

MODELING OF DYNAMIC PRICING OF ENERGY FOR A SMART GRID USING A  
MULTI-AGENT FRAMEWORK

A Paper  
Submitted to the Graduate Faculty  
of the  
North Dakota State University  
of Agriculture and Applied Science

By

Harvinder Kaur

In Partial Fulfillment of the Requirements  
for the Degree of  
MASTER OF SCIENCE

Major Department:  
Computer Science

April 2011

Fargo, North Dakota

North Dakota State University  
Graduate School

---

Title

**MODELING OF DYNAMIC PRICING OF ENERGY**

---

**FOR A SMART GRID USING A MULTI-AGENT FRAMEWORK**

---

By

**HARVINDER KAUR**

---

The Supervisory Committee certifies that this *disquisition* complies with North Dakota State University's regulations and meets the accepted standards for the degree of

**MASTER OF SCIENCE**

---

North Dakota State University Libraries Addendum

To protect the privacy of individuals associated with the document, signatures have been removed from the digital version of this document.

## ABSTRACT

Kaur, Harvinder, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, April 2011. Modeling of Dynamic Pricing of Energy for a Smart Grid Using a Multi-Agent Framework. Major Professor: Dr. Kendall E. Nygard.

The use of smart grids is being promoted to address issues such as energy independence, global warming and emergency resilience. A smart grid is a digitized form of the power grid and is comprised of an intelligent monitoring system that keeps track of the two-way digital communications in the system. A multi-agent system is a collection of interacting intelligent agents that can be used in problem solving for systems that are difficult or impossible to be solved by an individual agent. Applications of multi-agent systems can range from transportation, logistics, graphics, networking and mobile technologies to modeling real world scenarios to achieve automatic and dynamic load-balancing, pricing, and disaster response.

The goal of this project was to design and implement a multi-agent system to model dynamic pricing of electricity in a smart grid, thereby improving the overall efficiency of electricity consumption in a real world scenario. This project was accomplished by devising and implementing a multi-agent system for regulating automatic and dynamic pricing of electricity by monitoring power consumption periods and rising or falling prices accordingly. The system developed has the capability of rising and lowering the prices of electricity based on the availability of electricity from energy sources. This system will depict how much energy the consumers are using and how much it is actually costing them. We believe that the logistics analyzed above will help energy-consumption utilities and consumers to make better energy-efficient decisions.

## ACKNOWLEDGEMENTS

This paper is one of the most significant achievements of my life, and it would not be possible without the love and support of people who believed in me. I express my sincere gratitude to Dr. Kendall E. Nygard, whose guidance; advice and support have helped me achieve my academic goal. Without his continuous support, help and direction, it would not have been possible for me to finish this paper. I give sincere thanks to Dr. Brian Slator, Dr. Jun Kong and Dr. Karl Altenburg for serving on the committee.

I would like to thank the faculty and staff of the Computer Science Department for their help and guidance throughout my master's program. I would also like to thank all of my friends, Kuldeep, Vineet, Kanchan, Arti, Ankita, Vandana and Deepak, for their love and support which helped me feel at home in Fargo.

My sincerest gratitude to my mother, father, sister, brothers and cousins who provided me with wonderful memories, support and encouragement throughout my education for which I will always be indebted and would like to pay off by achieving great success in my career.

## TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	iv
LIST OF TABLES.....	vii
LIST OF FIGURES.....	viii
1. INTRODUCTION.....	1
2. BACKGROUND AND LITERATURE REVIEW.....	3
2.1. Smart Grid.....	3
2.2. Multi-Agent Systems.....	4
2.3. Negotiation in Multi Agent Systems.....	5
3. EVOLUTION OF THE SMART GRID.....	6
3.1. Emerging Need of Smart Grid.....	6
3.2. Dynamic Pricing.....	7
4. MULTI-AGENT SMART GRID SYSTEM FRAMEWORK.....	9
4.1. Multi-Agent Smart Grid System Architecture.....	9
4.2. Agents in the Smart Grid Framework.....	11
4.2.1. Device Agents.....	11
4.2.2. Control Agents.....	11
4.2.3. Management Agent.....	12
4.2.4. Distributed Resource Energy (DER) Agents.....	13
4.2.5. User Agent.....	14
4.2.6. Negotiation.....	14
4.3. Model for Dynamic Pricing in the Smart Grid Framework.....	17

4.4. Strengths .....	18
4.5. Limitations.....	19
5. SOFTWARE SIMULATION .....	21
5.1. Development Environment.....	21
5.1.1. Java .....	21
5.1.2. Java Agent DEvelopment Framework.....	22
5.1.3. Eclipse Platform.....	22
5.2. Multi-Agent Smart Grid System.....	22
5.3. Agent Classes in Multi-Agent Smart Grid System.....	23
5.3.1. ControlAgent Class.....	24
5.3.2. ManagementAgent Class .....	26
5.3.3. DeviceAgent Class.....	27
5.3.4. DERAgent Class .....	29
5.3.5. Class UserAgent .....	30
5.4. How to Run the Simulation .....	31
5.5. Simulator Execution .....	33
6. CONCLUSIONS AND FUTURE WORK.....	38
REFERENCES .....	41

## LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Environment information file structure and contents.....	24
2. ControlAgent class: member variables .....	25
3. ControlAgent class: member functions .....	25
4. ManagementAgent class: member variables.....	26
5. ManagementAgent class: member functions .....	27
6. DeviceAgent class: member variables .....	28
7. DeviceAgent class: member functions.....	28
8. DERAgent class: member variables.....	29
9. DERAgent class: member functions .....	29
10. UserAgent class: member variables .....	30
11. UserAgent class: member functions.....	31

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. Multi-Agent Smart Grid System Architecture. ....	10
2. Negotiation Between User Agents and DER Agents. ....	15
3. Dynamic Pricing Model. ....	18
4. Class Diagram for Smart Grid Multi-Agent System. ....	23
5. Screen Shot of the Tool Running. ....	32
6. JADE Main Container. ....	33
7. Screen Showing Control Agent Issuing Commands to Management Agent. ....	33
8. Screen After Adding User, Device and DER Agents. ....	34
9. JADE GUI After DER, Device and User Agents are Added. ....	34
10. Initiation of Negotiation Among User Agents and DER Agents. ....	35
11. Response of DER Agents on Pricing Requests from User Agents. ....	36
12. Final Results of Negotiation Between DER Agent and User Agent. ....	37



## 1. INTRODUCTION

There has not been a revolution in power distribution technology since Nikola Tesla's pioneering work published in the late 19th century [PEP04]. The technologies prevalent today are still influenced by what was first thought possible many decades ago. The smart grid enables the modernization of power grids to meet contemporary sensibilities and requirements. From a simple broadcast model the grid has now evolved to a more complex distributed system. Software systems are the key enabling technology that lets the grid be more efficient for the producers by enabling the routing of power and also enables the consumer to get the best pricing by allowing a dynamic pricing model.

The smart grid is also better equipped at dealing with failures and reacting to environmental factors that can impact supply or demand. The consumer side is also impacted in a beneficial way by letting the users control consumption and costs in real-time by considering factors like price of a unit of energy and anticipated load etc. [PRK10].

A multi-agent system consists of a number of entities capable of interaction with each other. Such a multi-agent system can enable the power grid to become a Smart Grid by letting the agents interact with each other and make decisions based on dynamic factors. A simpler monolithic system lacks the capabilities of handling such complex scenarios [MPH09].

The basic premise is that price of a unit of energy is not fixed but varies in real-time depending on pressures of demand and supply. The cost of electricity is high when the supply is low and the cost is low when the supply is plentiful. The consumers can react to this by varying usage to optimize usage when rates are low. The power grid is made aware

of various situations and factors that can impact supply or demand [BAT08]. These factors can help in determining the price that the suppliers can set for a unit of energy or also the help users decide how much to bid for or how to distribute usage over time to reduce costs.

The objective of this paper is to design and construct a Smart Grid simulator. We also developed a model for real-time, dynamic pricing of electricity that can be tested within the context of the Smart Grid simulator we have developed. The dynamic pricing model is implemented via the concept of negotiation agents which are responsible for all communication to enable setting of prices. Our dynamic pricing model takes into account fluctuations in demand cycle over time and its effect on pricing. Our design for the multi-agent system allows for a flexible system where agents can join or leave the system, communicate with other agents and make negotiation decisions for the dynamic pricing model. The simulation was achieved using JADE. JADE (Java Agent Development Framework) is a software framework for developing multi-agent systems in java [AVA00]. The system we designed has the following features:

- Is scalable and robust
- Is capable of dynamic addition and removal of agents, and
- Has a low cost of implementation.

The aim of this project is to find and remove inefficiencies in the power distribution system. As energy prices spiral upwards it is our belief that smart grid technology that enables the consumer to make choices that reduce expenses as well as reduce waste would be beneficial for the consumer as well as the producer [AUT06]. We hope that acceptance and usage of these technologies will increase moving the world towards a smarter and greener power grid.

## 2. BACKGROUND AND LITERATURE REVIEW

### 2.1. Smart Grid

The smart grid is the technology that brings the electric network to the digital age. For a long time the electric network followed the predominantly broadcast model wherein a few large generators supply energy to all the consumers of the grid. Software systems now enable the system to be 'smart' by allowing capabilities such as letting the route the power more efficiently and enabling a dynamic pricing model to be implemented instead of a flat fee structure [BAT08].

By enabling the electric grid to be aware of situations that can affect the supply or demand for power, the grid can respond to changes and at the same time enable these changes to be an input in determining pricing. Thus, situations, like transformer failures, or conditions that affect demand, such as heat waves, can be dealt with.

Smart grids do not act only on the distribution side but also on the energy user side. User-side devices such as smart meters or smart thermostats enable the user to control consumption leading to benefits such as lower energy bills, less load on the grid during peak hours and potentially lower impact on the environment by reduction of waste.

