Imperial College London

Department of Electrical and Electronic Engineering

Final Year Project Report 2015

Project Title:	**Simulator for Decentralised Energy Systems**
Student:	**Yuchen Wang**
CID:	**00683700**
Course:	**EE4 (EM)**
Project Supervisor:	**Dr. Jeremy V. Pitt**
Second Marker:	**Dr. Javier A. Barria**

Acknowledgements

I would like to thank my supervisor Dr Jeremy Pitt for taking the time to impart advice, guidance and supervision over the past academic year on this project.
I would like to thank e.quinox, all members past, present for giving me the opportunity of a lifetime and giving me the information to make this project relevant.
I would also like to thank my friends, housemates and fellow colleagues who have made my last 4 years the best experience it could possibly be.
Finally, I would like to thank my family, in particular my mother for their support and love.

Yuchen Wang
June 2015

Abstract

In many developing countries the electricity grid network is underdeveloped, causing low levels of rural electrification. However, electricity access in these rural communities are not non-existent. There exists a number isolated and independent electricity generators owned by individuals, NGOs and local government institutions. This project aims to explore the feasibility of a Decentralised Community Energy System which utilises these isolated sources of electricity to better serve communities with a micro grid.

Using a holonic system, collectives of energy providers can be modelled using a Multi-Agent System to pool electricity as a Common Pool Resource. Similarly, electricity allocations can be distributed fairly using holonic institutions and Rescher's Canons of Distributive Justice to guarantee all consumers a continuous access to electricity.

This project applies these principles to create a simulator for Decentralised Community Energy Systems, and demonstrates the feasibility of a community micro grid.

Contents

List of Figures

List of Tables

List of Acronyms

Introduction 1

Electricity is in wide-spread use in many developed countries and are delivered by a vast network of cables, overhead lines and other assets maintained by Distribution and Transmission Network Operators. Electricity access has 100% penetration in the UK,[2] and residents also benefits from a Transmission Network which operates at above 99% reliability.[3] In many areas of rural developing countries, there are often no wide-spread access to a continuous and reliable electricity supply which is provided by the country's electricity grid.[4] However there exists many isolated sources of electricity generation such as solar panels and standalone diesel generators utilised by relatively wealthy households, shops and buildings belonging to large organisations.

The aim of this project is to explore the idea of using a holonic institution to model a Decentralised Community Energy System which bring together the existing decentralised generation infrastructure to provide a reliable electricity for users in rural communities. The model was constructed as a Multi-Agent System (MAS) and simulated using Presage 2 with individual consumers such as households and providers such as generators modelled agents. The agents can be grouped together to form communities such as villages, which in turn can be grouped together to form a larger entity such as a district or a province.

With the vast majority of people living in communities such as villages, towns and cities, the structure of the decentralised energy system model will need to be designed in a fashion that is scalable and allows grouping of people to form communities. In a traditional electricity system operated by most developed countries is composed of centralised generation, transported around the country by transmission systems and distributed to consumers by the distribution network. Traditional models and structures do not apply to this project as they have been designed for mostly uni-directional power flow: from large generators to consumers. With many of the users of the Decentralised Community Energy System capable of both consuming and generating, a scalable micro-grid for this model needs to be designed with that in mind.

Designing the structure of the simulation on the design of holonic systems simplifies the scalability aspect of having multiple communities in a simulation. A holonic system is one which is formed of many smaller systems, which are in turn formed of many smaller systems and so on, until reaching the most "elementary" of systems. In the case of this project, the simulation will be designed to be able to distribute fairly electrical power between interrelated agents which are in turn composed of interrelated subagents recursively, until reaching lowest level of subagents (households and businesses).

A Multi-Agent System was selected to simulate as Agents are required to be able to act independently, to be unable to directly manipulate the environment and unable to control the actions of other Agents. In the case of this simulator, consumers are independent entities who can't change the environment conditions (e.g. have access to Grid Electricity) and unable to directly control the consumption and

provision of other consumers or providers. Presage 2 was selected as the Multi-Agent Simulation platform as it was possible to seek support from PhD students within the EEE department who currently use the system.

With Presage 2, there are also some issues and limitations. The simulation results can sometimes be inconsistent due to out of order parallel execution by the simulation platform, and the simulation can only execute in discrete time steps. Suitable design steps have been taken to mitigate these problems and limitations which have been detailed in the Design section of this report.

To allow the model to be realistic, the model has been based on real rural communities in Rwanda, with realistic demand and generation profiles. Due to the lack of available data in Rwanda, the demand and generation profile has been approximated using data from rural UK load centres.

This report outlines the design, implementation and testing of a holonic multiagent simulation of a decentralised energy system.

Electricity as a Common Pool Resource

A Common Pool Resource is a depletable resource which can be utilised by a group of people, characterised by a reduction in the availability of this resource as individuals withdraw or utilise this resource.[5] Electricity can be a Common Pool Resource if there exists a finite amount of electricity generation capacity. As users connect demand appliances to the generators, the availability of electricity supply for additional demand diminishes.

In developing communities with significant generation from renewable sources such as wind and solar, the availability of power is subject to variation between periods in time. This inherent volatility in the amount of available resource could increase the likelihood of selfish actions of by individuals in the community.

Ostrom showed that Common Property Regimes can be formed to maintain the Common Pool Resources by controlling the access to the resource.[5]

Decentralised Community Energy Systems as a Holonic System

A holonic system (or holarchy) is a system which is composed of interrelated subsystems or institution, each of which are in turn composed of sub-subsystems or institution and so on, recursively until reaching a lowest level of "elementary" subsystems. Each system, sub-system or institution has a well-defined set of goals or objectives which is achieved through enforcing a set of rules on its members (subsystems, sub-institutions and elementary entities).[6] It is this type of Common Property Regime that will be explored in this project to maintain the Common Pool Resource that is electricity.

In the context of a rural Decentralised Community Energy system, a network between households, communities and even villages to pool and share electricity as a common pool resource can be modelled as a holonic system. The holonic system in this case would be composed of communities such as Districts, Provinces, Sectors which are composed of sub-communities such as Towns and Villages. The sub-communities would be composed of many "elementary" subsystems such as households, businesses and other points of connection for electricity. Each community or institution has the goal of fairly allocating electricity to all members. This goal would be achieved with the assumption that they are provided with the necessary infrastructure and powers for enforcing quotas and contribute to a common pool of electricity.

Ostrom's principles

Distributive Justice and Fair Allocation

Given that the necessary infrastructure and powers exist for enforcing quotas and contribution to the Common Pool, the allocation needs to be fair. Being fair forms two of the necessary Ostrom's Principles for Enduring Institutions in a Common Property Regime.[5]

Rescher's Canon of Distributive Justice

Multi-Agent Simulation

The simulation will be designed as a Multi-Agent System (MAS). MASes are particularly suited for this kind of model as Agents in MASes have three very important characteristics:

- Autonomous: Agents act independently
- Local view: no Agent can see or manipulate the environment it is in
- Decentralised: There is no Agent which controls the action of all Agents

In reality, individual households which are represented by Agents in the simulator all perform actions according to their individual and unique needs, and not controlled by a third party. This makes Autonomy and Decentralisation a requirement for the Agents in the Simulator. Participants or households connected to the network can't directly control how other participants use or generate electricity for the Common Resource Pool, making it an requirement for the Agent to have a Localised view.

About Presage 2

Presage 2 is a simulation platform for multi-nodal or Agent simulation of societies. The platform was built by Sam Macbeth and is currently maintained by PhD students within Imperial. This platform was chosen for the simulator as it enables the investigation of the impact of agent design (such as household behaviour), network properties (constraints on access) and the physical environment on individual agent behaviour and long-term global network performance.[7] In the context of this Project, each Node/Agent can represent individuals, households, businesses or generators.

Presage 2 was chosen for this project as it is a platform which allows the rapid prototyping of complex Agent societies. Presage 2 Agents are only allowed to act during increments of time steps, which makes the simulation a discrete time driven one. Figures 2.1 and 2.2 illustrates general and simulation archtecture.

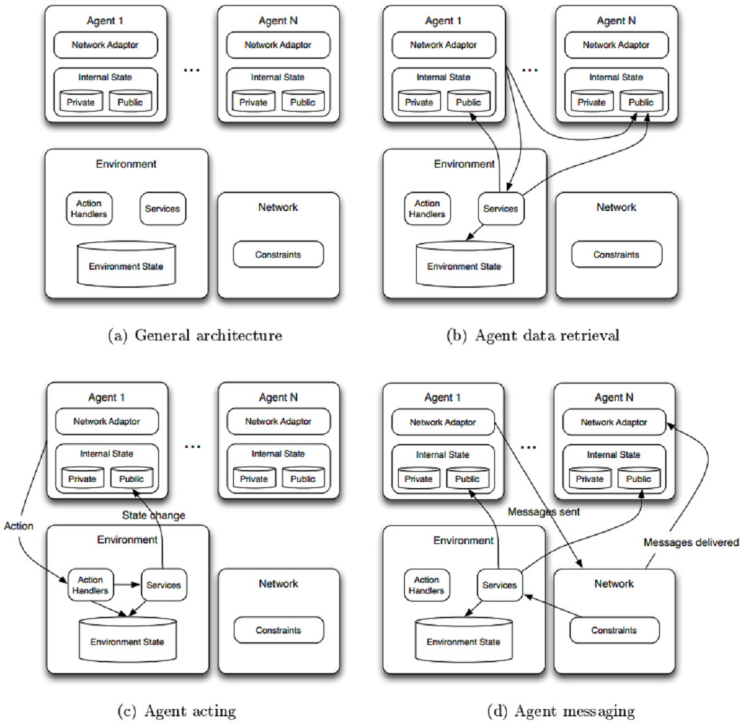

FIGURE 2.1: Presage 2 architecture[1]

