Autonomous Energy Management System in Next Generation Homes

DIPLOMARBEIT
zur Erlangung des akademischen Grades

Diplom-Ingenieur/in
im Rahmen des Studiums
Telecommunications Engineering
eingereicht von

Raul Charro Laino
Matrikelnummer 1029354

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Dipl.-Ing. Dr. Sandford Bessler
Mitwirkung: Dra. Slobodanka Tomic
The energy management system in next generation home has been developed in web semantic technology.

This technology is used in Smart homes. And for the monitoring of the power level and consumption of appliances, groups and rooms. Inside the autonomous energy system. The semantic web will give us the necessary interrelated to use autonomous.

For good management of energy consumption, we must create communication methods to interact with the user. We can refer to the possibilities offered by the internet and the new smartphones of the latest generation, which facilitates remote use of applications.

For this project we want to convey the importance of being aware of saving energy, starting with knowing how to manage it.
Acknowledgements

I would like to dedicate this thesis to my parents who inspired me in my life and work every day. Always more and better.

I wish to express my gratitude to my grandmother, who died during my period in Vienna. Without her support and concern, I would never have been as I am. Thanks YAYA!

During this period it was hard to go ahead but was easier thanks to my girlfriend Esther for her support every day. Thanks for being by my side.

I thank Dra. Slobodanka Tomic for the supervision of this thesis and the opportunity of writing the thesis at the FTW.

I thank Dr. Brian Walsh his help in my thesis and share their knowledge of technologic, politics and economic today.
# Contents

## 1 Introduction

1.1 Motivation and background ............................................. 1  
    Next Generation Home (GENIO) .................................... 1  
    Energy Management in Next Generation Home .................... 1  
    Smart Plugs for energy measurement ............................... 2  
    Home Gateway and web services .................................. 2  
    Semantic processing and Ontologies .............................. 3

1.2 Problem statement and goal ......................................... 4

1.3 Outline ............................................................... 4

## 2 High Level description of the system  

Connecting to the Plugwise system ................................. 5  
Creating an Ontology Model ......................................... 6

Creating Schedules .................................................. 6  
Building the Energy Management System ......................... 7

User Interface ....................................................... 7

Web Services interface ............................................. 7

## 3 EMS System implementation  

3.1 Interactions with the Plugwise system ......................... 8

Plugwise Source ..................................................... 8

XML Data Structure .................................................. 9

Methods of interaction ............................................. 11

Java classes within the energy management system that interact with plugwise .................. 13
B Foundations
B.1 Common protocols and standards ........................................ 72
B.2 Semantic Web ................................................................. 72
B.3 User interface - Web Page .................................................. 77
  HTML .............................................................................. 77
  Servlet ............................................................................ 77
  Javascript ....................................................................... 78
B.4 Web service ................................................................. 79
  JAX-WS ......................................................................... 79
B.5 Data base ................................................................. 81
  JDBC-ODBC .................................................................. 81
  MySQL ........................................................................... 82
C EMS - Analysis ...................................................................... 83
C.1 Using XML and HTML files .............................................. 83
C.2 Using Parsers, Action, Monitoring and Alarms ..................... 83
C.3 Using Data Base and RDF ................................................. 84
C.4 Using Ontology and Semantic Web ..................................... 84
C.5 Using Web Methods ....................................................... 84
D Index .................................................................................. 85
List of Figures ........................................................................ 85
List of Tables ........................................................................ 87
Listings ............................................................................... 88
1.1 Motivation and background

Next Generation Home (GENIO)

The GENIO\(^1\) project aims to define the home network of the future, developing an advanced self-management of the home network, facing the problem of heterogeneity of the devices and their interactions, and looking to maximize automation and to respond intelligently to events and alarms. The project intends also to resolve other challenges like access to the content from everywhere and personalization of the services.

Energy Management in Next Generation Home

The term Energy Management System refers here to a computer system which is designed specifically for the automated control and monitoring of the heating, ventilation and lighting needs of a building or group of buildings such as university campuses, office buildings or factories. The data obtained from these can then be used to produce trend analysis and consumption forecasts.

The Energy Management in Next Generation Homes is an important part for the future projects in Smart Homes, Smart Grids,... The next generation homes need to develop energy management systems (EMS) with more intelligence. Intelligent agents with the technology of semantic web need a specific ontology in order to operate.

Plugwise products made these technologies thus created the three main components.

- the autonomous building system (ABS) that provides control for appliances, groups and rooms.

\(^1\)GENIO: http://projects.celtic-initiative.org/genio/index.html
• the energy information system (EIS), which is a database of information technologies that works with the EMS to provide data and information to energy managers and other stakeholders.

• management and integration with user interface and the Genio project

Smart Plugs for energy measurement

Smart Plug is a ‘smart’ energy meter, developed to wireless monitor power consumption and to notify users when it observes drastic changes in the energy consumption pattern.

Quite often we do not realize that homes and industries consume much more energy than required due to inefficient electronics, faulty devices and our own negligence. This is especially true in sectors such as hospitals, healthcare and networking devices in homes. Studies show that we can save at least five to fifteen per cent electricity in homes and in industries by using smart meters to manage electricity demands.\(^2\)

Plugwise Products

Plugwise helps consumers and organisations realise quick results in their efforts to make their businesses and homes more sustainable. Plugwise has developed and produced wireless systems for energy management and appliance control since 2006.

Plugwise is a leading supplier in the market for wireless infrastructure and dynamic energy management for buildings. The Plugwise system is easy and safe to implement at low cost yielding a high return on investment.\(^3\)

Smart Plugs

The Circle/Circle+ plug measure the energy consumption of connected appliances and can switch them on or off. It communicates using the ZigBee wireless protocol with coordinator running at the USB stick.

Coordinator (USB Stick)

The Stick takes role of the ZigBee coordinator and reads the data from all Smart Plugs in the network. The coordinator transmits switching commands to the Circles.

Home Gateway and web services

Home Gateway, also called a residential gateway is defined as an intelligent network interface device located in the consumer premises. It provides the means for the residential user to access the Internet services delivered to home and also to access the


\(^3\)Plugwise About: http://www.plugwise.com/idplugtype-f/about-plugwise
different services offered by the various smart devices located within home. Essentially the home gateway device provides the necessary connectivity features to enable the consumer to exploit the advantages of a networked home.

Web services allow humans or automated agents to interact with a system via the web. They offer operations to send and retrieve data and it is possible to compose them in a workflow. The term web services was often associated with the WS* stack, a set of standards for description, lookup and communication regarding web services mostly based on exchanging SOAP messages.

**Semantic processing and Ontologies**

We define Semantic Processing as coding information into or decoding information from communication patterns (texts, informational protocols and streams, etc.) using a formal representation of semantics. Computer operations on data are well understood and can be done in automatic mode. Meaning of data (Information) is interpreted and processed by humans. Once this process is completed, humans can generate new data and programs for computers to process. So, strictly speaking, computers are doing information processing in cooperation with humans instead of independently. The reason for this is that computers cannot define what the data means to humans and they cannot understand human language.

Semantic Processing eliminates the semantic gap between the computer as a formal device on one side and the non-formal human language interface on the other side. This is the most common problem in computing and it affects the following computer applications:

- Software Methodologies and Tools
- Artificial Intelligence
- Human-Machine and Machine-Machine Interface
- Telecommunications
- Systems Integration
- Other areas

In computer and information science, an ontology formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain and may be used to describe the domain.

---

Ontologies are the structural frameworks for organizing information and are used in artificial intelligence, the Semantic Web, systems engineering, software engineering, biomedical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation about the world or some part of it. The creation of domain ontologies is also fundamental to the definition and use of an enterprise architecture framework.

1.2 Problem statement and goal

Current use of electricity in recent years has increased exponentially and has generated higher consumption. It needs new methods to save and manage energy.

At present there are no smart homes. We have multiple smart devices but not intelligent infrastructures. It is important to create smart homes, smart cities and smart grids to meet the needs of new devices.

The goal of this project is to explore existing concepts and implementations and to investigate the need of workflows using semantic technology. The focus will be on the creation of the ontology and rules for operating schedules and monitoring with priority lists.

1.3 Outline

The rest of the thesis is structured as follows: Chapter 2 gives the high level architecture of the designed system and identifies the major tasks in the work. Chapter 3 describes the design and implementation of the various components. A novel energy management system based on semantic technologies is introduced. Chapter 4 describes with an use case the operation and use of the developed energy management system. Chapter 5 will give an overview of related work and other systems that deal with similar problems. Finally chapter 6 concludes the document and outlines the findings and lessons learned during this project. Future aspects and open issues are discussed.

In the appendixB we give a technology overview of web services, semantic web and their architecture. Then also web service composition is covered where existing paradigms are examined that provide a foundation for this work. User interface and data base, are discussed and important modules in the semantic web.
High Level description of the system

This section describes the overall system and its integration in the Smart Home Genio architecture. We also state the major activities performed within this thesis.

The energy Management system to be developed in this work is not isolated, it must be integrated into a larger Energy remote monitoring and management architecture as described by the project GENIO. The communication to the outside world is performed via the Home gateway, using web service technology (SOAP protocol).

The Energy Management System itself consists of periphery sensors and actuators realized with the Plugwise system. Their state can be queried by the central EMS application. The role of the application is to monitor the consumption, perform semantic reasoning using defined rules, target values and priorities and then perform actions accordingly such as switching on and of appliances using priorities.

The home has of course a spatial organization in rooms and appliance groups, all of those being modeled in the ontology. The user participates in the decision of the reasoner by being able to change priorities, target values, etc. via a web interface on which she can also visualize the measured and recorded values.

The work described in this thesis is divided in the following activities:

Connecting to the Plugwise system

The aim of this activity is to create the corresponding Java classes for communication between Plugwise and the application and to realize relationship with the ontology through the use of schedules with power levels of priority lists.

