Interim Report

Full Unit Project

Benedict Wilkins

Supervisor: Zhiyuan Lou

2016

Table of Contents

Foreword

This report assumes the reader has knowledge about the structure of the project – this was discussed in the Project Plan and for convenience is presented in the *Appendices* section at the end of the report. If the Project Plan has not been seen it is recommended that a reader starts there.

Smart Meters and Energy Demand Management

Energy Demand Management

Energy Demand Management is in essence about managing the consumption/production of energy to ensure an effective energy network and to minimize cost and environmental damage. One of the main issues that EDM faces is how to efficiently supply energy to consumers. Peaks in energy demand arise when consumers – domestic or industrial, have synchronized habits. These lead to energy demand fluctuations (EDF), daily, weekly and seasonally [1]. These fluctuations must be dealt with by energy provides so not to result in damage to the energy network, blackouts and unpredictable service.

Techniques for Supply Management

There are two general categories of power plant; base load and peaking. Base power plants are used to supply the base load power – the minimum requirement of energy over a period of time. They are usually nuclear, coal or large hydro-electric plants [2]. These plants are only turned off for maintenance or upgrades and usually provide power to a large area. Peaking power plants are one way to deal with EDF. They produce a variable amount of power which is matched to the current demand. Peaking power plants are expensive, heavily dependent on the fossil fuel market (as they are usually gas/coal burning plants) and are only usually used as a last resort after employing the methods mentioned below. See Figure 1.

Other techniques include energy storage and energy purchasing. Energy storage involves storing a large amount of energy and releasing it back to the grid during peak times. One of the most common examples of large-scale energy store are hydroelectric dams/pumps. Water may pumped during non-peak times or naturally build up behind a dam, later to be let through the turbines when it is required. A lot of energy is lost during this process and so it is not ideal. Energy purchasing is when an energy provider buys in energy from a separated grid. An example may be a British energy provider buying power from the French natation grid. Depending on the energy market this may be more desirable than using their own grid infrastructure.

Demand Side Management and Smart Meters

Moving to the consumer side of EDM. Another way of approaching the problem (which will be the focus of this project) is to alter the habits of the consumers to reduce EDF. This will result in reduced cost for both energy providers and consumers. Demand response (DR) uses financial incentives to encourage consumers to alter their energy consumption habits. This method has been tested in the health sector; altering habits that relate to health by heavily taxing cigarettes and alcohol for example, has shown to be effective [3]. There have been various models that support DR in this projects context, see [4]. The National Grid (NG) in the UK has previously had meters to monitor electricity demand on a larger scale).

Now however, with the advent of smart meters and their increasing popularity in the market, it is possible to collect house-hold specific data making it possible to implement DR.

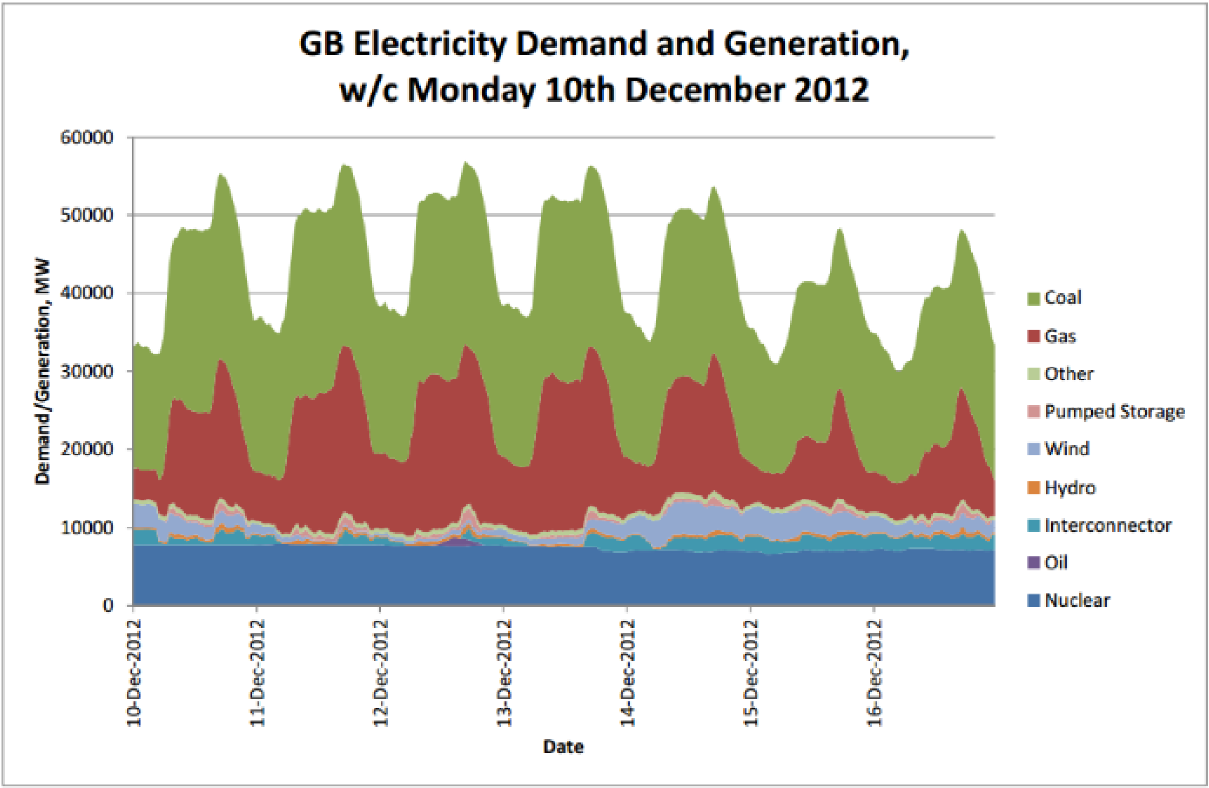


Figure 1. Taken from Special feature – Seasonal variations in electricity demand. See bibliography entry [1]. The above shows the power demand/generation on the GB National Grid from 10 – 16 of December 2012 starting at midnight (00:00). It illustrates daily energy demand fluctuations and shows the use of peaking and base load power plants. Nuclear power plants are shown here to be the base load power plants. It is clear to see that gas and coal fire power plants are being used to deal with the EDF and so are the peaking power plants.

Data Collection in Context

It is important to look at the type of data that the system to be developed will be using. According to the Smart Metering Implementation Program [5], *'GSME shall be capable of recording Consumption in each thirty minute period'.* (GSME - Gas Smart Metering Equipment), the same applies for Electric Smart Metering Equipment (ESME). There is also a daily recording option however this will not be suitable for the predictions that this project is concerned with. In principle the system will be able to support any reasonable time scale, but the most useful will be on an hourly time scale (because of the relatively large hourly EDF see figure 1.) The implementation details of the smart meter or what type of meter is irrelevant as long as the data is useable. With this in mind, the system isn't necessarily limited to house-holds. Data may be collected from businesses/industrial settings. However it may be useful to make the distinction between the two as the scale/fluctuation pattern of each may be quite different.

Motivations

Machine learning is a heavily researched area in computing and is well suited in achieving this projects goal. Machine learning models require a few things to work but in essence; lots of data and a pattern. Both of those things seem to be present; data will be readily available from a large number of smart meters around the country and there is a pattern, looking a figure 1 gives a good indication that there is at least a daily pattern. ANNs should be able to model this pattern effectively and provide good predictions as a result. ANNs have been used previous to forecast for arguably more complex system including the stock market.

Using a multi-agent framework will provide a flexible but robust structure for the system, it allows the system to be easily distributed which is key if the system is to scale well. The real smart meters can be considered agents – they operate in some environment and their goal is to record energy related measurements from their environment. It will be useful to model the intermediate ‘processing’ layers as agents within layered environments. It will be easier to experiment with different pre-processing methods by switching agent behaviours, agents could also automatically report errors and/or statistics using different behaviours.

In conclusion, DR may provide a cheaper alternative to the supply management solutions presented above. There are some problems it cannot solve – for example large EDF when weather/seasons change, but it shows potential in helping reduce EDF on a day to day scale. Utilising Smart meters and current computing technology to create a more efficient energy grid will help reduce our impact on the environment and reduce cost for everyone involved.

Introduction to Multi-agent Systems

Intelligent Agents

There are a variety of definitions for what a software agent actually is. A good way to look at an agent is as a metaphor – the real world. We may consider a human as a real world agent. We exist in an environment - the universe, we have beliefs, desires and intentions which lead us to make decisions and ultimately perform actions which may or may not affect our environment. In essence this is what a software agent does but in a software context; it and its environment will be running on one or more machines. [6]

Continuing the metaphor, we as humans have a means to affect and observe our environment though use of our body. Parts of our body are used to perform actions in our environment – for example our hands can be used to pick up objects. In agent terms these parts are known as actuators or effectors. Other parts of our body are used for observation – for example our eyes, we look at the world and gather information about our environment. These parts are known as sensors. See figure 2 for an illustration of this idea. It is interesting to see how far this metaphor can be taken and how well it represents software agents – this will be discuss further in the *Agent Architecture* section.

The environment is key aspect of agent based systems. An environment is just some space in which the agents in the system are housed. Agent environments have a number of properties [7], in the later section *MAS in context* some attempt will be made to classify the environment of the system to be developed with these properties in mind.

In a lot of cases agents are in environment where other agents are also situated and so the need for communication between agents arises. These kind of systems are known as multi-agent systems.

