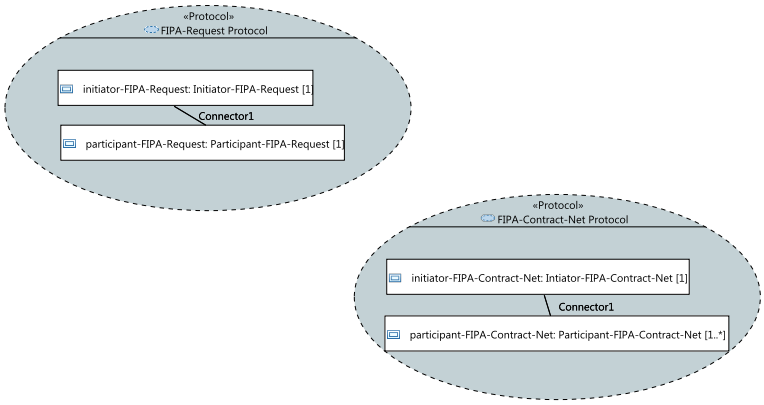

FIGURE 7.8: The Interaction diagram of the DR scenario

7.3.2.5 Modeling Environment Task

A CIM profile for the Demand Response problem is created to be used for the agent knowledge modeling (i.e Environment) and to be the base in designing the agent ontology (messages parameters). Figure 7.9 depicts a partial view of the environment diagram of the DR scenario. It includes objects, e.g. EquipmentContainer to represent a microgrid, ConformLoad to represent a controllable load and ConformLoadSchedule to represent the load's consumption schedule, used to store agent's knowledge on their environment's objects. The objects may have relations to other objects or to the primitive types Integer, String and Real.

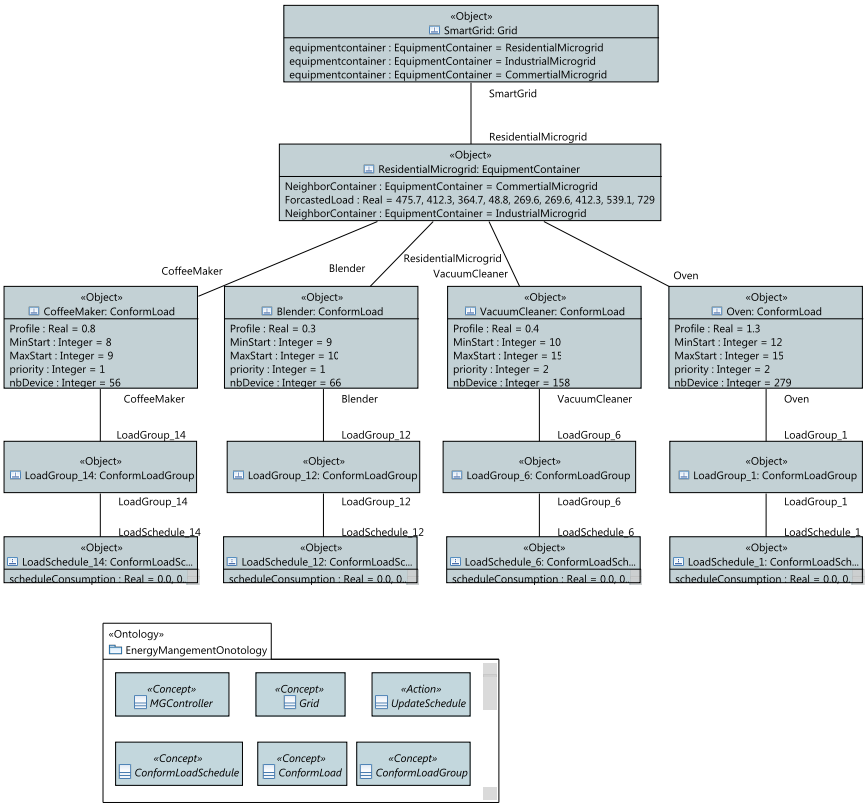

FIGURE 7.9: The environment diagram of the DR scenario

7.3.2.6 Modeling Domain Task

The CIM objects are used again as a base for the domain modeling, other concepts and actions we added to represent other domain entities that are not included in IEC CIM data model such as the UpdateSchedule action that is sent in the content of a message to request an agent to update it's controlled load's consumption schedule. The Figure 7.10 depicts the ontology model for the DR scenario.

7.3.3 The Design Phase

The process of the Design phase is depicted in Fig. 7.11.

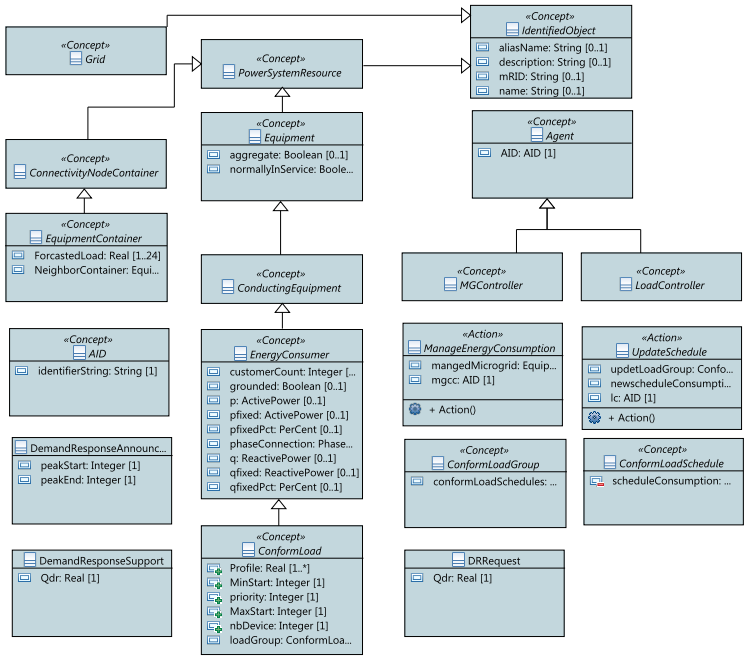

FIGURE 7.10: The domain model for DR scenario.

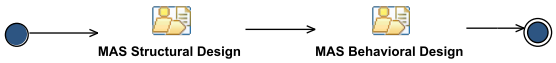

FIGURE 7.11: The process of the Design phase

7.3.3.1 MAS Structural Design Activity: Modeling Role Task

The identified roles in our use case match the USEF [2] framework, expanded with a role for managing a microgrid (MGCC) and another for a Device in a microgrid where micro-generation, load and battery are examples of device roles.

DSO role: The Distributed System Operator role is responsible for predicting the future occurrence of congestion points; if congestion is expected at a point, a special control request is delivered to the Load Aggregator (LA) such as a command for load reduction during peak hours.

Aggregator role: it is a middleware between DSM and Load, it is responsible to make a deal with the load in order to reduce the energy consumption to answer the DSM demand, and then it sells flexibility to the DSM. This role is responsible for monitoring possible congestion points raised by the DSO: if a congestion is expected, the AGGR tries to sell flexibility, bought from the MGCC or other AGGRs, to the DSO.

MGCC role: this role matches the prosumer role in USEF framework. The microgrid central coordinator (MGCC) coordinates multiple agents so that the overall microgrid can match the load reduction requested by the grid operator. More precisely, this role is responsible for locally optimizing energy consumption (and generation) and for bargaining flexibility with the AGGR. The agent is allowed to interact with the AGGR and the in-microgrids devices.

Device role: it is responsible for managing a controllable load consumption schedule in order to answer a demand from the aggregator to reduce the energy

[2] "Universal Smart Energy Framework", Available [Online]: https://www.usef.energy

peak; its decision depends on its flexibility, incentive, operation cost, etc; which are a private information. The Device role in USEF corresponds to a load controller.

Figure 7.12 depicts the role diagram for the DR scenario.

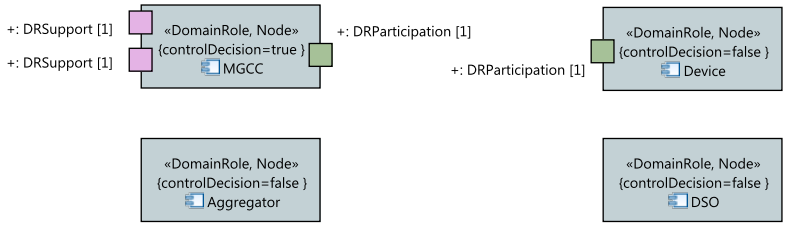

FIGURE 7.12: The role diagram of the DR scenario.

7.3.3.2 MAS Structural Design Activity: Modeling MAS Task

Figure 7.13 depicts the MAS diagram of DR scenario. It contains an overview on the (i) organization MicrogridEnergyMangementOrganization and (ii) the agents MGCCAgent, LoadAgent, AggregatorAgent and DSOAgent and (iii) domain roles MGCC, Device , Aggregator and DSO and (iv) the MicrogridManagement environment and (v) the MicrogridManagement ontology and (vi) part of the set of the exchanged messages between agents in the MAS;

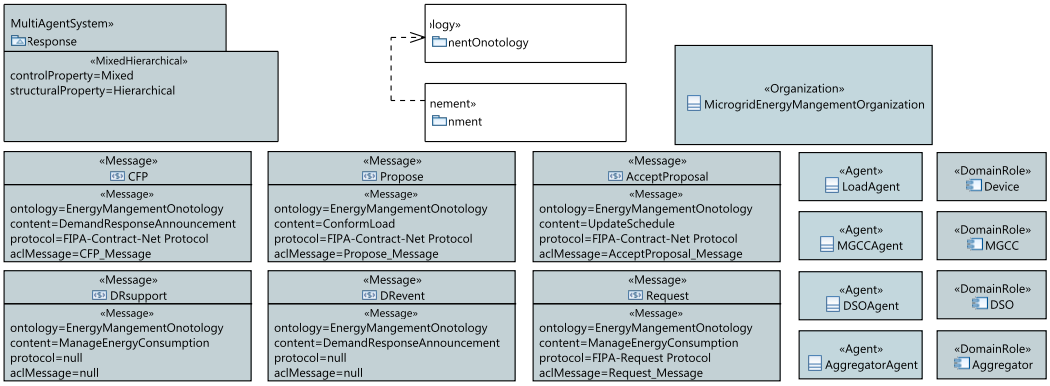

FIGURE 7.13: The MAS diagram of the DR scenario.

7.3.3.3 MAS Structural Design Activity: Modeling Organization Task

For the DR scenarioo, we design only one organization that requires four Domain Roles; MGCC, Device , Aggregator and DSO. Figure 7.14 depicts the organization diagram of DR scenario.

7.3.3.4 MAS Structural Design Activity: Modeling Agent Task

The Agent model identifies the agent types belonging to the system, and the agent instances that will be instantiated. In the present case, each role corresponds to an agent type. Thus, we have four types of agents: DSO, MGCC, Aggregator, and Load. This classification considers that the Load agent is extended for each type of conformed load. At the district level, there is a single DSO, whereas the other types

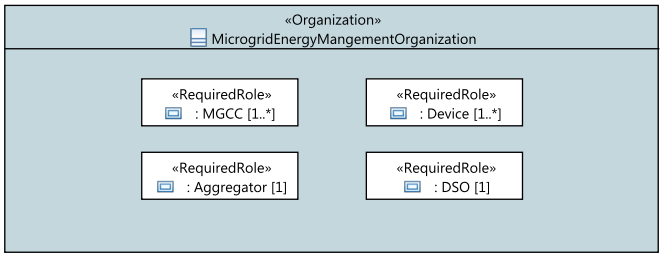

FIGURE 7.14: The Organization diagram of the DR scenario.

have a cardinality higher or equal to 1. Figure 7.15 depicts a partial view of the agent diagram of DR scenario.