The management of the elements that are provided by the equipment have to interact with the device for control and energy management according to the needs of the system.
Creating an Ontology Model

The development of semantic technology in the energy management system will require the definition of an ontology. We will create classes, subclasses, lists of objects and property data to incorporate all the information from the XML interfaces for connecting to Plugwise. We will use rules in the ontology to link appliances with groups and rooms, to be evaluated by Jena reasoners.

Creating Schedules

The objective of this activity is to create a class in the ontology mapping onto the Plugwise model, such that we can add, edit and delete new schedules in a user interface. for the realization of the schedule, we consider a week with 7 days, a day with 24 hours and every hour is divided into fractions of 15min. The reasoning rules will refer to the ap-
pliances, groups and rooms in the model Plugwise. When this goal is accomplished the EMS application logic can make queries and decisions and interact with the Plugwise using HTML methods periodically.

**Building the Energy Management System**

Once we have the ontology we can create the energy management system. It should also include monitors to capture information and set an alert if an appliance, group and room has exceeded the power level.

This way we get to have a full monitoring system with autonomous decision making. Finally we make the system available through a web server, thus being totally independent from any external system failures that communicates with ours.

**User Interface**

We propose a web based user interface, since the browser can be used from remote devices to easily access the system.

The graphical aspect of the user interface (UI) must be intuitive for the user. The UI must have a header, sidebars and main content to be more compact and agile interaction with the energy management system. We must use the tables and graphs to represent data and the representation of information about the devices, groups and rooms for monitors, priority lists and schedules.

**Web Services interface**

The methods have to be defined to replicate to a web service client. Depending on customer needs, in this case the GENIO system, specific methods will be included. The goal is to make a list of methods and mechanisms to create a web service client. Testing can done on various platforms PC, tablet, smartphone.
Chapter 3

EMS System implementation

3.1 Interactions with the Plugwise system

This section has been developed to implement the parse and action methods that interact with Plugwise products through the webserver into Plugwise Source.

Plugwise Source

Source is the software that one uses to control the Plugwise system and view the energy consumption of the user, displayed in well-organised charts. Source also calculates how much energy, CO₂ emissions and costs you actually save.

Figure 3.1: Plugwise Source

This webserver can be used to transmit information on the Plugwise system and switch appliances remotely by of HTML pages or XML feeds.¹

¹Plugwise Source Template Engine (PDF) http://www.plugwise.com
The location of the files to transmit information on the Plugwise system through the webserver, is located at:
Plugwise(folder main)/Source/www/ftw/

![Plugwise (XML & HTML)](image)

Figure 3.2: Schema files Plugwise XML & HTML

**XML Data Structure**

The XML structure used by Plugwise Source contain different parameters of appliances, groups and rooms. These parameters are defined in the document (Plugwise Source Template Engine) that provides Plugwise. These parameters are:

- **id** - appliance or group or room.
- **name** - appliance or group or room.
- **realstate** - Switch state appliance or group or room.
- **powerstate** - Lock state appliance.
- **powerusage** - Power level of the appliance or group or room.
- **totalusage** - Total Consumption of the appliance or group or room.
- **totalusagetoday** - Total Consumption today of the appliance or group or room.

When we make a request to the server, we will receive a xml schema with updated data. These data make the communication possible between the Smart Plugs and Energy Management System.
CHAPTER 3. EMS SYSTEM IMPLEMENTATION

Figure 3.3: XML Code Structure

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<appliance>
  <id>id</id>
  <name>name</name>
  <type>type</type>
  <image>image</image>
  <locked>locked</locked>
  <powerstate>powerstate</powerstate>
  <powerusage>powerusage</powerusage>
  <totalusage>totalusage</totalusage>
  <moduleid>moduleid</moduleid>
  <macaddr>macaddr</macaddr>
  <moduletype>moduletype</moduletype>
  <schedule>schedule</schedule>
  ...
</appliance>
```

Figure 3.4: XML - Resolved

```
<appliance>
  <id>1</id>
  <name>Computer</name>
  <type>Computer</type>
  <image>image</image>
  <locked>False</locked>
  <powerstate>on</powerstate>
  <powerusage>27.3</powerusage>
  <totalusage>8.08688</totalusage>
  <totalusagetoday>0.078</totalusagetoday>
  <moduleid>2</moduleid>
  <macaddr>00C6F00009909F</macaddr>
  ...
</appliance>
```
Methods of interaction

Below are shown the methods for the interaction with the plugs:

- **Switch On/Off** - appliance or group or room.
- **Lock On/Off** - appliance (only). If you can switch On/Off a group or room and have a plug/s locked. We can preserve the switch state of the plug/s.

```<%
<!-- Switch Off All--> if Request.get("option") == "switchoffall" Plugwise.Networks[Request.get("id")] .SwitchOff() /if
<!-- Switch On All--> if Request.get("option") == "switchonall" Plugwise.Networks[Request.get("id")] .SwitchOn() /if
<!-- Switch Off Appliance--> if Request.get("option") == "switchoff" Plugwise.Appliances[Request.get("id")] .SwitchOff() /if
<!-- Switch On Appliance--> if Request.get("option") == "switchon" Plugwise.Appliances[Request.get("id")] .SwitchOn() /if
<!-- Switch Off Room--> if Request.get("option") == "roomoff" Plugwise.Rooms[Request.get("id")] .SwitchOff() /if
<!-- Switch On Room--> if Request.get("option") == "roomon" Plugwise.Rooms[Request.get("id")] .SwitchOn() /if
<!-- Switch Off Group--> if Request.get("option") == "groupoff" Plugwise.Groups[Request.get("id")] .SwitchOff() /if
<!-- Switch On Room--> if Request.get("option") == "groupon" Plugwise.Groups[Request.get("id")] .SwitchOn() /if
<!-- Lock Off Appliance--> if Request.get("option") == "lockoff" Plugwise.Appliances[Request.get("id")] .SetDoNotSwitchoff(False) /if
<!-- Lock On Appliance-->
if Request.get["option"] == "lockon"
    Plugwise.Appliances[Request.get["id"]].SetDoNotSwitchoff(True)
    
/ef

echo Request.get
%
<html>
    <head></head>
    <body>
    <script type="text/javascript">
        if(true){

            history.go(-1);
        }
    </script>
    </body>
</html>

Listing 3.1: Codes of the methods of interaction - actions.html
Java classes within the energy management system that interact with plugwise

Parser

Here is described the different Java classes which interact with Plugwise through reading XML by using DOM.

The Document Object Model (DOM\(^2\)) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Aspects of the DOM (such as its "Elements") may be addressed and manipulated within the syntax of the programming language in use. The public interface of a DOM is specified in its application programming interface (API).

Three parsers have been developed with the same structure, which perform a) the connection with XML file through URI, b) authentification and c) storage in a buffer for later analysis by DOM.

```java
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;

import java.net.Authenticator;
import java.net.PasswordAuthentication;
import java.net.URL;
```

\(^2\) What is the Document Object Model? - W3C [http://www.w3.org/DOM/](http://www.w3.org/DOM/)
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public void connectionParser() {
    try {

        // --- Connection Plugwise XML --- //
        URL url = new URL("http://localhost:5357/ftw/statisticsApp.xml");
        URLConnection con = url.openConnection();

        // --- Authenticator Plugwise XML --- //
        Authenticator au = new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("admin", "admin".toCharArray());
            }
        };
        Authenticator.setDefault(au);
        BufferedReader in = new BufferedReader(
            new InputStreamReader(con.getInputStream()));

        // --- Buffer Plugwise XML --- //
        DocumentBuilderFactory dbFactoryPlugwise = DocumentBuilderFactory.newInstance();
        DocumentBuilder dBuilderPlugwise = dbFactoryPlugwise.newDocumentBuilder();

        //Parser "statisticsApp.xml"
        dBuilderPlugwise.getDocumentElement().normalize();

        } catch (Exception ex) {
            System.out.println("Exception connectionParser(): " + ex);
        }
    }
}

Listing 3.2: Parser of the Appliance. Connection. applianceParser.java
The following code (get power level) is the structure implemented to extract the parameters of the xml files:

```java
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;
import org.w3c.dom.Element;

public double getPowerLevel(int idapp) {
    double returnPowerLevel = -1;
    try {
        connectionParser();
        NodeList listaAppliances = docPlugwise.
            getElementsByTagName("appliance");
        for (int i = 0; i < listaAppliances.getLength();
            i++) {
            Node appliance = listaAppliances.item(i);
            if (appliance.getNodeType() == Node.
                ELEMENT_NODE) {
                Element elemento = (Element) appliance;
                int testid = new Integer(getTagValue("id",
                    elemento));
                if (testid == idapp) {
                    returnPowerLevel = Double.
                        parseDouble(getTagValue("powerusage",
                            elemento));
                }
            }
        }
    } catch (Exception ex) {
        System.out.println("Exception getPowerLevel(): " +
            ex);
        return -1;
    }
    return returnPowerLevel;
}

private static String getTagValue(String sTag,
    Element eElement) {
    NodeList nList = eElement.
        getElementsByTagName(sTag).item(0).
        getChildNodes();
    Node nValue = (Node) nList.item(0);
    return nValue.getNodeValue();
}
```

Listing 3.3: Parser of the Appliance. Get Power Level. `applianceParser.java`
EMS Actions

Here we give an example of an action called to switch on an appliance.

![Figure 3.6: EMS Actions](image)

```java
public String selectAction(int id, int action) {
    ...
    url = new URL("http://localhost:5357/ftw/actions.html?
                    option=switchon&id=" + id);
    con = url.openConnection();
    au = new Authenticator() {
        @Override
        protected PasswordAuthentication
            getPasswordAuthentication() {
            return new PasswordAuthentication("userName",
                      "password".toCharArray());
        }
    };
    Authenticator.setDefault(au);
    in = new BufferedReader(new InputStreamReader(con.
                      getInputStream()));
}
```

Listing 3.4: Action of the Appliance. Switch On. applianceAction.java
3.2 Semantic architecture in the Energy Management System

The model of the devices and plugs previously discussed is based on the data obtained through Plugwise server through parser. We divide in four classes to group the information and then link it. The relationship between the information is done through the object properties, to give meaning to the ontology.