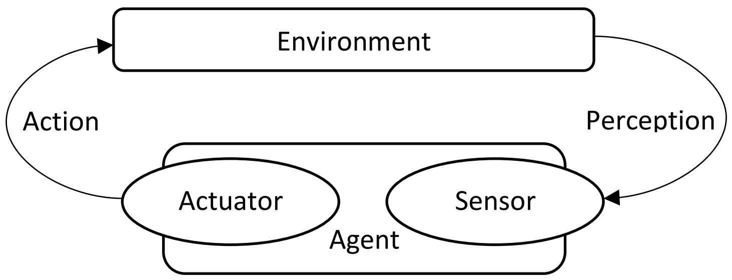


Figure 2. An illustration of an agent action on and observing an environment via is actuator and sensor.

Multi-agent Systems

Agents interact with others agents in a variety of different ways depending on context. These interactions can be classified as a result from the following two agent behaviours; cooperative and competitive. Cooperative agents work together to achieve a common goal while competitive agents work to achieve their individual goals. It is not unusual for cooperative agents to display competitive behaviour and vice versa [8]. If an agent is *clever* enough it is easy to see how a competitive agent may behave like it is a cooperative. It may temporarily work with other agents if it finds that this is a more effective way of achieving its own goal.

Communication between agents is an important aspect of a multi-agent systems (MAS) and it provides the basis for agent interaction. Cooperative agents must have a means of communication in order to complete their goal as a group. They may need to dynamically organise themselves or communicate information that may be useful to the other agents. Competitive agents may also need to communicate. An agent based e-commerce trading system is a good example of a need for competitive communication. Each agent in the system will represent a real world entity such as a person or corporation. Their goal is to negotiate the best deal for the entity they are representing. They will require some communication protocol in order to negotiation [9]. It is up to the agent designer to provide these communication protocols in a way that best suits the system.

Distributed agent environment

The nature of the smart grid forces the system to utilise some distributed agent environment. Smart meters are physically separate entities each with an environment that they are responsible for – the household of the consumer. The smart meter agents (SMA) will need to be capable of communicating information to physically separate systems. This can be done using an IP protocol. Each SMA will be capable of communicating in this way with its manager agent. There are a few different possible configurations of management agents all with different associated positives and negatives and are to be discussed in this section.

The hierarchical structure of the MAS implies groupings of the agents. The SMAs already have a natural grouping, by geographical location or address which is already used by all household service providers. The manager agent's job should be to forward data on behalf of its group as well as manage and control them by forwarding commands from higher in the hierarchy.

To keep the system as de-centralised as possible it may be desirable to have the SMAs doing management themselves. They could delegate one of the other agents in their group to be a manager. This configuration may avoid system failures; if the delegated agent fails, a different SMA can be delegated as the manager. There is a potential privacy issue with sending data to a Smart Meter in a household, some measures would have to be in place to prevent data theft or alteration at the management agent location.

The alternative configuration is to have the management agent reside further up the hierarchy. It would be more complicated to deal with system failures as the system would have no alternative management agent. However there will be no privacy issues with sending the data.

Multi-agent Systems in Context

There have been various other studies involving multi-agent systems and the smart grid [10] [11] [12]. W. *Multi-Agent Systems (MAS) controlled Smart Grid - A Review* in particular gives a good summary of relevant studies. Most are concerned with managing operations and control of the smart grid and some address problems such as the centralisation of power plants and service restoration.

This project is less concerned with the technology that will be used to implement the control/management of the smart grid after predictions have been made. However from the studies mentioned previously it seems likely that MAS will play a key role. This motivates the use of an extendable MAS in this project.

Classifying the agent environment:

(From afore mention standard environmental attributes (Russel and Norvig 1995))

To begin classifying the environment we must first define what exactly we mean in context. Each SMA is situated in its own 'house' environment in which it interacts (take readings). The house environment is however linked to the global environment; the environment containing all houses and any other agents in the system. This is the environment which we will try to classify as it contains all other environments.

Inaccessible – due to the nature of networking. Each SMA does not know what is happening in some other part of the environment. It could ask for more information from the other agents but again there is no guarantee of a reply. Similarly the manager agents (whether they are central or not) do not have direct access to the house environments.

Nondeterministic – again due to the nature of networking. It is possible that the system will fail at some point, an SMA may fail and so no data will be sent up the hierarchy. It should be noted that the receiving agents should be able to handle not receiving any data. We do not want the employed machine learning method to fail because of this.

Nonepisodic – This attribute is a little harder to pin down. It could be said that the SMAs will reside in an episodic environment, all they will do it send data regardless of success or failure in the previous episode. In the simulated system the agents will receive an energy usage value every logical[*] half hour interval. However it may be the case that in the real system the agent will be continuously monitoring the energy usage. Whether we consider the physical monitoring device part of the agent is up for debate and should be carefully considered in a fully deployable version of the system. The non-SMA (the agents higher up the hierarchy) are situated in a nonepisodic environment. Although data may be sent every half hour from the SMAs perspectives their time may be out of sync (this should be

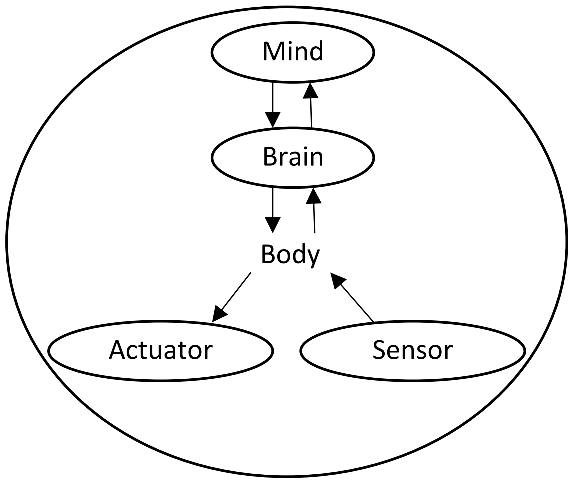
[*] The half hour interval will not be in real time, it will be scaled to simulation speed.

minimised as it will have an impact on prediction). The non-SMAs must be capable of continually receiving packets from the SMAs.

Dynamic – The environment will change independently of the each agent. Network failures and other agents may affect the environment.

Discrete – SMAs may only take an energy reading and send it. Non-SMAs may only receive messages, send messages.

Agent Architecture



The arrows represent the forwarding of events, results and actions.

- The mind generates an action based on some reasoning, this is forwarded to the brain, which forwards it to the body. The body then generates an event from the action which is forwarded to the actuator. The actuator then forwards the event to the environment.
- The sensor receives a result from the environment, this is the agent's perception. The sensor forwards it to the body and in the same fashion as above it reaches the mind as a result. The mind can then process the result as it sees it.

In Java: Each part of the agent is represented as a class, each class is an observer observable pair (each class can communicate via observer/observable design pattern with its respective parts). The environmental also uses the observer/observable design pattern in order to send and receive results and events respectively. The body class contains all of the parts of the agent and so is essentially the complete agent. A body may also have an appearance which is defined as being; the external appearance of the body – what the agent looks like to other entities in the environment.

The framework for the agents is given by the GAWL (Generic Agent World Library) package. This framework also includes packages for, actions, events, perceptions, environment, physics and the observer/observable implementation. GAWL is essentially a revised version of the GOLEM framework see [13] and [14].[*]

Perceive, Decide, Execute

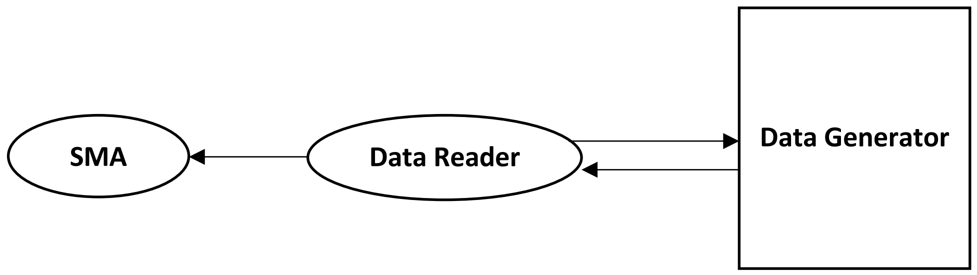
The architecture that is being used is an architecture used by the KGP model agency [15], where the agent control is based on a perceive-decide-act cycle, but implemented in Java rather than Prolog. In this project the agent control cycle will be explicitly referred to as the perceive-decide-execute cycle. The model is supported as a library within a revisited version of STARLITE [16] (STARLITE+), an agent platform that has been developed in the DICE lab at

[*] Note: GAWL was developed by Prof. Kostas Stathis, Emanuele Uliana and myself and is based upon work that Prof. Stathis (and colleagues) have worked on previously including GOLEM and STARLITE.

Royal Holloway (http://dice.cs.rhul.ac.uk) to support the deployment of communicating software agents. Additionally see [17].

