UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS Nr. 315

A SIMULATION PLATFORM FOR POWER TRADING

Jurica Babić

Zagreb, June 2012

The author wishes to express his gratitude to his supervisor, Assist. Prof. Vedran Podobnik who was abundantly helpful and offered invaluable assistance, support and guidance.

Special thanks go to Assist. Prof. John Collins and Assist. Prof. Wolfgang Ketter who were supportive about the author's work on the Power TAC Visualizer.

Also, the author wishes to acknowledge the valuable contributions of his Power TAC team members (Siniša Matetić, Marin Matijaš, Marija Mijić, Ivo Buljević, Ivana Pranjić, Mia Augustinović and Nikola Jakić) who have contributed to design and implementation of the CrocodileAgent 2012 and who have contributed to the article [1] *, the content of which was used in the "CrocodileAgent 2012" section of this thesis.*

Finally, the author wishes to acknowledge the great amount of support received from his family and friends during the author's master study.

Abbreviations

EU	*European Union*
DSM	*Demand Side Management*
ICT	*Information and Communications Technology*
EISA	*Energy Independence and Security Act*
NIST	*National Institute of Standards and Technology*
EEGI	*European Electricity Grid Initiative*
SET-PLAN	*Strategic Energy Technology Plan*
ENTSO-E	*European Network of Transmission System Operators for Electricity*
EDSO	*European Distribution System Operators for Smart Grids*
ETP	*European Technology Platform*
RD&D	*Research, Development and Demonstration*
EMCAS	*Electricity Market Complex Adaptive Systems Model*
MAIS	*Multi Agent Intelligent Simulator*
ACE	*Agent-based Computational Economics*
TAC	*Trading Agent Competition*
TAC AA	*Trading Agent Competition Ad Auctions*
TAC SCM	*Trading Agent Competition Supply Chain Management*
Power TAC	*Power Trading Agent Competition*
DU	*Distribution Utility*
Genco	*Generation Company*
JSF	*JavaServer Faces*
XHTML	*eXtensible HyperText Markup Language*
CSS	*Cascading Style Sheets*
URL	*Uniform Resource Locator*
POJO	*Plain Old Java Object*
JMS	*Java Message Service*

Contents

List of figures

List of tables

Introduction

The current electrical power systems are switching from a traditional grid to an advanced grid called *smart grid*. Changes that occur in the electricity sector will affect the way customers use electricity. It is believed that retail customers with smart metering equipment installed will be able to adjust their consumption habits, depending on a market price signal received from a smart grid. Thanks to new technological solutions, a user becomes an essential element in real-time alignment of energy demand and supply within the local area. In addition to technical aspects of smart grid, establishment of the retail electric energy market for supporting the market aspect of the smart grid systems is crucial. Consequently, what is lacking in addition to technical infrastructure of a smart grid is an efficient set of market mechanisms. In order to avoid bad market design once smart grids are going to be widely deployed, it is necessary to provide a risk-free environment for testing market regulative.

Power Trading Agent Competition (Power TAC) is an open, competitive market simulation platform that addresses the need for policy guidance based on robust research result on the structure and operation of retail electrical energy markets. The Power TAC is an international project created with the cooperation of six universities across Europe and North America, including the University of Zagreb. Responsibility of the University of Zagreb's team is twofold. First responsibility is to develop a visualization module for the Power TAC platform. Developed solution, presented in this paper, is not only scalable, robust and based on state-of-the-art technologies, but has a huge scientific value because it proposes a solution to seize dynamics of electric power markets through identifying, analysing and presenting stakeholders, processes and key interactions in a smart grid environment. Second responsibility is to develop an efficient agent for the Power TAC 2012.

Motivation for transition from an existing, traditional electrical grid to a smart grid, the economic aspect of the smart grid as well as the review of possible approaches for market modelling is given in the first chapter. The second chapter contains general information about the Power TAC project and provides description of the Power TAC platform design. The technical aspect of the Power TAC platform is given in the third

chapter. The fourth chapter describes a design of the Power TAC Visualizer, a visualization module for Power TAC platform developed by University of Zagreb team. The last chapter describes a design of the University of Zagreb's entry for the Power TAC 2012, a software agent called the CrocodileAgent 2012.

1. Background and motivation

Energy has an extremely important role in shaping human life. Throughout the history man has used various types of energy to improve the quality of life: power of their own muscle was used for soil treatment; the strength of domestic animals was used for transporting people and goods; a fire for heating, food preparation, lighting and metalwork; wind to power windmills; water to power watermills; fire and water for steam engine; and ultimately electrical energy. Today, life without electrical energy is almost unimaginable, what makes it a highly significant form of energy.

According to Agenda 21 [2], the majority of world production and consumption occurs in ways that are not sustainable if overall consumption continues to grow and if application of the same technology is going be continued. Consequence of electric power production is the negative impact on the environment: air pollution with combustion emissions, radiation burden on environment, solid and liquid waste disposal, etc.

In order to support the concept of sustainable development[1], in terms of production it is necessary to improve existing technologies for electricity production based on the exploitation of coal, oil, gas and nuclear fuel and to implement technologies that rely on renewable energy sources in existing energy sector. The share of renewable energy sources in total consumption within the EU-27[2] in 2010 was 12.4% [3]. The European Union with document Europe 2020 [4] proposes the strategic goals that are focused to universal progress and to achieve the following energy targets by 2020: reduce CO2 emissions by 20% compared to the level of the year 1990, increase the share of renewable energy to 20%, and increase energy efficiency by 20%.

The introduction of renewable energy sources is a challenge to the existing traditional grid. The mode of transmission of electricity to end users as we know it, was not changed significantly since its first designs based on the idea of scientist Nikola Tesla in

---

[1] According to [30], s**ustainable development** is a development that meets the needs of the present without compromising the ability of future generations to meet their own needs.

[2] **EU-27** – The European Union (EU) is an economic and political union of 27 member states.

the year 1888 [5]. Traditional power system (Figure 1) is designed as a centralized one-way transmission of electricity from producers to end consumers. Transmission of electricity is organized through high and medium-voltage networks to the low-voltage networks that enable energy delivery to end consumers.

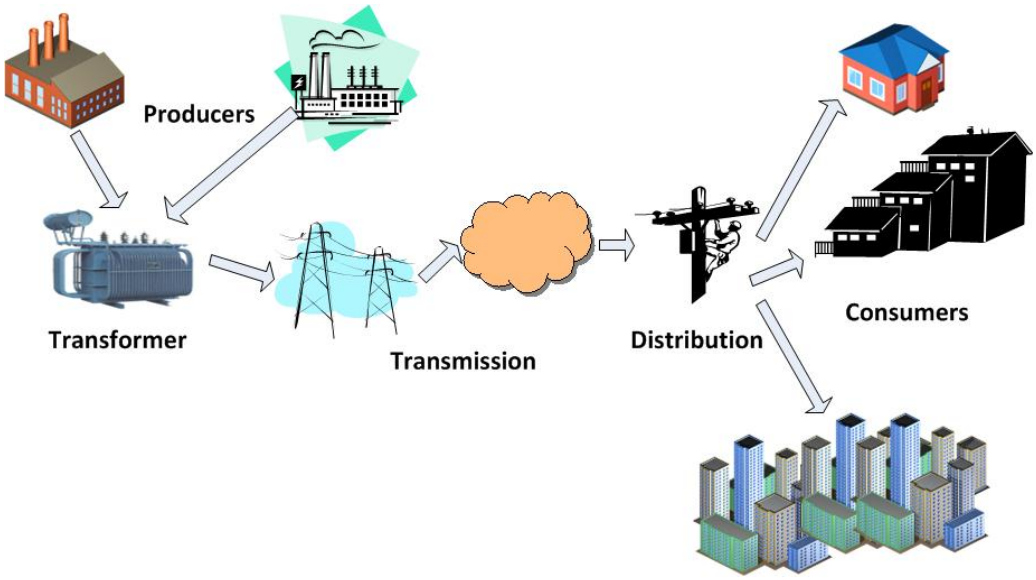

Figure 1: A traditional power system

Following problems can be identified from Figure 1:

- loss of energy due to big distances between producers and consumer, and
- centralized network topology.

A small number of control entities are responsible for managing the production capacity of several major power plants. Depending on the prediction of future demand, based on historical data on previously consumption, control entities schedule the available plants production.

The only form of renewable energy that is widely incorporated within a traditional power system is hydropower, used by hydroelectric power plants to produce electricity. In 2010, hydropower had the share in global electricity production of 16.1%, while other renewables, such as wind, geothermal, solar and biomass, accounted for only 3.3% [6]. The list of primary renewable energy sources as well as their various applications is depicted in Figure 2.