Plugwise model

The model consists of four Classes:

<table>
<thead>
<tr>
<th>Global Class</th>
<th><a href="">plugwise:Plugwise</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Class:</td>
<td><a href="">plugwise:appliance</a></td>
</tr>
<tr>
<td>Class:</td>
<td><a href="">plugwise:group</a></td>
</tr>
<tr>
<td>Class:</td>
<td><a href="">plugwise:room</a></td>
</tr>
<tr>
<td>Class:</td>
<td><a href="">plugwise:schedule</a></td>
</tr>
</tbody>
</table>

Table 3.1: List of Class

- Within the class we have the subclasses:

<table>
<thead>
<tr>
<th>Class</th>
<th><a href="">plugwise:group</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td><a href="">plugwise:group_appliance</a></td>
</tr>
<tr>
<td>Appliance of the group</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th><a href="">plugwise:room</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td><a href="">plugwise:room_appliance</a></td>
</tr>
<tr>
<td>Appliance of the room</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th><a href="">plugwise:schedule</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclass</td>
<td><a href="">plugwise:schedule_appliance</a></td>
</tr>
<tr>
<td>Appliance of the schedule</td>
<td></td>
</tr>
</tbody>
</table>

| Subclass       | <plugwise:schedule_group> |
|                | Group of the schedule   |

| Subclass       | <plugwise:schedule_room> |
|                | Room of the schedule     |

Table 3.2: List of SubClass
• The model has also four Object Properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="">plugwise:hasAppliance</a></td>
<td>class or subclass or element has associated a appliance</td>
</tr>
<tr>
<td><a href="">plugwise:hasGroup</a></td>
<td>class or subclass or element has associated a group</td>
</tr>
<tr>
<td><a href="">plugwise:hasRoom</a></td>
<td>class or subclass or element has associated a room</td>
</tr>
<tr>
<td><a href="">plugwise:hasSchedule</a></td>
<td>class or subclass or element has associated a schedule</td>
</tr>
</tbody>
</table>

Table 3.3: List of Object Properties

![Diagram of relationship with classes and objects]

Figure 3.7: Schema of relationship with classes and objects

The implemented model relates all the components of the system and manages the interactions in a specific or global way, thus being able to make easily rules performing autonomously. This model covers all the features to be implemented in the energy management system.
• Each Class/SubClass consists of different Data Properties:

<table>
<thead>
<tr>
<th>Class:</th>
<th><a href="">plugwise:appliance</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_appliance</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Appliance</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:isTooHigh</a></td>
</tr>
<tr>
<td></td>
<td>boolean - Appliance is too high with respect to the schedule</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:locked_appliance</a></td>
</tr>
<tr>
<td></td>
<td>boolean - Appliances is locked</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:name_appliance</a></td>
</tr>
<tr>
<td></td>
<td>string - Name of the Appliance</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:powerstate_appliance</a></td>
</tr>
<tr>
<td></td>
<td>double - Power level of the Appliance</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:powerusage_appliance</a></td>
</tr>
<tr>
<td></td>
<td>double - Consumption Today of the Appliance</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:priorty</a></td>
</tr>
<tr>
<td></td>
<td>int - Level of the priority the Appliance has</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:realstate_appliance</a></td>
</tr>
<tr>
<td></td>
<td>On or Off - Status Switch of the Appliance</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:totalusage_appliance</a></td>
</tr>
<tr>
<td></td>
<td>double - Consumption Total of the Appliance</td>
</tr>
</tbody>
</table>

Table 3.4: List of Data Properties (Appliance)

Figure 3.8: Schema of the Appliance
### Table 3.5: List of Data Properties (Group)

<table>
<thead>
<tr>
<th>Class:</th>
<th><a href="">plugwise:group</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td></td>
</tr>
<tr>
<td><a href="">plugwise:id_group</a></td>
<td>int - Id of the Group</td>
</tr>
<tr>
<td><a href="">plugwise:isTooHigh</a></td>
<td>boolean - Group is too high with respect to the schedule</td>
</tr>
<tr>
<td><a href="">plugwise:name_group</a></td>
<td>string - Name of the Group</td>
</tr>
<tr>
<td><a href="">plugwise:powerstate_group</a></td>
<td>double - Power level of the Group</td>
</tr>
<tr>
<td><a href="">plugwise:powerusage_group</a></td>
<td>double - Consumption Today of the Group</td>
</tr>
<tr>
<td><a href="">plugwise:priority</a></td>
<td>int - Level of the priority the Group has</td>
</tr>
<tr>
<td><a href="">plugwise:realstate_group</a></td>
<td>On or Off - Status Switch of the Group</td>
</tr>
<tr>
<td><a href="">plugwise:totalusage_group</a></td>
<td>double - Consumption Total of the Group</td>
</tr>
</tbody>
</table>

Figure 3.9: Schema of the Group

An important part of the ontology in the groups is to consider indexing the appliances. Using the XML received, we have to store and relate with rules.
Similarly to the groups, the ontology of rooms is also considering the indexing of the devices. Using the XML received, we have to store and map the rules, not to confuse the appliances of room with the appliances of groups, to keep the functionality of a system for autonomous decisions. Since we can have the same appliance in a room and in a group at a time.
The case of schedules is different for groups and rooms, as we may have the appliances, groups and rooms associated with the schedules. It is necessary to define the subclasses of the appliances, groups and rooms within the ontology schedule.
Subclass Data Properties:

- group_appliance, room_appliance
- day_x_hour_ij_min_xy, schedule_appliance, schedule_group, schedule_room

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:group_appliance</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_group</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Group</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:id_group_appliance</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Appliance</td>
</tr>
</tbody>
</table>

Table 3.8: List of Data Properties(Group_Appliance)

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:room_appliance</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_room</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Room</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:id_room_appliance</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Appliance</td>
</tr>
</tbody>
</table>

Table 3.9: List of Data Properties(Room_Appliance)

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:day_x_hour_ij_min_xy</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_schedule</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Room</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:Limit_power_level</a></td>
</tr>
<tr>
<td></td>
<td>double - Limit the power level</td>
</tr>
</tbody>
</table>

Table 3.10: List of Data Properties(day_x_hour_ij_min_xy)
CHAPTER 3. EMS SYSTEM IMPLEMENTATION

Table 3.11: List of Data Properties(Schedule_Appliance)

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:schedule_appliance</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_schedule</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Schedule</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:id_schedule_appliance</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Appliance</td>
</tr>
</tbody>
</table>

Table 3.12: List of Data Properties(Schedule_Group)

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:schedule_group</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_schedule</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Room</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:id_schedule_group</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Group</td>
</tr>
</tbody>
</table>

Table 3.13: List of Data Properties(Schedule_Room)

<table>
<thead>
<tr>
<th>Subclass:</th>
<th><a href="">plugwise:schedule_room</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data:</td>
<td><a href="">plugwise:id_schedule</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Room</td>
</tr>
<tr>
<td></td>
<td><a href="">plugwise:id_schedule_room</a></td>
</tr>
<tr>
<td></td>
<td>int - Id of the Room</td>
</tr>
</tbody>
</table>

Model of priority

The Priority scheme is created by a table of database, accessible by a user interface which then uses the ontology. The priority level is divided into 4 levels, level 1 is the lowest - level 4 is the highest, and it is assigned to appliances, groups and rooms.

- <plugwise:priority> int - Level of the priority has the Appliance or Group or Room

Lowest priority devices should be switched off first  It’s first condition to be met when the ontology <plugwise:isTooHigh> is true in group or room. Once the condition is fulfilled, this condition (isTooHigh) is checked until all devices are switched off.

If the priority level is the same, we can decide to switch off the devices with more power or less power.

Implementation based on groups or rooms
Groups / Rooms Within the ontology of the group or room associated appliances are selected, sorted and placed from lowest to highest priority within a java class (priorityAction.java - Methods: switchPriorityGroup[flag, idgroup] and switchPriorityGroup[flag, idgroup] - flag select lowest 0 or highest 1 power level) and switch off the appliance with lowest priority and lowest power level.

Power level
The power level in the ontology is represented in the appliances, groups and rooms with:

- `<plugwise:powerstate_appliance>` value (double) - Power Level of the Appliance.
- Is the same with groups(powerstate_group) and rooms(powerstate_room).

The values are updated periodically by Plugwise XML and are represented in the schedule by:

- `<plugwise:limit_power_level>` value (double) - Limit of Power Level [-1 = On, 0 = Off, xx.x Watts]

Later, will be mentioned the rules to interact with appliances based on their power level.

Consumption
The consumption in the ontology is represented in the appliances, groups and rooms with:

- `<plugwise:totalusage_appliance>` value (double) - Consumption Total of the Appliance.
- `<plugwise:powerusage_appliance>` double - Consumption Today of the Appliance.
- Is the same with groups(totalusage_group, powerusage_group) and rooms(totalusage_room, powerusage_room).

The values are updated periodically by Plugwise XML.

Data are stored in the Ontology Data are stored using the Java class developed with plugwise server. This Java class stored in the Ontology through files XML developed into Plugwise server. Also it is used to interact with the rules, and information queries for the user.
Rules are created to verify whether the appliances or groups or rooms comply with the schedule with limit power level on the day, hour and minute that are developing energy consumption. Also, the other functions of the rules are to check the level priority and select the appliances that need switch action.

3.3 Communication with Plugwise and population of the Ontology

System start

The system starts with the creation of classes, object properties and data properties in the ontology using Jena API, before obtaining the data and insert it into the ontology.