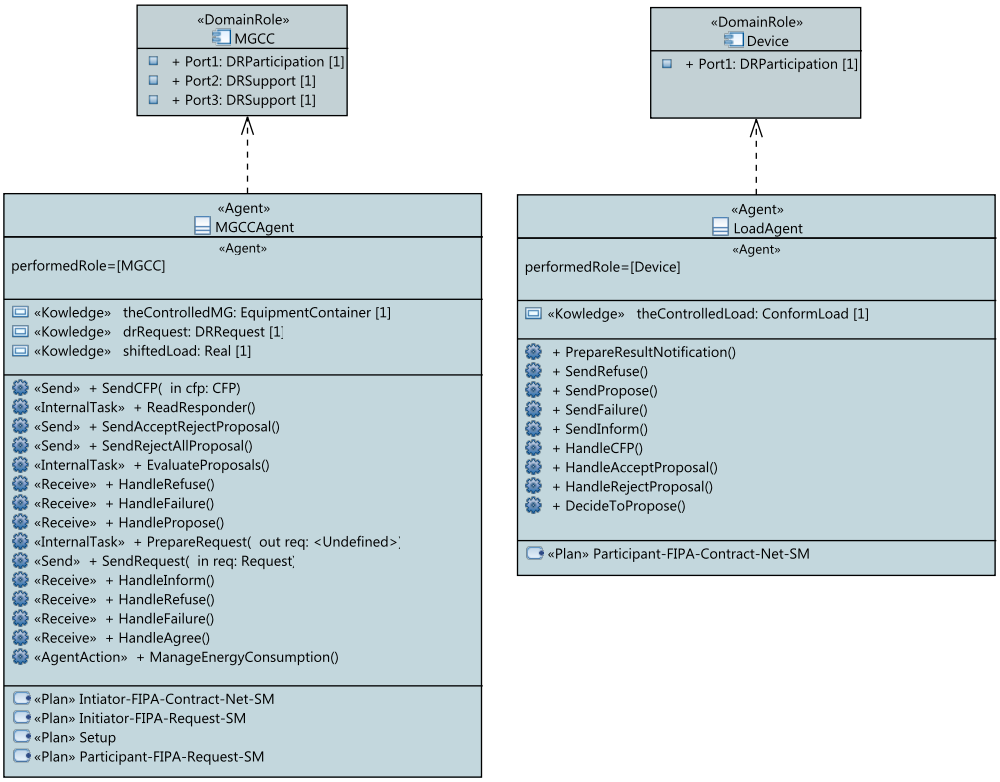

FIGURE 7.15: The Agent diagram of the DR scenario

7.3.3.5 MAS Structural Behavioral Activity: Modeling Collaboration Task

Figure 7.16 depicts the two collaborations OptimizeLoadConsumption and SupportDRrequest of the MicrogridEnergyManagementOrganization. The OptimizeLoadConsumption collaboration specifies the bindings initiator-FIPA-Contract-Net and participant-FIPA-Contract-Net of the FIPA Contract Net protocol and the domain roles MGCC and Device of MicrogridEnergyManagementOrganization. The SupportDRrequest defines the bindings between the actors Initiator-FIPA-Request and Participant-FIPA-Request of the protocol FIPA-Request Protocol and the domain roles MGCC and MGCC of MicrogridEnergyManagementOrganization.

The MGCC domain role can be bound to several actors of different protocols. It can be bound to an Initiator-FIPA-Contract-Net actor in the OptimizeLoadConsumption collaboration to ask loadAgent to participate in the peak load reduction, and can be bound to the Initiator-FIPA-Request actor to request another MGCCAgent to support it to answer the DR request when it fails to answer it by itself.

The domain role bindings are expressed through the corresponding domain role bindings dependency, whereas the actor binding are expressed through the protocol's properties such as the initiator-FIPA-Contract-Net and participant-FIPA-Contract-Net property of the FIPA-Contract-Net protocol (see fig. 7.8).

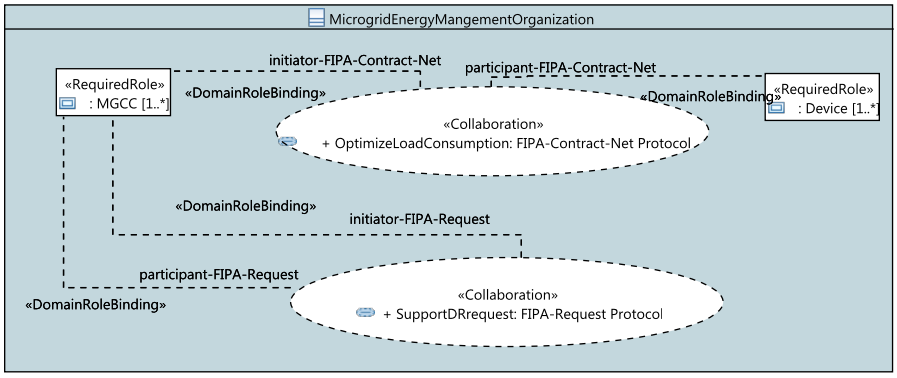

FIGURE 7.16: The collaboration diagram of the DR scenario

7.3.3.6 MAS Structural Behavioral Activity: Modeling Behavior Task

Figure 7.17 depicts the setup internal behavior of the MGCCAgent. It is a composite behavior type of Finite State Machine (FSM), it combines three internal behavior to reduce the complexity of the plan's body. The containing state machine (i.e setup behavior) contains three submachines which are: Intiator-FIPA-Contract-Net-SM, Intiator-FIPA-Request-SM, Participanr-FIPA-Request-SM. The MGCCAgent (i) initiates the CNP protocol when it receives a request for load reduction. If it fails in answering the received request it (ii) initiates the FIPA request protocol for asking support from other MGCCAgents. The MGCCAgent can be requested to help other MGCCAgents, thus it can (iii) participate also in a FIPA request protocol.

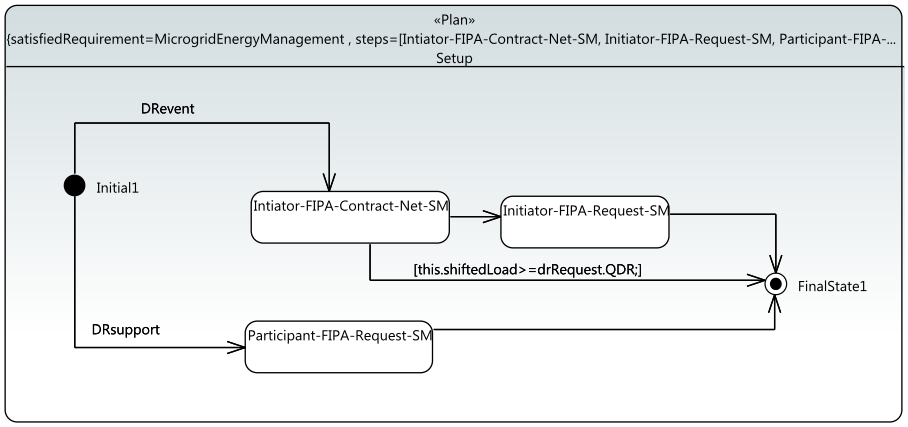

FIGURE 7.17: The behavior diagram of the MGCCAgent'setup behavior

Figure 7.18 depicts the Intiator-FIPA-Contract-Net-SM internal behavior of the MGCCAgent, which is provided by the MGCC domain role. It respects the protocolar behavior (protocol state machine) of the Intiator-FIPA-Contract-Net-PSM actor of the FIPA CNP protocol from Figure 6.6.

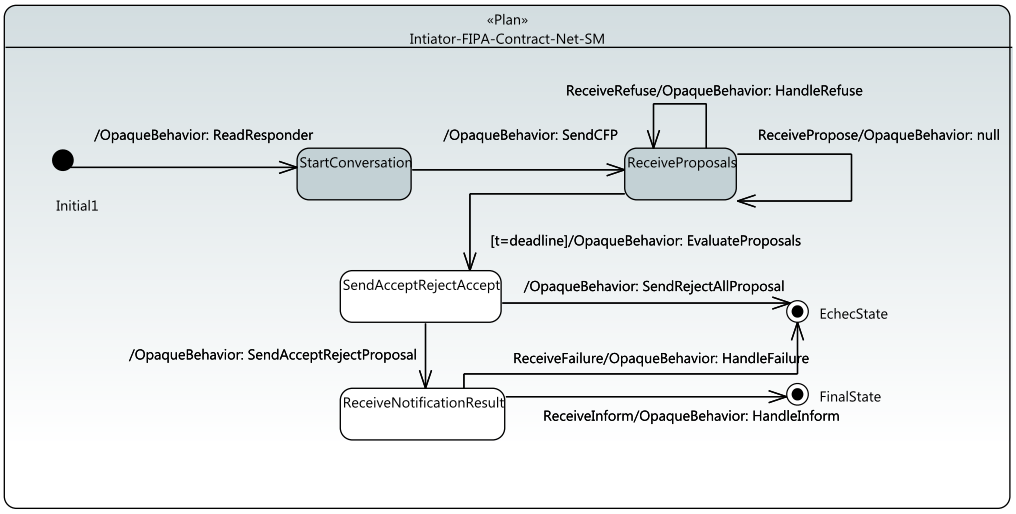

FIGURE 7.18: The behavior diagram of the Intiator-FIPA-Contract-Net behavior

7.3.4 The Deployment Phase

The process of the Deployment phase is depicted in Fig. 7.19.

FIGURE 7.19: The process of the Deployment phase

7.3.4.1 Modeling Deployment Task

Figure 7.20 shows the deployment diagram of the DR example. We modeled an instance of the MicrogridEnergyMangementOrganization. Furthermore, there are several agent instances, i.e., Dryer and Oven of type LoadAgent and MGCC of type MGCCAgent that controls a residential microgrid. The agent performs one or more domain roles, in the deployment phase the designer can specifies precisely which domain roles is played by an agent instance in a particular organization instance. The domain role an agent instance performs in an organization instance is specified by the membership dependency which is visualized as a link.

7.3.5 The Implementation Phase

The process of the Implementation phase is depicted in Fig. 7.21.

FIGURE 7.21: The process of the Implementation phase

As the implementation of the model transformation is not yet supported by the MAS4SG framework, we manually implemented the DR scenario in the JADE plateform as it is a FIPA platform. Now that we have created the models of the agents and the circuit objects, we manually implement the MAS code that shall run on the JADE platform. Figure 7.22 shows a part of the java code of the MAS solution (platform-specific for JADE), that allows managing the CIM components in our test case. Figure 7.23 shows part of the code of the *LoadAgent* class, where its controlled load is given through its arguments array. This equipment is sent as an argument from the main class shown in Figure 7.22. The results obtained from the proposed optimization algorithm for the residential smart grid is illustrated in Figure 7.24. The figure shows how the load shifting algorithm tried to bring the final consumption curve closer to the objective load curve. For instance, the amount of the forecasted load consumption between 12 and 2 p.m. was reduced after applying the shifting technique, since the selling price of energy at that time is expensive. The simulation outcomes show that proposed technique achieves sustainable saving by reducing the system peaks during the peak periods and as result the total consumption cost has decreased also from 2302.87$ to 2129.27$. The peak is reduced for more than 400 KW (the QDR) between 12 and 2 p.m., in this case the MGCC agent answers the DR request and thus it doesn't need any help from the MGCCs in its neighbor.