A side-benefit of the system is that the actual bandwidth required for the control system is a fraction of the total available bandwidth [FRO10]. This over-provisioning can be beneficial by letting consumer services subsidize smart grids.

It is important to keep in mind that the smart grid is not one single entity but involves a collection of multiple entities on both the producer and consumer sides communicating and working together to meet the goals outlined above.

What will be perhaps most influential in driving the adoption of these technologies is the potentially significant savings according to one report [LDK03] that estimates that the potential benefit over approximately the next two decades will be close to \$75 billion.

## **2.2. Multi-Agent Systems**

One way to enable a grid to be 'smart' is through the use of a multi-agent system. Such a system should be capable of state space search, communication and negotiation with other agents, and decision making (e.g., for pricing).

A multi-agent system is a system of multiple 'intelligent' entities that are capable of interaction with each other. In a smart grid the system has to interact with multiple entities and make decisions based on a number of dynamic factors [JHI10].

A multi-agent system has the following characteristics:

*Decentralized:* There is no single point of control. This is an essential characteristic that separates it from a monolithic system.

*Autonomous:* The agents retain some degree of autonomy and some decision making is transferred to the agent side.

*Demarcation:* Each agent is responsible for a subsystem and not the whole global system. This has implications in security by compartmentalizing subsystems as well as ensuring each agent is not bloated or having to perform complex operations [LAM03].

For developing a multi-agent based system there are many frameworks that aid development. The Java Agent Development Environment (JADE) is one such framework and we have used it in our system development.

### **2.3. Negotiation in Multi Agent Systems**

The decision making framework is an organization to support the multi-agent system or the multi-agent systems that are composed of different agents carrying out the goals, ideas, and decision. The decision making framework specifies how the different agents work together to achieve certain targets [JUR01]. Since different agents are involved in the decision making and action execution responsibilities negotiation come into picture. Negotiation enables these agents to agree on certain points or some decision. They are self-directed and independent, therefore, there is a need to agree on certain point or convince other agents.

Based upon the publication in [MAR99] these negotiations can be of many different forms including argumentation, which varies, from qualitative to quantitative arguments, auctions, and the use of different protocols like constructing of automated agents from the models as described in the publication [PEN02]. The specific scenario that was depicted in the paper was related to the crisis between Spain and Canada over access to a fishery in the North Atlantic. Therefore a need of negotiation was required which led to development of an automated agent to facilitate the negotiation.

Another scenario of negotiation was depicted in crisis management supply-chain in the paper [AYD09] where a new form of negotiation was presented to avoid a crisis. These negotiations have definitely helped in resolving various complicated issues with various approaches and strategies but with the emerging technologies and complicated concepts there is still a need to study of what is good strategy to be implemented by agents in order to have a successful negotiations.

### **3. EVOLUTION OF THE SMART GRID**

*“In the 21<sup>st</sup> Century, we know that the future of our economy and national security is inextricably linked to one challenge: Energy.”* – President Barack Obama, Chicago Press Conference, 16 Dec 2008.

Electric power has always been critically important in determining the economic growth of every nation. The Smart Grid is an upgraded form of the 20th century Electric Grid. Smart grid does more than serve power to utilities and consumers; it includes dynamic pricing. Dynamic pricing forces users to pay higher prices for using more energy during peak hours. The smart grid leverages common infrastructures to provide utilities integration into diverse systems. These systems can even address and respond to events which occur anywhere in the power generation, delivery and supply-demand chain [AUT06]. The smart grid concept has been promoted to facilitate efficient and smart use of energy in a cost-effective manner. This involves the end users making decisions to improve their energy-dependent needs by adopting smarter technology solutions.

#### **3.1. Emerging Need of Smart Grid**

Smart energy demand has been increasing in the past few years to respond to many conditions and events in demand and supply. The smart grid concept has not only touched the industry but, also the home electric consumers. The smart grid promotes the power delivery to power-consuming utilities and consumers by addressing real-time issues [FRO10].

The rapidly emerging need of Smart Grids can be viewed as:

- It provides solutions to enhance reliability, reduce peak hour energy demand, transfer energy demand to off-peak hours, and reduce overall energy consumption.
- It increases energy usage from renewable energy sources like: wind, solar energy, rain, tides, geothermal energy, and sunlight.
- The use of smart equipment, smart meters, smart networking, automated control of energy devices, and automatic and dynamic pricing along with real-time energy usage information for not only commercially spaced market but also real world individual users makes it favorable for the upcoming technological solutions [VEN09].

### **3.2. Dynamic Pricing**

One of the important aspects of smart grid concept is to make it visible to the energy consumers to analyze how much real-time cost-effectiveness is achieved when the energy is consumed at certain time periods. For example, energy consumers pay more for the same amount in peak hours and less in the off-peak hours. This is called dynamic pricing of the real-time energy usage based on the availability of the power.

As it has already been shown that energy play an important role in the overall economic growth. By adopting certain standards, the energy consumption can be optimized. Automatic and dynamic price generation in such situations can help consumers and utilities to schedule their everyday energy usage in a cost-effective manner [KAT11].

Pricing can be adjusted based on few factors:

- *Availability of the power:* the consumers in utility companies would need to pay more when the power is plentiful and less when it is not.

- *The overall power consumption:* Raise the price when it is consumed more so that consumers start reducing the consumption during those times. Similarly, reduce the pricing in quiet hours when the energy consumption is low.
- *The electricity usage:* Consumers will react to higher price periods of time by reducing their electricity usage, and adjust more electricity usage when prices are low. In a real system, these adjustments can be done directly by incorporating “smart” appliances.

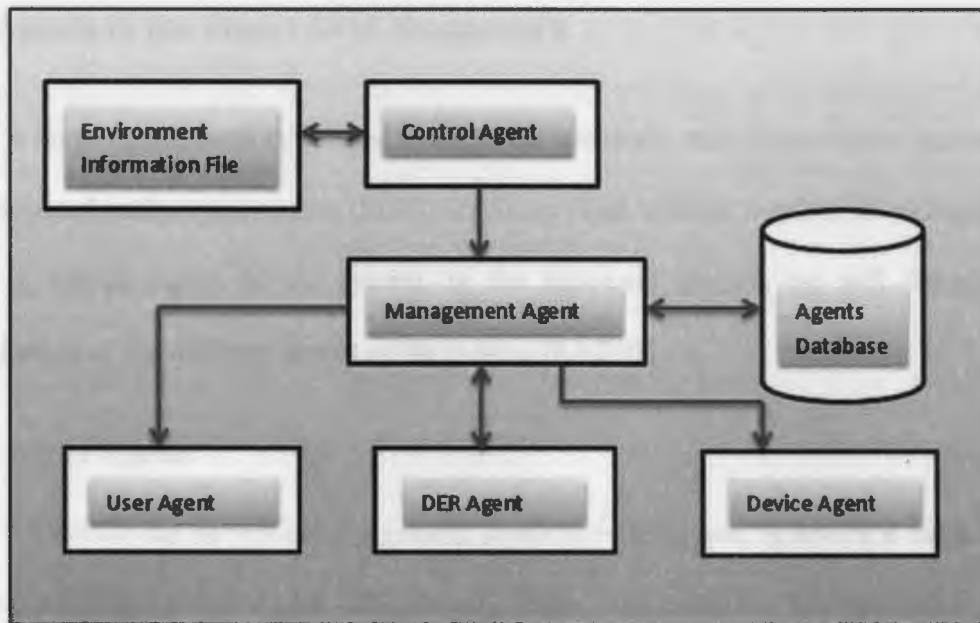


## **4. MULTI-AGENT SMART GRID SYSTEM FRAMEWORK**

In the proposed framework, the agents act as isolated units; they have incomplete information of properties and operations of their own world. These agents don't know much about the environment around them still they can get global information about other agents by querying the environment and communicating with other agents in the environment [JHI10]. For example, users can get power device information by a management system in the environment. Likewise, third party power service providers, i.e., distributed energy sources units, can also determine how much total power capacity is available to server consumers' request by getting certain knowledge from management system and other agents.

### **4.1. Multi-Agent Smart Grid System Architecture**

In multi-agent smart grid architecture, each power producing device is managed by a Device Agent. A power consuming unit is managed by a User Agent. When there is an additional need for power supply than the usual ongoing power supply, other power producing units are required. This additional power need request is met by third party power supply units called DER Agents (Distributed Energy Resources) [MPH09]. DER Agents manage their own power producing Device Agents and keep updated information on each of the Devices Agents which are registered with that DER Agent. When a request from a User Agent comes in to DER Agent, to meet additional Power need, it reads the information it already has and responds to a User Agent back with its offer. Figure 1 shown below is the architectural diagram which depicts the multi-agent smart grid system.



**Figure 1. Multi-Agent Smart Grid System Architecture.**

In addition to the above agents, there is a need for continuous monitoring of the environment for any unwanted disruption of power supply due to any sort of device failure, network failure or communication links failure etc. This monitoring operation is handled by a Control Agent. A Control Agent continuously queries the environment and looks for any damage to the Device Agents, system network, or communication links [PRK10]. In case of a failure or a negative result there are certain actions that need to be performed to quickly and effectively to overcome the situation, find a solution to minimize risk and provide uninterrupted power supply to all its consumers. A Control Agent's task as it seems can be much more complicated as the environment grows, or to avoid any delays in action and handle failure situations gracefully with minimal downtime and with maximum efficiency, Management Agent come into place. Management Agent can serve as the operation execution unit for Control Agent.

## **4.2. Agents in the Smart Grid Framework**

A set of agents were developed to solve the automatic and dynamic price generation. The proposed multi-agent system (MAS) for Smart Grid network requires the delegation of tasks to various agents in the system. In the following section, we will describe the organization of the different agents of the system [RAD05].