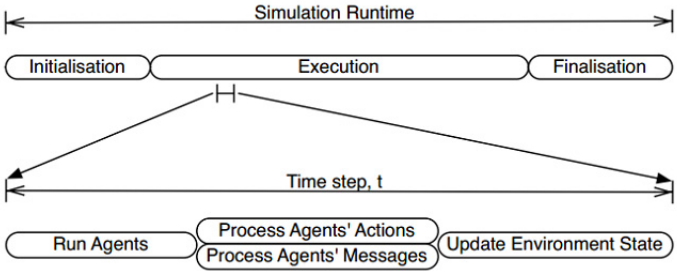

FIGURE 2.2: Presage 2 simulation architecture[1]

Decentralised Generation in Rural Communities of Rwanda

With an estimated 25% rural electrification rate in 2009,[4] rural Africa is one obvious candidate for simulation scenarios. Vast amounts of rural communities remain un-electrified. For a realistic simulation scenario, knowledge of existing infrastructure in place will need to be obtained. With many areas within many countries in Africa, Rwanda in particular has been identified as a good potential simulation scenario to research. Data is difficult to source for rural communities in developing countries, however in the case of Rwanda, some data can be easily from the student society e.quinox. e.quinox is a student-led society which aims to find a scalable solution for rural electrification who mainly operate in Rwanda.[8] The subsections below outline some of the ways remote rural communities are able to access electricity in Rwanda.

Electricity Generation

One of the solutions currently being implemented by e.quinox is the "Energy Kisok" model.[9] The "Energy Kiosk" model features an Energy Kiosk - a building where the generation, storage and distribution of electricity takes place. In e.quinox operated kiosks, electricity is generated from renewable sources. Traditionally, this has been with solar panels. However, hydro-electric generation has been demonstrated to be feasible with the recent construction of a "Hydro Kiosk" at Rugaragara Falls in Southern Rwanda.

Storage and Distribution

Within each kiosk, electricity that is generated is stored in storage batteries placed in the kiosks. The storage batteries regulate power output and allows access to electricity in the kiosk even during periods of no electricity generation.

Battery Box

In the absence of any electricity distribution infrastructure, e.quinox has traditionally provided a number of portable batteries for the purpose of electricity distribution. An example of the portable batteries can be seen in Figure 2.3.

Consumers from local community pay to hire the battery boxes under one of two payment schemes: pay-per-recharge and pay-per-month.[10] The biggest difference between the two schemes are that users can recharge as often as they would like with pay-per-month. Both payment schemes involve the recharge of the boxes at the energy kiosk when they are depleted of energy.

A potential use case of this project can be the simulation of charging of battery boxes at an energy kiosk. This could alleviate congestion and improve asset utilisation of the existing distribution systems by reducing the turn-around time of battery boxes for customers.

FIGURE 2.3: First Generation e.quinox Battery Boxes deployed at Rugaragara Falls Kiosk

Micro Grid

With the recent completion of a hydro-electric kiosk. e.quinox for the first time has a kiosk with access to an always-on generator. With a limited number of battery boxes in circulation and a constant generation available during the off-peak hours, there is excess capacity for electricity generation.

To improve utilisation of the generator in the kiosk, e.quinox has recently started conducting a feasibility study into constructing a transmission line and a distribution network which will serve a village near the kiosk with a view to make the most of the electricity generated.

Preliminary surveys conducted in the nearest village to the kiosk indicates the demand could exceed the amount of excess power generated by the kiosk.

The result of this project can be used in conjunction with e.quinox to conduct the feasibility study of implementing the Micro-Grid. Should a trading platform for energy also be built, the Micro-Grid could serve as a test-bed for this new system.

Standalone Solution

The Standalone Solution is an independent electrification solution which was recently developed by e.quinox for customers who live far from energy kiosks.

The Stand-alone solution consists of a pay-as-you-go solar electricity generation and storage kit, known as the Izuba.Box.[11] With the Izuba.Box, customers no longer have to travel regularly back to the Energy Kiosk for electricity. Solar panels are installed on the customerâĂŹs roof, and is connected to a sealed box which contains a large battery box. The attached large battery box allows a regulated power output and access to electricity during dark hours.

With the customers not returning to Energy Kiosks, the battery boxes are not hired out like the battery boxes are. The high capital costs of the independent solar

system is spread over typically a two year rent-to-own payment plan using a mobile payment system.

It is hoped that the Standalone Solution and additional generation from the Energy Kiosk can be complemented the battery boxes in circulation to provide a continuous access to electricity to all households in the village.

FIGURE 2.4: First Generation Izuba.Box deplyed in Minazi, Northern Rwanda

Rugaragara Falls as a Simulation Scenario

In developing countries such as Rwanda, poor communities with no access to grid electricity are often in isolated locations such as Rugaragara Falls. The local District Sector office estimates Grid Access won't be available before 2020. In these areas, access is often difficult, making fuel for generators often difficult to obtain. Therefore, locals often depend on other available sources of electricity such as solar and wind power. However, these methods of generation provides highly variable amounts of energy which depends on other variables such as weather. Without access to redundancies received from the national electricity grid to ensure continual access to electricty, it can be beneficial for prosumers within these areas to form a micro-grid, which in many cases would be cheaper than connecting to the national grid.

In this project, a model of a rural electricity network that is disconnected from the grid is expected to be modeled. To allow the network to be scalable, the network is holonic in structure and consist of communities of prosumer "smart households" which can generate and utilise electricity depending on their demand and generation profiles. Therefore the simulator was required to have the following features:

- Multiple Forms of Generation: Renewable and Non-renewable generators which can operate continuously or discontinuously
- Realistic Generator Models: Programmable variable generation power output to simulate wind and solar power
- Multiple demand centres: the simulation will be of one or more communities operating with a number of households/businesses requiring electricity
- Self organising by the system to allocate the available power fairly to all users
- Presage 2: The simulation will be programmed in Java using Presage 2.

As this was a simulation implemented in Presage 2, there were no specific requirements which must be adhered to with regards to speed, portability and performance.

The simulation was developed using Presage 2, with the hope that a network of Decentralised Community Energy Systems can be simulated, and an algorithm for fairly allocating available generation to demand will be implemented. It is assumed that no cheating will take place.

Model Assumptions

For the purpose of this project, all participants in the community micro-grid must be prosumers. This means all participants are expected to contribute and consume electricity. Some important assumptions will be made about how the system operates in order to simplify the implementation of both the model and the micro grid:

- No losses would be incurred by the network
- All load on the network will be purely resistive
- All generation will act as negative load
- Only basic appliances such as phones, lights and fridges will be connected to the vast majority of households
- All prosumer households will have a battery

- Demand requests are made automatically based on their consumption at the time without user intervention, and therefore there cheating will not take place
- A household will have multiple power outlets. Consumption is measured by the amount of power required to power all of these power outlets with appliances connected at full power
- If allocated power is below the required amount to power all of the outlets, some of the outlets will be automatically turned off by a automatic load shedding mechanism. The order which the outlets are turned off can be set by the user so the user can make sure the most important appliance is connected to the outlet that will be the last to switch off

Model Design

To allow the Community Energy System to work as a holonic system, the micro-grid has been designed akin to the simplified model in Figure 3.1. The Agents in this case are reprsented by the Circles labeled A-H, with various demand/generation equipment connected to the Agents. The Agents are connected to a Virtual Agent or a Parent Agent represented by the single black dot that all Agents on the periphery are connected to in Figure 3.1. As the simulator is a Multi-Agent System, A Virtual Agent is employed to easily represent Group Demand and Group Generation of a small community.

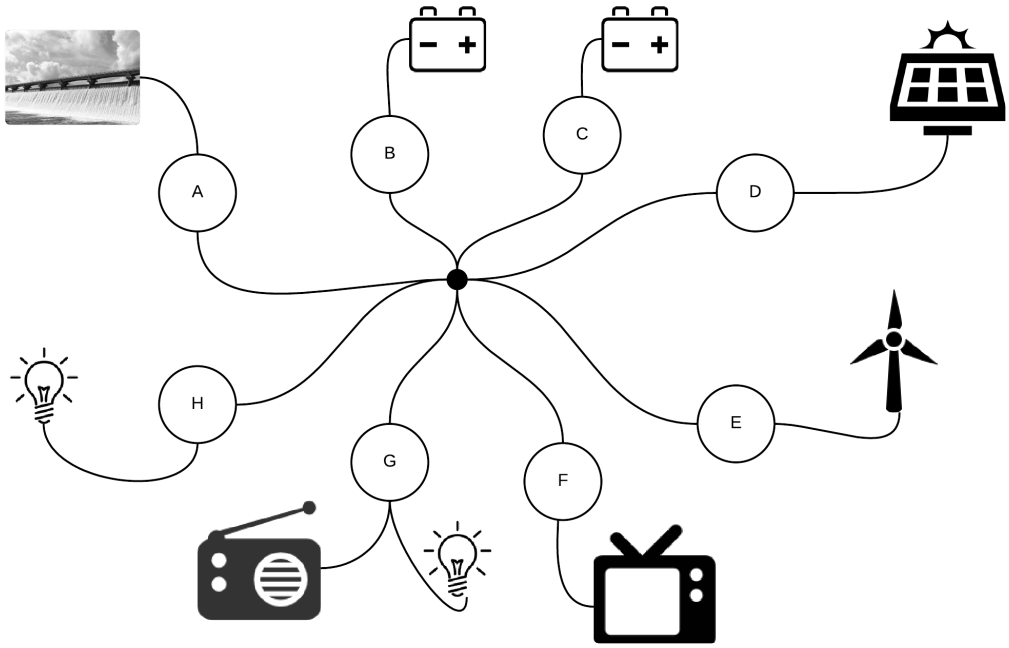

FIGURE 3.1: A simplified model diagram

Agents that group together are connected to a central virtual agent to allow the agents to form a community. These communities can further connected to another virtual agent to form even larger communities demonstrated in Figure 3.2.