7.4 Evaluation and Functionality validation of MAS4SG

This section evaluates the framework the benefits compared to the approach in (Logenthiran, Srinivasan, and Shun, 2012) that considers the same case of study. Furthermore, this section validates the MAS4SG's functionalities and discusses the requirements it covers.

7.4.1 Modifiability and reuse

In (Logenthiran, Srinivasan, and Shun, 2012), authors focus on optimization techniques that are evaluated through implementations. No methodology was followed

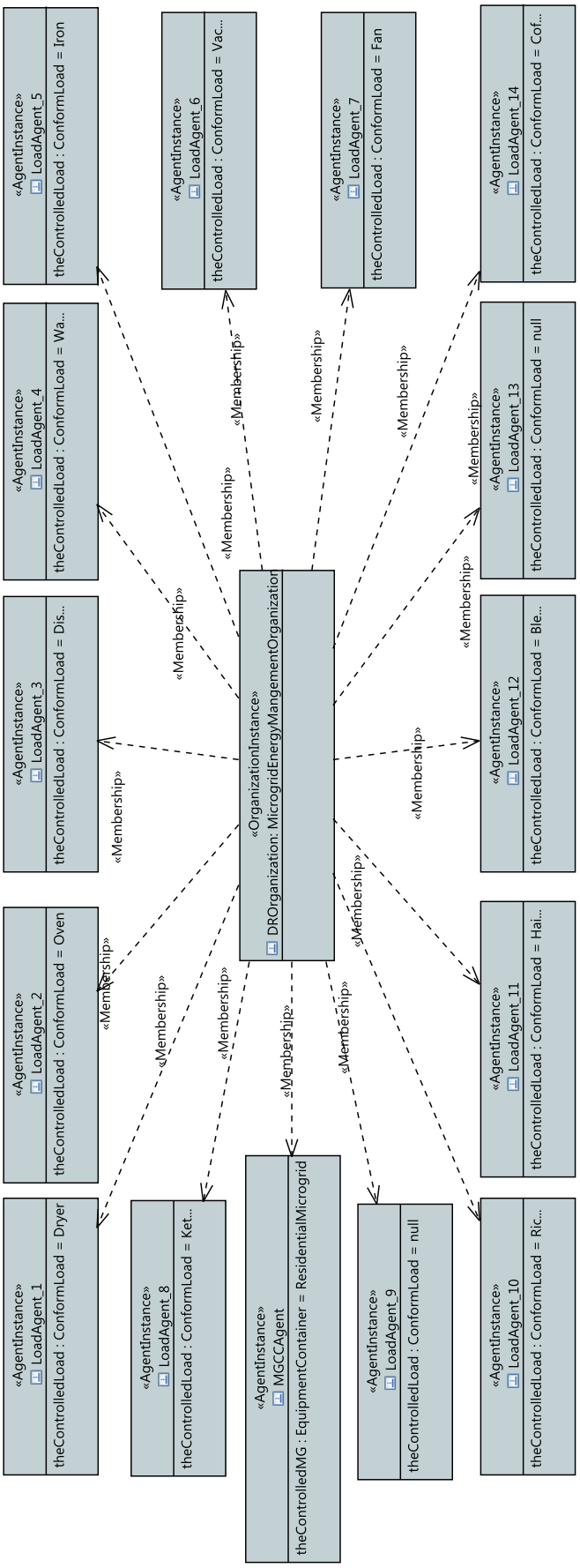

FIGURE 7.20: The deployment diagram of the DR scenario.

```
ConformLoad Kettle= new ConformLoad("Kettle",21,23,new double[]{0,0,0,0,0,0,0,0,
ConformLoad Toaster=new ConformLoad("Toaster",8, 9,new double[]{0,0,0,0,0,0,0,0,
ConformLoad RiceCooker= new ConformLoad("RiceCooker",12,13,new double[]{0,0,0,0,
ConformLoad HairDryer= new ConformLoad("HairDryer",8,9,new double[]{0,0,0,0,0,0,
ConformLoad Blender= new ConformLoad("Blender",9,10,new double[]{0,0,0,0,0,0,0,0
ConformLoad FryingPan= new ConformLoad("FryingPan",0,1,new double[]{1.1,0,0,0,0,
ConformLoad CoffeeMaker= new ConformLoad("CoffeeMaker",8,9,new double[]{0,0,0,0,
ConformLoad [] Loads =new ConformLoad[]{Dryer,DishWasher,WashingMachine,oven,Iro
double[] dataPrice=new double[]{8.65,8.11,8.25,8.10,8.14,8.13,8.34,9.35,12.0,9.1
double []ForcastedLoad=new double[]{475.7,412.3,364.7,348.8,269.6,269.6,412.3,53
ResidentialGrid grid= new ResidentialGrid(Loads);
grid.setForcastedLoad(ForcastedLoad);
ArgsDSM = new Object[2];
ArgsDSM[0]= ForcastedLoad;
ArgsDSM[1]= dataPrice;
```

FIGURE 7.22: Part of the Java Code of the Main Class

```
public class LoadAgent extends Agent  {
     /**
private static final long serialVersionUID = 1L;
private Codec codec = new SLCodec();
private Ontology ontology = EnergyMangementOnotology.getInstance();
private ConformLoad theControlledLoad= new ConformLoad();
     @Override
protected void setup() {
    Object[]args=getArguments();
    String name;
        if(args.length==1){
            theControlledLoad=(ConformLoad) args[0];
            System.out.println( "Hello I'am "+ getAID() +" I manage:
            getContentManager().registerLanguage(codec);
            getContentManager().registerOntology(ontology);
            name= theControlledLoad.name;
```

FIGURE 7.23: Part of the Java Code of the LoadAgent Class

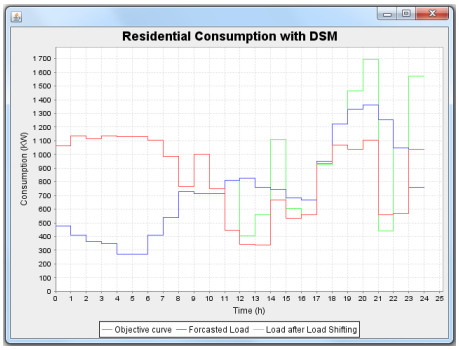

FIGURE 7.24: Results for the residential smart grid

for the MAS specification and thus, no models were proposed to be reused in the future. Thus, any extension should be done directly on the code which is very complicated to do and sometimes requires re-implementation of the whole application when new interactions should be handled with new agents or in case we want to add new capabilities for agents. However, the MAS specification we proposed for the present case is intended to be reused, i.e., the output model of each modeling task can be edited or extended to be reused to model applications in the energy management domain. This is very useful to facilitate future design tasks. Our specification can be extended to support interactions with other new agents by simply adding new collaboration between the played roles through any FIPA protocol. Furthermore, the use of the CIM ontology will guarantee understanding between new and existing agents. This can be done by editing collaboration and role models. Other capabilities can be assigned to agents by extending the roles they play with new functionalities. Thus, the MAS architecture specification we have developed for the presented case of study is flexible, extensible and open.

7.4.2 Best practice solution

The use of the MAS4SG framework in the engineering of the chosen case of study, shows the possibility for reusing existing work (or part of them) in the Smart Grid domain while modeling MAS solutions for ES problems rather then to start from scratch. For example, the system non-functional requirements could be derived from existing studies that elaborate list of quality attribute for a specific application domain such as (Davidsson, Johansson, and Svahnberg, 2005) that gives a set of quality attribute for the dynamic and distributed resource allocation domain. This set includes common characteristics for applications under this domain, e.g. power load management application. Furthermore, key roles, interactions and agent's behaviors could be designed following the Universal Smart Energy Framework (USEF). In addition, the ontology design is based on the Common Information Model (CIM) standard; and the knowledge design is based on the existing energetic data standards identified within the SGAM's information layer.

7.4.3 The MAS4SG's characteristics

- **Portability:** The designed MAS application model is independent from the agent platform we will used to execute our application, it can be transformed to many platform specific models which can be transformed on their turn to executable codes on different agent platforms.

- **Extension:** The base of the MAS architectural styles library we proposed can be extended with other MAS architectural styles patterns classified on other different aspects. The modeling language can be extended to cover the implementation phase and thus, transformation task could be automated.

- **Flexibility:** In this thesis, our aim was to propose a MAS architecture framework specific to the energy systems domain. However, our studies lead to propose an architecture framework specific to agent system engineering domain adapted to be appropriate for engineering MAS applications dedicated to the energy system sector. Thus, the proposed framework, the supported ML4Agents language and the connected methodology are flexible and could be used for other application domains if we take other choices about adhering standards to meet the specific domain requirements.

  Moreover, our framework can be supported by other MAS architecture evaluation methods besides the pair wise method we used to evaluate the MAS architectural styles.

7.4.4 Smart Grid's Requirements coverage

- **Interoperability**

  The developed MAS application uses FIPA protocols for the interaction that are specified in the proposed FIPA protocol library, . This shall guarantee the communication between agents from different agent systems and thus, solve interoperability issues among heterogeneous MAS. Furthermore, the use of the CIM profiles library to model ontology solves the issues of interoperability among the agents themselves, which are related to the communication semantics; The uptake of the CIM and FIPA standard by the MAS4SG framework assured the interoperability requirement of the smart grids systems.

- **Architectural Exploration** Generally, authors invent a new architecture or reuse one and apply it to a particular problem and conclude that it seems appropriate for this problem regarding the implementation results, without drawing any general conclusions.

  Using the MAS4SG framework, the non functional requirements are specified in the early phase of the MAS engineering process and used as a basis to evaluate the possible MAS architectural style for the present case, this is very optimal to select the best architecture solution that considers the intention of the designer regarding the quality of services of the developed application. The implemented constraints that are related to the architectural style concept ensure the detection of wrong choices in the design of the MAS architecture, which shall respect the selected style architecture. Thus, familiarity with the selected architectural style is not anymore necessary to achieve the potential of that particular MAS architectural style.

7.5 Conclusion

This chapter presents an agent-based solution for energy management in microgrids. A given use case for demand response have been tested to examine the operation of microgrid. The agents are programmed to flexibly communicate to the MGCC via the CNP and then finally find a solution of each unit corresponding to a certain DR request for peak shaving.

This chapter demonstrates the use of the MAS4SG framework and how to apply the contributions of this dissertation. Furthermore, it validates the MAS4SG functionalities regarding the smart grids's requirements to develop a flexible, extensible and open MAS architecture. The last section in this chapter, evaluates qualitatively the framework benefits and functionalities.

Chapter 8

Conclusion and future perspectives

8.1 Work review

In this dissertation, we proposed a MAS architectural framework called MAS4SG for the development of MAS solutions to manage ESs applying the IEEE recommendations for the use of energetic and MAS standards. MAS4SG helps power engineers to uptake the MAS technology in the power engineering field in order to develop agent based software to solve problems of Smart Grids where we tried to meet the energy system engineering major requirements: **Interoperability** and **Architectural exploration**. The building of MAS4SG framework's viewpoints reveals the need for a modeling language. Thus, we proposed an extension for the existing PIM4Agents metamodel to fulfill the different requirements such as:

- Including FIPA specification in the metamodel to guarantee interoperability
- Include new concept to model system requirement in the earlier phase and propose the architecture style metamodel with concepts to classify architecture style.