Read xml data into Ontology

The cycle of reading data from the ontology is possible through the Jena API or SPARQL queries. With these readings and queries we can store the power values with timestamp into a database, and update the ontology and replace the old power levels by new values.

Periodically 20 sec In our program we have selected a periodicity of 20 seconds for ontology updating. In the case of the monitors we can modify the frequency.

![Create OntClass](image)

*Create OntClass*

(appliance, group, room, group_app, room_app)

- Object Properties
  (hasAppliance, hasGroup, hasRoom)

- Data Properties
  (id, name, isTopHigh, priority, realstate, powerstate, powerusage, totalusage)

Figure 3.12: Creating class, object properties and data properties
CHAPTER 3. EMS SYSTEM IMPLEMENTATION

Query plugwise webserver for data xml

Query the information through the Java class (Parser). Then create the individuals and assign the values to the Ontology through the Jena API.

![Diagram of Plugwise (XML) to Ontology](image1)

Figure 3.13: Query of the appliances, groups, rooms

Call the reasoner

Finally call the reasoners and use the rules for each case. Previously, we read the ontology for associating the reasoner. In this example the reasoner associates the appliances with the groups and rooms.

![Diagram of Rules to Create Reasoner](image2)

Figure 3.14: Reasoner of the appliances, groups and rooms
Monitoring power level

Here we describe the Monitor threads that store the power level values. We can modify the time to capture power level, since not all devices are equal and for different cases we need a different control. Also, alarms can be generated if a power level is too high, we can specify a maximum power level as a parameter in the monitor or through schedulers.

In our system we will create different parameters for the monitors:

- **Time to capture** time between readings.
- **Power level Alarm** maximum alarm value.
- **Id** of the appliance or group or room.
- **Type** select appliance(type = 0) or group(type = 1) or room(type = 2)
- **Variance Power level Alarm** margin of caution before alarm.

![Figure 3.15: User interface of monitoring](image)
3.4 Scheduler architecture

The scheduler starts with the creation of classes, object properties and data properties in the ontology using Jena API.

![Diagram](image)

Figure 3.16: Creating class, object properties and data properties

Query the information through the Java class(Parser). Then create the individuals and assign the values to the Ontology for scheduler.

![Diagram](image)

Figure 3.17: Query of the schedules
Call the reasoners and use the rules for schedules. The reasoner associates the schedules with the appliances, groups and rooms.

Figure 3.18: Reasoner of the schedules

The next reasoner associates the appliances, groups and rooms when the power level is too high. If this is true it will put the isTooHigh data property to true. Through this simple implementation, we manage to meet a goal, determine if an appliance or group or room has a high power level.

Figure 3.19: Reasoner of isTooHigh
Now we can perform the query with SPARQL queries if isTooHigh is true and interact with the appliances through java class (action - Switch / Switch Priority).

![Query of isTooHigh](image)

Figure 3.20: Query of isTooHigh

The result is a webservice "Scheduler" that autonomously interacts with the appliances through user interface management schedules.

![User interface of scheduling](image)

Figure 3.21: User interface of scheduling

### 3.5 Web Service interfaces

The functionality supported the creation of web services to communication with an external system (GENIO) is shown below:

- Create a Web Service from a Java Class
- Add an Operation to the Web Service
- Deploy and Test the Web Service
- Generate WSDL (Web Service Definition Language). The XML Schema language for defining SOAP-based web services (XSD). The step 1 in Figure 4.22. (This action is performed once to establish service.)

- The client calls methods on the server. The step 2 in Figure 4.22.

- The server responds to client methods. The step 3 in Figure 4.22.

Figure 3.22: Communication SOAP client and server

**List of methods or operations to the web service**

**GENIO asks for data about devices**

**Get the Lists of the Appliances, Groups and Rooms**

getArrayListApplianceGroups, getArrayListApplianceRooms, getArrayListAppliances,
getArrayListGroupAppliances, getArrayListGroups, getArrayListRoomAppliances,
getArrayListRooms, getListApplianceGroups, getListApplianceRooms,
getListAppliances, getListGroupAppliances, getListGroups,
getListRoomAppliances, getListRooms.
Get Switch Status of the Appliances, Groups and Rooms getSwitchStatus.

Get Lock Status of the Appliances, Groups and Rooms getLockedStatus.

Get information about Monitors
getArrayListMonitor, getArrayAlarm, getArrayAlarmID, getArrayAlarmTimeNow.

Get information about Schedules
getArrayListSchedule, getArrayListScheduleIdType, updateScheduleJenaDay.

Information on the state of energy
getArrayConsumptionTodayAll, getArrayConsumptionTotalAll,
getArrayListScheduleIdType, getArrayPowerLevelAll, getConsumptionToday,
getConsumptionTodayAll, getConsumptionTotal, getConsumptionTotalAll,
getPowerLevel, getPowerLevelAll.

- GENIO sends an action command
createMonitor, createScheduleJena
deleteMonitor, deleteScheduleJena
setLockedAction, setSwitchAction, setSwitchActionAll.
updateMonitor, updateScheduleJenaDay.
3.6 Implementation

UI with logo - customized for Genio project

This section describes different parts in the User Interface (UI) for Plugswise server.

**Index page**  This page has three parts: header, left table and main contents.

- **Header(same all page):** Logo link FTW - Link Index with logo Genio - List of different links that interact with Energy management system.

- **Left table(same all page):** Index - About - Contact - Develop - List of priorities - List of different links that interact with Energy management system.

- **Main contents**  Logo Genio - List of different links that interact with Energy management system - Main Menu (Monitor Alarms, Monitor Ontology[Jena], Listener Alarm[Jena - isTooHigh] )

![User Interface: index.xhtml](image)

Figure 3.23: User Interface: index.xhtml
Appliances Plugwise page - Main Contents  List of appliances with Info appliance, info of power (Power Level, Consumption,...) into appliance, info of Schedule into appliance, State of appliance and options select( Switch, Lock ). This page has been developed in a Java servlet.

Figure 3.24: User Interface: Java Servlet - Appliance

```java
applianceServlet:: HttpServlet
- doGet(HttpServletRequest request, HttpServletResponse response)
- doPost(HttpServletRequest request, HttpServletResponse response)
- getServletInfo() : String
- getTagValue(String eTag, Element eElement) : String
- imageGraphAppliance(String IDApp, String fromDate, String toDate, PrintWriter out)
- processRequest(HttpServletRequest request, HttpServletResponse response)
- iCount : int
- refreshPage : boolean
```

Figure 3.25: Functions Java API: applianceServlet.java
Figure 3.26: Algorithm Description - Appliances UI
Groups Plugwise page - Main Contents  List of groups with Info group, Info of power (Power Level, Consumption,...) into group, Info of Schedule into group, State of group and options select(Switch). This page has been developed in a Java servlet.

![Image of Groups Plugwise page]

Figure 3.27: User Interface: Java Servlet - Group

```java
// groupServlet.java

public class groupServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
    public void doPost(HttpServletRequest request, HttpServletResponse response)
    public String getServletInfo()
    public String getTagValue(String sTag, Element eElement)
    public void processRequest(HttpServletRequest request, HttpServletResponse response)
    public int iCount
    public boolean refreshPage
```

Figure 3.28: Functions Java API: groupServlet.java
Figure 3.29: Algorithm Description - Groups UI
Rooms Plugwise page - Main Contents  List of rooms with Info room, Info of power (Power Level, Consumption,...) into room, Info of Schedule into room, State of room and options select( Switch ). This page has been developed in a Java servlet.

![Room Plugwise Interface](image)

**Figure 3.30:** User Interface: *Java Servlet - Room*

```
roomServlet:: HttpServlet
  doGet(HttpServletRequest request, HttpServletResponse response)
  doPost(HttpServletRequest request, HttpServletResponse response)
  getServletInfo() : String
  getTagValue(String sTag, Element eElement) : String
  processRequest(HttpServletRequest request, HttpServletResponse response)
  iCount : int
```

**Figure 3.31:** Functions Java API: *roomServlet.java*
Figure 3.32: Algorithm Description - Rooms UI
Module Plugwise page  List of modules with Info module, Info of power (Power Level, Consumption,...) into module and State of module. This page has been developed in a Java servlet.

Figure 3.33: User Interface: *Java Servlet - Module*

```java
moduleServlet :: HttpServlet
  doGet(HttpServletRequest request, HttpServletResponse response)
  doPost(HttpServletRequest request, HttpServletResponse response)
  doGetInfo() : String
  processRequest(HttpServletRequest request, HttpServletResponse response)
  iCount : int
  refreshPage : boolean
```

Figure 3.34: Functions Java API: *moduleServlet.java*
Figure 3.35: Algorithm Description - Modules UI
Scheduler User Interface

The Scheduler UI is a section that describes different options of schedule, graph and table to specify power level of appliances, groups and rooms depending on day, hour and minute during week. This page has been developed in a java servlet.

Figure 3.36: User interface of scheduling - Java Servlet - Schedule

```java
scheduleServlet : HttpServlet
```

- doGet(HttpServletRequest request, HttpServletResponse response)
- doPost(HttpServletRequest request, HttpServletResponse response)
- executeMe()
- getServletInfo() : String
- getTagValue(String sTag, Element eElement) : String
- outDatabaseDay(List<DBSchedule> lsDB, PrintWriter out, int postonDataBaseID, int Day)
- outDatabaseDayGraph(List<DBSchedule> lsDB, PrintWriter out, int postonDataBaseID)
- processRequest(HttpServletRequest request, HttpServletResponse response)
- iCount : int

Figure 3.37: Functions Java API: scheduleServlet.java
Each timetable is repeated each week. Of the seven days each day has 24 hours, each hour 4 groups of 15 minutes. In the Schedule UI, each fraction can be programmed with a specific power level [Watts] or switch on (-1 power level designee) and switch off (0 power level designee).

Each new schedule must have scheduled 672 specific power levels. Being able to update and modify these power levels in the user interface as shown in Figure 4.25.