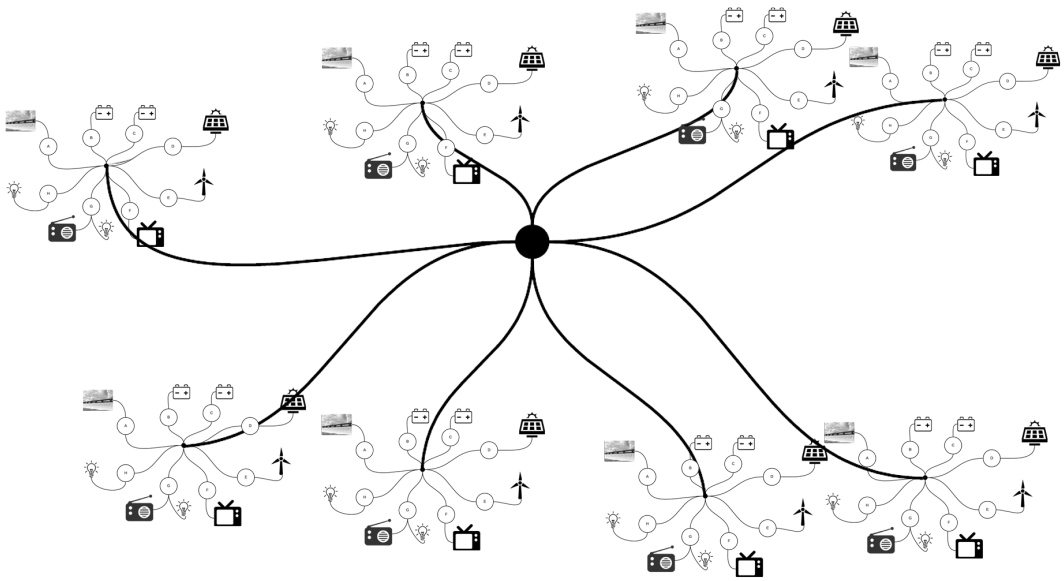

FIGURE 3.2: A simplified network model diagram

Types of Agents

A Multi-Agent System's main components are Agents, which are the actors which can act on the environment. Outlined below are the Agents that will be implemented as part of the simulator.

Virtual Agents represent all connected sub-communities or Agents in a community of Virtual Agents. These Agents will be responsible for enforcing quotas on connected sub-communities or Agents and collecting Generation for the Common Pool.

Prosumer Agents represent the most "elementary" system such as generators, households, businesses and other demand centres.

Agent Properties

Agent Properties represent the information that is to be relayed to the Community or Institution that the Agent is part of.

Demand - is a property which represents aggregated electricity consumption at a point of connection. Assumed demand curves will be produced from survey data of potential customers in the area for the initial testing. Should the survey data not be available for the area, a reasonable approximation will be made based a predicted usage habits of the wider local population.

It is anticipated that the final simulation will have a desired demand profile that each agent will aim to have.

For Virtual Agents, this property represents the aggregated Demand of all connected sub-communities or Agents and will not have any associated Demand profiles.

Generation - is a property which will allow all agents to generate power. For Virtual Agents, this property represents the aggregated Generation of all connected sub-communities or Agents. Four types of generator properties will be implemented in this simulation model for Prosumer Agents:

- Hydro-electric - a constant source of energy based on a mixture of historical data and projections.
- Solar - a source of energy following the typical output profile of a solar panel connected to households.
- Wind - a source of energy which will be highly variable in output.
- Diesel - a constant source of energy.

With the power output of renewable sources of energy such as wind and solar being non-predictable in nature, one of two approach will need to be undertaken to model these sources:

- A probabilistic generating factor is applied to the wind turbines. A constant amount of power each is assumed to be produced for a random amount of time at a random hour of the day that is to be determined. This can also happen for a random amount of times.
- An average generation power output curve is used for solar panels

A variation of both approaches are currently in use by distribution networks in the UK for assessing network congestion.[12]

Storage - Storage devices will be batteries of various types that can be connected to the network. For the purpose of this project, it will be assumed that all Prosumer Agents will have one of these to allow allocation of electricity on a hourly basis.

The Storage property should be designed have the capability to prioritise the allocation of its stored energy for certain Agents. For example, the energy output of Storage-only agents could be made to always prioritise the households they are attached to. If the battery box is communal or belongs to a centralised entity such as an e.quinox Energy Kiosk, then no priority will be attached.

Simulation with Presage 2

For the initial implementation and testing, the model will only contain two levels of Aggregation designed similar to 3.3 because the micro-grid in Rugaragara Falls will be only connecting one to two communities.

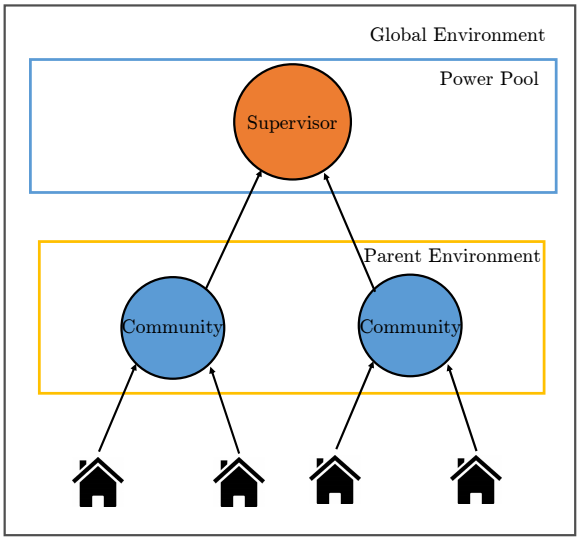

FIGURE 3.3: A simplified overview of the simulation structure

Similar to many other Multi-Agent Simulators, there can be no direct Agent-to-Agent communication. All communication must be done via the environment. As

such, the environments have been designed in such a way to enable multiple levels of holonic systems that can be seen in 3.4.

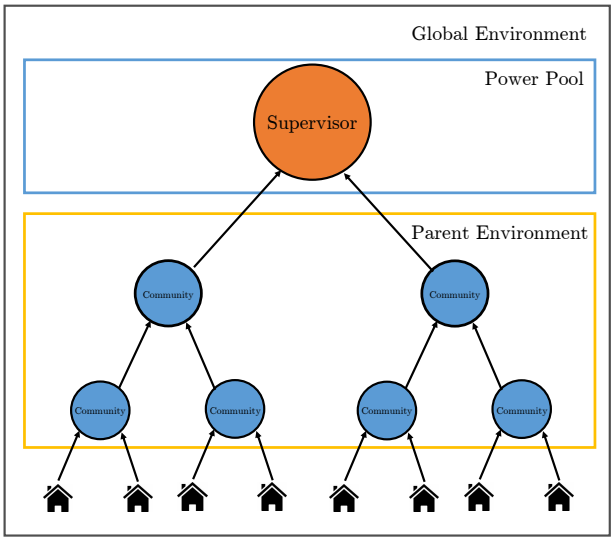

Figure 3.4: A simplified overview of the simulation structure

Architectually, Agents in the Simulation hold all of the information that is relevant to them. The Agents at the beginning of each allocation round submits their individual Demand and Generation request to the Virtual Agent that they are connected to. The Virtual Agents can submit their Group Demand and Group Generation Requests to other "Parent" Virtual Agents that they are connected to through the Parent Environment, and can do so recursively until the Group Demand and Generation have been sent to the Supervisor. By making the first initial allocation, the Supervisor is able to trigger a stream of allocation making by the Virtual Agents to the Virtual Agents and other Agents that are in their holonic sub-system.

During a round, Agents (or the houses in 3.3)are expected to publish their Demand/Generation request to the Virtual Agent that they are connected to. The Community that these parents are part of are represented by Virtual Agents. The Virtual Agent aggregates the connected Demand/Generation requests and publishes this information to the Supervisor. The Supervisor aggregates the total Demand/-Generation and appropriates the electricity fairly, and curtails the generation if there is an excess.

The basic algorithm for the simulation is outlined below in Algorithm 1:

for *each Agent* **do**
  request *Demand* and *Generation* from Virtual Agent;
end
for *each Virtual Agent* **do**
  request *Group Demand* and *Group Generation* from Supervisor Agent;
end
if *Group Generation >= Group Demand* **then**
  curtail Group Generation;
else
  allocate fairly Group Demand among Virtual Agents;
end
for *each Virtual Agent* **do**
  allocate *Group Demand* and *Group Generation* allocated from Supervisor Agent;
end

Algorithm 1: Basic Simulator Algorithm

Time

With the simulation being built on Presage 2, the simulation will be run in discrete time. Currently in the UK, energy is traded on a forward market at least 30 minutes ahead of consumption. In a similar fashion, the micro-grid will have the provision and allocation done on an hourly basis, one hour ahead of consumption. The Demand request will be made automatically based on their consumption at the time of allocation without user intervention. It is assumed that there is a device such a smart meter in place which can intelligently request Demand based on past user demand profiles and current electricity consumption.