Environment Architecture (Simulation)

House Environment

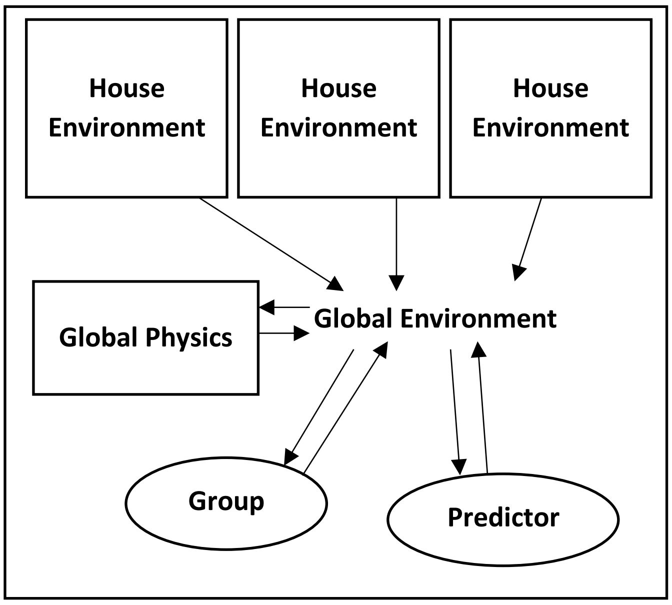


The SMA will receive a reading from the Data Reader agent – who in the simulation will be set to read a value from the Data Generator at a given time frame. The time frame will be decided at the beginning of the simulation, each frame will represent a half hour interval in the deployable system. The house environment/physics will be responsible for forwarding the Data Reader -> SMA message. Once the message is received the SMA will forward the data up the agent hierarchy. The Data generator module will be global and will depend on the set up of the simulation. It will provide data reading from all houses based on their properties e.g. what financial positon the house is in.

Environment and Physics

Each house has its own physics and environment, these are responsible for message forwarding within a house and will evaluate events that pass through them making sure they are valid in context. For example a house environment will receive an event from the Data Reader agent, it will forward it to the house physics which will evaluate it in context, make sure it is possible and valid. It will then execute the action (in this case executing the action means giving it back to the environment enabling a forward to the SMA).

Global Environment and Physics



The arrows represent the transfer of events. Each SMA in its respective House Environment will send a message to the Global Environment. This will be processed (sent to the physics) and given to the Group agent. The Group agent may be a neighbourhood, region, country or other defined at the start of the simulation. The Group agent will aggregate the data in some way and format it so that it may be useful to the Predictor agent. The aggregated data will be sent in the same manner from the Group agent to the Predictor agent.

There will be room in the architecture to provide multi group agents for different aggregations if needed. As an example we may have multiple neighbourhoods about which we want to make an aggregated prediction (a prediction regarding all houses in each neighbourhood). This can be done by simply having two Group (neighbourhood) agents with their respective House Environments. There is an obvious bottle neck at the Global environment/physics, this may be avoided by making a logical partition of the global environment e.g. for each group; a version of the global environment will be created to handle their communication specifically. In the real system the idea of Global environment will not hold as strong. The SMAs will forward their messages via IP communication – the machine who receives the message will essentially be the Global Environment.

Software Engineering in the Java Based Multi-Agent System

Design patterns

State

Allows an object to change its behaviour depending on its internal state by effectively changing its class.

The state design pattern has been used in many areas of my code. One example, the *SmartMeterAgentBody* class *brainHandler* attribute. The *brainHandler* attribute can be in one of two states – *NormalBrainHandler* or *IPBrainHandler*. This depends on the *Actuator* given to the agent at instantiation. One state allows the agent to handle *IPCommunicationActions* the other allows any non-IP related action e.g. *RecordAction*.

Observer/Observable

The observer design pattern is used for event parsing between objects without explicit method calls. In java the *update* and *notifyObervers* methods are used to do this.

A custom version of the java implementation is used in GAWL for event parsing between agent parts, agents and environments, and environments and their associated physics. For of these concepts there is one or more classes in the system that represent them e.g. *NationalGridUniverse, SmartMeterAgentBody, HouseEnvironmentPhysics* etc.

Factory

The factory design pattern provides an abstraction from object creation. No knowledge of the creation of the object is required, only the method in the factory for the desired object.

The *HouseModelFactory* class is one example of the factory design pattern in the system. It abstracts from house model creation, only an error term is required for creation.

Singleton

The singleton pattern is used to ensure that only one instance of a class can be created. This instance can be accessed in a static way, the class is responsible for the creation of this single instance.

The HouseModelFactory class is one example of the singleton design pattern. Factories are often singletons as there should only ever be one instance.

Check style

During the Java development process the google coding standards/check style was used to keep the formatting of code consistent and readable and also to make it easier to follow proper java coding standards.

Test Strategies and TDD

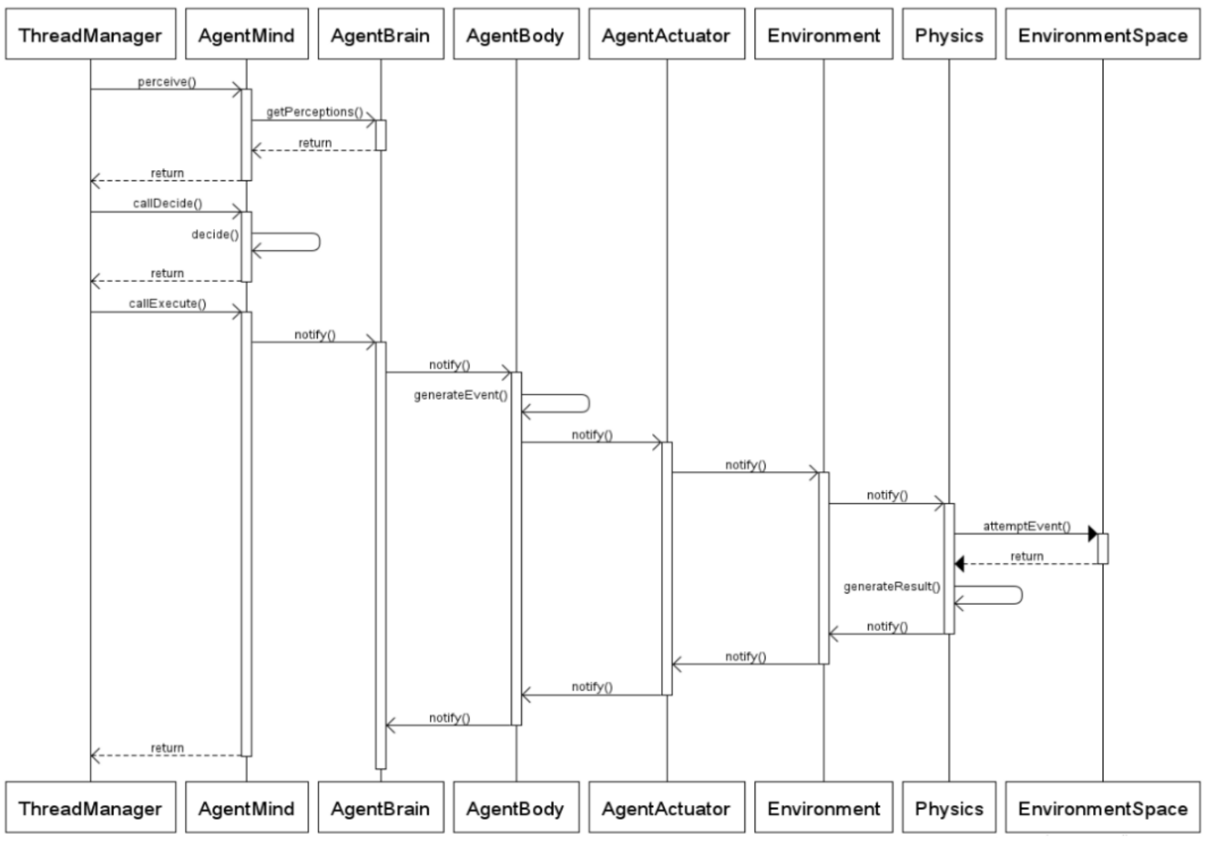
TDD is a software development strategy in which the requirements of the system are broken up into essentially atomic tests. The tests are written and then code is written so that tests succeed. All tests are re-run each time the system is tested. The system is built up in this way so that (hopefully) by the end everything will work. In java JUnit is used for TDD, JUnit and a TDD strategy was used when testing some important utility classes such as *TestArgumentUltilities*, *TestTimeDateTracker*, and *TestMathUtilities* all of which can be found in the *test* package.

TDD was not a development strategy that would provide any advantage in the rest of the project. It becomes difficult to create unit tests for classes that interact heavily with one another and in a multi-threaded system it was nearly impossible. Instead, I used an integration based test strategy, testing the interacting between Observer/Observable pairs in a chain using simple print statements. For example to test event parsing between all parts of an agent – place a meaningful print statement in each *update* method and evaluate their order/correctness manually.