![Figure 3.38: UI of scheduling - Update - Weekdays](image)

**Data are stored in the Ontology** Each program is created and updated in a database (*Microsoft Access .mdb*) and then read and stored in the ontology (RDF).

The structure of schedule in database is [ID, type, name, idtype, time, Day0, Day1, Day2, Day3, Day4, Day5, Day6]. Each DayX has 96 specific levels for every 15 minutes fraction.

The structure of schedule in Ontology is [id_schedule, name_schedule, type(schedule_appliance or schedule_group or schedule_room), day_x, day_x_hour_ij, day_x_hour_ij_min_zy].

---

3In section “4.2. Architecture - Semantic architecture in the Energy Management System” is explained in more detail.
CHAPTER 3. EMS SYSTEM IMPLEMENTATION

Power level Interface

This section is divided into data acquisition and data updating. The data acquisition is measured in Watts and is updated from the server each time plugwise refreshes the page or the UI.

The Power level UI is integrated into different sections Schedule UI(update limit power level) and Appliance UI or Group UI or Room UI (Info Power Level). The methods of parse have been designed to obtain information into Plugwise server. The interpretation of data done within the ontology using Jena API methods to get the algorithms decisions. The interpretation of data is done within the Ontology. With the realization of algorithms with Jena API methods.

Implementation of the flags.

For this section we have developed some flags to determine the action of the listeners to switch off the appliances. We can use four flags:

- **Switch off - Same priority.** When we have two or more appliances or groups or rooms with the same priority, we can determine whether to switch off an appliance or group or room with higher power (MAX) or lower power level (MIN).

- **Stored in the DB Alarm.** When we have an appliance or group or room that should be switched off, we can select if store in the database as an alarm and use it in communication with the web service.

- **Switch Priority.** With this flag, If a group or room should be switched off, the listener program uses the priority list of devices that make up the group or room to determine which to switch off.

- **Switch.** Selecting this flag the program detects to switch off the group or room, it will switch off all appliances in the group or room. Also if an appliance should be switched off individually only.
Activation of the listeners and monitors would be the following:

- When you press the start (Start Monitor Alarms) button, the monitor begins to store power levels and detect alarms of the appliances, groups and rooms. We can see this information on the monitoring interface.

- When you press the start (Start Monitor Ontology) button, the system starts architecture and communicates with Plugwise and fills in the Ontology.

- When you press the start (Jena Listener Alarm Schedule) button, the system queries the ontology in search of appliances, groups and rooms for switch off through using the selected options.
Priorities Plugwise page - Main Contents - Right Table  This page can create the list of priorities, modified and updated through the right table that shows all appliances, groups and rooms available with ID, and power level. It is also possible to select the flags described above.

Figure 3.41: User Interface: Java Servlet - Priority

Figure 3.42: Functions Java API: priorityServlet.java
3.7 Integration and interaction within the GENIO System

In this section we explain the overall integration with the GENIO project. We can see three parts, the first part is formed by Plugwise. The second part is formed by the energy management system. And the third part is formed by the GENIO system. The interaction in the first part with the second part will be using XML and HTML. For the second part, we can see the semantic system integration, database (RDF and MDB) and user interface (using Servlet). And the third part we can see the communication established by SOAP messages (implemented client using WSDL and XSD) within GENIO System.

Figure 3.43: Schema Integration and interaction within the GENIO System
Proving concepts

As a proof of concept and to bring the developments into practical use, we provide test cases within the GENIO system. The necessary requirements and the results of the tests.

4.1 Requirements

The following computer was used to create the energy management system:

- HP Compaq 8200 Elite
- Intel Core i5-2400 (3.10 GHz, 6 MB cache, 4 cores)
- 4 GB 1333 MHz DDR3 SDRAM
- 500 GB 7200 rpm SATA 6.0 Gb / s NCQ
- Windows 7 Professional 64.

The project has been implemented and tested within the Netbeans 7.0. The Glassfish server has been used to operate the energy management system and the web services. Google Chrome, Mozilla Firefox and Opera Browsers have been used to test the user interface.

For the part of semantic web, we have relied on the use of tools Protege and Jena, tested and implemented with Java applications and Java listeners.

To use the Web service of the Energy Management System we use a computer as a proxy to establish communication with the GENIO system. This computer communicates with an internal IP to the Energy Management System using the web service
client. Then the Web service methods are replicated with the client methods, and establish communication within the GENIO system through an external IP network.

### 4.2 Results

**Content Analysis and Workflow within the GENIO System**

The result obtained in the workflow for the GENIO system is shown in the figure below.

![User Interface Workflow Diagram](image)

Figure 4.1: User Interface Workflow

Energy Info shows a list of the appliances connected to the home. The interface lets the user see these appliances in one go or organized by rooms or personalized groups. Once the user has selected the desired appliance, from a list similar to the one seen in the Figure 5.2, the information about that appliance will be shown.
CHAPTER 4. PROVING CONCEPTS

Figure 4.2: Energy Info - Energy Management System

Figure 4.3: Appliance Info
CHAPTER 4. PROVING CONCEPTS

Figure 4.4: List of Appliances

Figure 4.5: List of Schedules
CHAPTER 4. PROVING CONCEPTS

Energy Information Web Service

- **Localhost Mode**
  File WSDL: \url{http://localhost:8080/PlugwiseDB2/PlugwiseWS?WSDL}
  Tester: \url{http://localhost:8080/PlugwiseDB2/PlugwiseWS?Tester}

![Figure 4.6: Localhost Mode](image)

- **Proxy Mode**
  IP External : 128.130.90.81
  Host External: genio or genio.ftw.at
  File WSDL: \url{http://genio.ftw.at:8080/ProxPlugwiseWS/PlugwiseWS?WSDL}
  Tester: \url{http://genio.ftw.at:8080/PlugwiseDB2/PlugwiseWS?Tester}

![Figure 4.7: Proxy Mode](image)
Figure 4.8: Methods of service

Component Test: Energy Autonomous System

Connection Web Server

1) Running Energy Management System and Web Server (Glassfish)
Initializing...
Deploying driver: C:/Users/rlaino/GlassFish_Server_01/glassfish/domains/
domain1/lib
Could not copy C:/Users/rlaino/GlassFish_Server_01/glassfish/domains/
domain1/lib to C:/Users/rlaino/GlassFish_Server_01/glassfish/domains/
domain1/lib
Starting GlassFish Server 3.1
GlassFish Server 3.1 is running.
Incrementally deploying PlugwiseDB2
Completed incremental distribution of PlugwiseDB2
run-deploy:
Browsing: http://localhost:8080/PlugwiseDB2
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 40 seconds)

2) Tester Web Server Open URL in localhost Tester: http://localhost:8080/
PlugwiseDB2/PlugwiseWS?Tester
Or connection proxy from FTW company: http://genio.ftw.at:8080/ProxyPlugwiseWS/
PlugwiseWS?Tester

PlugwiseWS Web Service Tester

This form will allow you to test your web service implementation [WSIDL File]
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods:
public abstract java.lang.String fswPlugwiseWS.hello(java.lang.String)

public abstract double fswPlugwiseWS.getPowerLevel()

public abstract double fswPlugwiseWS.getConsumptionToday()

public abstract double fswPlugwiseWS.getConsumptionTotal()

public abstract java.lang.String fswPlugwiseWS.getSwitchStatus()

public abstract java.lang.String fswPlugwiseWS.getLockedStatus()

public abstract java.lang.String fswPlugwiseWS.setSwitchAction(int,int)

Figure 4.9: Tester
3) Message WSDL (Request) – List Array of the Appliances
OUTPUT: Glassfish Server 3.1 [localhost]
INFO: parsing WSDL...
INFO: Generating code...
INFO: Compiling code...
INFO: Invoking wsimport with http://localhost:8080/PlugwiseDB2/PlugwiseWS?WSDL
INFO: wsimport successful
OUTPUT: Glassfish Server 3.1 [proxy FTW]
INFO: parsing WSDL...
INFO: Generating code...
INFO: Compiling code...
INFO: Invoking wsimport with http://genio.ftw.at:8080/ProxyPlugwiseWS/PlugwiseWS?WSDL
INFO: wsimport successful
List Array of the Appliances

```java
public abstract java.util.List<java.lang.String> getArrayListAppliances(java.lang.String)
getArrayListAppliances Method invocation
```

Method parameter(s)

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

SOAP Request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns2="http://service.plugwise.ftw.plainio/">
    <soap:Body>
        <ns2:getArrayListAppliances xmlns:ns2="http://service.plugwise.ftw.plainio/">
            <Tag1></Tag1>
        </ns2:getArrayListAppliances>
    </soap:Body>
</soap:Envelope>
```

Figure 4.10: Tester - getArrayListAppliances Method - SOAP Request
4) Message WSDL (Respond) – List Array of the Appliances

Method returned

```java
java.util.List : "[1;Computer, 2;Telephone]"
```

SOAP Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ns1="http://service.plugwise.ftw.at"
    xmlns:ns2="http://service.plugwise.ftw.at">
  <soap:Body>
    <ns1:getArrayListAppliancesResponse xmlns:ns1="http://service.plugwise.ftw.at"/>
  </soap:Body>
</soap:Envelope>
```

Figure 4.11: Tester - getArrayListAppliances Method - SOAP Response

5) Description Connection of WS

a) Configuration the host name in `C:/WINDOWS/system32/drivers/hosts` with “IP external `genio`“ or “IP external `genio.ftw.at`“.

   In the test is IP External: 128.130.90.81 then put into hosts(file) “128.130.90.81 genio”

b) It provides two files: `PlugwiseWS.wsdl` and `PlugwiseWS.xsd`. These files are for configuring the client server within the GENIO project.