### **4.2.1. Device Agents**

A device can be viewed as a generic power producing unit. A device is managed by a corresponding Device Agent. The Device Agent maintains the key attributes for its device and manages the communication between other agents. The Device Agent manages how much power capacity the device has, how much can be supplied and its available power capacity after the power has been supplied. Device agent also determines control strategies to access the device, modify key attributes on the device, and turn-on/turn-off the device. These control strategies may differ based on the device type and how the device is being used in the framework.

The Device Agent also determines the price of power supply based upon a certain amount of the energy demand.

### **4.2.2. Control Agents**

The environment can be complicated to maintain and control especially as the network grows with agents. The Control Agent keeps querying the environment to monitor any unwanted situations such as: failures, broken communication links, damaged network component, and broken network loop. The Control Agent analyzes any such situations, resolves the undesired situation, put the system back in functioning mode, and repairs any

of the damaged components or network areas. Since the internal structure of the Control Agent can become extremely complex, the efficiency may decrease. Hence the framework separates specific internal logic of Control Agent's operations by introducing Management Agent [PTA10].

The Control Agent monitors and analyses the environment at all times and in case of a failure situation, and communicates to a Management Agent to take appropriate action. In addition to analyzing failure situations, the Control Agent also sends commands to Management Agent to add/remove/update User, Device and DER Agents to handle high power demand load, high power supply situation, increase efficiency, minimize risk, increase robustness and scalability issues as well.

#### **4.2.3. Management Agent**

The Management Agent can be viewed a helper agent for a Control Agent. The Management Agent serves requests issued by Control Agent. The Management Agent provides the functionality to dynamically add a new agent to the network when there's need for additional agent. For example, in case of high energy demand, the Management Agent will add new Device Agents to the system to handle the requests from User Agents. Similarly, it can also remove an agent from the network based on the current need.

In this framework, a Management Agent maintains a complete list of all the agents available in the network, their key attributes, current status, transaction history, availability, performance parameters, and dependencies and/or relationships with other components in the system etc. By spreading this vast amount of data in distributed fashion across the entire network, the framework increases efficiency and minimizes data loses. The Management Agent also provides an effective way to access data as the request comes in

which increases the throughput of the overall system therefore results in a good performance. Maintenance of this data on distributed locations is not difficult as only Management Agent has the capability to modify and keep it updated.

#### **4.2.4. Distributed Resource Energy (DER) Agents**

DER Agents can be viewed as third party entities that offer services such as: power supply and communicate with other Device Agents and Management Agents to service any power energy request from User Agent. A DER Agent maintains a list of the Device Agents registered for them and offers User Agents a price and amount for a certain power supply need. The DER Agents manages rises and dips in the prices based on the power energy demand. In the peak hours, when the energy demand is high, the prices rise to control the energy usage by the consumers. Subsequently, when the energy demand becomes less, the price per unit energy goes down to match the available power energy amount [YUN02].

When a request for certain amount of power energy is received by DER Agent, it evaluates some factors below to make the decision:

- Is the device available to serve the request?
- If the device is available, how much available power does it have?
- Based on the peak or off-peak hours or the power demand (high or not high), at what price and what quantity can energy be offered to the user?
- In case the price or energy amount changes or the device becomes unavailable, after making an offer to the user, are there other devices that can serve the request?

Based on the factors above, a DER Agent determines how much energy will be supplied and directs the registered Device Agent handle the request [SHI03].

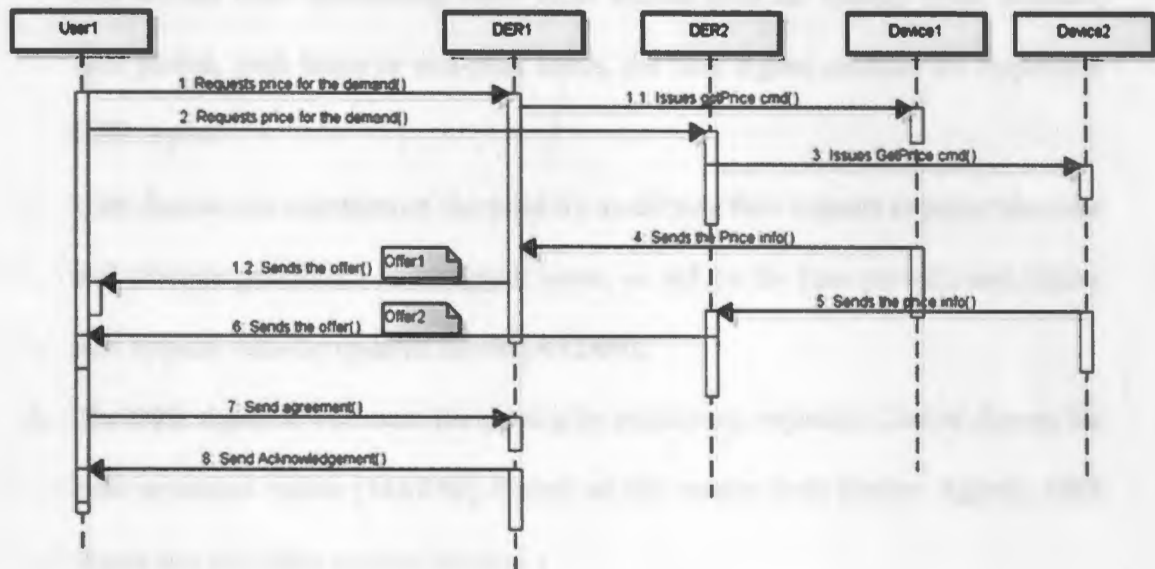
#### **4.2.5. User Agent**

A user is any consumer who is in need of energy. Each user is managed by a User Agent. A User Agent analyses its power need by evaluation power consumption by each resource that a user is using and how many such devices are consuming power. Whenever a user needs additional power, the corresponding User Agent communicates with the Management Agent to get a list of all the DER Agents who are the energy providers. Management Agent gets the User Agent with all the available DER Agents which are further contacted by User Agents. Once the agents are gathered, the agents can negotiate price for energy.

#### **4.2.6. Negotiation**

The framework allows a User Agent and DER Agents to negotiate a price for the requested amount of energy for a certain time period and allows them to agree or disagree on the offered pricing. The agents are connected to each other using contract net interaction protocol [ZAF03]. In contract nets, agents negotiate in a fully automated communication dialog through the use of contracts. Similar to contract nets, our User Agents and DER Agents negotiate on power demand and pricing in an automated manner. At any time, the initiator can start the negotiation; an initiator could be a User Agent wanting to buy power to meet its demand or a DER Agent who wants to sell power [YUN02]. During the negotiation, the DER Agent can specify energy quantity and price for that quantity. Similarly, a User Agent can specify an amount that the agent is willing to pay. There is a

time constraint in the process of negotiation; if the User Agent and the DER Agent agree on the price listed in that time interval, the contract is finalized. Otherwise the session times out. The DER Agent can send contract to other User Agents or a User Agent can query other DER Agents to meet its needs. Figure 2 below depicts the negotiation process between User Agent and DER Agents.



**Figure 2. Negotiation Between User Agents and DER Agents.**

Here are the steps for how the communication, negotiation and decision making is done among User Agents and DER Agents (as shown in Figure 2).

1. A User Agent sends a request with the amount and time period for power supply to all DER Agents in the network.
2. The DER Agents receives requests from various User Agents.
3. The DER Agents contact corresponding Device Agents to evaluate the total available energy and its price.

4. After getting an estimate on the amount and price, DER Agents responds back with the offer to all the User Agents they had received the request from. A DER Agent also sets the priority of each User Agent based on its request considering factors such as: price, time and time period.
5. The User Agents receives the offer from different DER Agents and based on the best pricing offer considering some other factors such as: offered price, amount, time period, peak hours or non-peak hours, the User Agent contacts the respective DER Agent.

User Agents can negotiate on the price by modifying their request to adjust the time slot (change peak hours to non-peak hours, or reduce the time period.) and send a new request with the updates values [AYD09].

6. The DER Agent re-evaluates the pricing by contacting respective Device Agents for new requested values [MAR99]. Based on the results from Device Agents, DER Agent can take three actions below:
  - a. Accept the request if there are no other higher priority agents in queue,
  - b. Reject the request if there are other User Agents to consume the energy or,
  - c. Send a new offer with updated values by modifying the offer based on some of the factors.

A User Agent can similarly do the following:

- d. Accept the offer and send an agreement message to DER Agent,
- e. Reject the current offer and send a new offer by modifying its request to either the same DER Agent or to next best DER Agent, or



- f. Terminate the current negotiation and start an altogether new offer after some interval.
7. When an agreement occurs on both the ends, an acknowledgement with the detailed deal is sent to User Agent by the DER Agent. Similarly, on closure, both agents acknowledge the completion of operation [YUN02].

### **4.3. Model for Dynamic Pricing in the Smart Grid Framework**

In practice, dynamic pricing is achieved by incorporating a host of diverse factors such as weather conditions, historic demand and supply patterns, and anticipated load as well as unpredictable factors such as system failures that impact demand such as transformer failures or issues in the power plant [PEN02].

The creation of a model encapsulating all these factors is beyond the scope of this project and for the purposes of this effort we are restricting ourselves to a simpler model described below.