UML

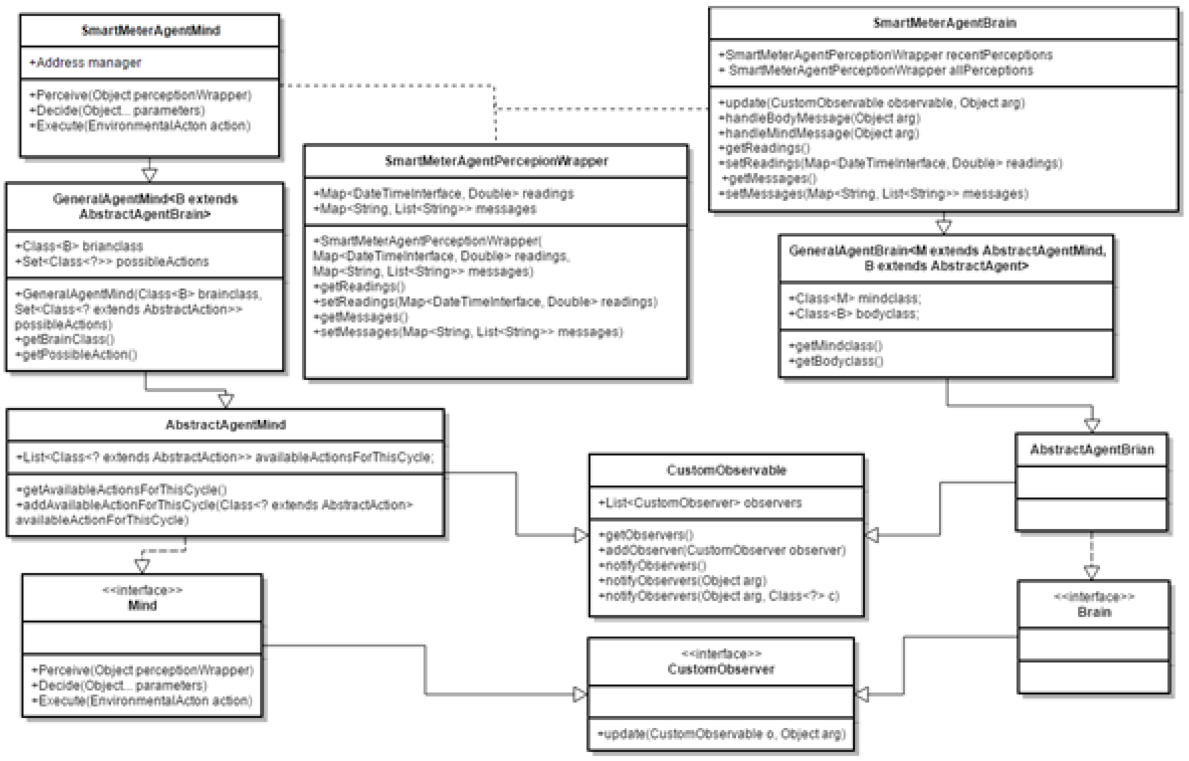
UML diagrams are used to explain and model a program graphically. They illustrate the design and architecture of the program and can also be used to identify architectural strengths/weaknesses. I created various UML diagrams for the project to illustrate the architecture of different areas. As the project is already quite large; I limited and divided them to only show the most useful areas.

UML Sequence diagram for an agent's interaction with its environment



The above shows the interaction via observer/observer able notification of the different parts of an agent with the environment and the environment with its physics. All messages (except the Physics → EnvironmentSpace interaction) are asynchronous. The diagram is for a single agent only. In the real system there may be multiple agents interacting with a physics and environment (for example in the NationalGridUniverse).

UML class diagram depicting the SmartMeterAgentMind and SmartMeterAgentBrain interaction/class structure



SmartMeterAgentMind
+Address manager
+Perceive(Object perceptionWrapper)
+Decide(Object... parameters)
+Execute(EnvironmentalActon action)
SmartMeterAgentBrain
+SmartMeterAgentPerceptionWrapper recentPerceptions
+ SmartMeterAgentPerceptionWrapper allPerceptions
+update(CustomObservable observable, Object arg)
+handleBodyMessage(Object arg)
+handleMindMessage(Object arg)
+getReadings()
+setReadings(Map<DateTimeInterface, Double> readings)
+getMessages()
+setMessages(Map<String, List<String>> messages)
SmartMeterAgentPercepionWrapper
+Map<DateTimeInterface, Double> readings
+Map<String, List<String>> messages
+SmartMeterAgentPerceptionWrapper(
Map<DateTimeInterface, Double> readings,
Map<String, List<String>> messages)
+getReadings()
+setReadings(Map<DateTimeInterface, Double> readings)
+getMessages()
+setMessages(Map<String, List<String>> messages)
GeneralAgentMind<B extends AbstractAgentBrain>
+Class<B> brianclass
+Set<Class<?>> possibleActions
+GeneralAgentMind(Class<B> brainclass, Set<Class<? extends AbstractAction>> possibleActions)
+getBrainClass()
+getPossibleAction()
GeneralAgentBrain<M extends AbstractAgentMind, B extends AbstractAgent>
+Class<M> mindclass;
+Class<B> bodyclass;
+getMindclass()
+getBodyclass()
AbstractAgentMind
+List<Class<? extends AbstractAction>> availableActionsForThisCycle;
+getAvailableActionsForThisCycle()
+addAvailableActionForThisCycle(Class<? extends AbstractAction> availableActionForThisCycle)
AbstractAgentBrian
CustomObservable
+List<CustomObserver> observers
+getObservers()
+addObserver(CustomObserver observer)
+notifyObservers()
+notifyObservers(Object arg)
+notifyObservers(Object arg, Class<?> c)
<<interface>>
Mind
+Perceive(Object perceptionWrapper)
+Decide(Object... parameters)
+Execute(EnvironmentalActon action)
<<interface>>
Brain
<<interface>>
CustomObserver
+update(CustomObservable o, Object arg)

Revision Control

Revision control systems such as Subversion and Git are used to store, manage, share code between developers working on a project. It will keep the entire history of the project (the versions) allowing more efficient code management and updating.

I have used Git as my revision control system (GitHub) out of personal preference. I have tried to regularly commit to this repository but my work on the project was quite modular so they tended to be fortnightly (see Planning below) and quite large. I have two branches: agentdemo and master. Agentdemo contains all the multi-agent system related code as well as the data generation code and testing. The master branch contains all reports as well as some R scripts for processing the London Low Carbon Dataset (see below) and for data analysis. The repository can be found at this address: https://github.com/BenWilkins20/3rd-year-project.git. Email – ZAVC926@live.rhul.ac.uk to receive viewing permissions.

Data Generation

Introduction

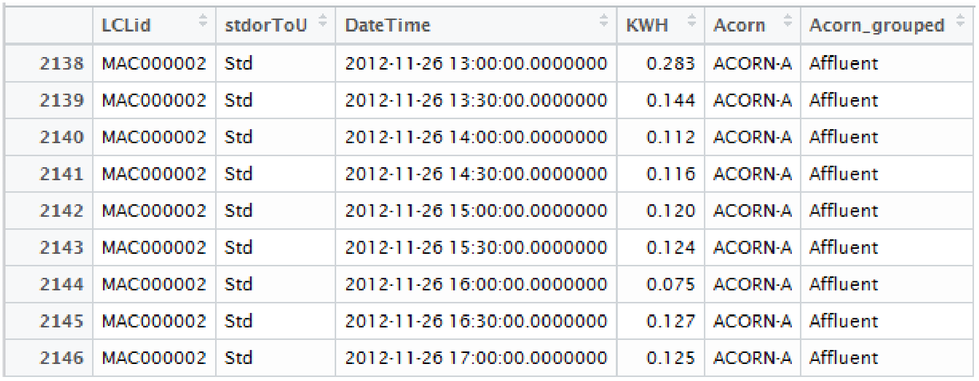
Data will be generated by a Data Generation module which will be global – it will be accessible to all House Environments. When generating data we want it to represent a real scenario as much as possible. The best way to do this is to acquire some real data; the energy usage of different households, and fit a representative model to it. We can then sample from the fitted model to generate realistic data. It is necessary to perform some analysis of the data (see below section Data Analysis) to find a good representative model. The analysis of the data gives some insight into desired properties of a prediction model e.g. linearity.

The below section contains analysis of the Low-Carbon-London dataset. This data set almost exactly the kind of data that the simulation will be using – half hourly data in different households for different financial situations.

Data analysis

Low-Carbon-London-Dataset

The data set includes KWH per half hour readings for a number of households. The data set is grouped into Affluent, Comfortable, Adversity and ACORN-U depending on the customer status.
The first section from this containing 1 million entries was used in the analysis. The complete dataset contains 167 million entries. The sample below shows 8 rows of the data set with the column names.



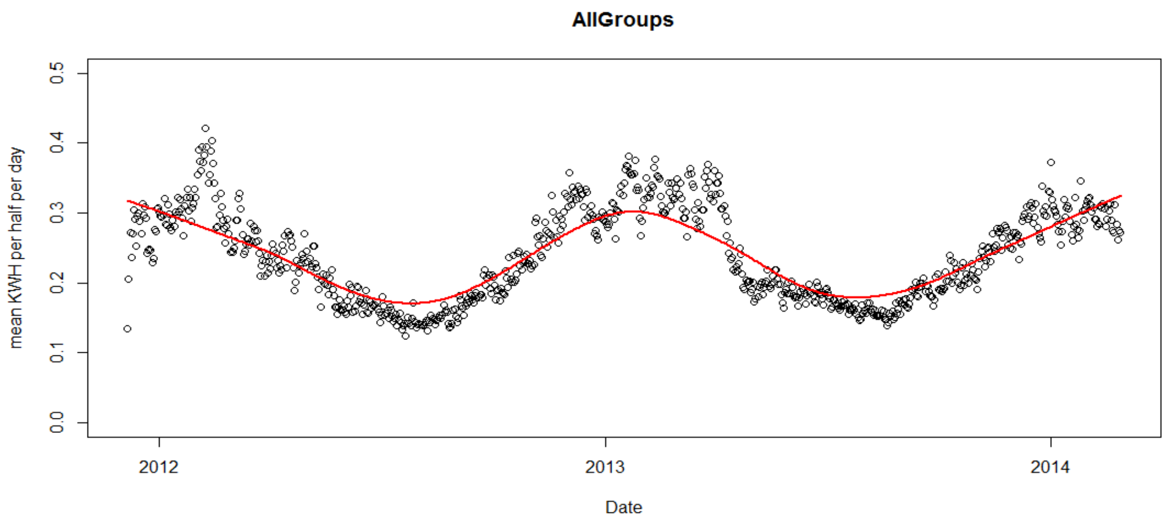
	LCLid	stdorToU	DateTime	KWH	Acorn	Acorn_grouped
2138	MAC000002	Std	2012-11-26 13:00:00.0000000	0.283	ACORN-A	Affluent
2139	MAC000002	Std	2012-11-26 13:30:00.0000000	0.144	ACORN-A	Affluent
2140	MAC000002	Std	2012-11-26 14:00:00.0000000	0.112	ACORN-A	Affluent
2141	MAC000002	Std	2012-11-26 14:30:00.0000000	0.116	ACORN-A	Affluent
2142	MAC000002	Std	2012-11-26 15:00:00.0000000	0.120	ACORN-A	Affluent
2143	MAC000002	Std	2012-11-26 15:30:00.0000000	0.124	ACORN-A	Affluent
2144	MAC000002	Std	2012-11-26 16:00:00.0000000	0.075	ACORN-A	Affluent
2145	MAC000002	Std	2012-11-26 16:30:00.0000000	0.127	ACORN-A	Affluent
2146	MAC000002	Std	2012-11-26 17:00:00.0000000	0.125	ACORN-A	Affluent