Figure 4.12: Client Service
Integration Test: Decision Autonomous System and Energy Autonomous System (or EMS)

The goal of this test is to check if the energy monitoring information sent by the Energy Autonomous system is received correctly in the Decision Autonomous System. Each data will be printed when sent and when received. The Decision Autonomous System has the possibility to take information from several sources and to merge information and give a final automatic decision or alarm.

Figure 4.13: Diagram of the system working as Decision Autonomous System inside Smart Home

Information generated in Decision Autonomous System

Energy consumption Information and home automation sensors: home sensors are mapped to numeric attributes inside the inference engine.
13:44:38,514 TRACE [PowerServiceProbe] Alarm[0]: '3498;1;1;6.3;17.7;2011-08-26 13:44:42.0'
detected in power service, the Appliance with id 1 has a power limit of 6.3 but it is using 17.7 watts'

- Method used for the test:

  ```java
  public java.lang.String[] getArrayAlarmTimeNow(java.lang.String Tag1, long timeBefore)
  Get alarms from a time before parameter(timeBefore)

  Parameters:
  Tag1 - separator string between elements -> ex. ","  
  timeBefore - last alarms produced in the time before.

  Returns:
  String[] - Array with struct -> [ idAlarm ; idMonitor ; idType ; type ; powerLevelAlarm ; powerLevelDetected ; time ]

  ex. Time: 11:30:20
  timeBefore(ms): 5 min = 5*60000 = 300000
  Alarms after the 11:25:20
  [ 497;18:1;0;30.0;31.5;2011-11-04 11:28:31.0 ,
  498;18:1;0;30.0;31.6;2011-11-04 11:27:41.0 , ...]
  ```

Figure 4.15: Information generated in Energy Autonomous System (Alarm)
Integration Test: User Interface and Energy Autonomous System

State of the system - Energy Management System
Initial state of Appliance with id (1) is ON

Figure 4.16: Appliances Plugwise - Energy Management System - Appliance with id (1) is ON

Getting some info from the device - User Interface within the GENIO System
UIServices: Showing data of Device with id: 1
UIServices: Device is on
UIServices: Device is using: 27.6

Switching off the device from the UI - User Interface within the GENIO System
UIServices: Setting status of 1 to off

Figure 4.17: Appliances Plugwise - Energy Management System - Appliance with id (1) is OFF
Semantic Test: Switching off appliances primarily with respect to a defined priority list and a power level of the schedule.

The test schedule is programmed with 80 watts. Initializes the Monitor Ontology the implementation section and also the Jena Listener Alarm Schedule. Every 10 seconds checks and updates the data.

When we have two or more appliances with the same priority, we can determine if to switch off an appliance with higher power (MAX) or lower power level (MIN). For this case we use higher power (MAX). During the test, the appliances are turned off in stages. And once a device is turned off the condition is checked to switch ON/OFF.
5.1 Smart Home

A smart home is: "Home automation, also called domotics, is the residential extension of "building automation". It is automation of the home, housework or household activity. Home automation may include centralized control of lighting, HVAC (heating, ventilation and air conditioning), appliances, and other systems, to provide improved convenience, comfort, energy efficiency and security." ¹

5.2 Smart Grid

“Smart grid” generally refers to a class of technology 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 communication technology 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 of electricity in homes and businesses. They offer many benefits to utilities and consumers – mostly seen in big improvements in energy efficiency on the electricity grid and in the energy users’ homes and offices².

The major driving forces to modernize current power grids can be divided in four, general categories.

- Increasing reliability, efficiency and safety of the power grid.

- Enabling decentralized power generation so homes can be both an energy client and supplier (provide consumers with an interactive tool to manage energy usage, as net metering).

- Flexibility of power consumption at the clients side to allow supplier selection (enables distributed generation, solar, wind, biomass).

- Increase GDP by creating more new, green-collar energy jobs related to renewable energy industry manufacturing, plug-in electric vehicles, solar panel and wind turbine generation, energy conservation construction.

From a solution perspective, the smart grid is characterized by:

- More efficient energy routing and thus an optimized energy usage, a reduction of the need for excess capacity and increased power quality and security

- Better monitoring and control of energy and grid components

- Improved data capture and thus an improved outage management

- Two-way flow of electricity and real-time information allowing for the incorporation of green energy sources, demand-side management and real-time market transactions

- Highly automated, responsive and self-healing energy network with seamless interfaces between all parts of the grid.

### 5.3 Electric Car

An electric car is an automobile which is propelled by electric motor(s), using electrical energy stored in batteries or another energy storage device.

Electric cars are a variety of electric vehicle (EV); the term "electric vehicle" refers to any vehicle that uses electric motors for propulsion, while "electric car" generally refers to road-going automobiles powered by electricity. While an electric car’s power source is not explicitly an on-board battery, electric cars with motors powered by other energy sources are generally referred to by a different name: an electric car powered by sunlight is a solar car, and an electric car powered by a gasoline generator is a form of hybrid car. Thus, an electric car that derives its power from an on-board battery pack is a form of battery electric vehicle (BEV). Most often, the term "electric car" is used to refer to pure battery electric vehicles.

A Smart grid allows BEVs to provide power to the grid, specifically:
- During peak load periods, when the cost of electricity can be very high. These vehicles can then be recharged during off-peak hours at cheaper rates while helping to absorb excess night time generation. Here the batteries in the vehicles serve as a distributed storage system to buffer power.

- During blackouts, as an emergency backup supply.

### 5.4 Smart Meter

A smart meter is usually an electrical meter that records consumption of electric energy in intervals of an hour or less and communicates that information at least daily back to the utility for monitoring and billing purposes. Smart meters enable two-way communication between the meter and the central system. Unlike home energy monitors, smart meters can gather data for remote reporting. Such an advanced metering infrastructure (AMI) differs from traditional automatic meter reading (AMR) in that it enables two-way communications with the meter.

### 5.5 Smart Sensor

A sensor is a device that converts a physical or biological quantity into electrical quantity. The measured electrical quantity should be calibrated, converted to digital format and sent to the microcontroller for further processing and control. Most of the sensors, irrespective of their types, can be included as part of a ubiquitous embedded system that has communication capabilities and backend connectivity. These types of sensors are called sometimes smart sensors. These enable software development and data analysis using embedded processing capabilities as well as sending remote processing by a computing system located at some other location.  

### 5.6 Wireless Sensor Network

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, vibration, pressure, motion or pollutants and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such as industrial process monitoring and control.
machine health monitoring, and so on.

The main characteristics of a WSN include

- Power consumption constraints for nodes using batteries or energy harvesting
- Ability to cope with node failures
- Mobility of nodes
- Dynamic network topology
- Communication failures
- Heterogeneity of nodes
- Scalability to large scale of deployment
- Ability to withstand harsh environmental conditions
- Ease of use
- Unattended operation
- Power consumption
Conclusion and Outlook

In this last chapter we recapitulate the work described in this thesis and point out plans for the future. First we will revisit the goals and objectives from chapter 2.

6.1 Evaluation

Connecting to the Plugwise

We have successfully created a communication through the java classes by using DOM. In this way, we get information and interact with the Plugwise through multiple computers or multiple users. Security parameters of login and password are used for this communication. We get a fast and secure data stream.

Creating an Ontology Model

We created an ontology that meets our needs. It was important to be able to create a model to establish our queries. Then we get to perform a system based on semantic technology and fully autonomous. This model can be reused for other applications for checking and interaction of appliances, groups and rooms.

Creating Schedules

We have successfully designed, a weekly calendar with fractions of hours and fractions of 15 minutes. This weekly calendar describes the power level an appliance, group or room should have. It is stored in a database and can create more calendars, update and delete. It has also been integrated into the ontology to interact with the Plugwise.
Creating an Energy Management System

Finally we realized the energy management system with the integration of user interface and web services.

User Interface  We elaborated on the usefulness of a workflow engine for user interface composition and integrated the different features that the energy management system must have. The interface monitoring and programming schedules have been successfully added with semantic technology.

Web service  We have defined and established the methods needed to replicate the functionality out of the energy management system. So it fulfills the requirements of being a flexible solution for different clients through implementation of a web service within the GENIO System. This method opens the possibility to integrate future decision-support, autonomous applications.

6.2 Future work

Although our objectives have been achieved, some small details in the implementation remain open as improvements and further work.

- Electric car: Include schedules for battery recharging vehicles. Using the excess production of the factories when the tariff is low.

- WSN: Add wireless sensors network to monitor the consumption of heating and air conditioning of rooms and buildings using the Energy Management System.

- Scalability for the methods used in the energy management system in Smart Grids.

- Add the ontology in a repository on local network or external network with OpenRDF (Sesame) or similar. To SmartHomes and SmartGrids.

- Create a custom application for Android

- Add a web server with REST technology.

- Add authorization through API Key or other security parameters for the web service.
6.3 Summary

We have created an application programming interface for communication and interaction with Plugwise. With this API one can create different software programs for managing energy.

With the use of ontology and the creation of an API, we can successfully perform an autonomous system. Adding a list of priority and schedules, we have a decision support for efficient energy consumption.

Finally, we provided a web based user interface front end and incorporated a web service using WSDL and SOAP messages to interact with the GENIO system.