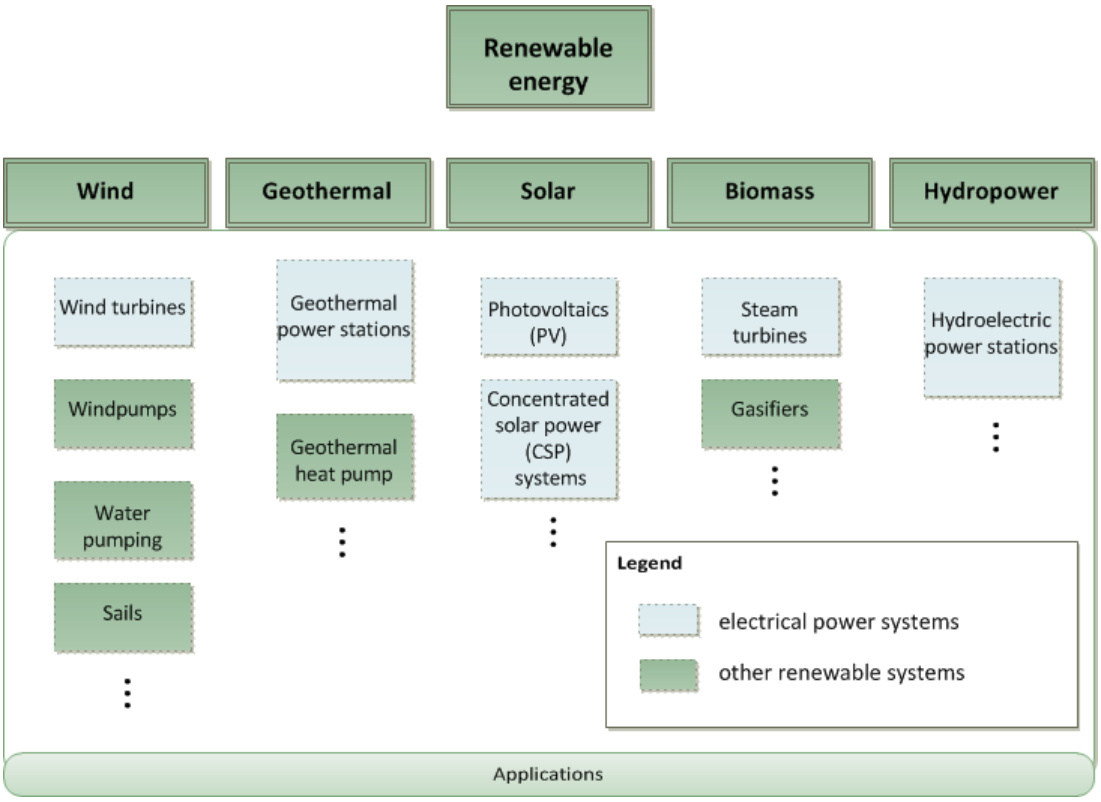

Figure 2: Various renewable energy sources

Some of the main characteristics of renewable energy sources are:

- their intermittent (due to variability of renewable energy sources) production, and
- their distribution in the network topology.

Wind power plants use power of wind what makes their contribution dependent on meteorological conditions. Solar cells are also dependent on the weather since it cannot produce electrical energy if there is not enough solar energy. Installation of numerous generators based on renewable energy sources like solar power and wind power, enriches network and contributes to its distribution. Renewable energy sources enable current customers to install their own generators, and consequently their participation in the electricity market in the additional role of producer. These characteristics also pose a problem for traditional electric power system, because there does not yet exist enough sophisticated management system that would adequately take into account the uncertainty in the production of a potentially large number of smaller plants. Another problem with the

existing network infrastructure is inefficient operation during periods of peak demand[3] leading to poor quality services, particularly temporary failure of electricity networks.

In order to enable network infrastructure to handle many changes in the electrical system, it is necessary to introduce certain changes. One of proposed solution is to transform the existing power networks in advanced energy networks called *smart grids*.

According to the definition from the European Strategic deployment document [7], a smart grid is defined as "an electricity network that can intelligently integrate the actions of all users connected to it – generators, consumers and those that do both – in order to efficiently deliver sustainable, economic and secure electricity supplies".

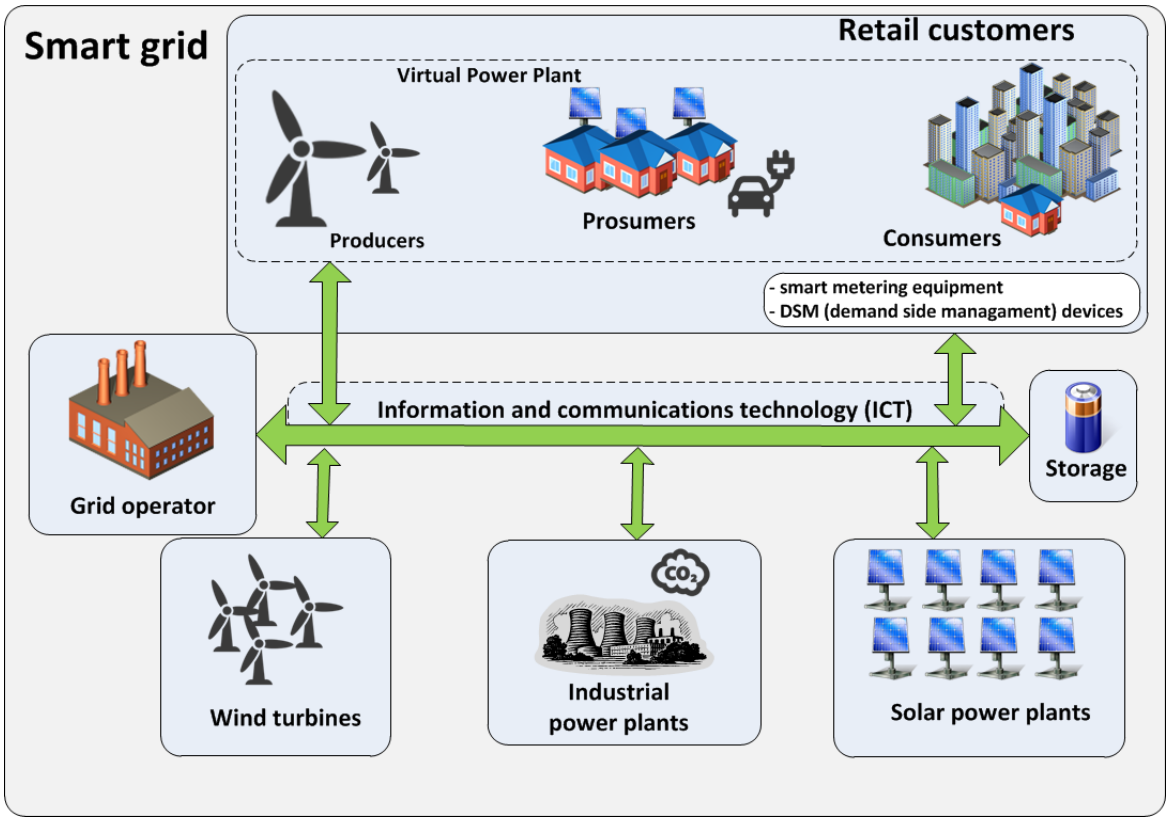

Figure 3: A smart grid deployment

[3] **Peak demand** – Power consumption is not constant, but it depends on the characteristics of consumers. Peak demand occurs at a time when energy demand is very high. One of examples when it occurs is during summer months, when sudden increase in power demand is recorded due to switching on many cooling devices.

Example of an implementation of advanced energy networks at the local level is shown in Figure 3. In the given scenario, active entities of network topologies are:

- *grid operator*,
- *retail customers*,
- *storage*,
- *industrial power plants*, and
- various types of *renewables*.

Grid operator has the property of the regulated entity that oversees the overall operation of energy networks: i) managing the distribution of power; and ii) real-time balancing of supply and demand of energy.

Retail customers may be:

- *consumers* (such as office buildings or households),
- *producers* (such as privately-owned wind turbines or production-based households equipped with solar panels), or
- *prosumers*[4] (such as households or electrical vehicles).

In order to allow constant monitoring and managing their own load, retail customers are using devices such as smart metering equipment and demand side management (DSM) devices. Thanks to new technologies, that can increase their flexibility in spending the total power load forecasting, based on synthetic load profiles, is difficult. For the grid operator, management of retail customers is highly complex job. To allow for efficient control, retail customers are aggregated in the *virtual power plant*, thanks to software and smart-grid infrastructure. An example of a planned virtual power plant project is the four-year, €21 million EcoGrid project for the Danish island of Bornholm [8].

Storage serves to store excess energy produced, and enables the management of minor deviations in the system equilibrium. Power plants using fossil fuels and nuclear

[4] **Prosumer** is a portmanteau formed by contracting the word producer with the word consumer.

power plants are examples of industrial power plants. Each of them brings different pollutants: combustion emissions of fossil and nuclear waste at nuclear power plants. Wind turbines and solar power plants, which all contribute to the intermittent production, were added in Figure 3 to emphasize the distribution of renewable power plants. They must be taken into account when balancing the whole network.

The introduction of information and communications technology (ICT) enables integration of the smart grid components and two-way communication between the entities. It is believed, the "Internet of energy" [9], will be developed due to use of ICT in energy distribution systems. This will serve as the basis for the development of advanced grid management, i.e. dealing with *energy layer* that includes production, transmission, distribution and consumption of energy. A smart grid extends a traditional power grid with various functionalities that are above the energy layer. Noticeable client-side functionalities are smart metering and demand-side management, while the grid operator can benefit from grid balancing and real-time monitoring of the grid. Multi-layered smart grid architecture along with its functionalities and correspondent flows is depicted in Figure 4.