LClid: the unique house identifier.
stdorToU: Tariff.
KWH: energy used KWH per half hour.
Acorn/Acorn grouped: the grouping of the customer.

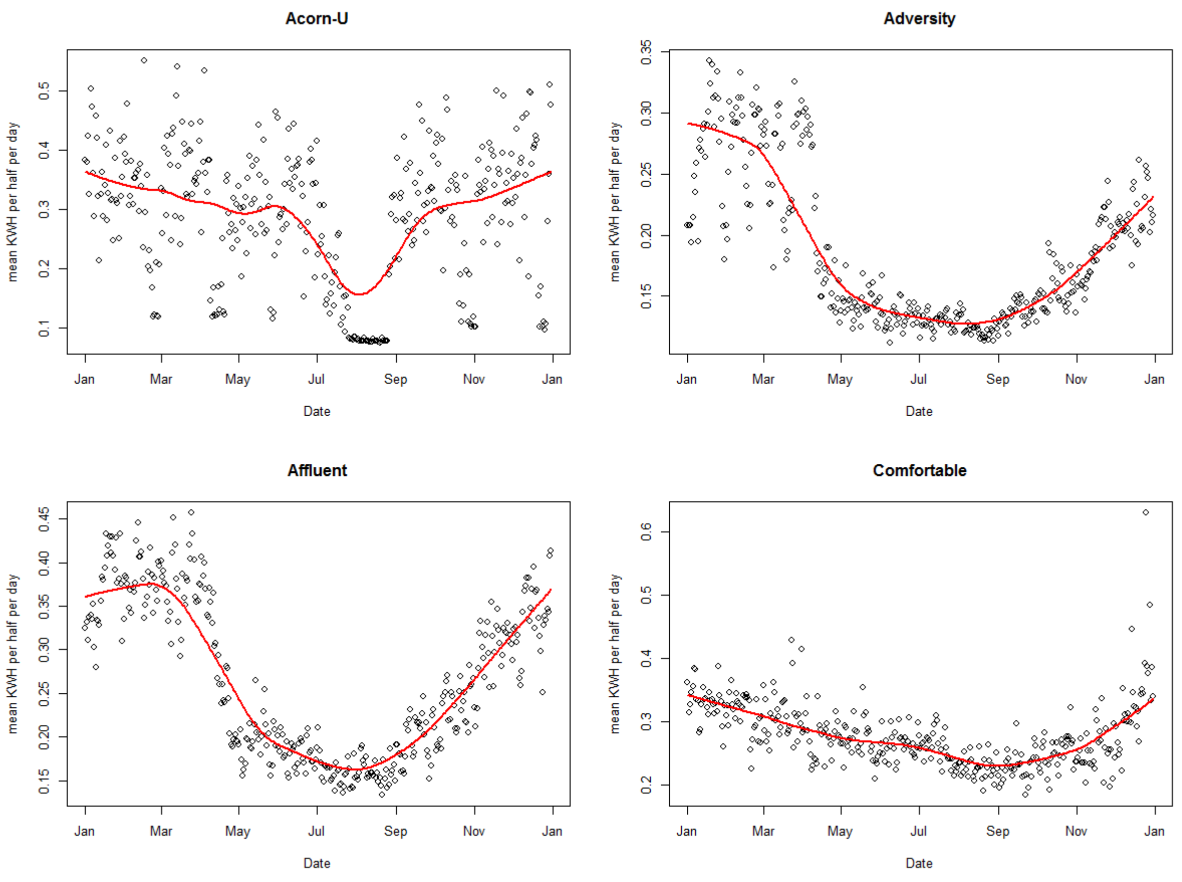
The data set can be found here https://data.london.gov.uk/dataset/smartmeter-energy-use-data-in-london-households. Details on the ACORN groups can be found here http://acorn.caci.co.uk/downloads/Acorn-User-guide.pdf.

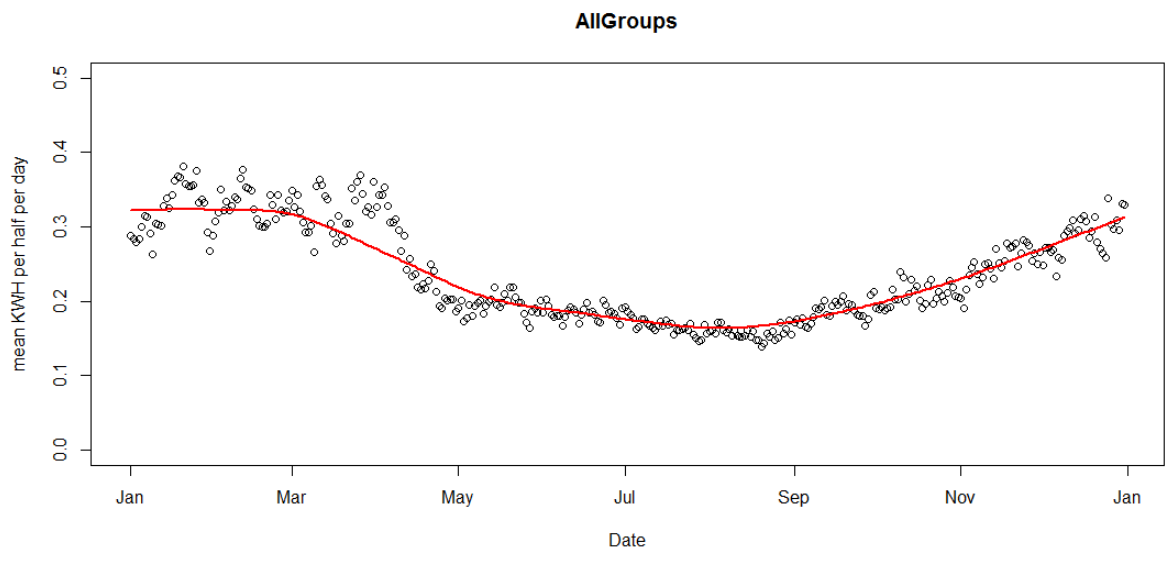
Seasonality

The plot below illustrates seasonality over the two years that data was collected. We can see that the average KWH usage is higher in the winter than in the summer. On the y axis: the mean KWH per half hour usage per day, on the x axis: the date (in days).



The seasonality of each group varies, however for the sampling model one seasonality function will be developed that represents all groups. This will be similar to the above graph (the average of all groups). The graphs below show the seasonality of each group in the 2013 year.



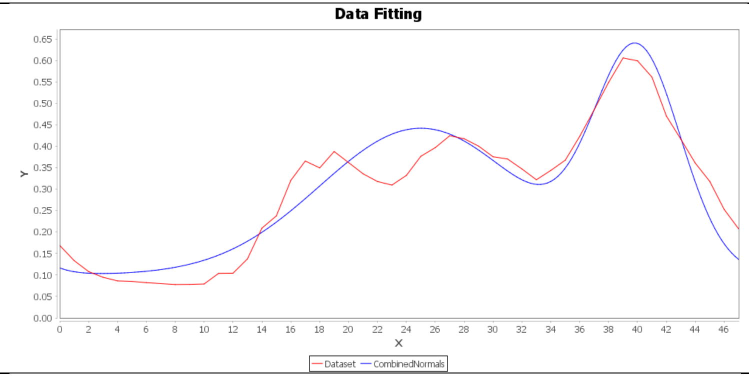
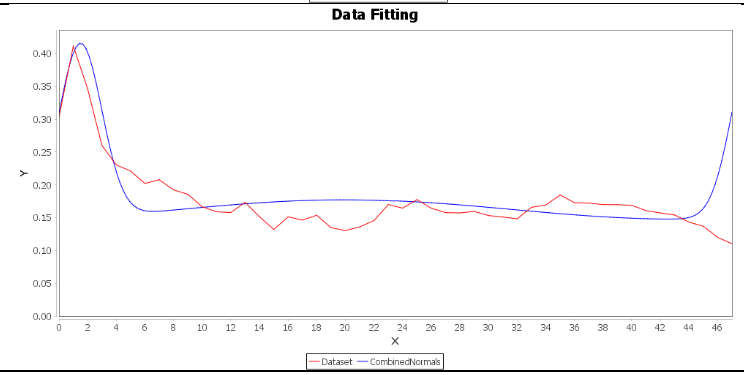