Fair Allocation of the Common Pool Resource

With a lot of generation provided by renewable sources, there will be occurences when the aggregated Demand requests will be exceeded by the aggregated Generation requests. In circumstances such as these, electricity will need to be allocated to all users fairly. Rescher had observed in 1966 that a fair allocation can be found by ranking participants according to the Rescher's Canon of Distributive Justice:

1. Treatment as equals
2. Treatment according to their needs

3. Treatment according to their actual productive contribution
4. Treatment according to their efforts and sacrifices
5. Treatment according to a valuation of their socially-useful services
6. Treatment according to supply and demand
7. Treatment according to their ability, merit or achievements

From Rescher's analysis, using each canon (criteria) alone as a basis for a claim was inadequate as it was deemed unfair. Instead, for a fair system, all canons need to be employed in conjunction with each other. In a democratic and self-organising system, the importance of each canon in the eventual allocation should be decided by the system participants. In this simulation, how the participants are ranked according to the canons are based on a similar allocation systems demonstrated in *LPG'*[13] , and are defined as below:

1. ***The canon of equality*** - Agents can be ranked in increasing order of their average allocations
2. ***The canon of needs*** - Assuming similar agents will have similar Demand requests, the Agent most in need is the one that has made the smallest Demand request. Therefore Agents can be ranked in decreasing order of their average Demand requests.
3. ***The canon of productivity*** - Agents can be ranked in decreasing order of their economic output
4. ***The canon of effort*** - All Agents have made similar sacrifices and will gain a similar amount by being a participant of the Micro-Grid. Therefore, this canon is not represented in the simulation
5. ***The canon of social utility*** - Some Agents will provide services to the community such as providing a venue to watch a UEFA Champions League football game (something that occurs surprisingly often in rural Rwanda). Agents will be given a score of social utility based on the services they provide, and ranked in decreasing order
6. ***The canon of supply and demand*** - When allocations are required, the Agents which are most "in demand" are those Agents who contribute the most to the common pool. Therefore to encourage contribution, Agents are ranked in decreasing order of their average provision
7. ***The canon of merits and achievements*** - The merits and achievements of the Agents can't be accurately depicted in this simulation, and therefore excluded
8. ***Weighting of the canon*** - The importance placed in each of these canons when used to decide the ranking of the legitimate claims made by the participants is decided by a vote. At the end of each allocation round, participants

submit a vote to decide the importance of each of the canons for the next round

9. ***Voting and multiple claims*** - To allow multiple claims to be allocated fairly, each canon and each Agent is treated as a voter in the Borda count protocol. During each allocation round, each Agent is given a number of votes by each of the canons. Under Borda count, each Agent receives a number of points corresponding to $n\text{-}r+1$ with n being the number of Agents being ranked, and r where the Agent placed in the ranking. For example, in a ranking of $n=5$ Agents, the Agent ranked first receves $n\text{-}0=5$ votes, and last ranked Agent receiving $n\text{-}4=1$ vote. Similarly, at the end of each allocation round, Agents rank the canons in a similar fashion to decide the points of each of the canons according to the Borda count protocol.

Implementation 4

In this section, how the design is implemented will be described in detail.

Discrete Time Simulation Steps

Demand and Generation requests are made and satisfied on a hourly basis. Due to the nature of Presage 2, actions and requests happen in discrete time steps. In this simulation, each hour has to be split into 5 discrete time-steps due to the out of order execution of Agent actions and the discrete time nature of Presage 2. What actions are performed in each of the time steps are outlined below:

1. During the first time step, Agents submit their Generation and Demand requests to the Virtual Agent or Community they are part of.

2. During the second time step, Virtual Agents or Communities aggregate the Generation and Demand requests received and submit their Generation and Demand requests to the Supervisor Agent

3. During the third time step, the Supervisor Agent gathers the total Demand and Generation requests and subsequently allocates the Demand and Generation to the Virtual Agents

4. During the fourth time step, the Virtual Agents or Communities receive the allocation given by the supervisor, and allocates that to the Agents

5. During the 5th time step, the Agents appropriates the Demand and Generation allocation that has been given to them

Agent Class Structure and Implementation

Agents in Presage 2 are created by extending the *AbstractParticipant* class as shown in the UML diagram in figure 4.1. The class *MasterAgent* is how a Supervisor Agent is defined within the Simulator. *MasterAgent* is a simple class and is required to only do two things:

- Keep track of the Virtual Agents in the community it is reponsible for
- Allocating the correct amount of Demand and Generation to the Virtual Agents

To satisfy those requirements the following methods in the class are implemented:

- *addChild()* method is used for keeping track of the Virtual Agents. This is called during the initial Simulation set-up as part of the Agent creation and initialisation process.

- *step(int)* method is called by Presage during the simulation to allow the Agent to act on the environment. Demand and Generation are allocated to the Virtual Agents when this method is called automatically by Presage

A Virtual Agent (*ParentAgent* class) is a more advanced *MasterAgent*. It is required to:

- Keep track of which Agent is the Supervisor Agent
- Keep track of the Agents that depend on it to submit Demand and Generation requests on their behalf
- Submit Generation and Demand requests to the Supervisor Agent it is connected to
- Allocating the correct amount of Demand and Generation to the Agents

To perform the duties outlined above, the following methods are implemented:

- The *constructor* of this object is responsible for keeping track of which Agent is the Supervisor Agent. The *constructor* is called when this object is created in the initial Simulation set-up as part of the Agent creation and initialisation process.
- *addChild()* method is used for keeping track of the Agents. This is also called during the initial Simulation set-up as part of the Agent creation and initialisation process.
- *step(int)* method is called by Presage during the simulation to allow the Agent to act on the environment. Demand and Generation requests are sent to the supervisor, and allocation of Demand and Generation are also made to the Agents when this method is called automatically by Presage.

Prosumer Agents are Virtual Agents but with no aggregation function, as a Prosumer Agent is the most elementary Agent is the most elementary Agent or sub-community in this holonic system simulation. A Prosumer Agent is required to:

- Keep track of which Agent is the Virtual Agent
- Submit Generation and Demand requests to the Virtual Agent it is connected to
- Appropriate the allocated amount of Generation and Demand allocated

The Prosumer Agent is implemented with the following methods to allow it to perform the requirements outlined above:

- The *constructor* of this object is responsible for keeping track of which Agent is its *ParentAgent*. The *constructor* is called when this object is created in the initial Simulation set-up as part of the Agent creation and initialisation process.

- *addChild()* method is used for keeping track of the Agents. This is called during the initial Simulation set-up as part of the Agent creation and initialisation process.
- *step(int)* method is called by Presage during the simulation to allow the Agent to act on the environment. Demand and Generation requests are sent to the supervisor, and allocation of Demand and Generation are also made to the Agents when this method is called automatically by Presage.
- *addProductivity(), addSocialUtility(), addProfileHourly()* are the methods that are called during the initial Simulation set-up as part of the Agent creation and initialisation process to define the properties of this Agent.

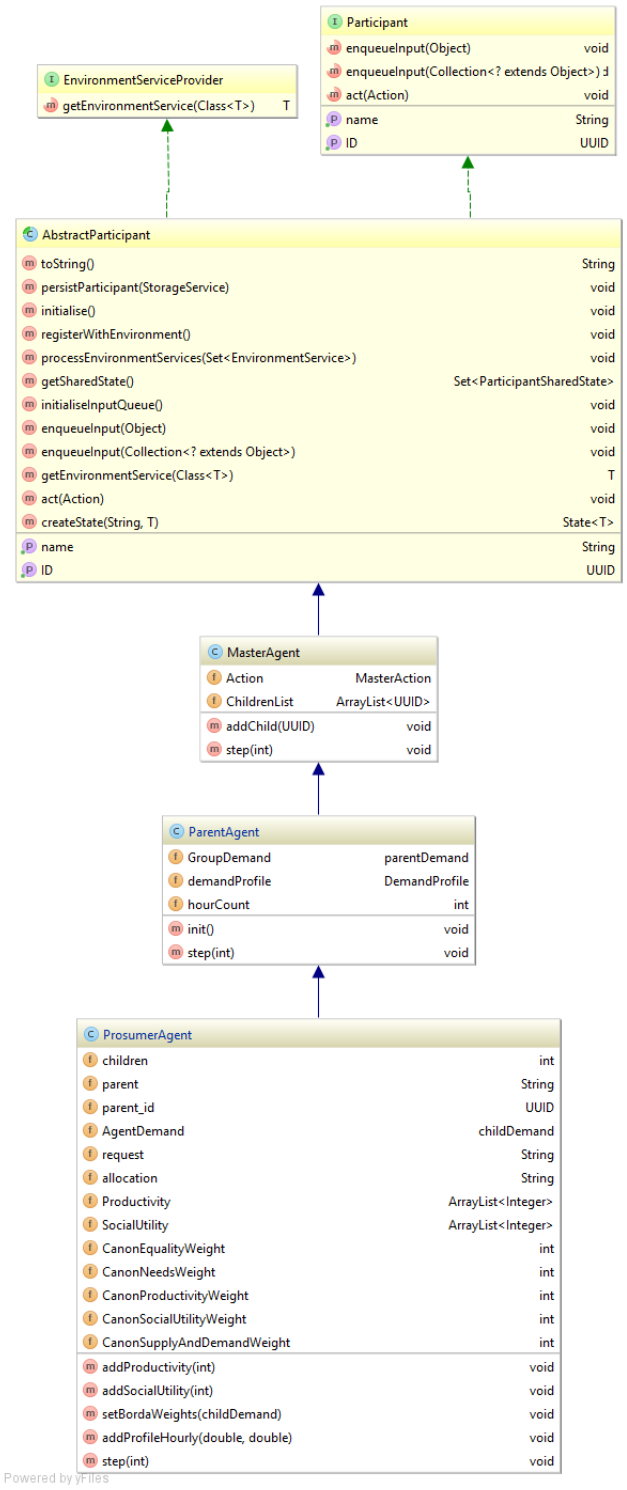

Figure 4.1: Agent UML Diagram

Environment Services

All Agents perform actions on Environments, which contains one or more "shared states". In Presage 2, all communication that happens between Agents are done via the environment by storing the data in a "shared state". In the context of this simulation shared state would be information such as Demand and Generation requests or the amount available in the Common Resource Pool. A visual diagram on how the Environment classes in this simulation are implemented can be found in figures 4.2 and 4.3.