We presented a methodology to support the use of the MAS4SG framework to analyze, design and develop MASs for energy system. It follows the MDE process in order to guide the power engineers in the design and implementation of MAS applications, and to simplify the design task by reusing models. Our methodology uses the CIM standard, is divided into five phases ,namely: requirement, analysis, design, deployment and implementation phases. The methodology was presented and validated by the development of a MAS optimization solution for a well-known application.

8.2 Perspectives

Considering the architectural exploration problem, the evaluation of possible MAS architectures can be done through an algorithm which gives as result the appropriate MAS architectural style for the modeled application. The selected architecture can then be applied on the application model by the means of a tool that takes as input the XMI file of the model application and the selected MAS architecture style and gives as result a first proposal of a UML model application with an architecture that corresponds to the selected architectural style.

In addition, a set of possible MAS architecture style shall be proposed and classified according to the domain application within the smart grid domain. For each MAS architecture style, a set of quality attribute satisfied by each architectural style can be fixed.

Implementation of specific diagrams, model transformations and code generation can be developed in order to give a tool for MAS development dedicated to Smart grids that covers all the MAS development phases (i.e., from requirement to implementation).

In this thesis, our basic idea was to propose a MAS architecture framework specific to the energy systems application domain. However, we ended up proposing an architecture framework specific to agent system engineering domain adapted to be appropriate for engineering MAS applications dedicated to the energy system sector. Thus, we conclude that the proposed framework, its supported ML4Agents language and the connected methodology are flexible and could be used for other application domains besides the energy systems domain by adhering standards to meet the target domain requirements.

Appendix A

Résumé

A.1 Introduction

Au cours des dernières années, les systèmes multi-agents (SMA) sont devenus l'une des technologies les plus prometteuses pour la conception et le développement des systèmes énergétiques intelligents nommés aussi Smart Grid McArthur et al., 2007b; Rohbogner et al., 2012; Ghribi et al., 2014. Les solutions basées sur les agents traitent les systèmes énergétiques complexes en permettant la mise en place d'entités de contrôle autonomes intégrées dans des environnements dynamiques et distribués, capables de se coordonner de manière coopérative et tolérante aux pannes pour résoudre des problèmes complexes de manière distribuée (e.g.,optimisation) ou pour résoudre un problème distribué (e.g.,contrôle distribué).

Actuellement, les SMA sont exploités de deux façons: comme une approche pour construire des systèmes extensibles et flexibles (i.e., la capacité de réagir correctement aux situations dynamique); et comme une approche de modélisation et simulation. Cependant, l'utilisation de la technologie d'agent dans l'ingénierie des systèmes pour modéliser, contrôler et simuler les comportements des systèmes énergétiques, pose encore de nombreux défis qui se résument en quatre problèmes majeurs: (i) problème méthodologique dû à la variété de méthodologies de conception existantes et les différentes anatomies des agents; (ii) problème technique liée à la diversité des approches d'implémentation et des plateformes d'agent;(iii) problème d'interopérabilité entre les SMA hétérogènes et entre les agents eux même au niveau de la sémantique de la communication; et (iv) problème d'exploration architectural lié à la sélection d'un style d'architecture approprié à l'intention du concepteur du système et aux exigences non fonctionnelles du système dédié.

Ces défis justifient le besoin d'un cadre architectural pour faciliter la conception et l'implémentation de solutions SMA dans le domaine de l'énergie qui répondent aux exigences fonctionnelles et non fonctionnelles du système. On peut distinguer deux exigences pour y répondre :

- L'interopérabilité entre les SMAs hétérogènes et entre les agents eux-mêmes, qui sont liés à la sémantique de la communication;
- Les architectures SMA doivent s'adapter à la complexité croissante des réseaux électriques. Les architectures décentralisées sont privilégiées pour favoriser l'extensibilité; toutefois, pour ajouter ou supprimer un fournisseur ou un consommateur, il sera un peu mieux de le faire avec des architectures centralisées, car les changements peuvent n'être nécessaires que dans une partie du système. Comme ces deux objectifs sont dans la plupart des cas contradictoires, surtout lorsque des résultats optimaux sont attendus, il faut trouver un compromis pour définir l'architecture SMA appropriée qui améliore mieux la

qualité de service et répond à l'intention centrale de conception concernant l'exigence non fonctionnelle (telle que l'extensibilité et la résilience).

Cette thèse porte sur la conception et le développement des systèmes multi-agents, et en particulier sur leurs architectures, afin de répondre aux exigences précédentes. Les questions de recherche (QR) sous-jacentes sont les suivantes:

1. (QR1) Comment concevoir et mettre en œuvre facilement et de manière efficace des solutions SMAs pour les réseaux intelligents?
2. (QR2) Comment se conformer aux normes dans la phase de conception pour assurer l'interopérabilité?
3. (QR3) Quel style d'architecture SMA est le plus adéquat pour fournir l'équilibre le plus approprié entre les attributs de qualité requis?

Ces préoccupations sont étudiées dans le cadre d'une application qui fournit des scénarios pour répondre aux questions précédentes, et répond également aux questions suivantes, spécifiques aux réseaux intelligents :

- Comment un système résidentiel de réponse à la demande devrait être conçu pour réduire la consommation en période de pointe?
- Quel style d'architecture SMA est le plus adéquat pour ce problème selon les attributs de qualité requis?

Les résultats de la recherche et du développement sur ces sujets de recherche, guidés par les QR1, QR2 et QR3, conduisent aux contributions majeures de cette thèse de doctorat.

La contribution de notre travail consiste à proposer une méthodologie de développement de SMA pour les systèmes énergétiques. Notre méthodologie outillée doit nous permettre de réaliser une analyse non fonctionnelle des systèmes énergétiques dans les premières phases de conception et de sélectionner le style architectural de SMA le plus approprié pour répondre aux exigences du système. Notre approche fournit un langage de modélisation dédié à la conception des SMAs indépendants des plateformes. Ce langage est basé sur le support fourni par le langage UML et étend la sémantique d'un métamodèle existant (i.e., métamodèle PIM4Agents).

Cette thèse propose un cadre architectural conforme à la norme ISO 42010 ISO, 2011, contenant l'ensemble des conventions, principes et pratiques pour la description des architectures multi-agents établies dans le domaine des Smart Grids, comme une solution adéquate pour résoudre les problèmes mentionnés ci-dessus.

Le cadre architectural s'appuie sur l'Ingénierie Dirigé par les Modèles (IDM) Gascueña, Navarro, and Fernández-Caballero, 2012 pour résoudre les problèmes techniques et méthodologiques et réduire la distance entre la spécification et l'implémentation des SMA "Multi-agent solutions for energy systems: A model driven approach". Il cible les plateformes d'agents conformes à la norme FIPA [1], ce qui assure l'interopérabilité entre les systèmes développés en se basant sur le fait que les systèmes existants dans le domaine des Smart Grids sont des systèmes interactifs. De plus, les modèles de données (e.g.,le standard du modèle commun d'information CIM Uslar et al., 2012) définis par la norme SGAM (Smart Grid Architectural Model) CEN-CENELEC-ETSI, 2015 peuvent être adaptés pour modéliser l'ontologie supérieure pour assurer l'interopérabilité

[1] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

au niveau de la sémantique de la communication inter-agents. Cela résout le troisième défi qui est le problème d'interopérabilité. Le cadre se base sur une approche d'évaluation des styles d'architecture multi-agents pour sélectionner le style d'architecture le plus approprié pour répondre à des exigences non fonctionnelles liées à un domaine d'application spécifique. Par ailleurs, un langage de modélisation indépendant des plateformes d'agents a été proposé permettant la modélisation des SMAs et l'analyse des modèles développés pour vérifier leur conformité au style d'architecture SMA sélectionné.

L'approche a été prototypée dans un environnement d'Ingénierie Dirigée par les Modèles et évalué sur un cas d'application représentatif du domaine des Smarts-Grids.

Le cadre architectural se base sur une approche d'évaluation des styles d'architecture multi-agents pour sélectionner le style le plus approprié pour répondre à des exigences non fonctionnel liées à un domaine d'application spécifique.

A.2 Contexte

Cette section détaille le contexte de notre thèse en introduisant les concepts fondamentaux qui sont fortement liés à notre travail.

A.2.1 Les réseaux électriques intelligents

Les réseaux intelligents ou "Smart Grids" sont des réseaux électriques qui, grâce à la technologie informatique, peuvent ajuster le flux d'électricité entre les fournisseurs et les consommateurs. Les Smart Grids sont l'une des dénominations des réseaux de distribution d'électricité intelligents qui utilisent la technologie informatique pour optimiser la production, le transport, la distribution et la consommation d'énergie. Selon l'Institut national des normes et de la technologie (NIST) Fang et al., 2011, les bénéfices anticipés des réseaux intelligents sont :

- Améliorer la fiabilité et la qualité de l'alimentation électrique
- Renforcer la capacité et l'efficacité du réseau électrique existant
- Améliorer la résilience aux perturbations et être auto-réparé
- Élargissement du déploiement des sources d'énergie renouvelables et distribuées
- Maintenance et exploitation automatiques
- Réduction des émissions de gaz à effet de serre
- Réduire la consommation de pétrole
- Permettre la transition vers les véhicules électriques rechargeables
- Augmenter le choix des consommateurs

Un réseau électrique est un réseau inter-connecté qui permet de fournir de l'électricité aux consommateurs par l'intermédiaire des fournisseurs. Il se compose des stations de production d'énergie électrique, des lignes de transmission à haute tension qui transportent l'énergie de sources éloignées vers les centres de consommation, et des lignes de distribution qui relient les clients individuels.

En termes de vision globale et selon le département américain de l'énergie [2], le réseau intelligent (ou Smart Grid) est: intelligent, efficace, accommodant, motivé, opportuniste, focalisé sur la qualité, résilient et écologique. En étudiant les caractéristiques et les définitions des réseaux intelligents, nous avons identifié un certain nombre d'exigences :

- Extensibilité: Les réseaux intelligents nécessitent une extensibilité, c'est-à-dire la possibilité d'être étendus pour satisfaire une demande de charge en augmentation constante.
- Flexibilité: Les réseaux électriques exigent la flexibilité afin de répondre à la demande d'énergie grâce à des sources renouvelables.
- Tolérance aux pannes: Les réseaux intelligents sont connus comme des systèmes résilients composés de diverses opérations numériques, notamment des compteurs intelligents, des appareils intelligents, des points utilisés pour la détection des défauts et la prévision des pannes afin d'éviter les défaillances du système.
- Interopérabilité: Les réseaux intelligents sont composés d'entités de communication hétérogènes et il faut donc aborder la question de l'interopérabilité pour permettre les communications entre eux.