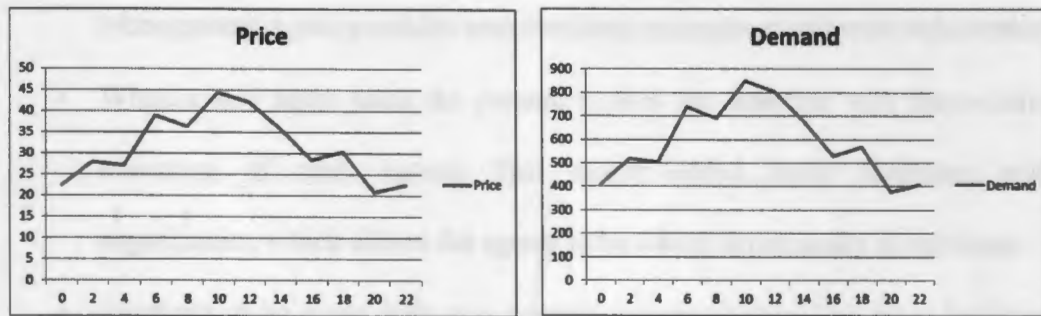
$$P = P_B + (D - D_B) / D_B * 30$$

$D_B$ : is the base demand; the average demand assuming a static pricing model.

$P_B$ : is the base price that is set assuming a constant base demand.

$D$ : Demand is modeled to vary with time, with its maxima at peak daytime hours and minimum at off-peak times. The demand is further adjusted by multiplying to a constant factor (relative change in demand times 30 above) to simulate real-world fluctuations in demand.

The graphs in figure 3 below depict the dynamic pricing model described above. X-axis in the graphs below represents time interval and Y-axis represents price and demand over time respectively.



**Figure 3. Dynamic Pricing Model.**

We assume a 24-hour cycle where the peak is at noon and least demand is at midnight. The demand keeps decreasing from noon till midnight then keeps increasing till the peak at noon. These changes are further adjusted by a random factor to add noise [LIS02].

#### 4.4. Strengths

This framework exhibits the following strengths:

- The approach of introducing another layer to handle additional tasks for a Control Agent provides a good way to handle isolation in monitoring and performing operations such as: managing other agents in the environment, reacting to undesired situations, and resolving conflicts of any kind of communication issues.
- The introduction of Management Agent allows easier, simpler and a more effective way to dynamically deploy other Agents with their components, removal of the failure prone agents, failed components, and damaged network

links. With this kind of clear functional division amongst all agents, the Management Agent provides structured and transparent software architecture.

- When a new agent joins the system, it does not interfere with the existing execution of other agents. This newly added agent facilitates self-organization, which allows the agents to be added dynamically at any time.
- Similarly, if an agent fails or a specific communication link stops working, the removal of an affected component or area does not halt the whole system; rather, it allows the existing execution to continue without disruption.
- Very simple rules have been implemented in all the decisions and algorithms. This requires a moderate amount of memory and even very small-scale computers can be used to run the program.

#### **4.5. Limitations**

The proposed system has been implemented to depict certain core feature in Smart Grid using a Multi-Agent framework. There could have been significant modifications to achieve additional functionalities such as automatically analyzing each User Agents' current energy consumption and regulating pricing individually [LDK03]. Similar limitations are listed in the following section:

- The automatic and dynamic pricing generation is regulated by a time interval.
- The dynamic pricing ignores peak and off-peak hours.
- Though multi-agent systems are capable of moving around the network, but this implementation limits the agents to run in the same network, they do not exhibit mobile agent behavior.

- The underlying architecture relies on running the Control Agent manually, the agents do not start running automatically as part of the environment until the Control Agent is run.
- Decision making algorithms are not mature enough to make the priority queue of User Agents in evaluating which User Agent's request to process first or reject as they come in.
- The negotiation process is restricted to a one step only, if the User Agent and DER Agents ALWAYS agree in the first transaction, they do not re-send exchange negotiation further.

## **5. SOFTWARE SIMULATION**

Software simulation and modeling plays an important role in helping design and improves the effectiveness and the efficiency of the system. A multi-agent system was implemented to provide a better understanding for Smart Grids in modeling the concept of real-time, dynamic pricing of electricity.

The application models the different Agents in a Smart Grid and represents coordination, communication and function of each entity as a whole [AUT06]. The goal of this work is to simulate a Smart Grid environment through a multi-agent framework system. Each agent is responsible for handling operations in the environment and representing as a complete unit to fulfill the need of its own and the other agent entities.

### **5.1. Development Environment**

The tool was developed in JAVA using Java Agent Development Framework (JADE) to simplify the implementation of multi-agent Smart Grid system through this middle-ware. The system was built on Eclipse JAVA platform to easily build and deliver integrated system. It runs on the Windows operating system.

#### **5.1.1. Java**

Java provides a powerful and flexible development environment for application developers for creating applications which follows “write once, run anywhere”. It combines the power of object-oriented programming with the efficiency to run the compiled code anywhere that supports the Java platform. A network-centric aspect of the language makes it unbelievably easy to work with resources across a network [JAM05].

### **5.1.2. Java Agent DEvelopment Framework**

JADE (Java Agent DEvelopment Framework) is an easy-to-use software Framework fully implemented in Java language. It can be used in building multi-agent systems through a set of graphical tools that supports the debugging and deployment of the system in more effective way. It creates multiple containers for the agents, each of which can be run on either same computing platform or different platforms [FAB07].

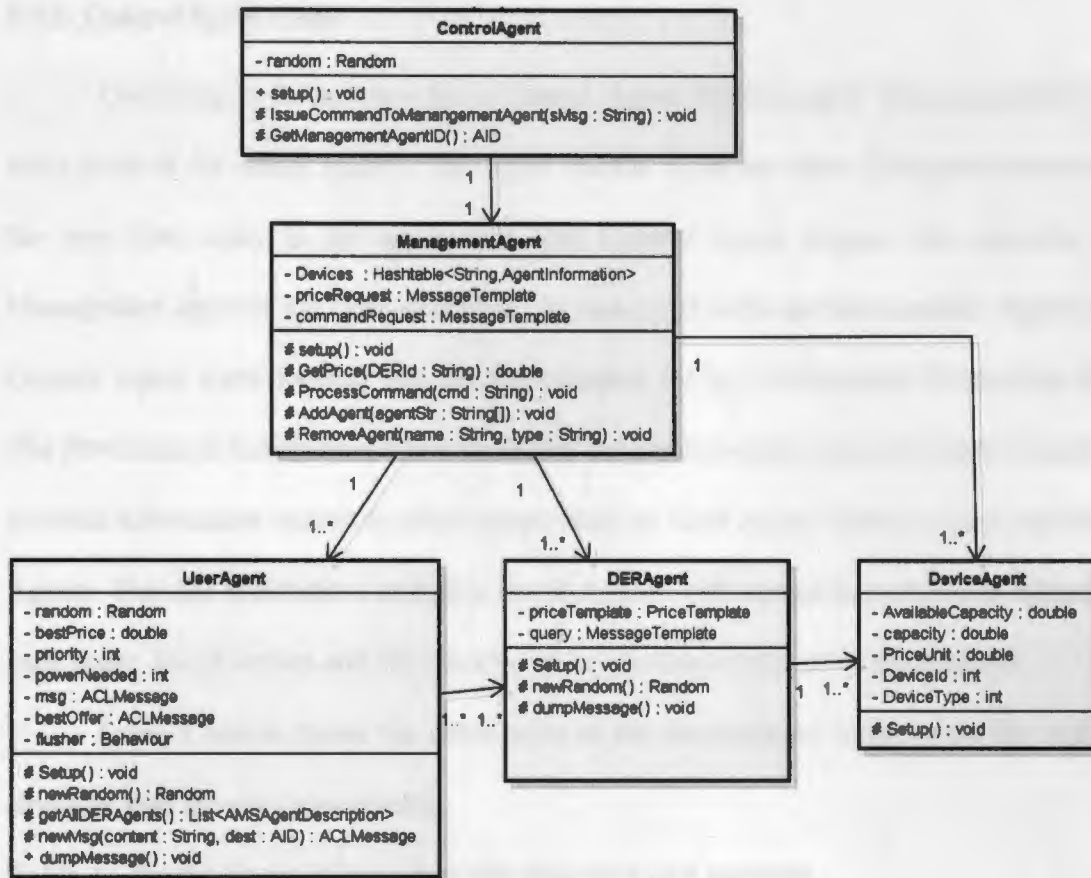
### **5.1.3. Eclipse Platform**

The Eclipse Project was originally developed by IBM in November 2001 [ECW08]. It is a multi-language software development environment, written mostly in java and can be used to develop applications in Java and, by means of various available plug-ins other programming languages. The Eclipse SDK includes the Java Development Tools (JDT) kit, offering advanced refactoring techniques and code analysis.

## **5.2. Multi-Agent Smart Grid System**

The Main Container of JADE framework represents the environment for multi-agent smart grid framework. The main container provides a platform for the agents to run various operations in the framework [AVA00].

The class diagram shown below in figure 4 depicts detailed information on how User Agents are not only connected to DER Agents in the environment, but also responsible for their communication about power needs and price limitations. Likewise, Device Agents can be added and removed dynamically by Management Agents to keep the communication going without interrupting the existing agents operation and the framework infrastructure holding the other agents together [KNU02].



**Figure 4. Class Diagram for Smart Grid Multi-Agent System.**

The following sections describe development environment, main components and the operation of each Agent in the system along with default Main Controller.

### 5.3. Agent Classes in Multi-Agent Smart Grid System

Each Agent class contains one or more methods to meet the requirements and the overall required communication with other agents. The description of each class below depicts the overall goal associated.

### 5.3.1. ControlAgent Class

*ControlAgent* is the class for a Control Agent. ControlAgent Class represents the entry point of the whole system. This agent resides inside the Main Container and run as the very first entity in the application. The Control Agent triggers the execution of Management agent in the same container. As soon as it starts the Management Agent, the Control Agent starts looking into the environment for an Environment Information file. The Environment Information file is stored on the local directory same as ControlAgent. It contains information related to other agents such as: User Agent, Device Agent and DER Agents. This file maintains a complete list of Agents with certain key attributes related to each agent, list of actions and other environment variables required by those agents.