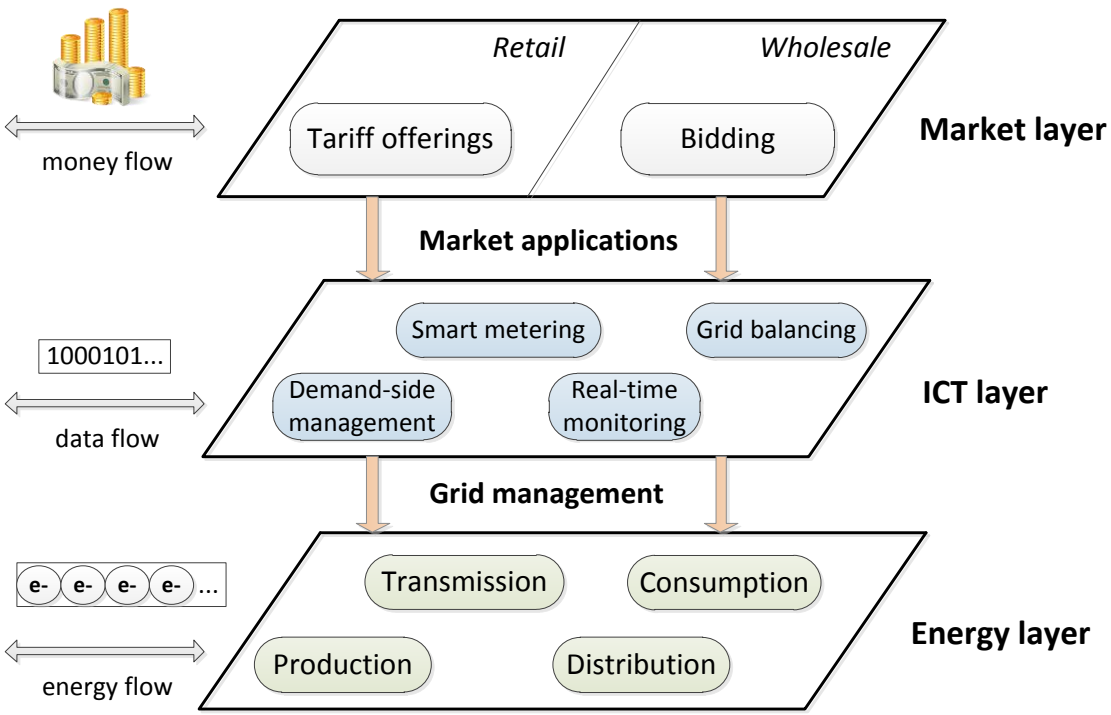

Figure 4: A multi-layered smart grid functional architecture

ICT layer provides the necessary infrastructure for wholesale and retail market applications and thus acts as a middleware between *energy* and *market layers* in smart grid architecture (Figure 4). The smart grid energy layer deals with the same activities as the traditional power grid although its implementation is far more complex. Production no longer ties to a couple of larger power plants; instead, it is consisted of numerous distributed energy sources. Limitations in transmission and distribution line capacity are now more critical, thanks to uncertainty in power production and consumption. Market layer consists of the *retail* and *wholesale* market. Retail customers use the extensive set of information provided by their ICT equipment to review and choose the appropriate tariff from the retail market offered by energy companies. The wholesale market represents a deregulated market that is used by competitive energy companies that want to obtain necessary capacity for their customers.

1.1. Smart grids

In the year 2007, United States prepared Energy Independence and Security Act (EISA) [10] with the aim of modernizing the transmission and distribution of electricity to the national level in order to achieve an advanced power grid. This act granted American National Institute of Standards and Technology (NIST) coordinating the development of accountability frameworks that includes protocols and model standards for information management to achieve interoperability[5] of smart grid devices and systems [11]. In February 2012, NIST publishes version 2.0 of document "NIST Framework and Roadmap for Smart Grid Interoperability Standards" [11], which specifies the progress made in developing smart grid backbone. NIST grouped its efforts into six key functional groups and put special attention on cyber-security and network communications mechanisms. Particular emphasis on the issue of demand response[6] and energy efficiency[7] from the user

[5] **Interoperability** is the capability of two or more networks, systems, devices, applications, or components to interwork, and to exchange and readily use information—securely, effectively, and with little or no inconvenience to the user.

[6] **Demand response** is a mechanism to manage customer consumption of electricity in response to supply conditions, for example, having electricity customers reduce their consumption at critical times or in response to market prices.

perspective was placed. The demand response is used to optimize the balance between supply and demand of electricity. Thanks to increased information about energy consumption offered by new technology, consumers devoted towards energy efficiency are able to save energy, and potentially earn more if they manage to learn where it pays to invest further [11, 12].

European Electricity Grid Initiative (EEGI) is an example of European industrial initiatives within Strategic Energy Technology Plan (SET-Plan). EEGI is a result of the industrial partners ENTSO-E[8], EDSO[9] and SmartGrids ETP[10]. The initiative proposes a nine-year program of research, development and demonstration (RD&D) that will encourage innovation and development of energy networks of the future in Europe. Implementation plan for the period from 2010 to 2012, and the guide for the period from 2010 to 2018 [13] describes the future solution as a user and market-oriented, interactive, reliable, flexible and sustainable electric power system. With a market point of view, the objectives of the initiative group EEGI include:

- active user participation in the electricity market,
- integration of national networks in the market-oriented European network, and
- opening of business opportunities and markets for new participants in the advanced energy networks [13].

---

[7] **Energy efficiency** is the goal of efforts to reduce the amount of energy required to provide products and services.

[8] **ENTSO-E (*European Network of Transmission System Operators for Electricity*)** – represents all electric TSOs in the EU and others connected to their networks, for all regions, and for all their technical and market issues (https://www.entsoe.eu/).

[9] **EDSO** (***European Distribution System Operators*** *for* ***Smart Grids*****)** is an international non-profit association committed to the development of Smart Grids in Europe. http://www.edsoforsmartgrids.eu/

[10] **SmartGrids ETP (*European Technology Platform*)** – is the key European forum for the crystallization of policy and technology research and development pathways for the smart grids sector, as well as the linking glue between EU-level related initiatives (http://www.smartgrids.eu/web/node/81).

1.2. Economic aspect of a smart grid

Previous chapters suggest there is a plan for implementing a technical infrastructure of a smart grid. Users of such networks will have at their disposal advanced equipment for recording energy consumption in real time, which are used to gain better control over their own consumption. Dynamic pricing of electricity, which reflects the state of energy balance on the market, motivates change in consumers' energy spending pattern to reduce their own costs and improves delivery capacity of the producer [14]. This behaviour contributes to the additional complexity of the production and distribution of energy: they must operate in real time to maintain the load balance of power grids.

Thanks to new technological solutions, a user becomes an essential element in balancing supply and demand of energy within the local area. In addition to technical aspects of smart grid (i.e. energy and ICT layer in Figure 4), establishment of the retail electric energy market for supporting a market aspect of smart grid systems is crucial.

Another problem relates to the centralized and regulated electricity markets, which currently undergo a restructuring. California energy crisis in 2000 [15] is one example of unsuccessful deregulation of the electricity market. Some of the causes of the crisis in the California are market manipulation of California market stakeholders and the partial deregulation of energy market. The disintegration of the market in California has shown that the success of competitive electricity markets depends on the design of the market, demand response, reserve capacities, financial risk management, and reliable management of electricity supply chain [13].

1.3. Modelling smart grid markets

In order to identify and limit the problems of the electricity market, prior to its establishment, it is necessary to offer a simulation environment to test ideas about the design of electricity markets. Several modelling methods [16] can be used for modelling electricity markets:

- *equilibrium models*,
- *game theory*, and
- *human-subject research*.

However, all mentioned methods have some shortcomings.

First, *equilibrium models*, do not take into account the strategic behaviour of market participants, or assume that parties have all relevant information about the characteristics and behaviour of other participants. In addition, equilibrium models neglect the consequences of the knowledge that a participant could get through the daily operation on the electricity market.

Second, *game theory* is largely limited to the specific situation in the market that depends on a few factors, and thus achieves stringent, sometimes unreal assumption of behaviour of participants.

Third, employing *human-subject research* can be rather difficult to research related to the electricity market since it takes great expertise to describe the behaviour of electricity generators to market in a realistic manner.

A possible solution which addresses all listed issues of other methods is market modelling based on software agents[11]. The following section provides a brief overview of several electricity market models based on software agents.

1.3.1. Agent-based market modelling

Considering the complexity of the electricity sector, as well as developed competitive electricity market economies, there is a need to explore methods of modelling market. Modelling based on agents is a methodology that offers solutions to some of the problems that are typical for the traditional models. There are number [16] of systems and tools based on agents that assist in analysis and design of electricity markets.

First, the Electricity Market Complex Adaptive Systems Model (EMCAS) [17], is agent simulation of electricity markets, which describes the behaviour of producers and consumers in the electricity market, simulates the activities of the electricity system and calculates the electricity price for each hour and location within the transmission network. Electricity prices depend on demand, production costs, congestion of the transmission system, and external factors such as delays in production or disturbances in the system and

[11] A **software agent** is a software program that autonomously acts for a user or other program in a relationship of agency.

strategies applied by the power company. The EMCAS model results contain information about the economic consequences for individual companies and consumer groups in different scenarios.

Second, the Multi Agent Intelligent Simulator (MAIS) [15] is a system based on agents for analysis and understanding of the dynamic changes in prices in the U.S. wholesale electricity market in the period before and during the energy crisis in California [16]. The proposed MAIS artificially generates commercial agents with different learning abilities and give them the bidding strategy, in order to simulate the California electricity market, applied during the crisis period in California's energy in 2001.

Third, software model of Agent-based Computational Economics (ACE), allows arithmetic study of economic processes that are modelled in dynamical systems with agents. In the ACE model, an agent is presented as an entity that consists of a set of data and computational behaviour in the arithmetically created world. Possible areas of research within the ACE [17] model include:

- understanding and evaluating of the market design,
- assessing interactions between automated markets and trading agents,
- development of rich environment for economic decision-making in research with human subjects as a subject of research, and
- proposing business policy based on expected market behaviour.

1.3.2. Agent-based simulation platforms

Although existing simulation models, tools and systems represent a useful tool to view many details about the activities of the market, they are usually limited in opportunities to participate in the consideration of different strategies of individual market participants [16]. Simulation platform that emphasizes competition of market participants can help to identify potential hazards and design flaws of the electricity market. The competitive approach proved successful in the study of numerous innovations [18, 19]. Trading agent competitions uses a multi-year competitive format for studying assembling and trading of travel packages (TAC Classic) [20], Supply Chain Management (TAC SCM) [21], trading with keywords in sponsored advertising (TAC AA) [22] and designing of market rules (TAC Market Design) [23]. A solution that combines market agents and

the element of competition in the electricity market simulation platform is Power Trading Agent Competition[12] (Power TAC) [24].