A.2.2 Les Systèmes Multi-Agents

Les SMAs sont utiles pour la conception de systèmes distribués qui nécessitent une autonomie de leurs entités tel que les systèmes énergétiques. Ils offrent une solution intéressante pour le développement de tels systèmes. Un SMA comprend généralement un environnement avec de multiples objets et agents, où les agents représentent des entités actives et intelligentes qui manipulent (contrôlent, perçoivent, modifient, ...) les objets de leur environnement. Un agent est capable de percevoir son environnement et de le manipuler, alors que son comportement tend à atteindre ses objectifs en fonction de ses perceptions Wooldridge and Jennings, 1995. Un agent peut communiquer avec d'autres agents, et il doit être capable de comprendre son environnement et le langage de communication afin de fonctionner convenablement ; c'est-à-dire qu'un agent n'est capable de communiquer que sur des faits exprimés dans une ontologie prédéfinie, qui décrit les concepts du domaine et la relation entre ceux-ci. Cela permet aux agents de comprendre les messages reçus des autres agents. L'interopérabilité entre les SMAs est très importante et les standards de la Fondation pour les agents physiques intelligents' (FIPA [3]) sont utilisés par les SMA McArthur et al., 2007a.

A.2.3 SMA pour les applications dans le domaine de réseaux énergétique intelligents

Comme indiqué précédemment, les réseaux électriques intelligents peuvent être considérés comme des systèmes complexes, alors que la technologie basée sur des agents sont adaptées pour modéliser, contrôler et simuler de manière efficace et fiable des systèmes aussi étendus et hétérogènes. De nombreux travaux prouvent le

[2] US Department of Energy, "The Smart Grid: An Introduction", Available [Online]: https://www.smartgrid.gov/thesmartgrid/smartgrid.html

[3] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", Available [Online]: http://www.fipa.org/

potentiel de la technologie d'agents dans l'ingénierie des systèmes électriques McArthur et al., 2007b; McArthur et al., 2007a; Moradi, Razini, and Hosseinian, 2016 montrant que la technologie SMA est adaptable et applicable dans le domaine de réseaux énergétique intelligents. En outre, l'IEEE recommande l'utilisation de la technologie d'agents pour relever les défis de l'ingénierie des applications pour les réseaux intelligents. Les propriétés des SMAs et des agents peuvent répondre aux exigences du réseau intelligent présentées ci-dessus dans (voir A.2.1). Les propriétés des SMA permettant de répondre aux exigences des réseaux intelligents sont les suivantes:

- **Autonomie/proactivité**: Les agents intelligents autonomes devraient être automatiquement *flexible*, en effet ils sont capables de programmer leurs propres actions afin d'atteindre ses objectifs. La *flexibilité* concerne la réception de nombreuses requêtes et ne peut pas les satisfaire toutes dans un délai raisonnable, à partir duquel il faut décider s'il faut satisfaire la requête et si d'autres actions doivent également être programmées.

  Le cadre des agents fournit les fonctionnalités de messagerie et de découverte de services, permettant aux nouveaux agents de s'intégrer et de communiquer facilement et sans effort. Ainsi, des fonctionnalités supplémentaires peuvent être ajoutées simplement en déployant de nouveaux agents. Cela permet aux systèmes d'être *extensible*.

- **Architectures SMA ouvertes**: Une architecture d'agent ouverte permet une communication flexible entre des agents hétérogènes provenant de différentes plateformes d'agents et implémentés dans différentes langues. La Fondation pour les agents physiques intelligents (FIPA) est l'un des standards les plus connus en termes d'architecture ouverte. L'architecture SMA ouverte n'impose aucune restriction quant au langage de programmation ou à l'origine des agents qui rejoignent le système. La norme FIPA supporte l'*interopérabilité* entre les systèmes à base d'agents développés par les différentes organisations. Le modèle de référence de gestion des agents du FIPA, fournit une architecture ouverte, à laquelle des agents peuvent facilement être ajoutés et retirés. Ainsi, l'architecture SMA ouverte contribue aux systèmes *extensibles*. Une architecture ouverte d'agents ayant une bonne capacité sociale supporte la *Flexibilité*. En effet, lorsqu'un agent ne peut pas remplir une tâche, un autre agent ayant la même capacité peut s'occuper de la tâche en son nom. Par conséquent, cette flexibilité conduit à la conception d'un système *tolérant aux pannes*.

- **Robustesse**: Selon Shehory, 1998, l'un des avantages du SMA est la distribution de l'exécution, qui permet d'augmenter la performance globale. En outre, l'échec d'un agent n'implique pas nécessairement un échec de l'ensemble du système. La robustesse fournie par un SMA est encore accrue par la réplication des capacités.

Généralement, il y a deux possibilités d'utiliser la technologie SMA : (1) comme une approche pour la modélisation et la simulation de systèmes complexes ; (2) comme une approche pour la construction de systèmes robustes, flexibles et extensibles. Ces méthodes sont utilisées pour l'ingénierie de solutions SMAs dédiées aux systèmes énergétiques. Le SMA peut être utilisé comme un système autonome pour contrôler le réseau intelligent au nom de l'être humain. Il s'applique principalement dans les secteurs de contrôle distribué telle que la réponse à la demande.

Le problème de la réponse à la demande appartient au domaine de la gestion de l'énergie. Afin de choisir le style architectural du SMA le plus approprié pour le

problème de la demande, nous pouvons explorer les styles architecturaux des SMAs qui sont recommandés pour être appropriés au domaine de la gestion de l'énergie. Dans la suite, nous présenterons les stratégies de contrôle et les infrastructures de communication possibles pour gérer des entités autonomes dans un micro-réseau pour les problèmes de gestion de l'énergie. L'ensemble des styles architecturaux du SMA possibles pour le domaine d'application de la gestion de l'énergie où les styles architecturaux sont classés en deux aspects (communication et contrôle) sont :

- Structure de communication pour la gestion de l'énergie des micro-réseaux:
  - Architecture centralisée (horizontale)
  - Architecture distribuée
  - Architecture hiérarchique (verticale)
- Stratégies de contrôle intelligent pour la gestion de l'énergie des micro-réseaux:
  - Architecture de contrôle centralisé
  - Architecture de contrôle distribuée
  - Architecture de contrôle hybride

A.2.4 L'ingénierie dirigée par les modèles

Les techniques de l'ingénierie Dirigée par les Modèles (IDM) fournissent des solutions qui améliorent la réutilisabilité, la portabilité et l'interopérabilité des conceptions et des implémentations. Par conséquent, l'application de l'IDM pour le développement des SMAs émerge naturellement. Actuellement, l'utilisation de l'approche de l'ingénierie dirigée par les modèles tout au long du processus de développement de logiciels est de plus en plus populaire Gaševic, Djuric, and Devedžic, 2009. L'IDM permet aux développeurs et aux intervenants d'utiliser des abstractions plus proches du domaine d'intérêt commercial que des concepts informatiques. Ainsi, il réduit la complexité et améliore la communication. Cela peut conduire à une meilleure portabilité et interopérabilité Gascueña, Navarro, and Fernández-Caballero, 2012.

Les auteurs dans Gascueña, Navarro, and Fernández-Caballero, 2012 ont abordé le sujet de l'utilisation des techniques MDE pour résoudre les problèmes d'ingénierie d'un SMA que nous avons résumés ci-dessous:

- Le processus est complexe, coûteux et long, c'est-à-dire que le processus d'ingénierie de SMA nécessite plus de temps pour analyser et concevoir les modèles et pour réaliser le codage.
- Il existe une lacune entre les modèles de conception et les langages d'implémentation existants qui nécessite l'utilisation les techniques de l'IDM qui introduisent des modèles de conception raffinés directement exécutables dans un langage de programmation.
- Intégration et maintenance du système difficiles.

Nous avons présenté dans cette section le contexte général dans lequel s'inscrivent les travaux de recherche de notre thèse. Cette section est essentielle pour comprendre le reste de ce document.

A.3 État de l'art

Cette section présente les travaux connexes qui ont été développés dans la littérature autour de l'ingénierie des SMAs dédiés aux réseaux électriques intelligents.

A.3.1 Normes de base des réseaux électriques intelligents

A.3.1.1 Le modèle d'architecture de réseau intelligent

Le modèle d'architecture de réseau intelligent (SGAM) a été introduit par le groupe de coordination du réseau intelligent en 2012 CEN-CENELEC-ETSI, 2015. Il est centré sur une description structurée d'un système de réseau intelligent distribué afin d'identifier les lacunes de la standardisation. Le cadre du SGAM permet également de valider les cas d'utilisation de réseaux intelligents. Il fournit un ensemble de concepts, de points de vue, comme ainsi qu'une méthode permettant de schématiser les informations sur les cas d'utilisation et donc une une approche structurée pour le développement de l'architecture des réseaux intelligents (Smart Grid).

L'**interopérabilité** est considéré comme le principal élément permettant la mise en place de réseaux intelligents CEN-CENELEC-ETSI, 2015. L'interopérabilité désigne la capacité de deux ou plusieurs réseaux, systèmes, dispositifs, applications ou composants à échanger et à utiliser facilement et efficacement des informations en toute sécurité.

Constitué de cinq niveaux d'interopérabilité, le cadre SGAM permet la représentation des entités et de leurs relations dans le contexte des domaines des réseaux intelligents, des hiérarchies de gestion de l'information et des aspects d'interopérabilité. Les niveaux d'interopérabilité représentent les objectifs et les processus commerciaux, les fonctions, l'échange et les modèles d'information, les protocoles de communication et les composants.

Le niveau d'information décrit les informations qui sont utilisées et échangées entre les fonctions, les services et les composants. Il contient des objets d'information et les modèles de données canoniques sous-jacents (par exemple, le modèle de données unifié IEC CIM 61970). Ces objets d'information et modèles de données canoniques représentent la sémantique commune permettant un échange d'informations interopérable. Ci-après, une description de la norme CIM.

A.3.1.2 Le modèle de données unifié

L'Institut des ingénieurs en électricité et en électronique (The Institute of Electrical and Electronics Engineers (IEEE)) est particulièrement positionné pour guider la normalisation de l'interopérabilité des réseaux intelligents, il existe plus de 100 normes IEEE disponibles ou en cours d'élaboration concernant le réseau intelligent dans divers domaines. La norme IEC CIM est utilisée dans le domaine de l'électricité, couvrant le transport, la distribution, les marchés, la production et les processus commerciaux connexes. L'idée clé de la norme CIM est de définir un langage commun afin de permettre à la fois : l'échange de données entre différentes entreprises, et l'échange de données entre les applications des entreprises. Les modules de base du CIM sont définis dans la norme CIM 61970-301, qui définit les composants du système électrique en utilisant le langage UML (Unified Modeling Language) Uslar et al., 2012. Le niveau d'information de la norme SGAM définit les objets d'information à échanger au sein des systèmes de réseaux intelligents ainsi que les modèles de données canoniques associés (par exemple, le CIM). Ainsi, le

groupe de travail IEEE PES MAS [4] a annoncé qu'une ontologie basée sur le CIM peut être proposée pour être considérée comme une ontologie supérieure pour les solutions SMAs de gestion de l'énergie afin d'assurer l'interopérabilité entre les agents de différents systèmes d'agents McArthur et al., 2007b; McArthur et al., 2007a.