Table 1 below shows the description of the Environment Information file and its structure with sample data contents.

**Table 1. Environment information file structure and contents**

<b>ACTION</b>	<b>ID</b>	<b>Type</b>	<b>DER</b>	<b>Price</b>	<b>Capacity</b>	<b>Demand</b>
ADD	DER1	DERAgent				
ADD	DER2	DERAgent				
ADD	DVC1	DeviceAgent	DER1	20	8000	
ADD	DVC2	DeviceAgent	DER1	25	7000	
ADD	DVC3	DeviceAgent	DER2	30	5000	
ADD	DVC4	DeviceAgent	DER2	35	2000	
ADD	USR1	UserAgent				3000
ADD	USR2	UserAgent				4000
ADD	USR3	UserAgent				9000

The Control Agent provides method to issue commands to Management Agent. The commands are represented by the actions in the environment information file above. Table 2 below shows the member variables used in the Control Agent class.



**Table 2. ControlAgent class: member variables**

<b>Data Type</b>	<b>Variable name</b>	<b>Description</b>
<b>ACLMessage</b>	<b>Msg</b>	Contains the message string as command to issue to Management agent.

The Control Agent class contains methods to query the environment by accessing environment information file. The Control Agent class member functions shown in Table 3 perform functions to add Management Agent to the network, issue commands to Management Agents and access environment information file.

**Table 3. ControlAgent class: member functions**

<b>Return Type</b>	<b>Function Declaration</b>	<b>Description</b>
<b>void</b>	<b>Setup()</b>	Creates Management Agent in the JADE Main Container to process commands.
<b>Void</b>	<b>IssueCommandTo ManagementAgent(String sMsg)</b>	Issues command to Management agent to add User, Device and DER Agents in the container.
<b>AID</b>	<b>GetManagementAgentID</b>	Gets the ManagementAgent ID to issue commands.

When the control Agent finds the file in its local directory, it reads the agents' information from the file and starts issuing commands to Management Agent. At this point, there are no agents such as: User Agent, Device Agent and DER agents exist on the environment. Part of sending commands to Management Agent is to have those agents created in the environment and start assigning the designated tasks for each Agent. Next, the Management Agent comes in place to take the execution to the next level.

### 5.3.2. ManagementAgent Class

*ManagementAgent* represents the class for Management Agent. Management Agent class handles the operations involved in managing other Agents and executing the commands issued by Control Agent. The following behavior depicts the overall flow of operations from the time the Management Agent is created by Control Agent in the system. Once the Management Agent starts running in the Main Container and receives the command issued by Control Agent, it reads the received command and acts appropriately.

The command is a string of texts read from the Environment Information file by the Control Agent. The Control Agent reads the file as one line at a time, which itself is a complete command containing ACTION, Agent ID, Agent Type and key attributes related to the corresponding agent. The control Agent then sends this command to Management agent and Management Agent performs the action specified in the command. The execution of each command results in adding User Agents, Device Agents and DER Agents in the system along with their key attributes such as: ID, Type, Price, Capacity and Demand. Table 4 below shows the member variables which hold the values for the key attributes of the agents.

**Table 4. ManagementAgent class: member variables**

Data Type	Variable name	Description
Hashtable<String, AgentInformation>	devices	A database containing all device agents information
MessageTemplate	priceTemplate	Acts as an ACLMessage response in the transactions
MessageTemplate	commandRequest	Acts as an ACLMessage command request in the transactions

The Management Agent provides methods to process command such as add agent and remove agent. Table 5 contains information on these methods exposed by Management Agent.

**Table 5. ManagementAgent class: member functions**

Return Type	Function Declaration	Description
void	Setup()	This method calls the constructor from parent class "Transaction" which extends "SequentialBehaviour" class. This class contains functionality to Query from a DER Agent about price and process command came from Control Agent.
Double	GetPrice()	Gets the prices from all Device Agents.
Void	ProcessCommand()	Processes command received through Control Agent. E.g. AddAgent and RemoveAgent by calling the functions below.
Void	AddAgent()	Adds an agent in the Main Container.
Void	RemoveAgent()	Removes an agent from the Main Container.

The Management Agent also maintains a database with the entire Agents' information such as: Agents name, the DER Agent it is associated to (only in case of a Device Agent, i.e. there's a one to one mapping between Device Agent and the DER Agent it is registered with) and other key attributes like demand, capacity, price etc. It also provides a method for the key attributes such as: GetPrice. At this point User Agents, Device Agents and DER agents are added in the system.

### 5.3.3. DeviceAgent Class

*DeviceAgent* represents a class for Device Agents. The primary goal of the Device Agents is to hold and maintain certain data as key attributes such as: DeviceId, DeviceType, PricePerUnit, Capacity and AvailableCapacity. Table 6 on the next page refers to these key attributes associated with each device managed by a Device Agent.

**Table 6. DeviceAgent class: member variables**

<b>Data Type</b>	<b>Variable name</b>	<b>Description</b>
<b>Int</b>	<b>DeviceId</b>	Initializes the Device Agents Id.
<b>Int</b>	<b>DeviceType</b>	Holds the DeviceType information
<b>Double</b>	<b>PriceUnit</b>	Sets price of the Device Agents per unit energy.
<b>Double</b>	<b>Capacity</b>	Contains total capacity of the energy for that Device.
<b>Double</b>	<b>AvailableCapacity</b>	Contains available capacity information at any time.

Table 7 below contains the method to initialize the key attribute properties associated for a Device Agent.

**Table 7. DeviceAgent class: member functions**

<b>Return Type</b>	<b>Function Declaration</b>	<b>Description</b>
<b>Void</b>	<b>Setup()</b>	This method initializes the certain properties described above for a DeviceAgent.

The database which Management Agent maintains is populated with Device Agents information by getting the values for this data from each Device Agent. This behavior reduces the cost for User Agent or DER Agent to go to each Device Agent to get the required information on each device and its attributes. This also minimizes the time elapsed in querying each device and especially in a case where any of the Device Agent meets a failure or becomes unavailable for some reason. It also reduces the additional layer between Management Agent and User and DER Agents. The next important Agent in the system is DER Agent.

### 5.3.4. DERAgent Class

*DERAgent* is the class for Distributed Energy Resources Agents. These are third party entities which provide the services of supplying Power to the consumers when the regular power supply is not sufficient for the consumers. In this framework, these third party entities are represented by DER Agents. DER Agents are responsible of obtaining power capacity Information for each device by querying Device Agents' information through Management Agent. Table 8 contains information of member variables of DER Agent class used in message exchange with other agents.

**Table 8. DERAgent class: member variables**

Data Type	Variable name	Description
MessageTemplate	query	Acts as an ACLMessage request in the transactions
priceTemplate	priceTemplate	Acts as an ACLMessage response in the transactions

Table 9 exposes a set of methods participating in the negotiation process and communication with other agents.

**Table 9. DERAgent class: member functions**

Return Type	Function Declaration	Description
void	Setup()	This method calls the constructor from parent class "Transaction" which extends "SequentialBehaviour" class. This class contains functionality to process requests from other agents and send replies back to the agents.
Random	newRandom()	Utility method for generating distinct Random generator used in dynamic price calculation.
Void	dumpMessage()	Utility method used to print the on-screen messages exchanged between agents.

This class contains functionality to process requests from other agents and sends replies back to the agents. DER Agent class also contains utility method for generating distinct Random generator used in dynamic price calculation.

### 5.3.5. Class UserAgent

*UserAgent* is to represent User Agents. The User Agent is responsible for contacting Management Agent and DER Agents when there is an additional need of power supply. As soon as the power demand becomes available, the User Agent contacts the Management Agent to get a list of all the DER Agents to communicate further to meet its power need. It populates certain key attributes such as: BestPrice, priority and PowerNeeded. Table 10 contains the information on member variables which depict these key attributes associated with a User Agent.

**Table 10. UserAgent class: member variables**

Data Type	Variable name	Description
Double	bestPrice	Initializes this property to set to 9999.0
int	priority	Sets the priority of the User Agent.
int	powerNeeded	Holds the value for power needed information for that User Agent.
ACLMessage	msg	Contains message to talk to DER agents.
ACLMessage	BestOffer	Holds message to evaluate bestOffer amongst offer messages from all DER agents.
ACLMessage	Flusher	Contains message to send out power requirement message to all the DER Agents.

The User Agent class also provides methods to get a list of all the DER Agents in the system. Table 11 shows the member functions exposed by a User Agent class.

**Table 11. UserAgent class: member functions**

<b>Return Type</b>	<b>Function Declaration</b>	<b>Description</b>
<b>void</b>	<b>Setup()</b>	Contains the negotiation logic between DER Agents.
<b>Random</b>	<b>newRandom()</b>	Utility method to generate random number to help make decisions in negotiation.
<b>List&lt;AMSAgentDescription&gt;</b>	<b>getAllDERAgents()</b>	Gets a list of all the DER Agents on the network.
<b>ACLMessage</b>	<b>newMsg( int perf, String content, AID dest)</b>	Utility method to initialize messages for communication with DER agents.
<b>Void</b>	<b>dumpMessage()</b>	Displays the messages on the screen console.

Once it receives the list of DER Agents response back from Management Agent, the User Agent issues commands with the amount of power it requires and a request to get the quote for pricing information for the requested power demand.

The important aspect of the communication between the User Agent and the DER Agents is the basis for our study which addresses the real-time scenario of modeling the dynamic pricing of power demand. In this framework, the rises and falls in the power supply prices are depicted directly to the availability of power from DERs Agents. User Agents will pay less when the power is in plentiful and more when it is not.