Power TAC is an example of agent-based competition simulation platform designed to allow exploration of the retail electricity market. The main goal of the Power TAC is to give a complete overview of the possibilities and limitations of open markets to identify good practice and legislation necessary for the management of energy networks of the future that will include a variety of distributed resources [16]. The concept of the Power TAC relies on the role of broker that serves as an aggregator of supply and demand for energy, which embodies software agent market at the level of program implementation. Detailed overview of simulation, including activities of brokers and other entities of competition, is given in the following chapters.

[12] Power Trading Agent Competition official site: www.powertac.org.

2. Power Trading Agent Competition platform

Power TAC is a competitive simulation platform that models a free[13], retail-level electrical energy market. Competitors in such simulation are business entities or "brokers" that can fulfil the real-world role of energy retailers in smart grid environment. Their task is to provide energy services to customers through tariff offerings, and then manage their customer portfolio loads by trading in a wholesale market.

2.1. Project stakeholders

The Power TAC is an international project created in the cooperation of six universities across Europe and North America:

- Aristotle University of Thessaloniki (Greece),
- Carnegie Mellon University (Pennsylvania, United States),
- Delft University of Technology (Netherlands),
- University of Minnesota (Minnesota, United States),
- Rotterdam School of Management, Erasmus University (Netherlands),
- University of Zagreb (Croatia).

Responsibility of the University of Zagreb's team is to develop a visualization module (called Power TAC Visualizer) for the Power TAC platform. The Power TAC Visualizer is a crucial component for the success of the whole project because it has a twofold role:

- enables real-time observing of Power TAC competitions, and
- provides enhanced analysis of stakeholders' behaviour in the Power TAC market.

[13] A **free market** is a market where prices are determined by supply and demand. Free markets contrast with controlled markets in which prices, supply or demand are directly controlled.

Therefore, developed Visualizer should not only be scalable, robust and based on state-of-the-art technologies, but also has a huge scientific value because it proposes a solution to seize dynamics of electric power markets through identifying, analysing and presenting stakeholders, processes and key interactions in the smart grid environment.

Figure 5: Power TAC Partners, retrieved from Power TAC official web site[14]

2.2. Competition overview

The major elements of the Power TAC scenario are shown in Figure 6. The main element, a competitive trading agent, is a self-interested *broker* that aggregates energy supply and demand with the intent of earning a profit. Brokers must build a good-quality portfolio of *retail customers* by offering carefully designed tariffs through *tariff market*. Good-quality portfolio implies having tariff subscriptions that are profitable and can be real-time balanced. However, the consumption and production capacities broker has acquired through the tariff market will almost certainly cause imbalance in broker's energy

[14] http://powertac.org/node/7

supply and demand. The energy imbalance has two negative impacts. First, it assists in imbalance of the whole power grid, causing serious problems in the power grid management and lowers the quality of energy service. The second problem is less-than-attractive balancing fees broker has to pay to distribution utility for causing imbalance of the power grid. Because of that, a profit-oriented broker will tend to use strategies that will contribute to low energy imbalance. Additionally, to tackle these problems, brokers are encouraged to trade in the *wholesale market* by placing bids to acquire some extra capacity or to sell an energy excess by placing asks.

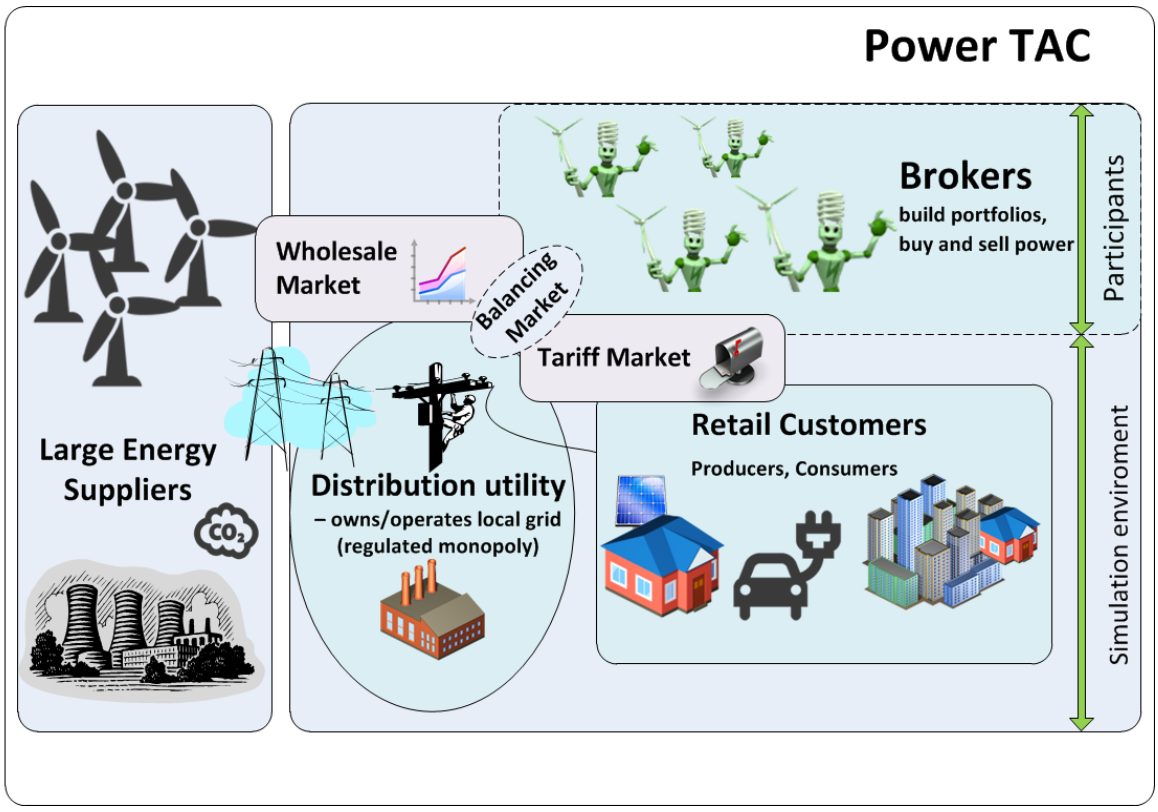

Figure 6: Major elements of the Power TAC scenario

Retail customers are essential entities connected to the grid. They can be *producers*, such as solar panels and wind turbines, *consumers*, such as offices, factories and private households or *prosumers*, such as electric vehicles and combined heat and power systems. Customers indirectly interact with brokers through tariff market or directly using individual contract negotiation process (latter option will be implemented for Power TAC 2013 competition). They have to choose the appropriate tariff by evaluating all available

tariffs on the tariff market. Analysing meter readings provided by smart metering equipment, contract negotiating and providing the balancing capacities are some of tasks customers also do within this simulation. If specified in tariff contract from a broker, customers can allow balancing capabilities that are used by *distribution utility* in *balancing market* during the grid balancing process. The Distribution Utility (DU) represents the regulated electric utility and has the similar role as the grid operator from smart grid deployment (Figure 3). Except for grid balancing, the DU is also in charge of making distribution fees for brokers and providing default subscriptions for customers that are offered before competing brokers. In order to ensure liquidity to the wholesale market, *large energy suppliers* sell large-scale quantities of energy in the role of wholesale market participants. In the actual simulation, they are called Generation Companies (or *Gencos*). Finally, there is also a *wholesale buyer* (not shown in Figure 6) that simulates a population of buyers and speculators.

The time domain in Power TAC is managed by discrete time values called *timeslots*. Each timeslot represents 1 hour of competition time and is compressed to 5 seconds of real time. With the default game duration of 1440 timeslots (or 60 days), a typical simulation will run just over two hours of real time. The actual game duration will be randomized number of timeslots that is close to the game duration setting. This randomized game duration feature discourage broker developers in developing destructive and unrealistic strategies that would exploit the remaining market time.

Competition setting will also specify the number of competing brokers, i.e. groups of two, four or eight brokers, which can be varied for each simulation. Different group sizes serve to examine broker behaviours in different market positions, such as oligopolies[15], or high-competition markets.

2.2.1. Brokers

Broker, represented by a trading agent, is the main actor of the simulation. Figure 7 provides an overview of tasks brokers need to cope with in each timeslot. Typically, this is a three step process: *trading* in the wholesale market, *portfolio development* and *balancing*

[15] An **oligopoly** is a market form in which a market or industry is dominated by a small number of sellers (oligopolists).

of energy supply and demand. An important fact is that the specific order of some activities is not as rigid as it is shown in Figure 7. Description of activities from the three step process can be found in the following chapters.