Global Environment Service

All Agents are registered to the Global Environment Service, *GlobalEnvService* class. Like all Environment service classes in Presage, this was extended from the *EnvironmentService* class built into Presage. This class contains all of the methods that are called when allocations are being made. During time step 3 of a simulated hour, the *allocate()* method is called from the *MasterActionHandler* to perform allocations on behalf of the Supervisor Agent. *allocate()* method by default satisfies all the requirements of connected Virtual Agents if there is enough Generation to support it. If there is excess Generation, the excess is curtailed proportionally. If however, there isn't enough Generation, method *allocate_fairly()* is called, and the Virtual Agents are ranked according to the five applicable *Rescher's Canons of Distributive Justice*. These rankings are computed by calling the methods such as *canon_of_equality()*. The allocations are stored in the *PowerPoolEnvService*. At the end of the allocation process, the data about the allocation is stored in the environment ready for time step 3 of the next simulated hour via the method *environmentStore*.

Supervisor Environment Service

The *PowerPoolEnvService* is the Environment Service accessible by both the Supervisor Agent and the Virtual Agents. The *PowerPoolEnvService* inherits from *GlobalEnvService* class, and does a lot of the same things on a smaller scale. The *PowerPoolEnvService* is used to store information about the Virtual Agents, such as their aggregated Demand and Generation requests, and contains the same methods that are called when allocations are being made by the Virtual Agents. During time step 2 of a simulated hour, Virtual Agents sum up their Prosumer Agent Demand and Generation requests, and store them in *PowerPoolEnvService*. During time step 4 of a simulated hour, the *allocate()* method is called from the *parentDemandHandler*. *allocate()* method by default satisfies all the requirements of Agents if there is enough Generation to support it. If there is excess Generation, the excess is curtailed proportionally. In a holonic system, Agents are not able to access information concerning other Agents that it is not directly connected to. A separate Environment Service is used to prevent the Supervisor Agent from being able to access and modify "shared states" about Prosumer Agents that are not directly connected to the Supervisor Agent.

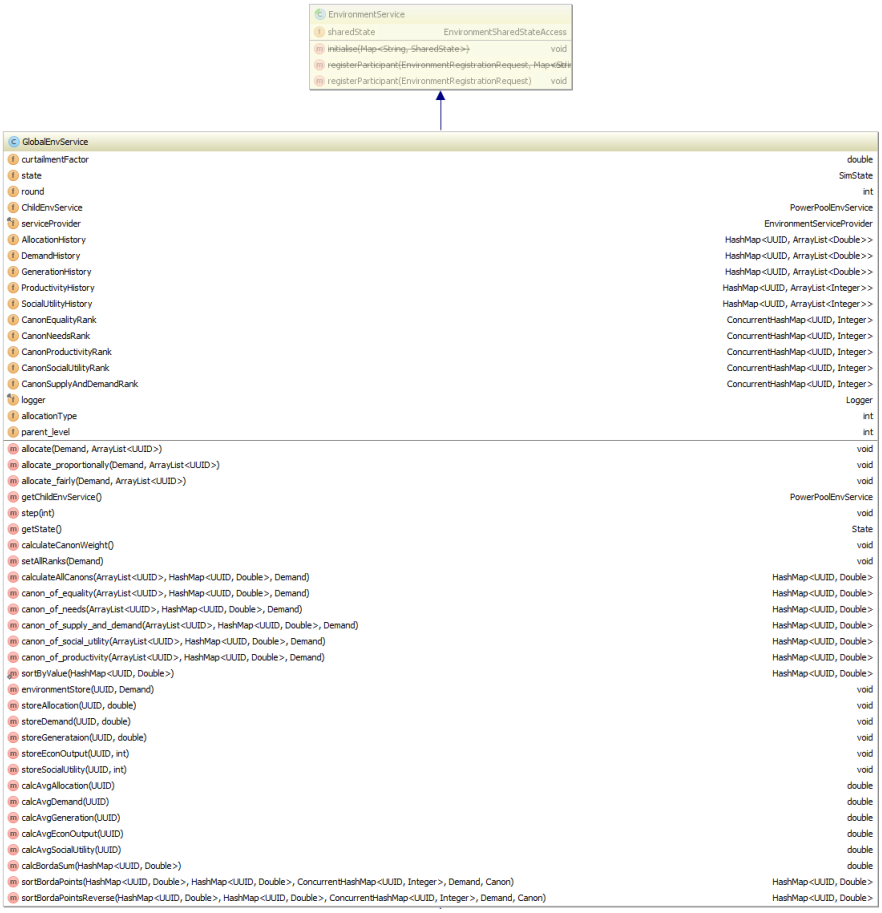

Figure 4.2: Environment Services UML Diagram

Simiar to the *GlobalEnvService* class, if there isn't enough Generation, method *allocate_fairly()* is called, and the Agents are ranked according to the five applicable *Rescher's Canons of Distributive Justice*. These rankings are computed by calling the methods such as *canon_of_equality()* which are inherrited from the *GlobalEnvService*. *PowerPoolEnvService* contains *Overriding allocate()* and textitallocate_fairly() methods and do not inherrit these from the *GlobalEnvService*, because the allocation data is required to be stored in the *ParentEnvService*. At the end of the allocation process, the data about the allocation is stored in the environment ready for time step 4 of the next simulated hour via the method *environmentStore*.

Virtual Agent Environment Service

The *ParentEnvService* is the Environment Service accessible by both the Virtual Agents and the Prosumer Agents. The *ParentEnvService* inherits from the *PowerPoolEnvService*, and does the same things on a smaller scale. The *ParentEnvService* is used to store information about the Prosumer Agents, such as their individual Demand and Generation requests, and also contain information about their allocations.

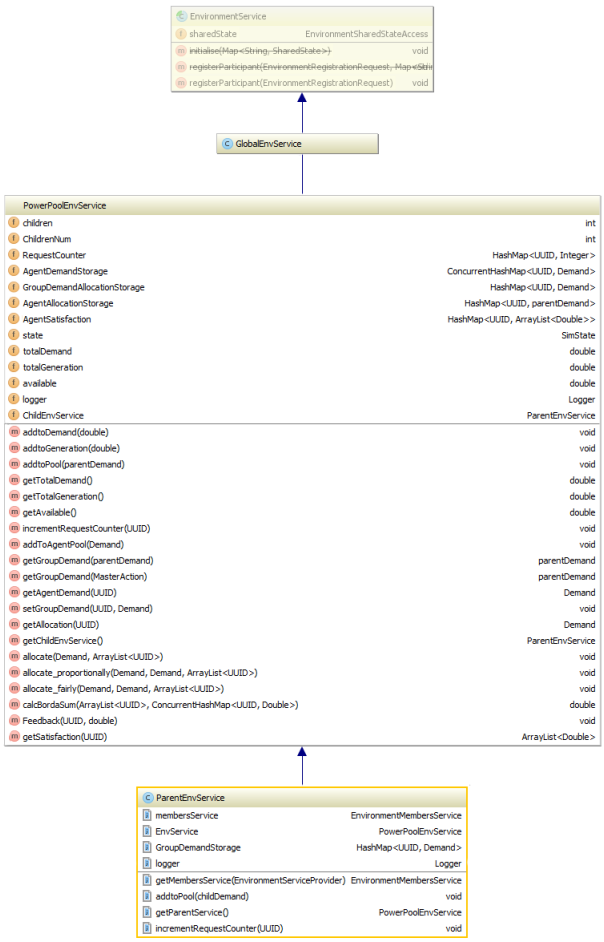

FIGURE 4.3: Environment Services UML Diagram

Action

To act on the Environment or acess a shared state in the Environment, Agents are expected to perform an Action. In the context of this simulation, Action would be Demand/Generation requests. As Generation can be modelled as a negative Demand, a single Action can be defined to allow the contribution and appropriation of electricity.

Action Handlers

To enable the Environment to be able to process the Action requests, Action Handlers need to be created to tell the simulation how to deal with Actions from Agents.