As result, we have fully completed an energy management system, versatile and open for improvement.
Acronyms

AJAX  Asynchronous JavaScript + XML
AMI  Advanced Metering Infrastructure
AMR  Automatic Meter Reading
API  Application Programming Interface
BEV  Battery Electric Vehicle
CCK  Content Construction Kit
CMIS  Content Management Interoperability Services
CMS  Content Management System
CRUD  Create Read Update Delete
EMML  Enterprise Mashup Markup Language
EV  Electric Vehicle
FTP  File Transfer Protocol
GNU  GNU’s Not Unix
GDP  Gross domestic product
HTML  HyperText Markup Language
HTTP  HyperText Transfer Protocol
IDE  Integrated Development Environment
APPENDIX A. ACRONYMS

IT  Information Technology
JSON  JavaScript Object Notation
OASIS  Organization for the Advancement of Structured Information Standards
PDO  PHP Data Objects
RDF  Resource Description Framework
REST  Representational State Transfer
RFC  Request for Comments
ROA  Resource Oriented Architecture
RPC  Remote Procedure Call
RSS  Really Simple Syndication
SOA  Service Oriented Architecture
SOAP  Simple Object Access Protocol
UDDI  Universal Description, Discovery and Integration
UI  User Interface
URI  Uniform Resource Identifier
URL  Uniform Resource Locator
W3C  World Wide Web Consortium
WEKA  Waikato Environment for Knowledge Analysis
WADL  Web Application Description Language
WSDL  Web Services Description Language
WSN  Wireless Sensor Network
WS-BPEL  Web Services Business Process Execution Language
WS-CDL  Web Services Choreography Description Language
WWW  World Wide Web
XHTML  eXtensible HyperText Markup Language
XML  eXtensible Markup Language
XSD  XML Schema Definition
In this appendix I will lay out some technology foundations that are necessary for my project. It will cover existing concepts and approaches that form a basis for the developments I am going to present in chapter 3.

**B.1 Common protocols and standards**

There are a lot of standards around the web, so I will cover some common ones, which will later be mentioned and referenced.

**XML** The eXtensible Markup language is a data format or more generically a way to define data formats. It has consistent and clean text tagging, it separates content from format and allows hierarchical data structures. It also has facilities for user-definable data structures, which is a central feature needed by web services.

**HTTP** The Hypertext Transfer Protocol is the standard application layer protocol to exchange hypermedia and other resources on the web. It is designed for client-server style request-response communication patterns and is stateless, which means that every request-response interaction is independent from any other. HTTP is a lightweight protocol and is widely used and implemented on many systems.

**B.2 Semantic Web**

The main purpose of the Semantic Web is driving the evolution of the current Web by enabling users to find, share, and combine information more easily.

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative
effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF).

The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly whereas concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. This allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing. The term “Semantic Web” is often used more specifically to refer to the formats and technologies that enable it. The collection, structuring and recovery of linked data are enabled by technologies that provide a formal description of concepts, terms, and relationships within a given knowledge domain. These technologies are specified as W3C standards and include:

- Resource Description Framework (RDF), a general method for describing information
- RDF Schema (RDFS)
- Simple Knowledge Organization System (SKOS)
- SPARQL, an RDF query language
- Notation3 (N3), designed with human-readability in mind
- N-Triples, a format for storing and transmitting data
- Turtle (Terse RDF Triple Language)
- Web Ontology Language (OWL), a family of knowledge representation languages

The Semantic Web Stack illustrates the architecture of the Semantic Web. The functions and relationships of the components can be summarized as follows:

- XML provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within. XML is not at present a necessary component of Semantic Web technologies in most cases, as alternative syntaxes exists, such as Turtle. Turtle is a de facto standard, but has not been through a formal standardization process.

- XML Schema is a language for providing and restricting the structure and content of elements contained within XML documents.

---

1 [http://www.w3.org/2001/sw/]
• RDF is a simple language for expressing data models, which refer to objects ("resources") and their relationships. An RDF-based model can be represented in a variety of syntaxes, e.g., RDF/XML, N3, Turtle, and RDFa. RDF is a fundamental standard of the Semantic Web.

• RDF Schema extends RDF and is a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes.

• OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g., disjointness), cardinality (e.g. "exactly one"), equality, enhanced typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

• SPARQL is a protocol and query language for semantic web data sources.

Below are several parts of the semantic web technology used in this project:
Jena

Jena is a Java framework for building Semantic Web applications. Jena provides a collection of tools and Java libraries to help you develop semantic web and linked-data apps, tools and servers.\(^2\)

The Jena Framework includes:

- an API for reading, processing and writing RDF data in XML, N-triples and Turtle formats;
- an ontology API for handling OWL and RDFS ontologies;
- a rule-based inference engine for reasoning with RDF and OWL data sources;
- stores to allow large numbers of RDF triples to be efficiently stored on disk;
- a query engine compliant with the latest SPARQL specification
- servers to allow RDF data to be published to other applications using a variety of protocols, including SPARQL

Protégé

Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies. \(^3\)

OWL Ontologies  Ontologies are used to capture knowledge about some domain of interest. An ontology describes the concepts in the domain and also the relationships that hold those concepts. Different ontology languages provide different facilities.

OWL ontologies have similar components to Protégé frame based ontologies. However, the terminology used to describe these components is slightly different from that used in Protégé. An OWL ontology consists of Individuals, Properties and Classes, which roughly correspond Protégé frames Instances, Slots and Classes.


\(^3\)Welcome to the Protégé wiki! : http://protegewiki.stanford.edu/wiki/Main_Page
Individuals, represent objects in the domain in which we are interested, also known as the domain of discourse. Properties are binary relations (relation between two things). It can be either object properties (relation between classes) or data properties (relation between classes and values). OWL classes are interpreted as sets that contain individuals.

Standard backend plugins include support for storing and importing ontologies in RDF Schema, XML files with a DTD, XML Schema files, DAML+OIL and OWL. The OWL Plugin provides a comprehensive OWL Editor environment that supports loading and writing of OWL files.

**SPARQL**

SPARQL (pronounced "sparkle", a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format\(^4\). As a query language, SPARQL is "data-oriented" in that it only queries the information held in the models; there is no inference in the query language itself. Of course, the Jena model may be ‘smart’ in that it provides the impression that certain triples exist by creating them on-demand, including OWL reasoning. SPARQL does not do anything other than take the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings or an RDF graph.\(^5\)

**Query forms** The SPARQL language specifies four different query variations for different purposes.

- **SELECT query** Used to extract raw values from a SPARQL endpoint, the results are returned in a table format.

- **CONSTRUCT query** Used to extract information from the SPARQL endpoint and transform the results into valid RDF.

- **ASK query** Used to provide a simple True/False result for a query on a SPARQL endpoint.

- **DESCRIBE query** Used to extract an RDF graph from the SPARQL endpoint, the contents of which is left to the endpoint to decide based on what the maintainer deems as useful information.

\(^4\) [SPARQL - W3C Link: http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/](http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/)

Each of these query forms takes a WHERE block to restrict the query although in the case of the DESCRIBE query the WHERE is optional.

### B.3 User interface - Web Page

**HTML**

HTML is, as the name Hyper Text Markup Language suggests, a markup language. It is predominantly used for creating web pages and as such has become very famous and widespread.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like `<html>`), within the web page content. HTML tags most commonly come in pairs like `<h1>` and `</h1>`, although some tags, known as empty elements, are unpaired, for example `<img>`. The first tag in a pair is the start tag, the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, tags, comments and other types of text-based content. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts in languages such as JavaScript which affects the behavior of HTML webpages.

Web browsers can also refer to Cascading Style Sheets (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS standards, encourages the use of CSS over explicitly presentational HTML markup. This project has developed a specific CSS for our user interface.  

**Servlet**

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by Web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.

---

6 HTML™ 1.0 The Extensible HyperText Markup Language - Link: [http://www.w3.org/TR/html/](http://www.w3.org/TR/html/)

7 HTML 4.01 Specification - Link: [http://www.w3.org/TR/1999/REC-html401-19991224/](http://www.w3.org/TR/1999/REC-html401-19991224/)

8 The J2EE Tutorial (Java Servlet Technology) - Link: [http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html](http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html)
The `javax.servlet` and `javax.servlet.http` packages provide interfaces and classes for writing servlets. All servlets must implement the `Servlet` interface, which defines lifecycle methods.

When implementing a generic service, you can use or extend the `GenericServlet` class provided with the Java Servlet API. The `HttpServlet` class provides methods, such as `doGet` and `doPost`, for handling HTTP-specific services.

**Javascript**

JavaScript is a prototype-based scripting language that is dynamic, weakly typed and has first-class functions. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript is used in billions of Web pages to add functionality, validate forms, communicate with the server, and much more.

JavaScript was formalized in the ECMAScript language standard and is primarily used in the form of client-side JavaScript, implemented as part of a Web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.\(^9\)

JavaScript\(^{(jqPlot)}\) was used in this project for the graphic representation of results. For the schedules and monitors of power level. So getting a clearer interpretation for the user.

**Graphs - jqPlot**

jqPlot\(^{10}\) is a plotting and charting plugin for the jQuery Javascript framework. jqPlot produces beautiful line, bar and pie charts with many features:

- Numerous chart style options.
- Date axes with customizable formatting.
- Up to 9 Y axes.
- Rotated axis text.
- Automatic trend line computation.
- Tooltips and data point highlighting.
- Sensible defaults for ease of use.

---


\(^{10}\)jqPlot pure java plotting - Link: [http://www.jqplot.com/](http://www.jqplot.com/)
Computation and drawing of lines, axes, shadows even the grid itself is handled by pluggable "renderers". Not only are the plot elements customizable, plugins can expand functionality of the plot too. There are plenty of hooks into the core jqPlot code allowing for custom event handlers, creation of new plot types, adding canvases to the plot, and more.

B.4 Web service

JAX-WS

JAX-WS stands for Java API for XML Web Services. JAX-WS is a technology for building web services and clients that communicate using XML. JAX-WS allows developers to write message-oriented as well as RPC-oriented web services.

In JAX-WS, a web service operation invocation is represented by an XML-based protocol such as SOAP. The SOAP specification defines the envelope structure, encoding rules, and conventions for representing web service invocations and responses. These calls and responses are transmitted as SOAP messages (XML files) over HTTP.

Although SOAP messages are complex, the JAX-WS API hides this complexity from the application developer. On the server side, the developer specifies the web service operations by defining methods in an interface written in the Java programming language. The developer also codes one or more classes that implement those methods. The client programs are also easy to code. A client creates a proxy (a local object representing the service) and then simply invokes methods on the proxy. With JAX-WS, the developer does not generate or parse SOAP messages. It is the JAX-WS runtime system that converts the API calls and responses to and from SOAP messages.