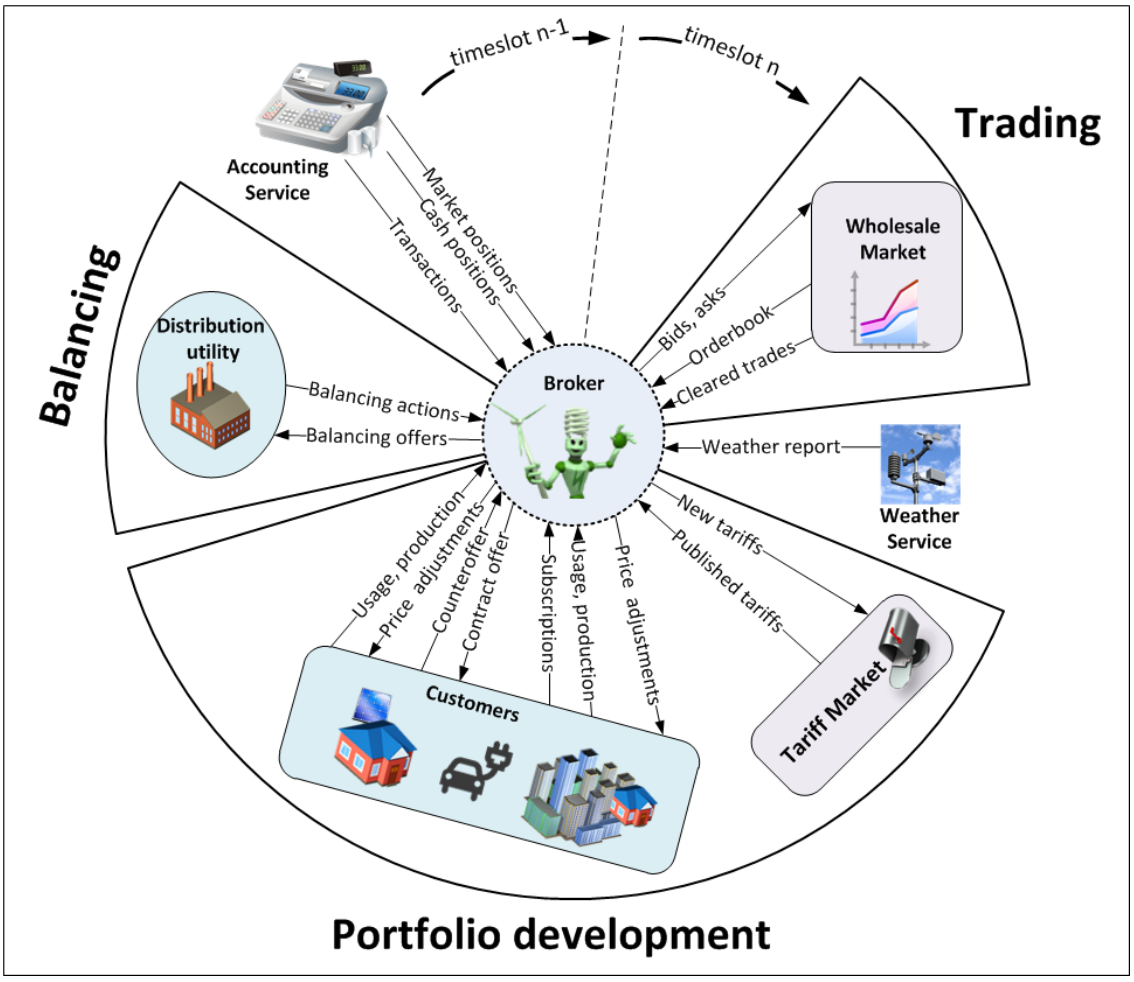

Figure 7: Power TAC broker's per timeslot activities

Portfolio management

To manage their portfolios in each timeslot, brokers may proceed with two parallel activities (Figure 8): *contract negotiation* and *tariff offering*. With contract negotiation, broker can try to make individual contracts with customers that do not want to use tariff-like contracts. The real life examples of such large customers are factories and large-sized enterprises. In parallel with individual contract negotiation, broker must develop a good-quality set of tariff subscriptions with customers who will sell or purchase energy through the use of tariff offerings. Brokers can design and offer new tariff specifications and update or withdraw an existing tariff by interacting with customers through tariff market. Power TAC platform provides customers with various consumption and production behaviours.

Each customer type can have a distinct production and consumption logic, which is briefly addressed in chapter 2.2.2.

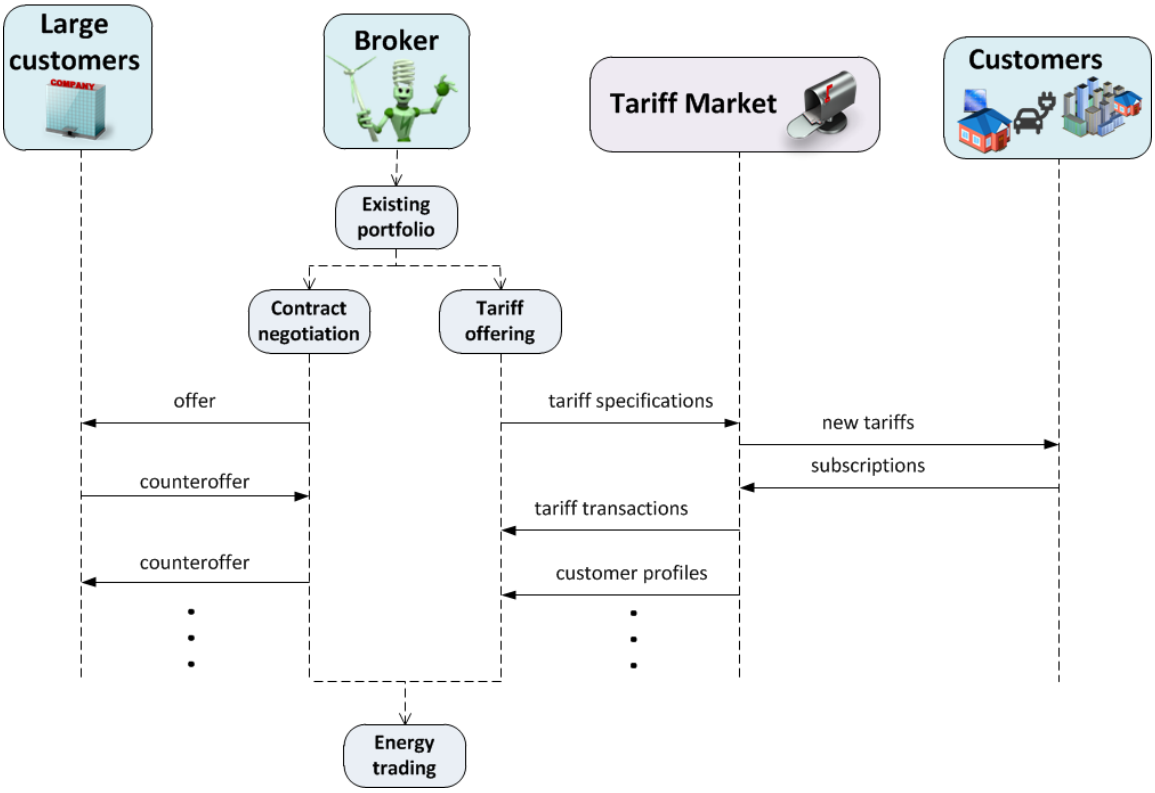

Figure 8: Portfolio management process – contract negotiation and tariff offering

Brokers can target a specific group of customers thanks to reasonably detailed tariff specification features, including:

- time related features, such as expiration date, minimum contract duration,
- rate specification, which enables the definition of time-of-use, weekday/weekend rates or tiered rates,
- two-part tariffs (fixed daily fee plus usage fee),
- balancing capacities offers, using variable rates with minimum and maximum values, estimated mean values, and notice intervals,
- signup fees or bonuses, and
- early withdrawal penalties.

Before performing the actual interaction with the tariff market, a successful broker will need to estimate and reason about future consumer and producer behaviour. This will

allow it to maintain a portfolio that is well-balanced and that assures an acceptable low risk of imbalance with acquired balancing capacities.

Energy supply and demand balancing

From broker's point of view, an energy imbalance is rather undesirable state that is caused by numerous factors, including:

- imbalanced portfolio,
- variable power generation based on weather conditions, and
- other brokers' strategies.

At the end of each timeslot, the distribution utility will charge broker with less-than-attractive balancing fee that is proportional to broker's contribution in power grid imbalance. To minimize the DU final balancing charge calculated on the balancing market, a reasonable broker can use dynamic adjustment of prices for consumers and producers who are on variable-price tariffs or offer balancing capacities to the DU that were acquired through the period of portfolio management process. Dynamic prices are typically communicated to the customers some number of timeslots before the timeslot to which they apply. Because of that, brokers must have forecasting ability to determine the optimal prices to set for the target timeslot. In order to help setting up the optimal prices for future timeslots, broker's forecasting module may try to predict demand and supply of customers. Broker can even try to forecast the prices in the wholesale market as well as the DU's balancing market price.

Balancing capacities that can be offered to the DU during the period balancing process include:

- retail customers with interruptible consumption feature (such as households equipped with DSM devices),
- extra energy storage or extra on-demand energy production (such as bio-gas units).

Trading in the wholesale market

Other way to reduce energy imbalance is by trading in the wholesale market. The Power TAC platform implements the wholesale market in a periodic double auction

format. Brokers can submit bids and asks to the wholesale market for delivery between one and 24 hour in the future.

The wholesale market will clear for each of the enabled timeslots when the simulation clock is advanced to a new timeslot. The clearing process will construct supply and demand curves from received bids and asks to determine the clearing price of each timeslot markets. The clearing price is the price that maximizes turnover and is set at the intersection of demand and supply curves.

2.2.2. Retail customers

Retail customers are models of numerous customer types, including electric vehicles, combined heat and power systems, solar panels, wind turbines, offices, factories and private households. Customer models ultimately represent the entities connected to the grid. They carry out tasks of choosing the appropriate tariff from the tariff market, contract negotiating for larger individual customers, recording meter readings for data analysis and optionally providing balancing capacities. Customer models are also designed to address responsiveness to price changes of customers in the real life applications. Each customer model is defined with one of the following power types:

- *consumption* – power flow from grid to customer,
- *interruptible consumption* – power flow from grid to customer that can be interrupted by the DU within certain bounds, typically characterized by heat-storage capacity,
- *production* – power flow from customer to grid; this power type is further split into subtypes that allow differentiation of power sources,
- *storage* – power flow to and from the grid; continuous operation in one direction is limited by storage capacity.