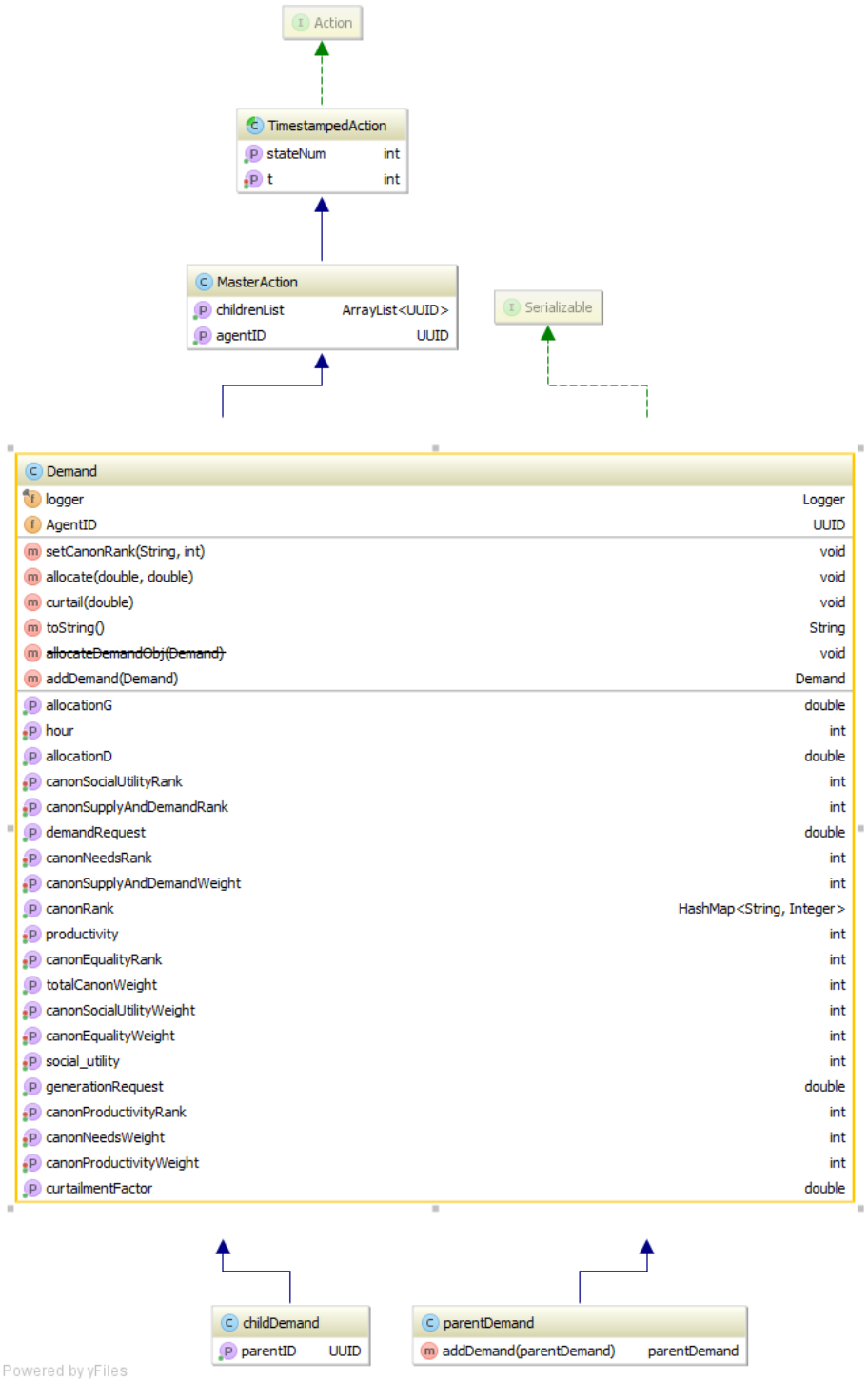

Figure 4.4: Actions UML Diagram

Simulation

Issues

28 Out of order parallel execution meant that Agents need to submit their individual Demands to the SharedState, and have that summed at the end of each timestep. It

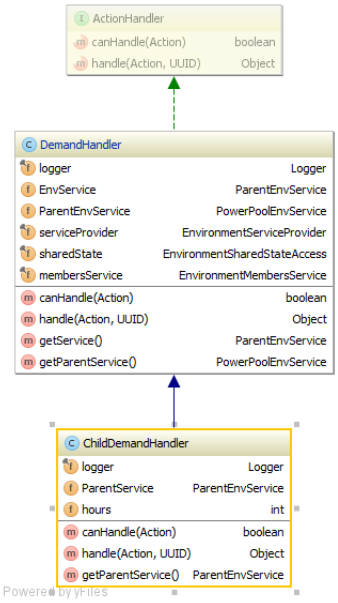

FIGURE 4.5: Action Handler UML Diagram

is not possible to sum the Demands on the fly.

Being new to both Java and Presage presented problems of its own. It was difficult to understand how simulations could be run and therefore create our own.

One action per time step meant that it takes 4 time steps to simulate one round of request and appropriation of electricity. It would therefore take 24*4 time steps to simulate a full day of requests and appropriation.

Difficulty with initialising Agents with arrays meant that Agents had to be created with no Demand or Generation Profiles, and the Demand and Generation Profiles were added one by one via a for-loop and the addProfileHourly() method.

Testing 5

The final product of this project should be a realistic simulation of a Decentralised Community Energy System. To produce the simulation, each of the following features will be tested to ensure they work individually and together:

- Agents are able to submit their requests to the Virtual Agent they are connected to.
- Virtual Agents are able to submit their requests to the Supervisor.
- Virtual Agents and their Agents are allocated Generation dispatch and an amount of electricity to use

Due to time constraints, unit testing was not implemented. A series of tests was done manually by exporting the data using Excel to prove that the simulator was working as intended. The testing has been outlined in the sections below. Each of the tests below has been conducted for the following simulation cases:

- 1 Supervisor, 1 Virtual Agent and 1 Agent
- 1 Supervisor, 2 Virtual Agent and 2 Agents (Each Virtual Agent connected to 1 Agent)
- 1 Supervisor, 2 Virtual Agents and 4 Agents (Each Virtual Agent connected to 2 Agents)
- 1 Supervisor, 5 Virtual Agents and 25 Agents (Each Virtual Agent connected to 5 Agents)

Testing Request Submission

To test that Agents are able to submit their Demand and Generation requests, the individual Demand and Generation request of each Agent was recorded in a CSV file. The allocations in the CSV file was summed and compared to that of the computed Global Demand and Generation request.

An example can be seen in figure 5.1 of testing the contribution of 25 Agents during hour 23 of a simulation

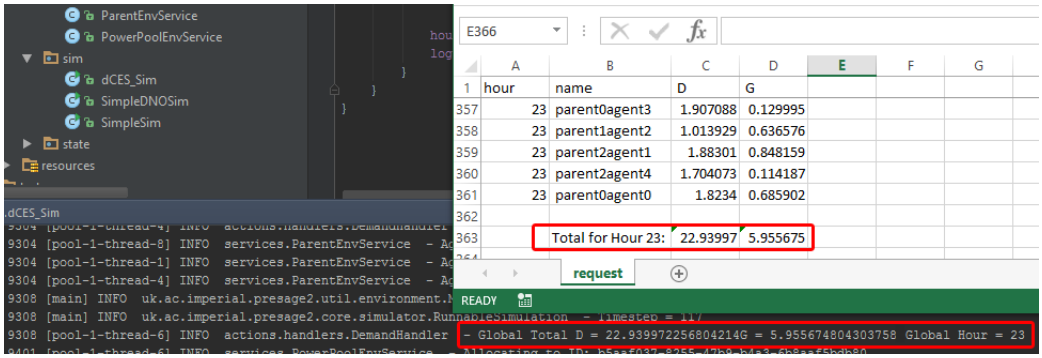

Figure 5.1: Testing the contribution at hour 23 of the simulation for 25 Agents

Testing Allocation Algorithm

When the global aggregated Generation requests exceed the global aggregated Demand requests, all Agents are expected to receive their requests. Globally, Generation is curtailed to not exceed total Demand Request. Generation is curtailed proportionally for everyone. In this case, we should expect all Agents to be allocated less Generation than they have requested, and all Agents to be allocated their Demand requests. An example of this test being passed can be seen in figure 5.2.

When the global aggregated Demand request exceed the global Demand request, the fair allocation method will need to be used. To ensure this is working as intended, we need to ensure the following tests are passed:

1. Total allocated Demand is equal to total allocated Generation
2. Total allocated Generation is equal to the total Generation request
3. Demand allocation proportion is correct
4. Borda ranking is correct for each of the canons
5. Borda voting by the Agents are correct
6. Borda voting is taken into consideration in the next round

Tests 1 and 2

To check that total allocated Demand is equal to total Generation dispatch, and the total Generation dispatch is equal to the total Generation request, the total Generation Request, Generation Dispatch and Demand Allocation were summed up for each hour of the simulation. Figure 5.3 shows the data for one of such tests. The summation at the bottom shows that the total allocated Demand (Allocated D) and Generation Dispatch (Allocated G) is equal to the total requested Generation (Request G).