#### **5.4. How to Run the Simulation**

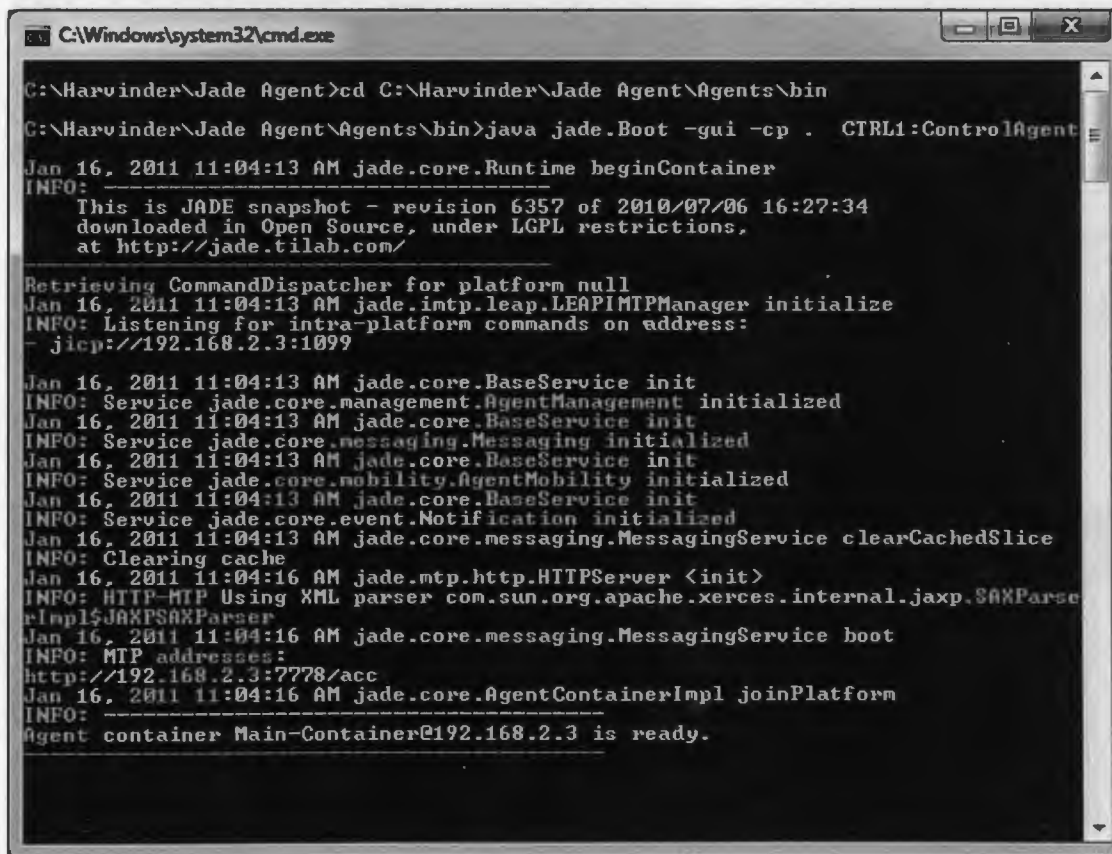
To start the simulation, click on ControlAgent.bat file. This will start the console application. It will also start the JADE main container and add Control Agent and Management Agent [FAB07]. Now the application is ready to add other agents such as: User Agent, Device Agent and DER Agents. We have a pre-built environment information file which had commands to add two DER Agents, four Device Agents (two for each of the

DER Agents added above) and three User Agents. As soon as the User Agents are added to the Main container, they start the negotiation process with DER Agents. The communication, negotiation and decision making protocols have been explained in Chapter2.

The command below on the command prompt initiates the tool.

```
java jade.Boot -gui -cp . CTRL1:ControlAgent
```

A sequence of events gets fired when the command above is entered. The command kicks off the initialization of the JADE Main controller in the first place. Figure 5 shows the detailed information of actions taking place, once the tool starts running.



```
C:\Windows\system32\cmd.exe
C:\Harvinder\Jade Agent>cd C:\Harvinder\Jade Agent\Agents\bin
C:\Harvinder\Jade Agent\Agents\bin>java jade.Boot -gui -cp . CTRL1:ControlAgent
Jan 16, 2011 11:04:13 AM jade.core.Runtime beginContainer
INFO:
-----
This is JADE snapshot - revision 6357 of 2010/07/06 16:27:34
downloaded in Open Source, under LGPL restrictions,
at http://jade.tilab.com/
-----
Retrieving CommandDispatcher for platform null
Jan 16, 2011 11:04:13 AM jade.intp.leap.LEAPIMTPManager initialize
INFO: Listening for intra-platform commands on address:
- jicp://192.168.2.3:1099
Jan 16, 2011 11:04:13 AM jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
Jan 16, 2011 11:04:13 AM jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
Jan 16, 2011 11:04:13 AM jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
Jan 16, 2011 11:04:13 AM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
Jan 16, 2011 11:04:13 AM jade.core.messaging.MessagingService clearCachedSlice
INFO: Clearing cache
Jan 16, 2011 11:04:16 AM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParser
Impl$JAXPSAXParser
Jan 16, 2011 11:04:16 AM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://192.168.2.3:7778/acc
Jan 16, 2011 11:04:16 AM jade.core.AgentContainerImpl joinPlatform
INFO:
-----
Agent container Main-Container@192.168.2.3 is ready.
```

Figure 5. Screen Shot of the Tool Running.



Figure 6 below shows the JADE Main container, which is launched after running the command described above to run the tool.

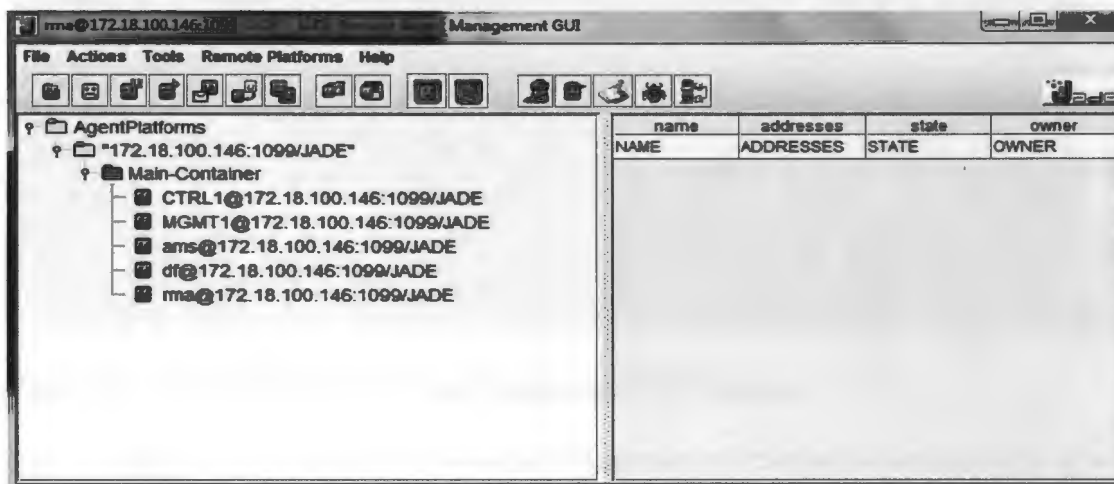


Figure 6. JADE Main Container

## 5.5. Simulator Execution

Once the JADE Main container is ready, Control Agent and Management Agents are added. Soon after the Control Agent is added, it starts reading the environment information file and starts issuing commands to Management Agent. Figure 7 below shows that the Control Agents and Management Agents are added and Control Agent issuing commands to Management Agent.

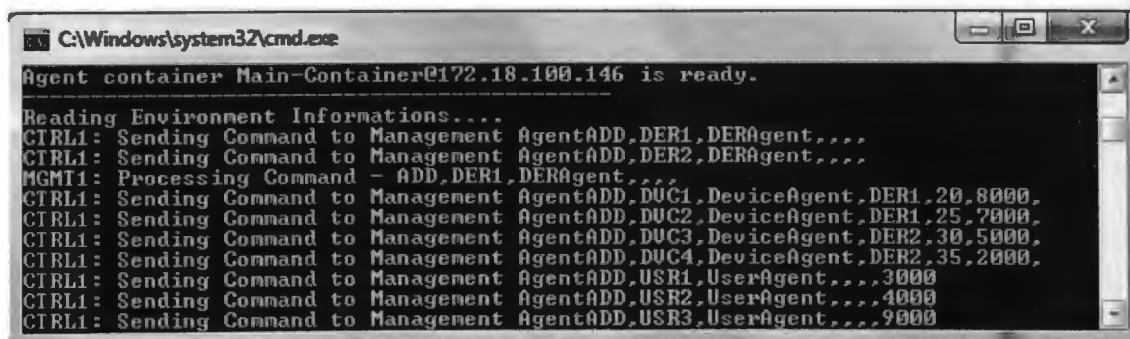


Figure 7. Screen Showing Control Agent Issuing Commands to Management Agent.

Once the User Agents, Device Agents and DER Agents are added (shown below in Figure 8).