Tariff selection

Customers indirectly interact with brokers through tariff market or directly using individual contract negotiation process (latter option is not available yet). They are able to choose an appropriate tariff through periodic evaluation of the brokers' tariffs available on

the tariff market. In order to evaluate listed tariffs, customer models need to derive the utility of each tariff u_i:

$$u_i = -(c_v + c_f)\alpha_{cost} - r_i\alpha_{risk} - I_i\alpha_{inertia} \tag{1}$$

Parameters α_{cost}, α_{risk} and α_{risk}, are customer-specific weighting parameters for tariff evaluation that deal with cost, risk and customer's inertia factor respectively. Variable tariff costs c_v are calculated using consumption payments: consumption payments are determined by sampling k random days, deriving each day's optimal consumption under the tariff to be evaluated and finally averaging the realized cost. For variable tariffs, this calculation is performed using the average realized values. Fixed tariff payments c_f consist of sign-up fees/bonuses of the new tariff $c_{sign-up}$, daily periodic payments c_{daily} as well as exit fees of current tariff c_{exit}. These costs are to be normalized to a one-day time span with the expected tariff life $\tilde{t}$ for the other payments. The normalized values of the fixed payments are summed to obtain the fixed other payments value:

$$c_f = c_{daily} + \frac{c_{sign-up} + c_{daily}}{\tilde{t}} \tag{2}$$

Tariff risk r_i is the risk of unfavorable rate developments under dynamic contract. It is evaluated using the variance of the realized prices. Finally, customer's behavioural cost of changing a tariff is defined as customer inertia I_i. This parameter has value of one for all tariffs except the current one; in that case, the value is zero.

The utility is an essential part of the customer tariff selection. However, an overall decision might not pick the highest utility tariff. Instead, a *logit choice model* is used to either mimic not perfectly rational tariff choice in case of single customer models or to assign population shares to different tariffs in case of a population customer model. Instead of providing a discrete tariff decision, a choice probability P_i is obtained for each tariff i from the set of tariffs considered $\mathbb{T}$:

$$\mathbb{P}_i = \frac{e^{\lambda u_i}}{\sum_{t\epsilon\mathbb{T}} e^{\lambda u_t}} \tag{3}$$

The parameter $\lambda \geq 0$ is a measure of how rationally a customer chooses tariffs: $\lambda = 0$ represents random, irrational choice, while $\lambda = 1$ represents perfectly rational customers always choosing the tariff with the highest utility.

The effect of tariff choice on realized load patterns

Customers are an integral part of the simulation. Their effect on the simulation is quantified by per-timeslot meter readings for both consumption and production. The actual outcome of timeslot metering may depend on the following group factors:

- *static factors* that include model primitives such as the number of household members, work shift hours and equipment. They create the base load profile that is independent of developments in the game,
- *broker-dependent factors* that are caused by broker actions, including the tariff (load shifting caused by time-of-use pricing) and balancing capability actions,
- *game-dependent factors* include the game environment factors such as randomization and weather conditions.

The effect of tariff choice on realized load patterns is the essential part of the Power TAC research setting. In order to provide different types of customer's load influence, multiple consumption and production logic were implemented, including *fully static*, *static amount with flexible timing*, *flexible amount with static timing* and *fully dynamic* logic.

First, *fully static* logic is characteristic for customers whose meter-readings are independent of their selected tariff. Examples of such customers may include rich or industrial customers that do not have shift load capabilities or treat electricity cost as insignificant part of their business.

Customers who will not change their consumption amount but are willing to change the timing of their loads are described with *static amount with flexible timing* logic. They will tend to minimize their cost by scheduling the activities appropriately.

Then, customers that have flexible amount with static timing use simple demand behaviour: they will not time-shift their load capacity but may reduce consumption if the electricity cost becomes more expensive. Such consumption logic is used to reflect the impact of synthetic load profiles and controllable generation with well-defined cost functions such as micro-CHP system[16].

---

[16] **Micro-CHP (micro combined heat and power) system** is an extension of the now well established idea of cogeneration to the single/multi family home or small office building.

Finally, *fully dynamic* logic implies both flexible consumption amounts as well as flexible pricing. Bottom-up models that employ such consumption logic are taking into account prices and available income, while top-down models specify cross-price elasticity between timeslots [25].

Balancing capabilities

Some customers may have balancing capabilities and thus be involved in the real-time grid balancing process. Broker can acquire interruptible consumers by offering a tariff with interruptible consumption feature and use it to reduce its own supply and demand imbalance. Other option is to attract customers that have energy storage capacity, which is limited by discharge power and level of charge. The last retail option for grid balancing is the controllable micro generation units that can pledge an extra energy production, such as micro-CHPs and bio-gas units.

Implementation

From implementation point of view, retail customers are realized in the forms of plugins that instantiates a population of a customer type. That allows researchers to customize experiments and do the research about the specific consumer types by using only relevant customer models.

2.2.3. Distribution utility

The Distribution Utility (DU) represents the regulated electric utility and has the similar role as the grid operator from smart grid deployment (Figure 3). Within simulation, it deals with the range of different grid-operating activities:

1. It is a natural monopoly in charge of power distribution from the transmission grid to the customers. Consequently, it also makes distribution fees that brokers must pay, proportionally to the quantities of the net load their customers supply to the grid. In the real life, instead of natural monopoly, this can be a cooperative, a for-profit regulated corporation, or a government.
2. It is responsible for real-time distribution grid balancing, carried out on the balancing market. Consequently, it makes balancing fees for brokers, based on the supply/demand energy imbalance broker has at the end of each timeslot,

3. It simulates the electric utility in a non-competitive regulated tariff prior to market liberalization by offering default tariffs for energy consumption and production. In an actual simulation, distribution utility is represented as “default broker” that initially provides access to power for customers. The default tariffs also help in limiting the possible unfair profitability of brokers caused by overpriced tariffs.

3. Power Trading Agent Competition platform deployment

Power TAC is conceived as an annual competition between university research teams, following the successful examples of other competitions [18, 19] that have encouraged a number of innovations through the research work. Each year, research teams are preparing programming solutions for software agents that compete against the software agents of other research teams.

Schematic representation of a platform deployment is shown in Figure 9. From a technical point of view, the Power TAC platform is Java-based platform based on the *Spring framework*, deployed on a servlet container (such as Tomcat or Jetty servlet containers).

Power TAC competition entities were implemented in the form of independent configurable plugins. Since it performs multiple roles, distribution utility is represented with two plugins: the logic of market balancing was implemented in the *Distribution-utility* plugin, while its role as a default tariff provider was carried out in the *Default-broker* plugin. Generation companies, whose logic is based on interactions with a wholesale market, were modelled with the *Genco* plugin. Both Broker agents and generation companies are participating in a wholesale market, an entity represented by the *Auctioneer* plugin. The financial aspect of the competition and the logic of tariff market were implemented in the *Accounting* plugin. The behaviour of retail customers was implemented with the *Customer* plugin. Each customer plugin describes one customer model. This plugin feature allows users of the Power TAC platform to customize an experiment by excluding desired customer-specified models.

Simulator uses previously mentioned plugins to simulate the market aspect of a smart grid. To simulate the impact of weather conditions, the simulator fetches the weather forecast data from a remote *Weather data server*. The simulator also contains set of mechanisms for communication with brokers, which can be distributed over the Internet.

For the Power TAC platform there are several different groups of stakeholders with different needs. Competitive teams are preparing for the competition through their own

empirical research on the Power TAC platform. They should be able to easily use the platform in order to configure an experiment and they need to have a rich data representation in order to analyse behaviour of brokers and markets. Administrator is a person responsible for management of the competition. The group of external *observers* includes market regulators, industry partners and project sponsors. Such group will benefit with a high-level overview of the competition.