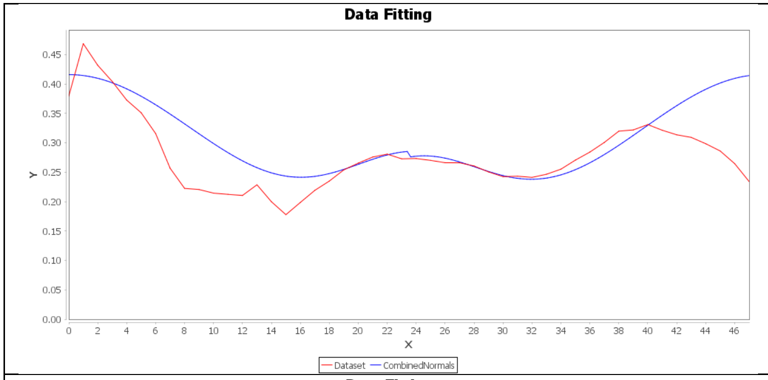
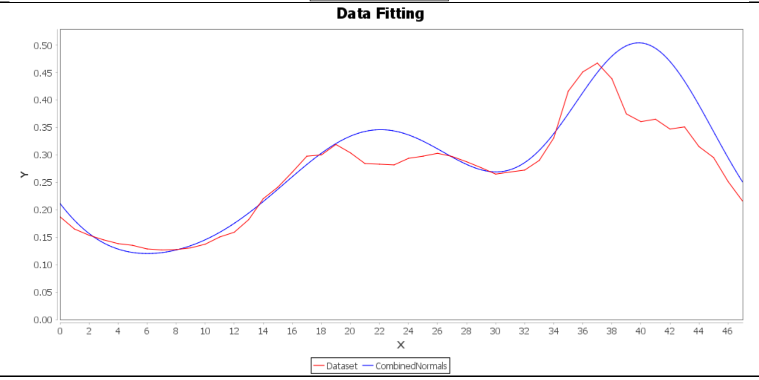
Building a sampling model

When building the sampling model the equations for each group will take the form:

$$b_1*\text{normal}(\mu_1, \sigma_1) + b_2*\text{normal}(\mu_2, \sigma_2) + c$$

Manually fitting two normal curves to the mean KWH per half hour of the ACORN-U group. The blue line shows the fitted combined normal curve, the curve loops around the boundary, this is because hours are continuous – it is time series data. The red line shows a calculated mean values of the data group per half hour. The Y axis shows KWH per half hour, the X axis shows 48 half hourly time intervals which covers one day (24 hours) starting at 00:00:00 and ending at 11:30:00. (The graphs below were generated in the java class DataFitter using the JFreeChart graphing library).

	**Group: Acorn-U** $b_1 = 3.8$ $\mu_1 = 40.0$ $\sigma_1 = 3.0$ $b_2 = 6.0$ $\mu_2 = 25.0$ $\sigma_2 = 7.0$ $c \;\; = 0.1$
	**Group: Adversity** $b_1 = 1.0$ $\mu_1 = 1.5$ $\sigma_1 = 1.5$ $b_2 = 2.0$ $\mu_2 = 20.0$ $\sigma_2 = 14.0$ $c \;\; = 0.12$

	**Group: Affluent** $b_1 = 6.0$ $\mu_1 = 48.0$ $\sigma_1 = 9.0$ $b_2 = 1.5$ $\mu_2 = 24.0$ $\sigma_2 = 5.0$ $c \ \ = 0.15$ discrepancy in normal curve *
	**Group: Comfortable** $b_1 = 5.0$ $\mu_1 = 40.0$ $\sigma_1 = 5.0$ $b_2 = 4.0$ $\mu_2 = 22.0$ $\sigma_2 = 6.5$ $c \ \ = 0.1$

* In the Affluent graph there is a small discrepancy in the normal line. This is because the equation wraps around the boundaries up to half of the range in each direction. It occurs when there is a significant different in the middle values (for each wrapped half range), in most cases the difference is negligible because the normal curve tends to 0, it has negligible effect when added. In this case however the stand deviation of the curves is sufficient enough so that the line is not close to 0 and so the addition is noticeable. It will have a very small effect on the data sampling at the centre point but as the curve is an approximation and some error term will be added anyway it is not something to be too concerned about.

Proceedings

After looking at a good representative example of the data that the system will be dealing with (above). It has come to attention that an ANN prediction model may not be the most suitable. Changing it in favour of a simpler and more intuitive model will benefit the final result of the project. Having a simulation of the system including the representative model formed above will allow testing of different prediction models. The selection of models to test does not necessarily exclude an ANN model.

Testing Prediction Models

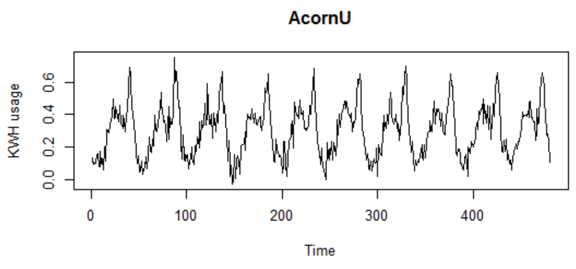
Introduction

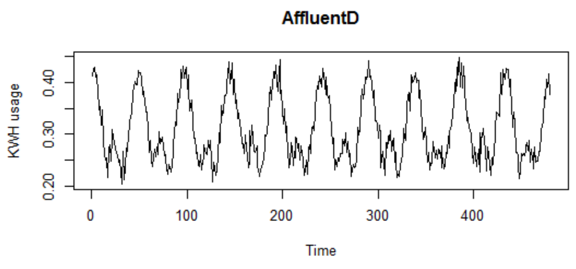
ANNs have their advantages, but they also have their disadvantages. They are essentially a black box; the interpretability of their results is very bad. It will be advantageous in a business oriented environment to have at least some interpretability. There is an opportunity to test the use different machine learning techniques with the success of the data generation module. If a prediction model can be found that works as well or better than an ANN model then it should be seriously considered. Taking this opportunity, the next section will begin to test some alternative machine learning models.

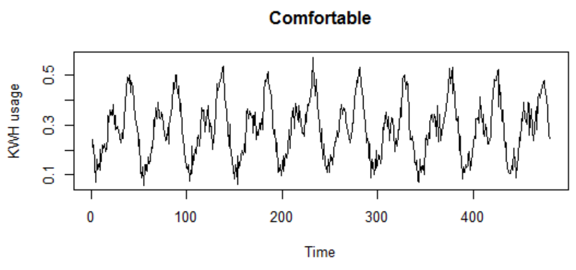
It should be noted that this testing is completely separate from the multi-agent system as generating data through a simulation is unnecessary. Data will be generated similarly to the *TestCombinedNormalHouseModel* class in the *test* package – it will be generated on mass. A dedicated data generator will provide enough data to perform the tests (See *ExperimentDataGenerator* in the *demo* package). Different combinations of house models will be used to see if they will affect the prediction performance of the alternative methods. The testing will be done using R and using time series packages (R stats library).*

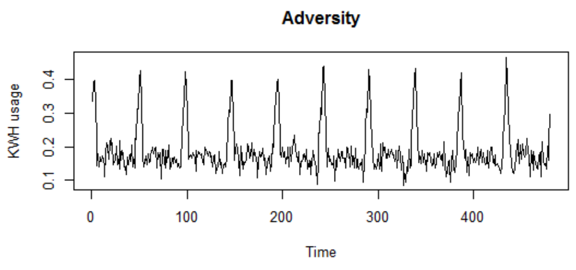
Generated Data Analysis

Below illustrates generate data for each group over a 10 day period.