I13 =IF(F13<D13, "yes", "no")

	A	B	C	D	E	F	G	H	I	J
1	hour	name	Request D	Request G	Allocated D	Allocated G		Request D == Request G?	Is Generation Curtailed?	
2	0	parent0agent0	1.592401359	2.311111995	1.592401359	1.453660808		yes	yes	
3	1	parent0agent0	1.995160167	2.229004158	1.995160167	1.367390779		yes	yes	
4	2	parent0agent0	1.769509059	2.445681676	1.769509059	1.39819329		yes	yes	
5	3	parent0agent0	1.648368572	2.823361438	1.648368572	1.624280561		yes	yes	
6	4	parent0agent0	1.698531447	2.098485929	1.698531447	1.275250851		yes	yes	
7	5	parent0agent0	1.245672833	2.868123024	1.245672833	1.793544947		yes	yes	
8	6	parent0agent0	1.198098745	2.056996387	1.198098745	1.300946125		yes	yes	
9	7	parent0agent0	1.844929509	2.098140893	1.844929509	1.350585918		yes	yes	
10	8	parent0agent0	1.558576104	2.897400312	1.558576104	1.833657969		yes	yes	
11	9	parent0agent0	1.030802763	2.149425156	1.030802763	1.31492169		yes	yes	
12	10	parent0agent0	1.350170927	2.778337346	1.350170927	1.64977437		yes	yes	
13	11	parent0agent0	1.910639238	2.753283402	1.910639238	1.72097032		yes	yes	
14	12	parent0agent0	1.491592782	2.606799055	1.491592782	1.545341026		yes	yes	
15	13	parent0agent0	1.924255335	2.006983643	1.924255335	1.156448361		yes	yes	
16	14	parent0agent0	1.254589004	2.313282333	1.254589004	1.379315387		yes	yes	
17	15	parent0agent0	1.815323723	2.306176827	1.815323723	1.501893006		yes	yes	
18	16	parent0agent0	1.857218884	2.114889386	1.857218884	1.291346395		yes	yes	
19	17	parent0agent0	1.480209778	2.681540784	1.480209778	1.657888309		yes	yes	
20	18	parent0agent0	1.307517911	2.067918309	1.307517911	1.110023994		yes	yes	
21	19	parent0agent0	1.309522885	2.220584418	1.309522885	1.261815735		yes	yes	
22	20	parent0agent0	1.199242606	2.915590237	1.199242606	1.691612538		yes	yes	
23	21	parent0agent0	1.068311668	2.975277496	1.068311668	1.634058805		yes	yes	
24	22	parent0agent0	1.232676004	2.221910139	1.232676004	1.414685194		yes	yes	
25	23	parent0agent0	1.322745932	2.257147764	1.322745932	1.391876975		yes	yes	
26	0	parent0agent1	1.983027954	2.950334956	1.983027954	1.85572413		yes	yes	
27	1	parent0agent1	1.990649072	2.777340226	1.990649072	1.703769552		yes	yes	
28	2	parent0agent1	1.091557528	2.654721342	1.091557528	1.517701017		yes	yes	
29	3	parent0agent1	1.027610958	2.316036187	1.027610958	1.332416214		yes	yes	
30	4	parent0agent1	1.779710396	2.701054557	1.779710396	1.641432079		yes	yes	
31	5	parent0agent1	1.663057514	2.463041132	1.663057514	1.54023204		yes	yes	
32	6	parent0agent1	1.117376148	2.399024928	1.117376148	1.51726187		yes	yes	
33	7	parent0agent1	1.849330187	2.827728518	1.849330187	1.820225863		yes	yes	
34	8	parent0agent1	1.582549611	2.086502422	1.582549611	1.320470553		yes	yes	
35	9	parent0agent1	1.549677387	2.857919857	1.549677387	1.7483469		yes	yes	
36	10	parent0agent1	1.123213878	2.235238446	1.123213878	1.327282702		yes	yes	
37	11	parent0agent1	1.120824313	2.575954255	1.120824313	1.61012877		yes	yes	
38	12	parent0agent1	1.469979531	2.822010251	1.469979531	1.672920744		yes	yes	
39	13	parent0agent1	1.100733124	2.147354561	1.100733124	1.23733179		yes	yes	

FIGURE 5.2: Testing the allocation of 25 Agents when total Generation Request Exceeds total Demand Request

Test 3

To test that the allocation proportion is correct, Borda ranking and point allocation for each Agent under each of the canons were checked to be correct. Figure 5.4 shows the previous requests and allocations made by Agents under Virtual Agent *Parent0* in one of these tests. The data for the requests and allocations were gathered from *"allocation.csv"* and *"request.csv"* , which are generated when the simulation is running. Figure 5.6 shows the ranking that is based on historical data of the Agents, which matches with what the data shows in figure 5.5.

With the Ranking checked to be correct, the Borda Point allocation for each of the Agents were also checked. The Borda point allocation for the example dataset above can be found in figure 5.7. The allocations according to the weighted Borda point allocation can be found in figure 5.8.

	A	B	C	D	E	F	G	H	I
1	hour	name	Request D	Request G	Productivity	Social Utility	Allocated D	Allocated G	Satisfaction
352	14	parent0agent0	1.472868	1.71	1	2	1.062328767	1.71	0.721265427
353	14	parent0agent1	1.472868	1.71	4	4	1.203972603	1.71	0.817434151
354	14	parent0agent2	1.472868	1.014166667	0	4	0.708219178	1.014166667	0.480843618
355	14	parent0agent3	1.472868	1.014166667	0	3	1.203972603	1.014166667	0.817434151
356	14	parent0agent4	1.472868	1.014166667	4	0	0.991506849	1.014166667	0.673181065
357	14	parent1agent0	1.472868	1.71	0	4	1.2925	1.71	0.877539603
358	14	parent1agent1	1.472868	1.71	2	0	1.384821429	1.71	0.940221003
359	14	parent1agent2	1.472868	1.014166667	1	0	1.107857143	1.014166667	0.752176803
360	14	parent1agent3	1.472868	1.014166667	3	4	1.384821429	1.014166667	0.940221003
361	14	parent1agent4	1.472868	1.014166667	0	0	1.2925	1.014166667	0.877539603
362	14	parent2agent0	1.472868	1.71	1	0	1.444201878	1.71	0.980537209
363	14	parent2agent1	1.472868	1.71	2	4	1.547359155	1.71	1.050575581
364	14	parent2agent2	1.472868	1.014166667	3	4	1.444201878	1.014166667	0.980537209
365	14	parent2agent3	1.472868	1.014166667	0	4	1.237887324	1.014166667	0.840460465
366	14	parent2agent4	1.472868	1.014166667	3	4	1.650516432	1.014166667	1.120613953
367	14	parent3agent0	1.472868	1.71	1	2	1.698357488	1.71	1.153095517
368	14	parent3agent1	1.472868	1.71	1	2	1.379915459	1.71	0.936890108
369	14	parent3agent2	1.472868	1.014166667	2	4	1.273768116	1.014166667	0.864821638
370	14	parent3agent3	1.472868	1.014166667	0	0	1.379915459	1.014166667	0.936890108
371	14	parent3agent4	1.472868	1.014166667	3	0	1.592210145	1.014166667	1.081027047
372	14	parent4agent0	1.472868	1.71	2	3	1.07413242	1.71	0.729279488
373	14	parent4agent1	1.472868	1.71	3	4	1.322009133	1.71	0.897574754
374	14	parent4agent2	1.472868	1.014166667	2	0	1.404634703	1.014166667	0.953673176
375	14	parent4agent3	1.472868	1.014166667	3	4	0.991506849	1.014166667	0.673181065
376	14	parent4agent4	1.472868	1.014166667	2	0	1.239383562	1.014166667	0.841476332
602									
603		Sum:	36.8217	32.3125			32.3125	32.3125	

Figure 5.3: Checking that Allocated Demand is less than total Allocated Generation

	A	B	C	D	E	F	G	H
1	hour	name	Request D	Request G	Productivity	Social Utility	Allocated D	Allocated G
2	0	parent0agent0	0.422494228	1.143316614	4	0	0.422494228	1.058670544
3	1	parent0agent0	1.167911784	1.516214737	3	0	0.60758766	1.516214737
4	0	parent0agent1	0.422841507	1.139822124	0	0	0.422841507	1.05543477
5	1	parent0agent1	1.158514651	1.519551714	2	3	0.364552596	1.519551714
6	0	parent0agent2	0.422375141	0	0	4	0.422375141	0
7	1	parent0agent2	1.163226822	0	2	2	0.60758766	0
8	0	parent0agent3	0.42307295	0	2	3	0.42307295	0
9	1	parent0agent3	1.16789018	0	0	1	0.394931979	0
10	0	parent0agent4	0.42266157	0	1	2	0.42266157	0
11	1	parent0agent4	1.159284888	0	1	4	0.455690745	0

Figure 5.4: Historical Parent0 Agent requests and allocations

	Average Allocation	Demand Req	Productivity	Social Utility	Supply and Demand
parent0agent0	0.515040944	0.795203006	7	0	1.329765675
parent0agent1	0.393697052	0.790678079	2	3	1.329686919
parent0agent2	0.514981401	0.792800982	2	6	0
parent0agent3	0.409002465	0.795481565	2	4	0
parent0agent4	0.439176157	0.790973229	2	6	0

Figure 5.5: Historical Parent0 Agent requests and allocations data processed

	A	B	C	D	E	F	G	H
1	hour	name	id	CanonEqualityRar	CanonNeedsRar	CanonProductivityRar	CanonSocialUtilityRar	CanonSupplyAndDemandRar
58	2	parent0agent0	dd98f973-	5	4	1	5	1
65	2	parent0agent1	b72cd33d-	1	1	2	4	2
68	2	parent0agent2	eea2a064-	4	3	2	1	3
71	2	parent0agent3	2836b64b-	2	5	2	3	3
76	2	parent0agent4	f0495904-	3	2	2	1	3
502								