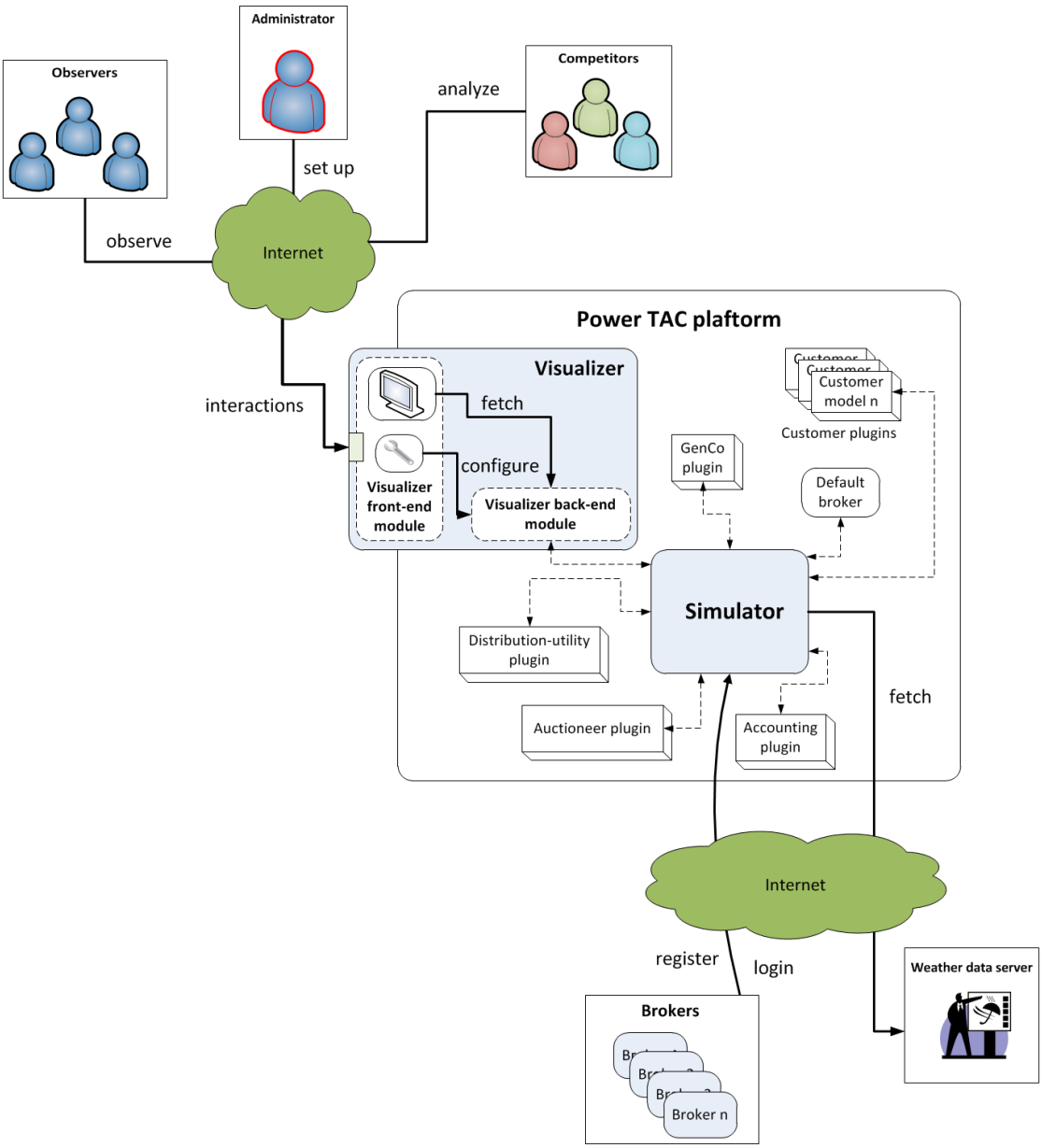

Figure 9: Power TAC platform deployment

To allow users interact with the platform, the Power TAC Visualizer is used. The Visualizer is a standard Power TAC component, implemented as a web-application.

Bearing in mind the above user requirements, a special emphasis is placed on the visualization of the competition, as well as support for analysis of the competition. A detailed review of the Visualizer is given in the next chapter.

4. Power Trading Agent Competition Visualizer

The Power TAC Visualizer is a crucial component for the success of the whole Power TAC project because it has a twofold role:

- enables real-time observing of Power TAC competitions, and
- provides enhanced analysis of stakeholders' behaviour in the Power TAC market.

Therefore, developed Visualizer should not only be scalable, robust and based on state-of-the-art technologies, but also has a huge scientific value because it proposes a solution to seize dynamics of electric power markets through identifying, analysing and presenting stakeholders, processes and key interactions in the smart grid environment.

4.1. Technological groundings

The Visualizer consists of the two following modules as shown in Figure 10:

- *back-end module*, and
- *front-end module*.

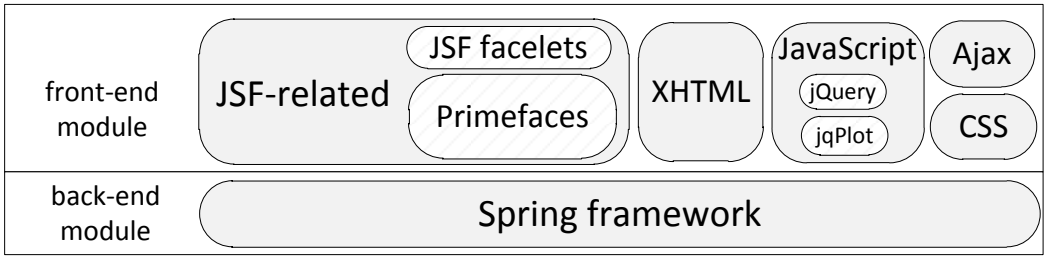

Figure 10: Technology stack for Visualizer's development

The main purpose of the back-end module is to interact with the simulator. Functionalities for the back-end module are implemented with the *Spring framework*[17], which is a Java-based technology.

The actual visualization of the Power TAC platform is achieved in the front-end module. In order to develop a rich user interface, the following two groups of technologies were used:

- *JavaServer Faces*[18] (JSF) technologies, and
- standard web technologies.

JSF technologies provide a server-side event model for dispatching events and attaching listeners to core system functionality. In particular, the Visualizer design exploits possibilities of a following set of JSF technologies:

- *JSF Facelets*[19] were used to build web-design template for the Visualizer,
- the rich set of components from the lightweight library *Primefaces*[20] is an important part of Visualizer's rich and visually attractive design.

A Visualizer's web-template is backed by a standard set of web technologies that are common in the web-design development:

- *EXtensible HyperText Markup Language* (XHTML) is one of the prerequisites for using *Facelets*. As a consequence, Visualizer has web-pages that are well-formed i.e. satisfy a list of syntax rules provided in the XHTML specification,

---

[17] The **Spring Framework** is an open source application framework and Inversion of Control container for the Java platform (http://www.springsource.org/).

[18] **JSF** is a request-driven MVC web framework for constructing user interfaces using components (http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html).

[19] **Facelets** is an open source web template system under the Apache license and the default view handler technology (aka view declaration language) for JavaServer Faces (JSF) (http://facelets.java.net/).

[20] **PrimeFaces** is an open source JSF component suite with various extensions (http://primefaces.org/).

- Other than JSF-based *Primefaces* library, for a rich visual design, two JavaScript libraries were used. Graph data representation was achieved using the *jqPlot*[21] plotting library. Competition animations were implemented with a help of the *jQuery*[22] library,
- One of requirements for Visualizer is to provide a dynamic visualization. Manual page refresh is not a user-friendly solution. Therefore, an automatic mechanism for page update is needed. Such mechanism should only update relevant parts of a page to minimize the unnecessary network traffic load. A periodic partial page update is one of solutions for these requirements. To exercise such mechanism the Visualizer takes advantage of the *Asynchronous JavaScript and XML* (AJAX) technology,
- Visualizer's visual design is additionally stylized with the usage of custom-made *Cascading Style Sheets* (CSS).

4.2. Visualizer use-case scenarios

The main purpose of the Visualizer is to provide rich and intuitive display of relevant information about the Power TAC competition. This will help research community in evaluating market rules for policy guidance in the retail-level markets. Additionally, Visualizer as a multi-purpose web-application also provides the necessary infrastructure for competition management. This enables competitors to set up a custom experiment with various different parameters.

A high-level overview of interactions between actors and the Visualizer is shown in Figure 11. The Visualizer interacts with four different types of actors:

1. *Observer* is an actor that wants to observe the simulation. Observer is a casual user that does not require the Visualizer to have an advanced data-analysis features. Instead, Observer is more interested in visually attractive design and high-level overview of a running Power TAC competition. In the real life, such

[21] **jqPlot** is a plotting and charting plugin for the jQuery Javascript framework (http://www.jqplot.com/).

[22] **jQuery** is a cross-browser JavaScript library designed to simplify the client-side scripting of HTML (http://jquery.com/).

actors are researchers at an academic conference, market regulators, Power TAC project sponsors or industry partners.

2. *Competitor* is an actor that provides a competing broker for the competition. In similar to Observer, Competitor demands visually attractive and rich design. What differs these two actors is Competitor's need for extensive data-analysis. Electricity market researcher involved in Power TAC project is a real-life example of this actor.

3. *Administrator* is an actor that is in charge of setting up the competition. Administrator wants to be able to configure competition options, to specify the list of brokers for a new competition and to run the competitive simulation. A game master is a real-life example of this actor.

4. *Simulator* is an actor that provides the competition. It is a simulation instance that communicates with the Visualizer by sending messages. The Visualizer uses these messages to create visualization of the competition.