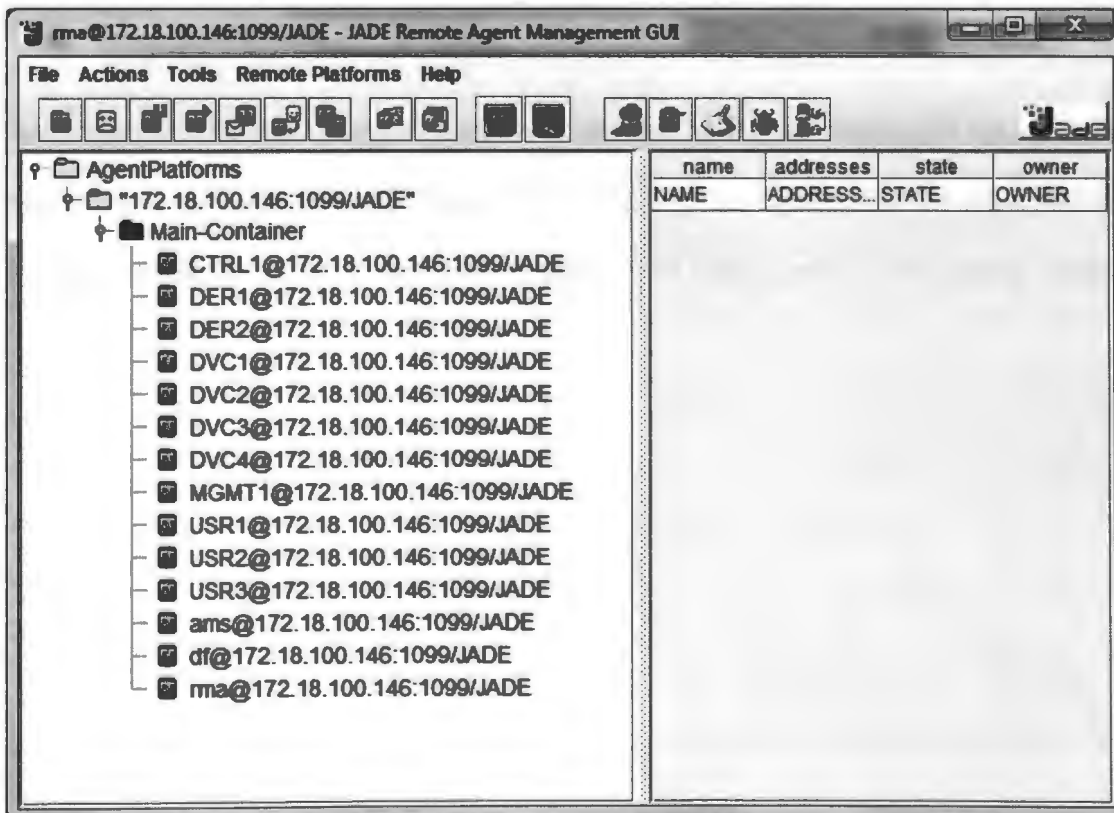
```

C:\Windows\system32\cmd.exe
CTRL1: Sending Command to Management AgentADD,DUC1,DeviceAgent,DER1,20,8000,
MGMT1: DERAgent DER1 has been added
MGMT1: Processing Command - ADD,DER2,DERAgent,,,,
MGMT1: DERAgent DER2 has been added
CTRL1: Sending Command to Management AgentADD,DUC2,DeviceAgent,DER1,25,7000,
MGMT1: Processing Command - ADD,DUC1,DeviceAgent,DER1,20,8000,
MGMT1: DeviceAgent DUC1 has been added
CTRL1: Sending Command to Management AgentADD,DUC3,DeviceAgent,DER2,30,5000,
MGMT1: Processing Command - ADD,DUC2,DeviceAgent,DER1,25,7000,
MGMT1: DeviceAgent DUC2 has been added
CTRL1: Sending Command to Management AgentADD,DUC4,DeviceAgent,DER2,35,2000,
MGMT1: Processing Command - ADD,DUC3,DeviceAgent,DER2,30,5000,
CTRL1: Sending Command to Management AgentADD,USR1,UserAgent,,,,,3000
MGMT1: DeviceAgent DUC3 has been added
MGMT1: Processing Command - ADD,DUC4,DeviceAgent,DER2,35,2000,
MGMT1: DeviceAgent DUC4 has been added
MGMT1: Processing Command - ADD,USR1,UserAgent,,,,,3000
CTRL1: Sending Command to Management AgentADD,USR2,UserAgent,,,,,4000

```

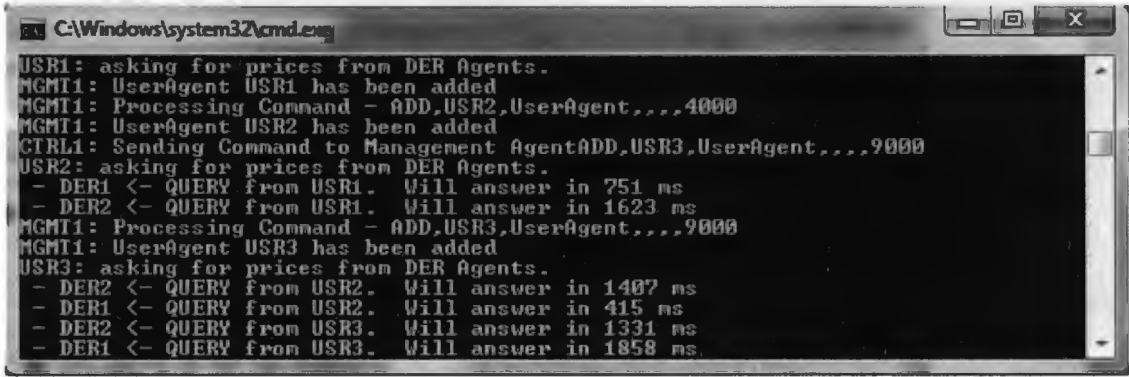
**Figure 8. Screen After Adding User, Device and DER Agents.**

DER Agent queries the Management Agent to get information on available power quantities with each Device Agent. DER Agent querying Management Agent for the information mentioned above is shown in Figure 9.



**Figure 9. JADE GUI After DER, Device and User Agents are Added.**

The User Agent then initiates the negotiation by preparing a contract specifying the total power it needs and sending it to the DER Agents in the network for the pricing information on the request quantity. Figure 10 shows the initiation of negotiation amongst User Agents and DER Agents.



```
C:\Windows\system32\cmd.exe
USR1: asking for prices from DER Agents.
MGMT1: UserAgent USR1 has been added
MGMT1: Processing Command - ADD,USR2,UserAgent,,,4000
MGMT1: UserAgent USR2 has been added
CTRL1: Sending Command to Management AgentADD,USR3,UserAgent,,,9000
USR2: asking for prices from DER Agents.
- DER1 <- QUERY from USR1. Will answer in 751 ms
- DER2 <- QUERY from USR1. Will answer in 1623 ms
MGMT1: Processing Command - ADD,USR3,UserAgent,,,9000
MGMT1: UserAgent USR3 has been added
USR3: asking for prices from DER Agents.
- DER2 <- QUERY from USR2. Will answer in 1407 ms
- DER1 <- QUERY from USR2. Will answer in 415 ms
- DER2 <- QUERY from USR3. Will answer in 1331 ms
- DER1 <- QUERY from USR3. Will answer in 1858 ms
```

**Figure 10. Initiation of Negotiation Among User Agents and DER Agents.**

DER Agent receives the requests from various User Agents in the network. Similar to contract net protocol, the agent prepares a contract stating the pricing for the requested quantity and publishes this onto the network to all the User Agents who has requested this information. Figure 11 below shows the response from DER Agents on the pricing requests.

```
C:\Windows\system32\cmd.exe
MGMT1: DeviceAgent DUC4 has been added
CTRL1: Sending Command to Management AgentADD,USR2,UserAgent,...,4000
MGMT1: Processing Command - ADD,USR1,UserAgent,...,3000
MGMT1: UserAgent USR1 has been added
USR1: asking for prices from DER Agents.
CTRL1: Sending Command to Management AgentADD,USR3,UserAgent,...,9000
MGMT1: Processing Command - ADD,USR2,UserAgent,...,4000
MGMT1: UserAgent USR2 has been added
USR2: asking for prices from DER Agents.
MGMT1: Processing Command - ADD,USR3,UserAgent,...,9000
- DER2 <- QUERY from USR1. Will answer in 358 ms
- DER1 <- QUERY from USR1. Will answer in 1752 ms
- DER2 <- QUERY from USR2. Will answer in 1247 ms
USR3: asking for prices from DER Agents.
MGMT1: UserAgent USR3 has been added
- DER1 <- QUERY from USR2. Will answer in 1753 ms
- DER2 <- QUERY from USR3. Will answer in 1174 ms
- DER1 <- QUERY from USR3. Will answer in 704 ms
USR1: Got quote $51.449999999999996 from DER2
USR3: Got quote $24.72 from DER1
USR2: Got quote $50.160000000000004 from DER2
USR3: Got quote $53.190000000000005 from DER2
USR1: Got quote $22.480000000000004 from DER1
USR2: Got quote $25.36 from DER1
```

**Figure 11. Response of DER Agents on Pricing Requests from User Agents.**

Under the negotiation, the User Agent receives the pricing information from the DER Agents and finds the lowest amount for the requested power quantity. Since the framework is following real-time pricing, the pricing changes in every time interval. The same quantity can cost \$X at the moment and it can automatically change to \$Y the very next moment. If the User Agent agrees to the price offered by DER Agent and sends him the acknowledgement, but the price changed after he sent the acknowledgement, DER Agent prepares a new contract reflecting new pricing. User Agent can either accept the deal or reject the deal based on the good lowest amount that he is willing to pay for the request power. The negotiation continues until DER Agent terminates the negotiation with the current User Agent in case he finds another User Agent for the same offer or User Agent finds a better deal from some other DER Agent. Figure 12 shows the negotiation agreement steps between User Agent and DER Agent.

```
C:\Windows\system32\cmd.exe
USR2: Got quote $27.02 from DER1
USR1: Got quote $31.58 from DER1
USR2: Got quote $38.73 from DER2
USR1: Got quote $43.77 from DER2
USR3: Got quote $35.78 from DER1

USR2: Best Price $27.02 from DER1
USR2: ORDER Placed to DER1 at 27.02

USR1: Best Price $31.58 from DER1
USR1: ORDER Placed to DER1 at 31.58
DER1: Got proposal $31.58 from USR1 & my price is $31.58
== AGREE
USR1: Got AGREE from DER1
----- Finished -----

DER1: Got proposal $27.02 from USR2 & my price is $27.02
== AGREE
USR2: Got AGREE from DER1
----- Finished -----
```

**Figure 12. Final Results of Negotiation Between DER Agent and User Agent.**

## 6. CONCLUSIONS AND FUTURE WORK

We have described an approach to modeling the idea of real-time, dynamic pricing of electricity in a Smart Grid. We use the concept of negotiating agents to model the dynamic pricing in a Smart Grid. The negotiation agents play a vital role in carrying out the appropriate communication [PEN02]. The focus of this study is dynamic pricing of electricity together with the decision making capability plus direct communication among the agents in real-time scenario. The system we designed allows agents to join the system, leave the system, exchange information with other agents, generate dynamic pricing for energy demand at certain time period, and make decisions to agree on the negotiation or reject a negotiation in based on the pricing of energy.