A.3.2 Étude des méthodologies et des cadres d'ingénierie des SMA

L'approche de développement des SMA basé sur l'IDM est au centre de plusieurs approches Amor, Fuentes, and Vallecillo, 2004; Pavón, Gómez-Sanz, and Fuentes, 2006. Pavón, Gómez-Sanz, and Fuentes, 2006 a introduit la plateforme de développement INGENIAS, qui est un ensemble d'outils pour la modélisation, la vérification et la transformation de modèles d'agents. Il fournit des outils de développement dirigé par le modèle pour le développement de SMA basés sur le métamodèle INGENIAS. Certains auteurs ont essayé de proposer un métamodèle standardisé pour le développement de SMA Beydoun et al., 2009; Bernon et al., 2004. Beydoun et al., 2009 a proposé le modèle dénommé FAML, qui vise à résoudre les problèmes d'interopérabilité entre les méthodologies orientées agent. Hahn, Madrigal-Mora, and Fischer, 2009 a proposé un métamodèle indépendant de la plateforme dénommé PIM4Agents, qui est conçu pour contribuer à l'interopérabilité entre les architectures spécifiques aux domaines et les plates-formes d'agents. Le métamodèle proposé fournit le langage de base à utiliser dans un processus de développement d'un SMA, qui est conforme aux principes de l'IDM. On constate une large utilisation des protocoles d'interaction par les méthodologie proposées orientées agent. Par exemple, PASSI Cossentino and Potts, 2002 utilise les protocoles d'interaction de FIPA [5] pour la spécification de l'interaction des agents.

Parmi les solutions présentées ci-dessus, certaines résolvent les problèmes méthodologiques du processus de développement des SMA, tandis que d'autres traitent les problèmes méthodologiques et techniques en suivant les perspectives de développement dirigées par les modèles. La plupart de ces solutions visent tous les domaines d'application et aucune d'entre elles n'est spécifique à l'ingénierie des systèmes énergétiques où les normes énergétiques doivent être respectées pour restreindre les choix de conception.

A.3.3 Étude des approches d'ingénierie des SMA pour les réseaux électriques intelligents

Plusieurs études ont été réalisées sur l'application du SMA pour la modélisation des systèmes énergétiques McArthur et al., 2007b; Kremers, 2013. Les solutions basées sur agents proposent généralement des solutions spécifiques; c'est-à-dire qu'elles ont été créées pour résoudre des problèmes spécifiques qui n'étaient pas destinés à être réutilisés Kremers, 2013. En outre, ces solutions ont été conçues pour fonctionner sur une plate-forme d'agents spécifique, sans prendre en considération l'interopérabilité entre les différents systèmes d'agents, ni avec d'autres technologies. Plusieurs études ont été réalisées sur l'application de la technologie d'agent pour les systèmes énergétiques McArthur et al., 2007b; Kremers, 2013. Une approche SMA a été proposée pour le diagnostic des perturbations du système énergétique Hossack et al., 2002

[4] IEEE PES Multi-Agent Systems Working Group, Available [Online] : https://site.ieee.org/pes-mas/

[5] FIPA 2007, "Foundation for Intelligent Physical Agents (FIPA)", disponible [en ligne] : http://www.fipa.org/

où les auteurs ont suivi leur propre méthodologie. Cette méthodologie est brièvement décrite dans McArthur, McDonald, and Hossack, 2003 et a été recommandée pour développer des solutions SMA dans le secteur de l'énergie McArthur et al., 2007a. Récemment, Constantin et al., 2017 a présenté une méthodologie orientée agent pour les bâtiments non résidentiels, en fournissant une ontologie et un modèle de données pour l'application aux bâtiments non résidentiels, tous les deux ont été développés sur la base des normes existantes et qui sont réutilisables et extensibles.

A.3.4 Synthèse

À notre connaissance, il n'existe aucune proposition d'approche méthodologique orientée agent et basée sur les techniques de l'IDM dédiée aux systèmes énergétiques qui réponde aux exigences des Smart Grids (c'est-à-dire l'interopérabilité entre les systèmes d'agents et entre les agents) et qui tienne compte des exigences non fonctionnelles du domaine d'application cible pour améliorer la qualité de service de la solution SMA développée.

Dans cette thèse, nous relevons les quatre défis présentés dans l'introduction en proposant un cadre architectural pour le développement des SMAs, conforme à la norme ISO 42010 ISO, 2011, pour développer des solutions SMAs dédiées aux aux systèmes énergétiques intelligents (Smart Grids). Ce travail contribue de la manière suivante : il définit une **méthodologie**, pour soutenir l'utilisation du cadre architectural MAS4SG afin de résoudre les problèmes méthodologiques (i) ; basé sur un méta-modèle indépendant de la plate-forme appelé ML4Agents pour résoudre les problèmes techniques (ii) ; le cadre proposé adhère à la norme FIPA et à la norme CIM pour résoudre les problèmes d'interopérabilité (iii) ; et pour résoudre les problèmes d'évaluation des styles d'architecture SMA (iv) il permet, avec les concepts du ML4Agents, de modéliser les styles d'architecture SMA qui seront évalués pour sélectionner le style d'architecture le plus approprié qui répond le mieux aux exigences non fonctionnelles modélisées.

On a comparé certains des travaux présentés. Nous avons a défini six critères fondamentaux sur les capacités et les possessions des approches proposées, qui sont énumérés ci-dessous :

- Spécifique au Smart Grids: solution dédiée au développement des SMAs dans le domaine des systèmes énergétiques
- Documenté: Documentations et guides explicatifs
- Interopérabilité au niveau de la communication: utilisation de protocoles d'interaction normalisés pour la communication
- Interopérabilité au niveau de l'application: utilisation d'une ontologie standardisée ou d'une ontologie basée sur une norme pour la spécification des connaissances échangées
- Qualité de service (Quality Of Service QoS): proposer un mécanisme pour répondre aux besoins non fonctionnels (c'est-à-dire la qualité de service)
- Portabilité: fourniture d'un métamodèle pour la métamodélisation indépendante de la plate-forme.

Seulement parmi les travaux évalués, notre proposition qui répond à tous les critères mentionnés ci-dessus.

A.4 Un Cadre Architectural pour la Conception des SMAs dédiés aux Réseau Intelligent

Cette section présente un cadre architectural, conforme à la norme ISO 42010 ISO, 2011, comme solution aux défis rencontrés dans l'ingénierie des solutions multi-agents dans le secteur de l'énergie.

A.4.1 Conformité aux normes

Si l'application de la technologie d'agent doit être largement appliquée dans le domaine de l'ingénierie énergétique, il est indispensable d'adopter des normes qui favorisent l'interopérabilité entre les différents systèmes. Le cadre architectural proposé est conforme aux normes existantes pour surmonter le problème de l'interopérabilité entre différents systèmes multi-agents qui se présente à deux niveaux différents.

Tout d'abord, l'interopérabilité entre les agents de différents concepteurs, c'est-à-dire leur capacité à se découvrir les uns les autres pour maintenir des communications, quel que soit la plateforme sur laquelle ils se trouvent. Deuxièmement, au niveau sémantique de la communication, l'interopérabilité doit être assurée entre les agents, qu'ils appartiennent aux mêmes plates-formes ou non, en disposant du même dictionnaire de données pour se comprendre.

La norme FIPA a été adoptée pour résoudre les problèmes d'interopérabilité au niveau de la communication car elle favorise l'interopérabilité entre des agents interactifs hétérogènes et des systèmes à base d'agents développés par les différentes organisations. Pour assurer l'interopérabilité entre les agents sur différentes plateformes au niveau sémantique de la communication, nous adoptons la norme CIM Uslar et al., 2012 pour construire l'ontologie utilisée par les agents dans la communication. Ceci est conforme à la recommandation de l'IEEE McArthur et al., 2007a d'utiliser une ontologie CIM de haut niveau.

A.4.2 La spécification du cadre MAS4SG

A.4.2.1 Les intervenants du cadre MAS4SG

Nous considérons les intervenants suivants pour le développement des SMAs dans le domaine de l'ingénierie énergétique:

- *Expert du domaine de l'énergie* : Les experts de domaine ont les connaissances concernant le domaine d'application qui est le système énergétique. Un expert du domaine de l'énergie est responsable de : (i) la spécification des exigences fonctionnelles et non fonctionnelles ; (ii) la conception du modèle de données de l'agent, et (iii) la conception de l'ontologie pour le domaine d'application.

- *L'architecte du système d'agents* : Les architectes des systèmes d'agents sont responsables de la création d'une bibliothèque de styles d'architecture SMA largement appliqués dans le domaine des réseaux énergétique intelligents, à évaluer pour sélectionner le style architectural qui répond mieux aux exigences non fonctionnelles du système.

- *Experts du protocole* : Les experts en protocole sont chargés de spécifier les protocoles de communication et de négociation. Ils offrent une bibliothèque de protocoles d'interaction FIPA pour faciliter la modélisation de l'interaction au sein d'une application SMA.

- *Ingénieur des agents* : L'ingénieur des agents est l'utilisateur final de l'environnement de développement. Il utilise une méthodologie d'agent pour développer l'application basée sur agents.
- *Programmeur:* Le programmeur est responsable de l'application des techniques de transformation pour générer un code exécutable sur une plate-forme d'agent spécifique. Il est chargé d'affiner le code généré si nécessaire.

A.4.2.2 Les différents vues du cadre MAS4SG

Les systèmes multi-agents peuvent être observés sous plusieurs aspect et de nombreux vues possibles peuvent donc être proposés. Comme pour les vues du cadre architectural SGAM, nous considérons les vues suivants:

- *Point de vue commercial* : cette vue connote la description de l'exigence du domaine. Elle précise les exigences du système et se concentre sur le contexte du système sans tenir compte de sa structure ou de son traitement.
- *Point de vue analyse*: cette vue traite de l'analyse détaillée des exigences et identifie les rôles et les interactions nécessaires pour atteindre l'objectif global. Elle considère également deux concepts principaux; la conception de l'ontologie et les connaissances nécessaires pour répondre aux exigences définies par la vue commercial.
- *Point de vue conception*: cette vue se concentre sur l'identification des types d'agents et des fonctions requises pour remplir leur rôle, qui est identifié par les communications inter-agents.
- *Point de vue déploiement*: cette vue identifie les cas concrets d'agents et d'organisations définis par l'analyse et les vues de conception.
- *Point de vue implémentation*: les transformations du modèle sont utilisées pour générer un modèle de conception concret dépendant de la plate-forme multi-agents et des raffinements manuels sont appliqués si nécessaire.

Afin de pouvoir décrire ces points de vue, un langage de modélisation indépendant de la plate-forme d'agent doit être proposé pour la modélisation du SMA.

A.4.2.3 Le langage de modélisation ML4Agents

Le langage de modélisation ML4Agents (Modeling Language For Agents) proposé est basé sur le métamodèle PIM4Agents. Le métamodèle ML4Agents qui définit la sémantique de notre langage de modélisation comprend (i) les concepts du métamodèle PIM4Agent qui manipulent de nombreux concepts communs aux agents; (ii) les concepts des contraintes FIPA pour être adaptés à la plate-forme d'agents conforme au FIPA, et (iii) les concepts pour la description du style d'architecture SMA. Le langage de modélisation envisagé supporte la conception du modèle indépendant de la plateforme d'agents. Nous avons considérablement étendu le métamodèle PIM4Agents avec des éléments de modélisation spécifiques qui permettent la conception des spécifications du FIPA. Nous avons défini un métamodèle qui comprend des concepts spécifiques pour modéliser l'ontologie et les exigences. En particulier, nous considérons que les spécifications du FIPA décrivent le protocole d'interaction des agents et le langage de communication des agents. Le métamodèle

ML4Agents définit les principaux concepts et relations utilisés pour concevoir des SMAs exécutables sur une plate-forme d'agent conforme au FIPA. Nos choix pour adapter et étendre le concept de PIM4Agents sont motivés par la nécessité d'intégrer les propriétés spécifiques des systèmes énergétiques et de répondre à ses exigences.