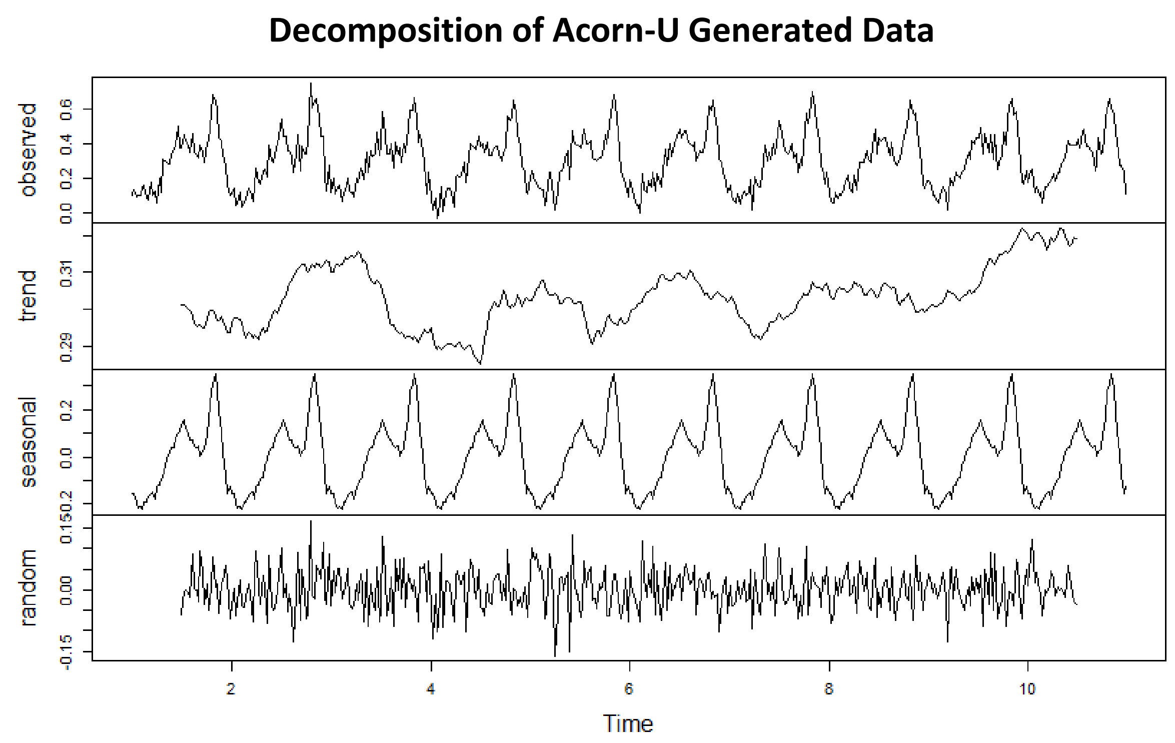








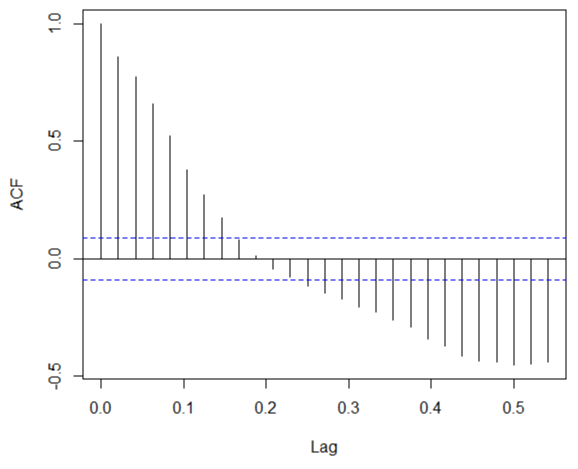
* It should be noted here that the details of the forecasting approach will not be discussed here due to time constraints. Reports about the different models/methods will be provided for next term. This section is just to get an idea of the different models that may or may be suitable.

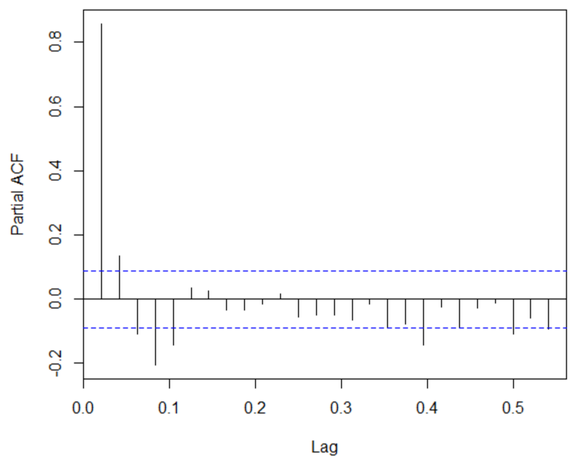


The above is a sample of generated data over 10 days. It is nice to see that the seasonal trend (daily) looks somewhat similar to the Figure 1 (first section). The next step it to forecast for future data. Note that the data can be described by an additive time series model (which is required for the R function that did the decomposition (uses the 'TTR' R library)). At this moment the generated data does not include any yearly seasonality. Because the series is stationary – the mean, variance and covariance (of the ith and ith + n) term do not change with time (this may change when yearly seasonality is added). We can apply a basic ARIMA (or AR/MA) model for forecasting without having to employ any methods to make the series stationary (differencing, de-trending etc). The study below will use the Acorn-U generated series as its example.

Find a time series model

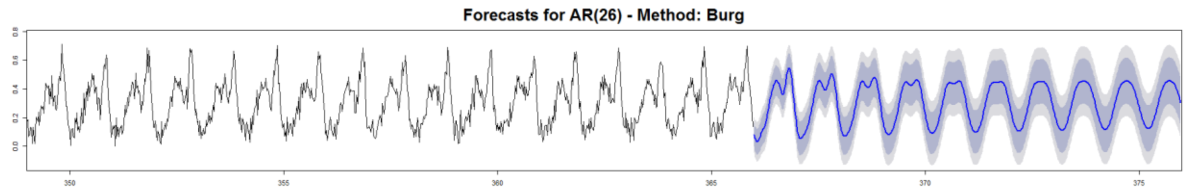
Checking the auto-correlation function (AFC) and partial auto-correlation function (PAFC)

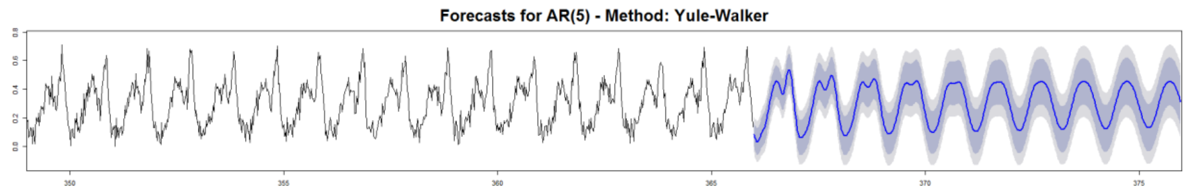


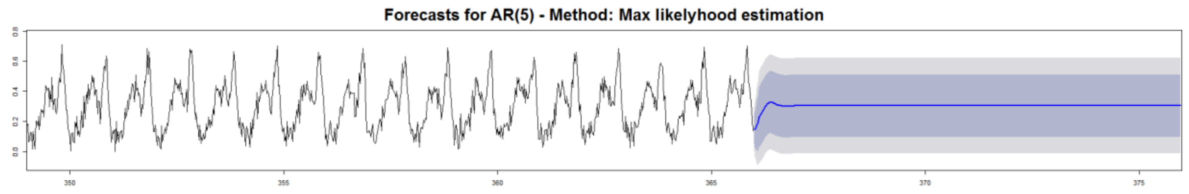


The PACF drops significantly after the 1st lag this implies that the series is an Auto Regressive (AR). Although the drop is significant, it does not level out until roughly lag 5. So the order can be between 1 and 5.

The AR model was tested with 3 different methods (R built in) – Burg, Yule-Walker (YW) and Maximum likelihood estimator (MLE). The models use 1 year of generated data (17520 half hour intervals).







The Burg and YW methods perform quite well, they smooth out over time but the forecast is fairly accurate for the first few days. (The R code can be found in the *Analysis.R* file)

Proceedings; Over the Christmas holidays the aim will be to test some more models, compare them with each other – this will include the first ANN test. The data generator will be updated to include the yearly seasonality mentioned earlier in the Data Generation section. This may affect the result of the above AR experiment and so it will be redone when this functionality is added.

Planning, Time Scale and Project Diary

Planning

I have not followed the original project plan, especially the sections related to ANNs. I was too ambitious with the time frames that I designated for each report/program, it would have been wiser to move the machine learning related tasks to the second term. Now knowing that an ANN based model may not be the way to go, it is fortunate that I had not already implemented the relevant proof of concept programs/written relevant reports. Most of the machine learning related work will be done next term. I have however had the opportunity to do some fairly extensive background reading on ANNs for time series prediction and machine learning in general [18] [19] [20] [21] [22] [23] which will give me a very good base for next term. I have progressed quite far with the simulation system. Data generation is complete, the multi-agent system is almost complete all that needs to be done is to create some integrated machine learning capability in an agent mind.

Diary

I had meetings with my supervisor – Dr Zhiyuan Lou and occasionally Prof Kostas Stathis every fortnight on Wednesday. The meetings were very useful in evaluating the progress of the project. Because of these fortnightly meetings I tended to work in chunks every two weeks – which is why my revision control (git repository) sees fortnightly commits. The meetings are as noted below:

12th October: General discussion about the project and how to proceed. Discussed the multi-agent architecture to be used and what reports I will be writing this term.

26th October: Kostas was involved with discussions on the project. Talking about the different layers of the architecture – including where in the agent system the predictors should reside. Talked about having area/neighbourhood agents having prediction capabilities for the houses they are responsible for. Discussed moving away from the ANN model in favour of a simpler model. This may be the way to go as ANN can be difficult to use/explain. I have put the multi-agent report on hold to work on the data generation section of the project. We agreed that this section was more appropriate as a means to continue the project effectively as the data will be relied upon when starting the prediction section. A good portion of the Multi-agent system has been completed by this point, agent communication via sockets, house environments with (after data generation section has been completed – will be implemented) the capability of holding and retrieving data from a generic data generator. The presentation at the end of term should have a demonstration graphic of the prediction working – e.g. A graph with 2 lines, one for real data and one for the prediction. The lines will extend across the graph with time.

9th November: Kostas was involved in the meeting again. I briefly demonstrated the multi-agent system to Zhiyuan and Kostas as well as the data generation program and report. We discussed the architecture of the House Environment, Kostas suggested that a data reading agent should be used to get readings from the generator and forward them to the Smart

Meter Agent. This method seems better than the current one – where the smart meter agent reads at a clock tick on a global timer. We again spoke about the ANN implementation and after looking at the data analysis section of the Data Generation report decided that testing different prediction models would be a good addition to the project. Having this meeting allowed me to finish the Introduction to Multi-Agent Systems report, specifically the sections about the architecture used in the project.