With JAX-WS, clients and web services have a big advantage: the platform independence of the Java programming language. In addition, JAX-WS is not restrictive: a JAX-WS client can access a web service that is not running on the Java platform, and vice versa. This flexibility is possible because JAX-WS uses technologies defined by the World Wide Web Consortium (W3C): HTTP, SOAP, and the Web Service Description Language (WSDL). WSDL specifies an XML format for describing a service as a set of endpoints operating on messages.11

WSDL

The Web Service Description Language is an XML vocabulary to specify metadata for web services like where and how clients can invoke the service and what operations and arguments are available. WSDL is extensible and is designed as a machine-readable

11The Java EE 5 Tutorial (Oracle)- Link: http://docs.oracle.com/javaee/5/tutorial/doc/index.html
format, so that service consumer agents can pick up the necessary information about the service automatically. Currently WSDL 1.1 is the dominant version that is widely accepted, however WSDL 2.0 has been released as W3C recommendation in 2007, but has not been adopted by the industry yet. An alternative to WSDL is the Web Application Description Language (WADL), also an XML based description standard but intended specifically for RESTful web services.

- **WSDL 1.1** is the most used standard for WS* services, but lacks capabilities to fully describe RESTful service characteristics.

- **WSDL 2.0** is the new standard and provides great flexibility to also describe RESTful services, but it is not in wide spread use and can be considered unsupported by most platforms.

- **WADL** The Web Application Description Language is an XML based standard as well and was specifically developed for RESTful services as counterpart to WSDL in the WS* world. It is well founded but is also not that common in real world service implementations.

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types** a container for data type definitions using some type system (such as XSD).

- **Message** an abstract, typed definition of the data being communicated.

- **Operation** an abstract description of an action supported by the service.

- **Port Type** an abstract set of operations supported by one or more endpoints.

- **Binding** a concrete protocol and data format specification for a particular port type.

---


13 WSDL Current Status - Link: [http://www.w3.org/standards/techs/wadl](http://www.w3.org/standards/techs/wadl)
• **Port** a single endpoint defined as a combination of a binding and a network address.

• **Service** a collection of related endpoints.

**SOAP**

The Simple Object Access Protocol is a standard to issue remote procedure calls and send/receive messages over the Internet. Commonly it uses HTTP as underlying transport protocol, but can be used with others as well. Messages are encoded with XML and consist of an envelope for namespace definitions, an optional header for additional information (e.g. security, addressing etc.) and a body containing the message data itself, i.e. service operations and their arguments. There are two types of messages: service requesters send SOAP Requests and service providers send back SOAP Responses.

**B.5 Data base**

**JDBC-ODBC**

JDBC is a Java-based data access technology (Java Standard Edition platform) from Sun Microsystems, Inc.. This technology is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database. JDBC is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the JVM host environment.\(^\text{14}\)

JDBC connections support creating and executing statements. These may be update statements such as SQL’s CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. JDBC represents statements using one of the following classes:

• **Statement** the statement is sent to the database server each and every time.

• **PreparedStatement** – the statement is cached and then the execution path is predetermined on the database server allowing it to be executed multiple times in an efficient manner.

• **CallableStatement** – used for executing stored procedures on the database.

Open Database Connectivity (ODBC) is a standard software API specification for using database management systems (DBMS). ODBC is independent of programming language, database system and operating system.

The ODBC API is a library of ODBC functions that let ODBC-enabled applications connect to any database for which an ODBC driver is available, execute SQL statements, and retrieve results.

The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system (DBMS) is handling the data. ODBC achieves this by inserting a middle layer called a database driver between an application and the DBMS. This layer translates the application’s data queries into commands that the DBMS understands.  

**MySQL**

MySQL is the world's most used relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language.

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC. Any system that works with ODBC or JDBC can use MySQL.

In this project was used Connector/J which is standardized database driver for Java platforms development and Connector/ODBC which is Standardized database driver Windows, Linux, Mac OS X, and Unix platforms.

---

15 Microsoft ODBC Overview - Link: [http://support.microsoft.com/kb/110093](http://support.microsoft.com/kb/110093)

C.1 Using XML and HTML files

The power management system must start from the creation of a communication via XML. Defining the structure and parameters. We must make a division of the appliances, groups and rooms. And determine what types of actions or interactions can be done calls through URI of a HTML.

The uniform resource identifier (URI) is a string of characters used to identify a name or a source. Such identification enables interaction with representations of the source over a network (typically the World Wide Web) using specific protocols. Schemes specifying a concrete syntax and associated protocols define each URI.

Within the URI we have a parameter query. The query is an optional part, separated by a question mark ("?"), that contains additional identification information that is not hierarchical in nature. The query string syntax is not generically defined, but it is commonly organized as a sequence of <key>=<value> pairs, with the pairs separated by a semicolon or an ampersand. Its use will help us to generate actions through the website.

C.2 Using Parsers, Action, Monitoring and Alarms

We will use Java classes for the extraction of the structure of XML, to help us get the parameters for the energy management system. Using the information of the parser can enable actions to appliances, groups and rooms. We may also make use of monitors and alarms to monitoring of appliances, groups and rooms. We get new features for use in new applications.
C.3 Using Data Base and RDF

The use of database and will be useful for interpretation for both the computer and the user. Using RDF we can get an autonomous system for managing energy, without the continuous intervention of the user. This advantage greatly improves the use and interpretation of data continuously.

C.4 Using Ontology and Semantic Web

This section discusses the importance of a model for the ontology, we need a model and rules for the devices, groups and rooms. Then we can create a schedule for using automatic queries made periodically to the ontology.

C.5 Using Web Methods

For communication outside the local use will be important to use a Web server and be able to perform methods of communication to customers. For this project we used a client within the GENO system.
List of Figures

2.1 Overall system architecture and interaction within the GENIO System . . . 6
3.1 Plugwise Source . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
3.2 Schema files Plugwise XML & HTML . . . . . . . . . . . . . . . . . . . . 9
3.3 XML Code Structure . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.4 XML - Resolved . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
3.5 Arquitectue Parser . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
3.6 EMS Actions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.7 Schema of relationship with classes and objects . . . . . . . . . . . . . . . 18
3.8 Schema of the Appliance . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
3.9 Schema of the Group . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
3.10 Schema of the Room . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
3.11 Schema of the Schedule . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
3.12 Creating class, object properties and data properties . . . . . . . . . . . . . 26

85
LIST OF FIGURES

3.13 Query of the appliances, groups, rooms .................................. 27
3.14 Reasoner of the appliances, groups and rooms .......................... 27
3.15 User interface of monitoring .................................................... 28
3.16 Creating class, object properties and data properties ................. 29
3.17 Query of the schedules ............................................................ 29
3.18 Reasoner of the schedules ....................................................... 30
3.19 Reasoner of isTooHigh ............................................................ 30
3.20 Query of isTooHigh ................................................................. 31
3.21 User interface of scheduling ...................................................... 31
3.22 Communication SOAP client and server .................................... 32
3.23 User Interface: index.xhtml ....................................................... 34
3.24 User Interface: Java Servlet - Appliance ................................. 35
3.25 Functions Java API: applianceServlet.java .............................. 35
3.26 Algorithm Description - Appliances UI .................................. 36
3.27 User Interface: Java Servlet - Group ...................................... 37
3.28 Functions Java API: groupServlet.java ................................... 37
3.29 Algorithm Description - Groups UI ........................................ 38
3.30 User Interface: Java Servlet - Room ....................................... 39
3.31 Functions Java API: roomServlet.java ...................................... 39
3.32 Algorithm Description - Rooms UI .......................................... 40
3.33 User Interface: Java Servlet - Module ...................................... 41
3.34 Functions Java API: moduleServlet.java .................................. 41
3.35 Algorithm Description - Modules UI ....................................... 42
3.36 User interface of scheduling - Java Servlet - Schedule ............. 43
3.37 Functions Java API: scheduleServlet.java ................................. 43
3.38 UI of scheduling - Update - Weekdays ................................... 44
3.39 priorityServlet.java - Main Contents - Options Flags Jena ........ 46
3.40 index.xhtml - Main Menu - Start Monitor and Listener ............. 46
3.41 User Interface: Java Servlet - Priority .................................... 47
3.42 Functions Java API: priorityServlet.java ................................. 47
3.43 Schema Integration and interaction within the GENIO System .... 48

4.1 User Interface Workflow ......................................................... 50
4.2 Energy Info - Energy Management System ............................... 51
4.3 Appliance Info ................................................................. 51
4.4 List of Appliances ............................................................... 52
4.5 List of Schedules ............................................................... 52
4.6 Localhost Mode ................................................................. 53
4.7 Proxy Mode ................................................................. 53
4.8 Methods of service ........................................................... 54
4.9 Tester ................................................................. 55
List of Tables

3.1 List of Class ......................................................... 17
3.2 List of SubClass .................................................... 17
3.3 List of Object Properties ........................................... 18
3.4 List of Data Properties (Appliance) ............................... 19
3.5 List of Data Properties (Group) .................................. 20
3.6 List of Data Properties (Room) ..................................... 21
3.7 List of Data Properties (Schedule) ................................. 22
3.8 List of Data Properties (Group_Appliance) ....................... 23
3.9 List of Data Properties (Room_Appliance) ....................... 23
3.10 List of Data Properties (day_x_hour_ij_min_xy) .................. 23
3.11 List of Data Properties (Schedule_Appliance) .................. 24
3.12 List of Data Properties (Schedule_Group) ...................... 24
Listings

3.1 Codes of the methods of interaction - actions.html .................................. 11
3.2 Parser of the Appliance. Connection. applianceParser.java .................. 13
3.3 Parser of the Appliance. Get Power Level. applianceParser.java ............ 15
3.4 Action of the Appliance. Switch On. applianceAction.java ................... 16