A.5 Approche Méthodologique

Cette section présente une méthodologie dirigée par les modèles pour supporter l'utilisation du cadre architectural proposé y compris les cinq phases: (1) la phase de spécification des exigences, (2) la phase d'analyse, (3) la phase de conception, (4) la phase de déploiement et (5) la phase d'implémentation.

A.5.1 Pré-requis de la méthodologie

Une méthodologie complète pour l'ingénierie des systèmes énergétiques intelligents implique de nombreuses disciplines. Tout d'abord, la méthodologie doit pouvoir répondre aux besoins particuliers des intervenants concernés, tels que les analystes commerciaux, les experts du domaine et les ingénieurs des agents. La méthodologie permet de réutiliser les travaux existants (ou une partie de ceux-ci) dans le domaine des réseaux intelligents et de modéliser des SMAs pour résoudre les problèmes des systèmes énergétiques plutôt que de repartir de zéro.

De plus, le processus de développement doit inclure la tâche de modélisation des exigences. La disponibilité d'un modèle d'exigences doit fournir une base pour l'évaluation des styles architecturaux. Il est donc recommandé d'inclure également la possibilité d'évaluer les styles architecturaux. Le processus de développement doit inclure aussi la tâche de modélisation de l'ontologie.

Finalement, pour un besoin d'interopérabilité entre les systèmes d'agents développés, (1) la conception de l'ontologie devrait être basée sur la norme énergétique existante IEC CIM, et (2) la conception de l'interaction devrait réutiliser la spécification des protocoles FIPA.

A.5.2 Vue globale de la méthodologie

La méthodologie proposée commence par la *phase de spécification des exigences*, qui aboutit au modèle d'exigences. Cette phase vise à définir les exigences fonctionnelles et non fonctionnelles d'un système particulier et implique généralement des experts de domaine.

La *phase d'analyse* implique généralement des architectes de systèmes d'agents et des analystes de systèmes (c'est-à-dire des modélisateurs des interactions, d'environnement et du domaine) et se concentre sur les interactions impliquées dans le système multiagent et sur les ressources de l'environnement (comme les objets et l'ontologie).

Dans la *phase de conception*, les ingénieurs des agents (tels que les modélisateurs de rôle, les modélisateurs d'organisation, ...) poursuivent la spécification des architectures SMA qui est composée de deux activités : la conception structurelle et la conception comportementale. Les rôles, les organisations et les agents sont conçus dans la conception structurelle du SMA et les comportements et les collaborations sont conçus et affinés dans la conception comportementale du SMA.

Dans la *phase de déploiement*, si les instances des agents en exécution sont connues, alors le modèle de déploiement doit être conçu avant de commencer la phase finale du processus qui est la phase d'implémentation, sinon on peut passer directement à la phase d'implémentation et faire le déploiement sur une des plate-formes

d'exécution. Dans la *phase d'implémentation*, le programmeur peut exécuter les transformations du modèle indépendant de la plate-forme en un modèle spécifique à la plate-forme d'agent et la transformation de ce modèle en code. Le code généré peut enfin être affiné dans le langage de programmation d'agent considéré.

Dans cette section, nous avons présenté une méthodologie pour appuyer l'utilisation du cadre MAS4SG pour analyser, concevoir et implémenter des SMAs pour les systèmes énergétiques. Elle se base sur les techniques de l'IDM afin de guider les ingénieurs en énergie dans le développement des solutions multiagent, et de simplifier la tâche de conception en réutilisant les modèles. La méthodologie adhère aux normes énergétiques. Elle permet d'évaluer et de choisir le style architectural le plus approprié pour le système d'agents développé afin de répondre à ses exigences non fonctionnelles et contribue à l'amélioration des performances du système en termes de qualité de service.

A.6 Implémentions

La section précédente a présenté le métamodèle ML4Agents et sa sémantique. L'implémentation de ce langage de modélisation dans un outil UML nécessite la définition d'un profil UML. Cette section explique comment le métamodèle ML4Agents est implémenté.

A.6.1 Langage spécifique au domaine (ML4Agents)

Lorsque nous devons définir un nouveau langage spécifique à un domaine qui restreint le nombre de concepts UML (métaclasses) ou leur ajoute certaines contraintes, tout en respectant la sémantique originale, nous n'avons pas besoin de créer un nouveau langage à partir de zéro en utilisant le langage MOF. Par contre, UML peut être facilement personnalisé grâce au mécanisme d'extension de profil. Dans la suite, nous présentons un extrait du profil UML qui implémente le métamodèle ML4Agents. L'extension UML pour la modélisation des agents est présentée dans le tableau A.1.

A.6.2 Librairie des protocoles FIPA

Le cadre MAS4SG est conforme à la norme FIPA, nous avons donc développé une bibliothèque de spécifications de protocoles FIPA à réutiliser pour la spécification des interactions. La norme FIPA est utilisée pour normaliser la communication entre les agents. La norme propose pour chaque protocole une séquence de messages envoyés et reçus et leur acte communicatif associé. La bibliothèque de protocoles proposée fournit une spécification UML pour les protocoles d'interaction du FIPA. Le concepteur d'un SMA peut l'utiliser pour concevoir la collaboration entre les rôles de domaine au sein d'une organisation et peut utiliser la spécification du comportement des rôles d'interaction pour concevoir le comportement interne des agents participant à une interaction.

Nous proposons une approche à trois niveaux pour la spécification UML de tout protocole FIPA.

- La vue de l'interaction : ce niveau représente l'interaction globale entre les agents définie par un protocole. Nous utilisons le diagramme de séquence UML pour créer cette vue.

ML4Agents Stereotype	**UML Base Class**
MultiAgentSystem	Model
Agent	Class
DomainRole	Component
ACLMessage	Message
Organization	Collaboration
Interaction	Interaction
Actor	Lifeline
Service	Class
Capability	Interface
Knowledge	Property
Initiator	Interface
Participant	Interface
Collaboration	CollaborationUse
Plan	StateMachine
Task	BehavioralFeature
AgentAction	BehavioralFeature
Environment	Package
Object	Class
Message	Signal
Ontology	Package
Protocol	Collaboration
Concept	Class
Predicat	Class
Action	Class
NonFunctionalRequirement	Class
FunctionalRequirement	Class
ArchitecturelStyle	Model, Collaboration

TABLE A.1: UML extensions for agent modeling

- La vue du service : ce niveau représente le service pour lequel le protocole est établi et les différents rôles d'interaction qui fournissent et demandent ce service. Pour ce niveau, nous utilisons le diagramme de classes UML.
- La vue comportementale : ce niveau représente le traitement interne de l'agent. Il donne une spécification pour le comportement de chaque rôle d'interaction. Ce comportement est spécifié avec un automate fini de protocole qui définit, quand et à quelles conditions, les comportementales individuelles (fonctionnement de l'envoi et de la réception de messages) peuvent être invoquées. Cela facilite la conception du comportement interne de l'agent. Pour ce niveau, nous utilisons le diagramme de la machine à états UML.

La bibliothèque de protocoles FIPA peut alors être utilisée pour la modélisation de n'importe quelle application SMA.

A.6.3 Librairie des profils IEC CIM

Cette bibliothèque fournit le modèle de données CIM (spécification UML) des différentes normes CIM telles que (IEC61970, IEC61968 et IEC62325) et fournit un modèle de données CIM restreint (profils CIM) classé par domaines d'application tels que l'application de modélisation et de simulation, l'application de surveillance et de diagnostic, l'application de contrôle distribué et l'application de protection.

Dans cette section, nous avons présenté une implémentation du cadre architectural MAS4SG dédié aux systèmes énergétiques. Ces implémentations ont été mises en place pour supporter notre méthodologie pour l'ingénierie des SMAs pour résoudre les problèmes dans le secteur des systèmes énergétiques intelligents.

A.7 Conclusion

A.7.1 Évaluation

Dans cette thèse, nous avons proposé un cadre architectural appelé MAS4SG pour le développement de solutions SMA pour gérer les systèmes énergétiques intelligent en appliquant les recommandations de IEEE pour l'utilisation des normes énergétiques et d'autres proposés pour les systèmes à base d'agents. MAS4SG aide les ingénieurs spécialisés dans l'énergie à adopter la technologie d'agent pour développer des solutions SMAs pour résoudre les problèmes des réseaux intelligents où nous avons essayé de répondre aux exigences majeures de l'ingénierie des systèmes énergétiques: **Interopérabilité** et **Exploration architecturale**. La construction des points de vue du cadre MAS4SG révèle la nécessité d'un langage de modélisation. Ainsi, nous avons proposé une extension pour le métamodèle PIM4Agents existant. Nous avons présenté une méthodologie pour soutenir l'utilisation du cadre MAS4SG pour analyser, concevoir et développer des SMAs pour les systèmes énergétiques. Elle suit le processus MDE afin de guider les développeurs dans la conception et la mise en œuvre des applications d'agents, et de simplifier la tâche de conception en réutilisant les modèles. Notre méthodologie est basée sur la norme CIM et est divisée en cinq phases: la spécification des exigences, analyse, conception, déploiement et d'implémentation. La méthodologie a été présentée et validée par le développement d'une solution d'optimisation SMA pour une application bien connue.

A.7.2 Perspectives

Considérant le problème de l'exploration architecturale, l'évaluation des styles d'architectures SMA possibles peut être effectuée par un algorithme qui sélectionne le style architectural SMA le plus approprié pour l'application considérée. Le style d'architecture sélectionné peut ensuite être appliqué sur le modèle de l'application au moyen d'un outil qui donne comme résultat une première proposition de modèle d'application avec une architecture qui correspond au style architectural déjà sélectionné.

De plus, un ensemble de styles d'architecture SMA possibles peut être proposé et classé en fonction du domaine d'application dans le domaine du réseau intelligent. Pour chaque style d'architecture SMA, un ensemble d'attributs de qualité satisfaits par chaque style d'architecture peut être fixé.

La mise en œuvre de diagrammes spécifiques, de transformations de modèles et de génération de code peut être développée afin de donner un outil de développement SMA qui couvre toutes les phases de développement d'un SMA.

Dans cette thèse, notre idée de base était de proposer un cadre d'architecture SMA spécifique au domaine d'application des systèmes énergétiques. Cependant, nous avons fini par proposer un cadre d'architecture spécifique au domaine de l'ingénierie des systèmes d'agents, adapté pour être approprié à l'ingénierie des applications SMA dédiées au secteur des systèmes énergétiques. Ainsi, nous concluons que le cadre proposé, le langage ML4Agents qu'il supporte et la méthodologie qui lui est associée sont flexibles et pourraient être utilisés pour d'autres domaines d'application

que celui des systèmes énergétiques intelligents en respectant les normes liés au domaine d'application cible pour répondre à ses exigences.

Bibliography

Agents, JACK Intelligent (2006). "The Agent Oriented Software Group (AOS)(2006)". In: AGENTS.

Amor, Mercedes, Lidia Fuentes, and Antonio Vallecillo (2004). "Bridging the gap between agent-oriented design and implementation using MDA". In: *International Workshop on Agent-Oriented Software Engineering*. Springer, pp. 93–108.

Arnold, George W et al. (2010). *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0*. Tech. rep.