23rd November: Final meeting this term. With Zhiyuan only, we spoke about the presentation, interim viva and how to proceed with the project. He advocated that I had at least some machine learning work to present for the interim report. I will now work on some basic forecasting on generated data. The forecasting will be done using prebuilt time series machine learning packages (probably in R). At least some experimentation will be complete by the time the report is due, the rest will be done over the Christmas break. I will be implementing the most successful machine learning methods next term.

Bibliography

[1] C. Gaven, “Seasonal variations in electricity demand,” Department of Energy & Climate Change, 2014.

[2] European Nuclear Society, “Base load power plants,” [Online]. Available: https://www.euronuclear.org/info/encyclopedia/baseloadpowerplants.htm.

[3] S. R. E. M. F. F. S. J. A. Emma L. Giles, “The Effectiveness of Financial Incentives for Health Behaviour Change: Systemtic Review and Meta-Analysis,” 2014.

[4] C. D. A. L. S. P. A. M. Antonios Chrysopoulos, “Agent-based small-scale energy consumer models for energy portfolio management,” *2013 IEEE/WIC/ACM International Joint Conferences on (Vol. 2, pp. 94-101). IEEE.,* 2013.

[5] Department of Energy & Climate Change, “Smart Metering Implementation Programme, Smart Metering Equipment Technical Specifications,” Department of Energy and Climate Change, 2014.

[6] M. Wooldridge, An introduction to multiagent systems, John Wiley & Sons, 2009.

[7] S. Russell and P. Norvig, Artifical Intelligence. A modern apprach, Prentice Hall, 1995.

[8] P. J. '. Hoen, K. Tuyls, L. Panait, S. Luke and J. A. La Poutré, “An Overview of Cooperative and Competitive Multiagent Learning,” *Lecture Notes Computer Science,* vol. 3898, pp. 1-46, 2006.

[9] I. Rahwan, R. Kowalczyk and H. H. Pham, “Intelligent Agents for Automated One-to-Many e-Commerce Negotiation,” *Autralian Computer Science Communications,* vol. 24, no. 1, pp. 197-204, 2002.

[10] C. Yilmaz, S. Albayrak and M. Lutzenberger, “Smart Grid Architectures and the Multi-Agent System Paradigm,” in *Proceedings of the 4th International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies.*, Chamonix, 2014.

[11] A. Rogers, S. D. Ramchurn and N. R. Jennings, “Delivering the Smart Grid: Challenges for Autonomous Agents and Multi-Agent Systems Research,” in *Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI-12)*, Toronto, 2012.

[12] M. S. Narkhede, S. Chatterji and S. Ghosh, “Multi-Agent Systems (MAS) controlled Smart Grid - A Review,” in *Proceedings on International Conference on Recent Trends in Engineering and Technology*, 2013.

[13] B. Alrayes, O. Kafali and K. Stathis, “RECON: A Robust Multi-agent Environment for Simulating COncurrent Negotiations,” *Advances in Agent-base Complex Automated Negotiation,* pp. 157-174, 2016.

[14] S. Bromuri and K. Stathis, “Situating Cognitive Agents in GOLEM,” *Lecture Notes in Computer Science,* vol. 5049, pp. 115-134.

[15] A. Kakas, P. Mancarella, F. Sadri, K. Stathis and F. Toni, “Computational Logic Foundations of KGP Agents,” *Journal of Artificial Intelligence Research 33 (2008) 285-348,* 2008.

[16] J. Hopkins, O. Kafali and K. Stathis, “Open Game Tournaments in STARLITE.,” in *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[17] K. Stathis and F. Toni, “The KGP model of agency for decision making in e-negotiation,” 2005.

[18] H. Allende, C. Moraga and R. Salas, “Artifical neural networks in time series forcasting: A comparative analysis,” *Kybernetika,* vol. 38, pp. 685 - 707, 2002.

[19] MachineLearner, “CS231n Lecture 10 - Recurrent Neural Networks, Image Captioning, LSTM,” 14 June 2016. [Online]. Available: https://www.youtube.com/watch?v=iX5V1WpxxkY.

[20] B. Oancea and S. C. Ciucu, “Time series forecasting using neural networks,” *arXiv,* p. 1, 2014.

[21] S. Miller, “Mind: How to Build a Neural Network (Part One),” 10 August 2015. [Online]. Available: http://stevenmiller888.github.io/mind-how-to-build-a-neural-network/.

[22] T. M. Mitchell, Machine Learning, MIT Press, The McGraw-Hill Companies Inc, 1997.

[23] PyData, “Alex Rubinsteyn: Python Libraries for Deep Learning with Sequences,” 4 December 2015. [Online]. Available: https://www.youtube.com/watch?v=E92jDCmJNek.

[24] B. Davito, T. Humayun and U. Robert, “The smart grid and the promise of demand-side management,” McKinsey and Company, 2009.

[25] British Gas, “Smart Meters,” 2016. [Online]. Available: https://www.britishgas.co.uk/smarter-living/control-energy/smart-meters.html.

[26] Smart Energy GB, “About the rollout,” [Online]. Available: https://www.smartenergygb.org/en/the-bigger-picture/about-the-rollout.

[27] nelsonminar, “Energy usage notes,” 31 Janurary 2012. [Online]. Available: https://nelsonslog.wordpress.com/2012/01/31/energy-usage-notes/.

[28] “BBC News,” [Online]. Available: http://news.bbc.co.uk/1/shared/spl/hi/guides/456900/456991/html/.

Appendices

Appendix 1. Project Plan Abstract

Energy demand management (EDM) has become an increasingly pressing issue in our technological age. EDM, specifically demand-side management (DSM) is concerned with planning and implementing models that allow consumers to alter their habits. This leads to reduction in energy demand peaks and overall energy consumption [24]. Energy providers have a number of ways of managing fluctuations in energy consumption that include buying energy, using peaking power plants and energy storage techniques.

This project aims to provide an alternative solution by integrating two related sub-fields in computer science; machine learning and multi-agent systems. This has become possible with the advent of Smart Meters that are now being offered by providers such as British Gas. [25] Modelling these Smart Meters as agents and placing them in a distributed multi-layered environment allows automated data collection and pre-processing. The processed data can then be given to a top level agent with specialised prediction capabilities.

Artificial neural networks (ANNs) will provide these prediction capabilities. ANNs have proven to be an effective method of prediction and classification. They have been used in time series forecasting to model complex systems such as financial markets and foreign exchange markets and so will be suited to this problem [20]. ANNs have the ability to approximate highly non-linear functions and so have been used instead of more traditional statistical models. Using ANNs in this case will further explore their forecasting capabilities.

Appendix 2. Project Plan MAS Structure

The agents will be placed in a multi-layered environment, each layer will contain one or more agents and each agent in a layer may be a manager of some agents in the layer below illustrated in figure 1. The system will be developed to have arbitrary layers in this fashion with the top layer agent(s) as prediction agents, the bottom layer as data collection agents and intermediate data pre-processing agents. Depending on the amount of data pre-processing to be done it may be preferable to distribute the system in terms of layers or agents. The agents will have the capability to communicate via an internet protocol of choice. This can be done using the Starlite framework as it allows sensor/actuator modules to be attached to the agents, these modules can be set up to accommodate any protocol. The full details of the Starlite framework will be discussed in the Introduction to Multi-agent Systems report mentioned below.

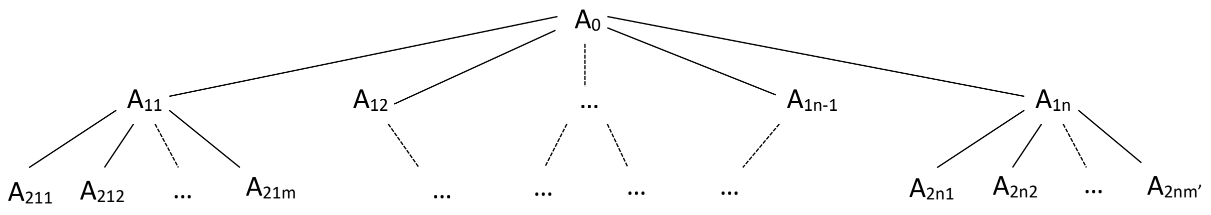


Figure 1. A_L is an agent at layer L. A_0 will typically be the prediction agent – in this case the one that uses the ANN. Agents directly below A_0 will be supplying it with clean formatted data. The bottom layer of the hierarchy will be reserved for data collection – in this case the smart meter agents. The lines represent communication channels between the agents. These channels will be used to send the collected data up the hierarchy. The channels will be bi-directional as an agent may want to send control data to an agent it is managing. See figure 2 for an illustration of communication channels.

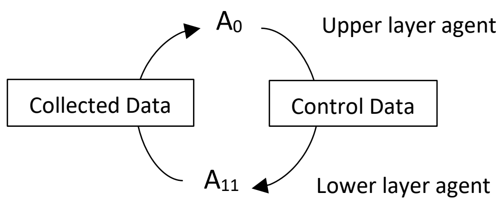


Figure 2. Control data is sent down the hierarchy and the collected data – in this case the smart meter data is sent up the hierarchy. Control data may be any data and will depend on the systems use. In this system it may be instructions for data pre-processing.