In addition to the simulation itself, the study also aimed to evaluate a model for dynamic pricing of power based on demand and supply pressures. We believe that dynamic pricing of electricity is an important element in a Smart Grid. Dynamic pricing of electricity and negotiation among agents is handled by using a combination of facts, rules and commands transfer. Using the proposed model, the real-time energy demand and supply requests can be solved by following the concept of offering a higher price when the supply is low or/and the energy will be consumed during peak hours and less when the supply is high and the energy will be consumed during off-peak duration. To this end we simulated a system using JADE multi-agent framework to model dynamic pricing of electricity in a smart grid system.

One of the salient features of the framework is that the agents are isolated from each other and are capable of querying the environment and/or other agents to collect information they need for processing. The system utilizes a Control Agent that in

conjunction with a Management Agent performs tasks such as continuous monitoring of the environment. These along with other components provide a truly distributed system that maximizes performance and minimizes data loss.

Some of the key advantages of the framework are:

- Structured and transparent software architecture.
- Use of delegation for better performance and robustness
- Self-organizing and scalable system supporting dynamic addition of agents.
- Scale not limited by the architecture but by available resources.
- Graceful failure, system is capable of functioning even if one agent is not working.
- Use of simple architecture and design means inexpensive and efficient hardware can be used to implement the system. Driving down costs and maintenance issues.

Possible improvements:

- The use of simple design means some features were left out such as temporal pricing (peak/off-peak pricing models)
- Agents are limited to being in the same network.
- The agents are dependent on the control agent to be run before they can run.
- Simple decision making algorithm that can be improved.
- Simplified single step negotiation. A more advanced system should be able to handle multi-step bidding/negotiations.

The motivation behind the project was to aid in developments of systems that, in the face of growing energy demand and rapidly escalating prices, will enable a more efficient and cost-effective system for both producers and consumers of energy.



## REFERENCES

- [AUT06] Akira Uehara and Takeshi Nagata, "A Study of Trading Electricity in Microgrid", PP.153-155, Oct. 30, 2006.
- [AVA00] Henri Avancini and Analia Amandi, "A Java Framework for Multi-agent Systems", SADIO Electronic Journal of Informatics and Operations Research, pp. 1-12, 2000.
- [AYD09] Ayda Kaddouci, Hayfa Zgaya, Slim Hammadi and Francis Bretaudeau, "PAAN: Partial Agreement Negotiation Network based on Intelligent Agents in Crisis Situation", Issue 4, Volume 3, 2009.
- [BAT08] A. Battaglini, J. Lilliestam, C. Bals and A. Haas, "The SuperSmart Grid", the energy and climate challenges, pp. 5-8, 2008.
- [DAR02] Dargar, Anup, Ahmed Kamel, Gordon Christensen and Kendall E. Nygard, "An Agent Based Framework for UAV Collaboration", in Proceedings of the ISCA 11th International Conference on Intelligent Systems, pp. 54-59, Boston, MA, 2002.
- [ECW08] "Where did Eclipse come from?", Eclipse Wiki,  
[http://wiki.eclipse.org/FAQ\\_Where\\_did\\_Eclipse\\_come\\_from%3F](http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F). Retrieved 16 Dec 2010.
- [FAB07] Fabio Luigi Bellifemine, Giovanni Caire and Dominic Greenwood, "Developing Multi-Agent Systems with JADE",  
[http://www.wiley.com/legacy/wileychi/bellifemine\\_jade/](http://www.wiley.com/legacy/wileychi/bellifemine_jade/) February 2007.
- [FRO10] Frost & Sullivan's Anthony Miller, "The Evolution of the Smart Grid", Issue-2 AMI and Smart Grid, 2010.

- [HOH09] Hohm Utility Support, "Microsoft® Hohm Fact Sheet for Utilities February 2010", On average, how much money will Microsoft Hohm help consumers save on their energy bills?, 2009.
- [JAM05] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha, "The Java language specification", third edition. Addison-Wesley, 2005.
- [JHI10] Jinhee Ko, In-Hye Shin, Gyung-Leen Park, Ho-Yong Kwak and Khi-Jung Ahn, "Design of a Multi-agent System for Personalized Service in the Smart Grid", Security-Enriched Urban Computing and Smart Grid Communications in Computer and Information Science, 2010, Volume 78, 267-273, 2010.
- [JUR01] Prof. Dr. Jürgen Dix, "Decision Making in Multiagent Systems", <http://www-poleia.lip6.fr/~charif/seminaires.html>, 2001.
- [KAT09] Katie Fehrenbacher at GIGAOM, "Microsoft Reveals Its Energy Management Tool: Hohm", <http://gigaom.com/cleantech/microsoft-reveals-its-energy-managent-tool-hohm/>, Jun 24, 2009.
- [KAT09] Katie Fehrenbacher, "Utility Perspective: Why Partner With Google PowerMeter?", <http://gigaom.com/cleantech/utility-perspective-why-partner-with-google-powermeter/>, 2009.
- [KAT11] Kat Shoa, "Smart Grid Dynamic Pricing: Behavior Change Easier Said than Done", <http://smart-grid.tmcnet.com/topics/smart-grid-fa/articles/87826-smart-grid-dynamic-pricing-behavior-change-easier-said.htm>, June 2010, ret. on Feb 2011.
- [KNU02] Knublauch, H., "Extreme programming of multi-agent systems", 2002, ACM Press, pp. 201-285, 2002.

- [LAM03] Lam, K. and H. Leung, "Rational communication in multi-agent semi-competitive environments", *Autonomous agents and multi-agent systems*, pp. 142-221, 2003.
- [LDK03] L. D. Kannberg, M. C. Kintner-Meyer, D. P. Chassin, R. G. Pratt, J. G. DeSteele, L. A. Schienbein and S. G. Hauser, W. M. Warwick, "The Benefits of a Transformed Energy System". Pacific Northwest National Laboratory under contract with the United States Department of Energy. pp. 25, 2003.
- [LIS02] Lisa Brouwers, Karin Hansson, Harko Verhagen, Magnus Boman, "Agent Models of Catastrophic Events", pp. 123-201, 2002.
- [MAR99] Martin Beer, Mark d'Inverno, Michael Luck, Nick Jennings, Chris Preist and Michael Schroeder, at Microsoft Academic Research Publication, "Negotiation in Multi-Agent Systems", pp. 285-289, 1999.
- [MPH09] M. Pipattanasomporn, H. Feroze, S. Rahman, "Multi-Agent Systems in a Distributed Smart Grid: Design and Implementation", *IEEE PES 2009 Power Systems Conference and Exposition (PSCE'09)*, Seattle, Washington, March 2009.
- [NRJ98] N. R. Jennings and M. Wooldridge, "Applications of Intelligent Agents", in *Proceedings of International Agent Technology Conference - IAT*, 1998.
- [PAU11] Paul Avery, "Multiagent Systems Engineering", Southwest Research Institute, Robotics and Automation Engineering,  
<http://www.swri.org/4org/d10/msd/automation/mase.htm>, 2011, retrieved on March 2011.

- [PEN02] Penina Hoz-Weiss<sup>1</sup>, Sarit Kraus, Jonathan Wilkenfeld and Tara E. Santmire , “An Automated Negotiator for an International Crisis”, AAAI/IAAI pp. 1000-1001, 2002.
- [PEP04] Pepe, "Pepe's Tesla Pages",  
[http://peswiki.com/index.php/PowerPedia:Nikola\\_Tesla](http://peswiki.com/index.php/PowerPedia:Nikola_Tesla), pp. 12-25, 2004.
- [PRK10] Prakash Ranganathan and Kendall Nygard, “A Multi-Agent framework in a Smart Grid”, International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 1, No. 4, December 2010.
- [PTA10] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers, and N. R. Jennings, “Intelligent Agents for the Smart Grid”, 1649-1650, 2010.
- [RAD05] Radziah Mohamad and Safaai Deris, “Pattern-Oriented Design for Multi-Agent System: A Process Framework”, Agent Design Improvement Process, pp. 1-8, 2005.
- [SHI03] Shih-Fen Cheng, Evan Leung, Kevin M. Lochner, Kevin O’Malley, Daniel M. Reeves, L. Julian Schwartzman, and Michael P. Wellman, “Walverine: A Walrasian Trading Agent”, AAMAS-03, 2003.
- [VEN09] Venkat Pothamsetty and Saadat Malik, “Smart Grid Leveraging Intelligent Communications to Transform the Power Infrastructure”, Cisco Systems, Inc. 2009.
- [YUN02] Ye Chen, Yun Peng, Tim Finin, Yannis Labrou, and Scott Cost, “Negotiating Agents for Supply Chain Management”, Negotiating agents and system framework, 2001.
- [ZAF03] Zafeer Alibhai, “What is Contract Net Interaction Protocol?”, FIPA Contract Net Interaction Protocol Specification, 2003.

[ZHA04] Zhang, Zhong, "Distributed decision-making in electric power system transmission maintenance scheduling using Multi-Agent Systems (MAS)", TR-2004-06-1, 2004.