FIGURE 5.6: Parent0 Agent ranking

	A	B	C	D	E	F	G	H	I
1	hour	canon	agent	id	Borda Pts	Borda pts norm	Borda Proportion	Borda Votes for Canon	Total Borda Votes
442	2	equality	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	1	0.1875	0.1875	15	80
444	2	equality	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	5	0.9375	0.1875	15	80
446	2	equality	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	2	0.375	0.1875	15	80
448	2	equality	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	4	0.75	0.1875	15	80
450	2	equality	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	3	0.5625	0.1875	15	80
453	2	needs	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	2	0.375	0.1875	15	80
454	2	needs	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	5	0.9375	0.1875	15	80
457	2	needs	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	3	0.5625	0.1875	15	80
458	2	needs	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	1	0.1875	0.1875	15	80
461	2	needs	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	4	0.75	0.1875	15	80
931	2	productivity	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	5	1	0.2	16	80
933	2	productivity	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	4	0.8	0.2	16	80
939	2	productivity	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	4	0.8	0.2	16	80
944	2	productivity	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	4	0.8	0.2	16	80
949	2	productivity	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	4	0.8	0.2	16	80
953	2	social_utility	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	1	0.2	0.2	16	80
957	2	social_utility	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	2	0.4	0.2	16	80
965	2	social_utility	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	5	1	0.2	16	80
969	2	social_utility	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	3	0.6	0.2	16	80
976	2	social_utility	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	5	1	0.2	16	80
983	2	supply_and_demand	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	5	1.125	0.225	18	80
985	2	supply_and_demand	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	4	0.9	0.225	18	80
994	2	supply_and_demand	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	3	0.675	0.225	18	80
996	2	supply_and_demand	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	3	0.675	0.225	18	80
1001	2	supply_and_demand	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	3	0.675	0.225	18	80
2052									

FIGURE 5.7: Parent0 Hour 2 Borda Point Allocation

		Available Power	2.192999123	
		Weighted Points	%	
	Agent0	2.8875	0.169106881	0.37085124
	Agent1	3.975	0.232796486	0.51052249
	Agent2	3.4125	0.199853587	0.43827874
	Agent3	3.0125	0.176427526	0.38690541
	Agent4	3.7875	0.22181552	0.48644124
	Sum:	17.075		0

FIGURE 5.8: Parent0 Allocations according to the Borda Points

Tests 4 and 5

To check that the Borda voting mechanism is correct, the votes for each Agent was checked against where they were ranked the previous round. For the dataset used above, this can be found in figure 5.9.

	A	B	C	D	E	F	G	H	I
1	hour	agent	id	Equalityweight	ProductivityWeight	UtilityWeight	NeedsWeight	SupplyAndDemandWeight	
54	2	parent0agent0	dd98f973-3eaf-45ab-a149-f45c7d162d22	1	5	1	2	5	
65	2	parent0agent1	b72cd33d-1e34-441f-af01-d2235c52858e	5	4	2	5	4	
74	2	parent0agent2	eea2a064-62c7-4b4c-a07d-58b5a5b1000f	2	4	5	3	3	
75	2	parent0agent3	2836b64b-1cba-48df-a7e8-93cf1f128248	4	4	3	1	3	
76	2	parent0agent4	f0495904-0e2b-4fd9-a5e8-8ee92ba0c87c	3	4	5	4	3	
603									
604			Sum	15	21	16	15	18	
605			Total:	85					
606									

Figure 5.9: Parent0 Agent votes for the weight of the canons in the next round

To check that the voting mechanism was fed back into the system, the aggregate votes for each of the canons for all Agents were checked. For the data set above, this can be shown in figure 5.10. *Borda Proportion* represents the percentage of votes that went into a particular canon under the Borda Voting protocol for all of the Agents.

	A	B	C	D	E	F	G	H	I
1	hour	canon	agent	id	Borda Pts	Borda pts norm	Borda Proportior	Borda Votes for Canon	Total Borda Votes
495	3	equality	parent0agent0	dd98f973-	2	0.352941176	0.176470588	15	85
765	3	needs	parent0agent0	dd98f973-	2	0.352941176	0.176470588	15	85
1030	3	productivity	parent0agent0	dd98f973-	5	1.235294118	0.247058824	21	85
1043	3	social_utility	parent0agent0	dd98f973-	1	0.188235294	0.188235294	16	85
1091	3	supply_and_demand	parent0agent0	dd98f973-	5	1.058823529	0.211764706	18	85

Figure 5.10: Parent0 Agent votes in hour 3

Bugs

Under Presage 2, all communication between Agents are stored in the Environment Services by the sending party, and accessed by the receiving party. With all simulations under Presage 2 being multi-threaded, there are some occurrences where some Agents are in different time-steps in the simulation. If a Virtual Agent was in a time-step ahead of the Agents connected to it, the Virtual Agent will sometimes access the Demand and Generation request data before it is ready. This usually causes a Java concurrency error warning in the console. This bug rarely occurs, and usually disappears when the simulation is rerun.

Results 6

6.1 One Agent (No Participation)

6.2 Multi-Agent Fair Allocation

6.3 Multi-Agent Proportional Allocation

Evaluation 7

Conclusions 8

Since October, an attempt has been made on understanding the properties of Common Pool Resource, and how electricity can be a Common Pool Resource in the context of a Decentralised Community Energy System. An attempt has been made on building a simulator of a Decentralised Community Energy System in developing countries.

The simulator being built has the ability to self-organise and fairly allocate electricity to end-users.

This Simulator of a Decentralised Community Energy System can be further developed with more features to explore possible utilisation patterns within rural communities, and also explore controls for regulating the use of electricity.

Further Work 9

So far, the work carried out has been only be applicable for small rural communities in developing countries, and has the scope to be developed further to a larger scale to model future smart grid networks in developed nations. Currently, much of the costs surrounding set-up costs, network maintenance costs have not been included. Additional limitations include purely resistive loading and no network losses. Future work can improve the accuracy of the model by taking into consideration these additional costs and losses.

An attempt has been made at fairly allocating energy to end users. More data would be required to implement all 8 Rescher's Canons of Distributive justice.

Currently, only renewable sources of generation were included, as these were the most prevailent in the area studied. It would be realistic to include diesel generators and their costs in further works.

Currently, only one aspect has been explored in the holonic model. The model employed in this simulation can be further adapted to include other bottlenecks and opportunities such as government intervention, grid electricity and financial incentives. The Agents' behavioral model are very simple, and can be further improved.

In the immediate future, it is hoped that this tool could be used to aid the feasibility study of the Micro-Grid to be implemented in Rugaragara Falls.

Consider Diesel Generators' costs Consider satisfaction of individual agents for them to partake in this

In the simulation, carry forward unused generation because we have storage. Any excess generation is simply curtailed.

Trading platform.

User Guide 10

The github source code repository for this project can be found at: https://github.com/yuchen-w/DCES-Simulator

The code was developed in IntelliJ Idea 14.1.3 IDE and should work with other IDEs. More detailed instructions can be found in the README.md of the repository.

Bibliography

[1] H. e. a. Chou, *Group Project*, 2015 (accessed June 6, 2015). http://github.com/kyotoprotocol/kyoto/raw/development/report/report.pdf.

[2] M. O. Infrastructure, *Energy Sector Strategic Plan*, 2015 (accessed February 2, 2015). http://www.minecofin.gov.rw/fileadmin/templates/documents/sector_strategic_plan/Energy_SSP_June_2013.pdf.

[3] N. Grid, *Customer service and network reliability*, 2015 (accessed February 2, 2015). http://www2.nationalgrid.com/responsibility/how-were-doing/grid-data-centre/Customer-service-and-network-reliability/.

[4] IEA, *IEA - Access to Electricity*, 2015 (accessed February 2, 2015). http://www.worldenergyoutlook.org/resources/energydevelopment/accesstoelectricity/.

[5] Elinor Ostrom, "Governing the Commons: The Evolution of Institutions for Collective Action," 1990.

[6] J. Pitt, *Holonic Institutions for Multi-Scale Polycentric Self-Governance*, 2015. Draft Unpublished Paper.

[7] presage2.info, *Presage2: About*, 2015 (accessed February 2, 2015). http://www.presage2.info/w/Main_Page.

[8] e.quinox, *e.quinox About*, 2015 (accessed January 27, 2015). http://e.quinox.org.

[9] e.quinox, *e.quinox Energy Kiosks*, 2015 (accessed January 27, 2015). http://www.e.quinox.org/index.php/our-solutions/energy-kiosks.

[10] e.quinox, *Design and Construction of a Pico Hydro Power System in Rural Rwanda*, 2012 (accessed Februar 2, 2015). http://e.quinox.org/reports/equinox_hydro_2012.pdf.

[11] e.quinox, *The Stand Alone Model*, 2012 (accessed Februar 2, 2015). http://e.quinox.org/reports/e.quinox%20Stand%20Alone%20Model.pdf.

[12] TNEI, *Web Based Constraint Analysis - Lincolnshire Low Carbon Hub*, 2015 (accessed February 2, 2015). http://www.ipsa-power.com/education/2013-user-group-meeting/constraint-analysis.

[13] J. Pitt, J. Schaumeier, D. Busquets, and S. Macbeth, "Self-organising common-pool resource allocation and canons of distributive justice," in *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, pp. 119–128, Sept 2012.

Index