Bellifemine, Fabio, Agostino Poggi, and Giovanni Rimassa (1999). "JADE–A FIPA-compliant agent framework". In: *Proceedings of PAAM*. Vol. 99. 97-108. London, p. 33.

Bernon, Carole et al. (2002). "ADELFE: a methodology for adaptive multi-agent systems engineering". In: *International Workshop on Engineering Societies in the Agents World*. Springer, pp. 156–169.

Bernon, Carole et al. (2004). "A study of some multi-agent meta-models". In: *International Workshop on Agent-Oriented Software Engineering*. Springer, pp. 62–77.

Beydoun, Ghassan et al. (2009). "FAML: a generic metamodel for MAS development". In: *IEEE Transactions on Software Engineering* 35.6, pp. 841–863.

Bresciani, Paolo et al. (2004). "Tropos: An agent-oriented software development methodology". In: *Autonomous Agents and Multi-Agent Systems* 8.3, pp. 203–236.

CEN-CENELEC-ETSI, Smart Grid Coordination (2015). "Group: Smart grid reference architecture (November 2012)". In: URL: https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf.

Chella, Antonio et al. (2006). "Agile passi: An agile process for designing agents". In: *International Journal of Computer Systems Science & Engineering* 21.2, pp. 133–144.

Constantin, Ana et al. (2017). "Design, Implementation and Demonstration of Embedded Agents for Energy Management in Non-Residential Buildings". In: 10, p. 1106.

Cossentino, Massimo and Colin Potts (2002). "A CASE tool supported methodology for the design of multi-agent systems". In: *International Conference on Software Engineering Research and Practice (SERP'02)*.

Davidsson, Paul, Stefan Johansson, and Mikael Svahnberg (2005). "Characterization and evaluation of multi-agent system architectural styles". In: *International Workshop on Software Engineering for Large-scale Multi-agent Systems*. Springer, pp. 179–188.

Dehghanpour, Kaveh, Christopher Colson, and Hashem Nehrir (2017). "A survey on smart agent-based microgrids for resilient/self-healing grids". In: *Energies* 10.5, p. 620.

Fang, Xi et al. (2011). "Smart grid—The new and improved power grid: A survey". In: *IEEE communications surveys & tutorials* 14.4, pp. 944–980.

Felix Bachmann, Mark Klein (2003). "Understanding Quality Attributes". In: *Software architecture in practice*. Ed. by Len Bass, Paul Clements, and Rick Kazman. Addison-Wesley Professional. Chap. 4.

Fischer, Klaus and Stefan Warwas (2012). "A methodological approach to model driven design of multiagent systems". In: *International Workshop on Agent-Oriented Software Engineering*. Springer, pp. 1–21.

Fuentes-Fernández, Lidia and Antonio Vallecillo-Moreno (2004). "An introduction to UML profiles". In: *UML and Model Engineering* 2, pp. 6–13.

Gascueña, José M, Elena Navarro, and Antonio Fernández-Caballero (2012). "Model-driven engineering techniques for the development of multi-agent systems". In: *Engineering Applications of Artificial Intelligence* 25.1, pp. 159–173.

Gaševic, Dragan, Dragan Djuric, and Vladan Devedžic (2009). *Model driven engineering and ontology development*. Springer Science & Business Media.

Gérard, Sébastien et al. (2010). "19 Papyrus: A UML2 tool for domain-specific language modeling". In: *Model-Based Engineering of Embedded Real-Time Systems*. Springer, pp. 361–368.

Ghribi, Khawla et al. (2014). "A survey on multi-agent management approaches in the context of intelligent energy systems". In: *Electrical Sciences and Technologies in Maghreb (CISTEM), 2014 International Conference on*. IEEE, pp. 1–8.

Gungor, Vehbi C et al. (2011). "Smart grid technologies: Communication technologies and standards". In: *IEEE transactions on Industrial informatics* 7.4, pp. 529–539.

Hahn, Christian (2008). "A domain specific modeling language for multiagent systems". In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 233–240.

Hahn, Christian, Cristián Madrigal-Mora, and Klaus Fischer (2009). "A platform-independent metamodel for multiagent systems". In: *Autonomous Agents and Multi-Agent Systems* 18.2, pp. 239–266.

Hahn, Christian Steven (2013). "A PLATFORM-INDEPENDENT DOMAIN-SPECIFIC MODELING LANGUAGE FOR MULTIAGENT SYSTEMS". PhD thesis. Universität des Saarlandes.

Herbst, Andrea et al. (2012). "Introduction to energy systems modelling". In: *Swiss journal of economics and statistics* 148.2, pp. 111–135.

Hernandez, L. et al. (2013). "A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants". In: *IEEE Communications Magazine* 51.1, pp. 106–113. DOI: 10.1109/MCOM.2013.6400446.

Hossack, J et al. (2002). "A multi-agent approach to power system disturbance diagnosis". In:

ISO, IEC (2011). "IEEE: ISO/IEC/IEEE 42010: 2011-Systems and software engineering–Architecture description". In: *Proceedings of Technical Report*.

Jarraya, Tarek and Zahia Guessoum (2007). "Towards a model driven process for multi-agent system". In: *International Central and Eastern European Conference on Multi-Agent Systems*. Springer, pp. 256–265.

Jayasinghe, Sarinda Lahiru and Kullappu Thantrige Manjula Udayanga Hemapala (2015). "Multi Agent Based Power Distribution System Restoration—A Literature Survey". In: *Energy and Power Engineering* 7.12, p. 557.

Kantamneni, Abhilash et al. (2015). "Survey of multi-agent systems for microgrid control". In: *Engineering applications of artificial intelligence* 45, pp. 192–203.

Katiraei, F. et al. (2008a). "Microgrids Management". In: *Power and Energy Magazine, IEEE* 6, pp. 54 –65. DOI: 10.1109/MPE.2008.918702.

Katiraei, Farid et al. (2008b). "Microgrids management". In: *IEEE power and energy magazine* 6.3, pp. 54–65.

Kleppe, Anneke G et al. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.

Koritarov, Vladimir S (2004). "Real-world market representation with agents". In: *IEEE Power and Energy Magazine* 2.4, pp. 39–46.

Kremers, Enrique Alberto (2013). *Modelling and simulation of electrical energy systems through a complex systems approach using agent-based models*. KIT Scientific Publishing.

Lasseter, Robert H (2011). "Smart distribution: Coupled microgrids". In: *Proceedings of the IEEE* 99.6, pp. 1074–1082.

Logenthiran, Thillainathan, Dipti Srinivasan, and Tan Zong Shun (2012). "Demand side management in smart grid using heuristic optimization". In: *IEEE transactions on smart grid* 3.3, pp. 1244–1252.

McArthur, Stephen DJ, James R McDonald, and John Hossack (2003). "A multi-agent approach to power system disturbance diagnosis". In: *Autonomous Systems and Intelligent Agents in Power System Control and Operation*. Springer, pp. 75–100.

McArthur, Stephen DJ et al. (2007a). "Multi-agent systems for power engineering applications-part 2: technologies, standards and tools for building multi-agent systems". In: *IEEE Transactions on Power Systems* 22.4, pp. 1753–1759.

— (2007b). "Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges". In: *IEEE Transactions on Power systems* 22.4, pp. 1743–1752.

Moaven, S. et al. (2008). "A Fuzzy Model for Solving Architecture Styles Selection Multi-Criteria Problem". In: *2008 Second UKSIM European Symposium on Computer Modeling and Simulation*, pp. 388–393. DOI: 10.1109/EMS.2008.45.

Moradi, Mohammad H, Saleh Razini, and S Mahdi Hosseinian (2016). "State of art of multiagent systems in power engineering: A review". In: *Renewable and Sustainable Energy Reviews* 58, pp. 814–824.

OMG, SPEM and O Notation (2008). "Software & systems process engineering metamodel specification". In: *OMG Std., Rev* 2, pp. 18–71.

Palensky, Peter and Dietmar Dietrich (2011). "Demand side management: Demand response, intelligent energy systems, and smart loads". In: *IEEE transactions on industrial informatics* 7.3, pp. 381–388.

Pavón, Juan, Jorge Gómez-Sanz, and Rubén Fuentes (2006). "Model driven development of multi-agent systems". In: *European Conference on Model Driven Architecture-Foundations and Applications*. Springer, pp. 284–298.

Pipattanasomporn, Manisa, Hassan Feroze, and Saifur Rahman (2009). "Multi-agent systems in a distributed smart grid: Design and implementation". In: *Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES*, pp. 1–8.

Rohbogner, Gregor et al. (2012). "What the term Agent stands for in the Smart Grid Definition of Agents and Multi-Agent Systems from an Engineer's Perspective". In: *Computer Science and Information Systems (FedCSIS), 2012 Federated Conference on*. IEEE, pp. 1301–1305.

Romdhane, L. Ben et al. "Multi-agent solutions for energy systems: A model driven approach". In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Schmidt, Douglas C (2006). "Model-driven engineering". In: *COMPUTER-IEEE COMPUTER SOCIETY-* 39.2, p. 25.

Shehory, Onn M (1998). *Architectural properties of multi-agent systems*. Carnegie Mellon University, The Robotics Institute.

Silva, Carla TLL, Jaelson Castro, and Patricia Azevedo Tedesco (2003). "Requirements for Multi-Agent Systems." In: *WER* 2003, pp. 198–212.

Silva, Viviane Torres da and Carlos JP de Lucena (2007). "Modeling multi-agent systems". In: *Communications of the ACM* 50.5, pp. 103–108.

Sturm, Arnon and Onn Shehory (2014). "Agent-oriented software engineering: revisiting the state of the art". In: *Agent-Oriented Software Engineering*. Springer, pp. 13–26.

Trencansky, Ivan and Radovan Cervenka (2005). "Agent Modeling Language (AML): A comprehensive approach to modeling MAS". In: *Informatica* 29.4.

Tsikalakis, Antonis G et al. (2006). "Management of microgrids in market environment". In:

Uslar, Mathias et al. (2012). *The Common Information Model CIM: IEC 61968/61970 and 62325-A practical introduction to the CIM*. Springer Science & Business Media.

Wang, Qiushi and Zhao Yang (2012). "A method of selecting appropriate software architecture styles: Quality Attributes and Analytic Hierarchy Process". In:

Warwas, Stefan et al. (2012). "BOCHICA: A Model-driven Framework for Engineering Multiagent Systems." In: *ICAART (1)*, pp. 109–118.

Wooldridge, Michael (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Wooldridge, Michael and Nicholas R Jennings (1995). "Intelligent agents: Theory and practice". In: *The knowledge engineering review* 10.2, pp. 115–152.

Wooldridge, Michael, Nicholas R Jennings, and David Kinny (2000). "The Gaia methodology for agent-oriented analysis and design". In: *Autonomous Agents and multi-agent systems* 3.3, pp. 285–312.

Xiao, Zhe et al. (2010). "Hierarchical MAS based control strategy for microgrid". In: *Energies* 3.9, pp. 1622–1638.

Yoldaş, Yeliz et al. (2017). "Enhancing smart grid with microgrids: Challenges and opportunities". In: *Renewable and Sustainable Energy Reviews* 72, pp. 205–214.

Yoo, Cheol-Hee et al. (2013). "Intelligent control of battery energy storage for multi-agent based microgrid energy management". In: *Energies* 6.10, pp. 4956–4979.