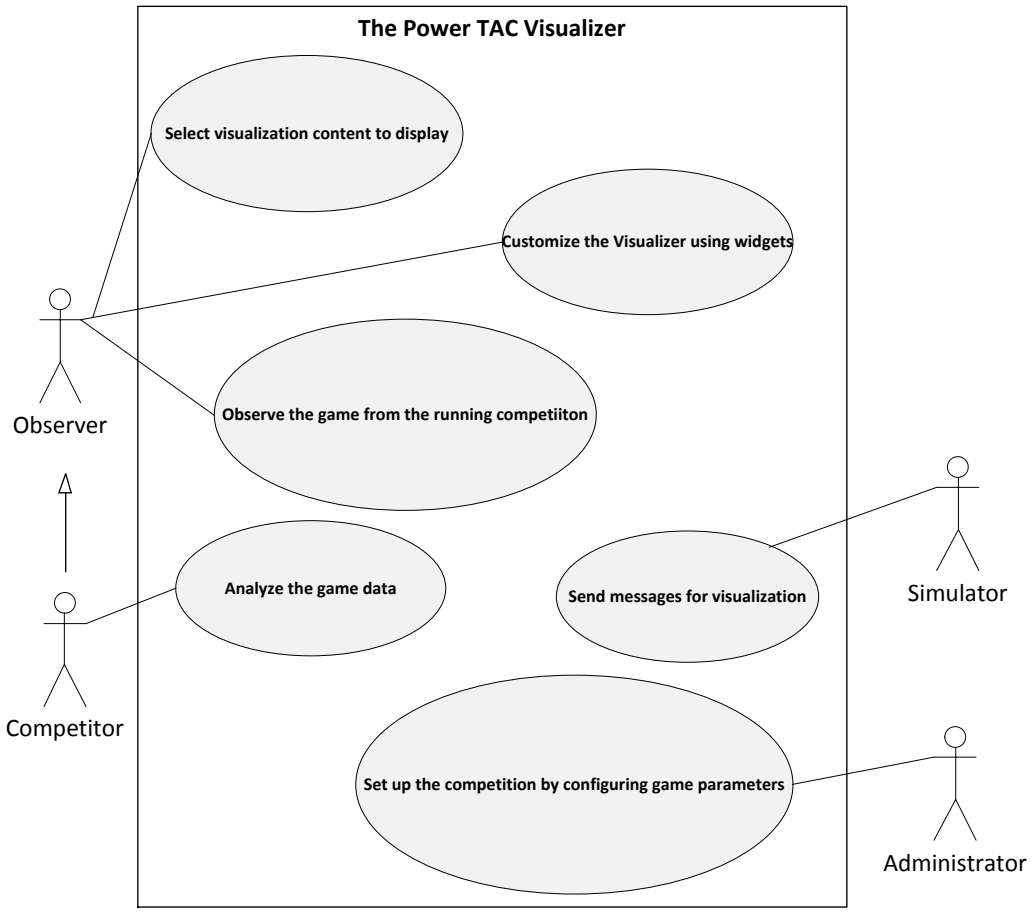

Figure 11: Use-case diagram – the Power TAC Visualizer

These four types of actors can be placed in a various use-case scenarios. The most important ones are described in the following chapters.

4.2.1. Use-case scenario (1): competition observing

The first use-case scenario describes a basic interaction between Observer actor and the Visualizer (Table 1). In order to be able access the Visualizer, Observer needs to have a web-browser and a valid Internet connection. Since Observer can use an arbitrary web-browser, a special emphasis was placed on cross-browser development of the Visualizer. Once the Visualizer is deployed, Observer can access the Visualizer through a web-link using web-browser. With an intuitive navigation system, Observer is able to pick a desired content. The Visualizer retrieves needed data from the back-end, generates the content for display and returns it to Observer as response. If Observer visits the Visualizer while there is no running simulation, an empty layout page will be rendered.

Table 1: Use-case scenario – Observer observes the competition

Title:	Observer observes the simulation
Description:	Observer accesses the Visualizer using a web-browser, selects a content to display and observes the competition.
Primary actor:	Observer
Preconditions:	The Visualizer is deployed and the competition is running. Observer has a valid Internet connection.
Post conditions:	Observer's web-browser has a valid content displayed.
Main success scenario:	1. Observer enters the Visualizer through a web-link using web-browser. 2. Using navigation dock, Observer selects a page with a desired content. 3. Visualizer retrieves data, generates and returns the content. 4. Observer's web-browser renders an expected result.
Variations:	3a. No data to display. - Generate and return an empty layout page.

4.2.2. Use-case scenario (2): content view customization

This scenario (Table 2) follows the first use-case scenario (Table 1), after which Observer is already connected to the Visualizer and is able to observe the competition. Observer now might want to get a customized content view from the Visualizer. With the help of various widgets, Observer modifies the Visualizer to get a customized content view. Notable widgets include rich buttons and tabs. Rich buttons allow Observer hiding a specific layout unit and thus freeing more space for better user-experience. Tabs are extensively used in order to provide rich and intuitive content navigation. More details about the front-end design are given in the front-end module chapter.

Table 2: Use case scenario – Observer customizes the Visualizer using widgets

Title:	Observer customizes the Visualizer using widgets
Description:	Observer customizes the Visualizer to get a customized content view.
Primary actor:	Observer
Preconditions:	The Visualizer is deployed and the simulation is running. Observer is already observing the simulation.
Post conditions:	Observer successfully observes a customized content view from the Visualizer.
Main success scenario:	1. Using widgets, Observer modifies the Visualizer to get a customized content view. 2. The Visualizer processes Observer's modification. 3. The Visualizer retrieves, generates and returns a customized content view. 4. Observer’s web-browser renders a customized content.
Variations:	1a. Using rich buttons, Observer hides particular Visualizer layout units. 1b. Using checkboxes, Observer modifies the set of graphs to be displayed. 1c. Using tabs, Observer requests for a specific content (such as individual broker and customer model display) 1d. Using clickable wholesale table, Observer requests for a specific market view.

4.2.3. Use-case scenario (3): data analysis

In this scenario (Table 3), Competitor is interested in the exhaustive information about the running competition. The Visualizer provides the information about Competitor's broker, other brokers and competition environment. The information is contained within various graphs and data tables. This allows Competitor to get valuable input for the empirical research. Among other, Competitor can use the Visualizer to evaluate broker's strategies and reason about the market design.

Table 3: Use case scenario – Competitor analyses the game data

Title:	Competitor analyses the game data
Description:	By analysing the data, Competitor analyses his broker's performance and a competition environment (such as other brokers' behaviour, wholesale market interactions and customers' behaviour).
Primary actor:	Competitor
Preconditions:	The Visualizer is deployed and the simulation is running. Competitor is already observing the simulation.
Post conditions:	Competitor gets valuable input for the empirical research.
Main success scenario:	1. Competitor customizes the Visualizer to get a customized content. 2. Competitor proceeds with the analysis.
Variations:	2a. Competitor analyses his/her broker's performance. 2b. Competitor analyses a competition environment.

4.2.4. Use-case scenario (4): simulator set up

The Visualizer also has a competition management feature. It is used by Administrator to set up the simulator and run the competition (Table 4). Prior to competition start, Administrator has to configure various simulation parameters (such as list of authorized brokers and competition configuration). With an appropriate button click, Administrator configures and starts the new competition. The Visualizer also provides exception handling, i.e. returns an error message for wrong competition parameter.

Table 4: Use case scenario – Administrator sets up the simulator

Title:	Administrator sets up the simulator
Description:	Prior to competition start, Administrator has to configure various simulation parameters.
Primary actor:	Administrator
Preconditions:	The Visualizer is deployed and the simulation is not running.
Post conditions:	Administrator successfully started the new competition.
Main success scenario:	1. Administrator navigates to competition control page. 2. Administrator enters a configuration file to be used. 3. Using pop-up dialog, Administrator specifies a list of authorized brokers for the simulation, displayed as a table. 4. Optionally, Administrator enters other parameters using appropriate forms. 5. Administrator clicks an appropriate button to start the competition. 6. Visualizer processes Observer's input and starts the competition.
Variations:	2a. Wrong game configuration filename entered. - The Visualizer returns an error message. 3a. Administrator specified a wrong list of authorized brokers. - Administrator edits a table of brokers.

4.2.5. Use-case scenario (5): message sending

This scenario describes an interaction between Simulator and the Visualizer (Table 5). Simulator is an actor that provides the competition. It is a simulation instance that communicates with the Visualizer by message sending. Upon new message receive, the Visualizer will call the appropriate message handlers and update its state. If the received message is of unknown type, the Visualizer will report a warning message to logger and resume to work.

Table 5: Use case scenario – Simulator sends a message to the Visualizer

Title:	Simulator sends a message to the Visualizer
Description:	Simulator runs the simulation and submits messages to the Visualizer. The main task for Visualizer is to update its state based on a received message.
Primary actor:	Administrator
Preconditions:	The Visualizer is deployed and registered for message listening. The simulation is running.
Post conditions:	The Visualizer successfully updated its state.
Main success scenario:	1. Simulator sends a message. 2. Visualizer receives a message. 3. Dispatches a message to appropriate message handlers. 4. Message handlers update the Visualizer's state.
Variations:	3a. Unknown message type for a received message. - The Visualizer reports a warning message to logger.

4.3. Back-end module

The back-end module of the Visualizer is a module that contains the application logic for visualization of the Power TAC platform. Its main purpose is to communicate with both the simulator and the front-end module.

In order to communicate with the simulator, the front-end module implements a necessary infrastructure needed for registration with a proxy provided by the simulator. After the Visualizer has been successfully registered, the back-end module needs to receive a message from the simulator and update Visualizer's state based on received message. To keep the complexity of communication within reasonable boundaries, the simulator does not require any input from the Visualizer except for registration process. Thus, communication with the simulator is achieved as a one-way message-based communication: the simulator is a message sender and the Visualizer is a message receiver. One of the main requirements for the Visualizer was to design for seamless integration with the Power TAC platform. This was the main guidance for choosing an appropriate technology for development of the back-end module. The back-end module is entirely developed using the Spring framework, the same technology used by all other Power TAC

projects. This technology choice allows the avoidance of using additional technologies to glue the Visualizer and the simulator.

The back-end module will update Visualizer's state for each received message. Visualizer's state is represented by the most recent values of the Visualizer model objects. Information about customer's net usage of electricity is an example of such values. In order to allow front-end to build a representation of the Visualizer's state, the back-end module acts as a content repository for the front-end module. It is also responsible for configuring the simulator, based on an input from the front-end module that is used by the Administrator.

4.3.1. Back-end module architecture

The front-end module overview of the core architecture is given in Figure 12. Structure of the core architecture is separated into the following components:

- *competition management component*,
- *middleware component*,
- *message handling component*, and
- *model component*.

Each of these contains a distinct set of functionalities. Competition management component provides access to the simulator through use of the front-end user interface. Middleware component is responsible for registering Visualizer with the simulator and receiving messages. It also dispatches received messages to message handling component.

Components responsible for Visualizer's state update are message handling component and model component. Message handling component extracts the valuable information from received message and calls methods from model component in order to update Visualizer's state. Finally, model component is the set of domain objects that describe Visualizer's state.

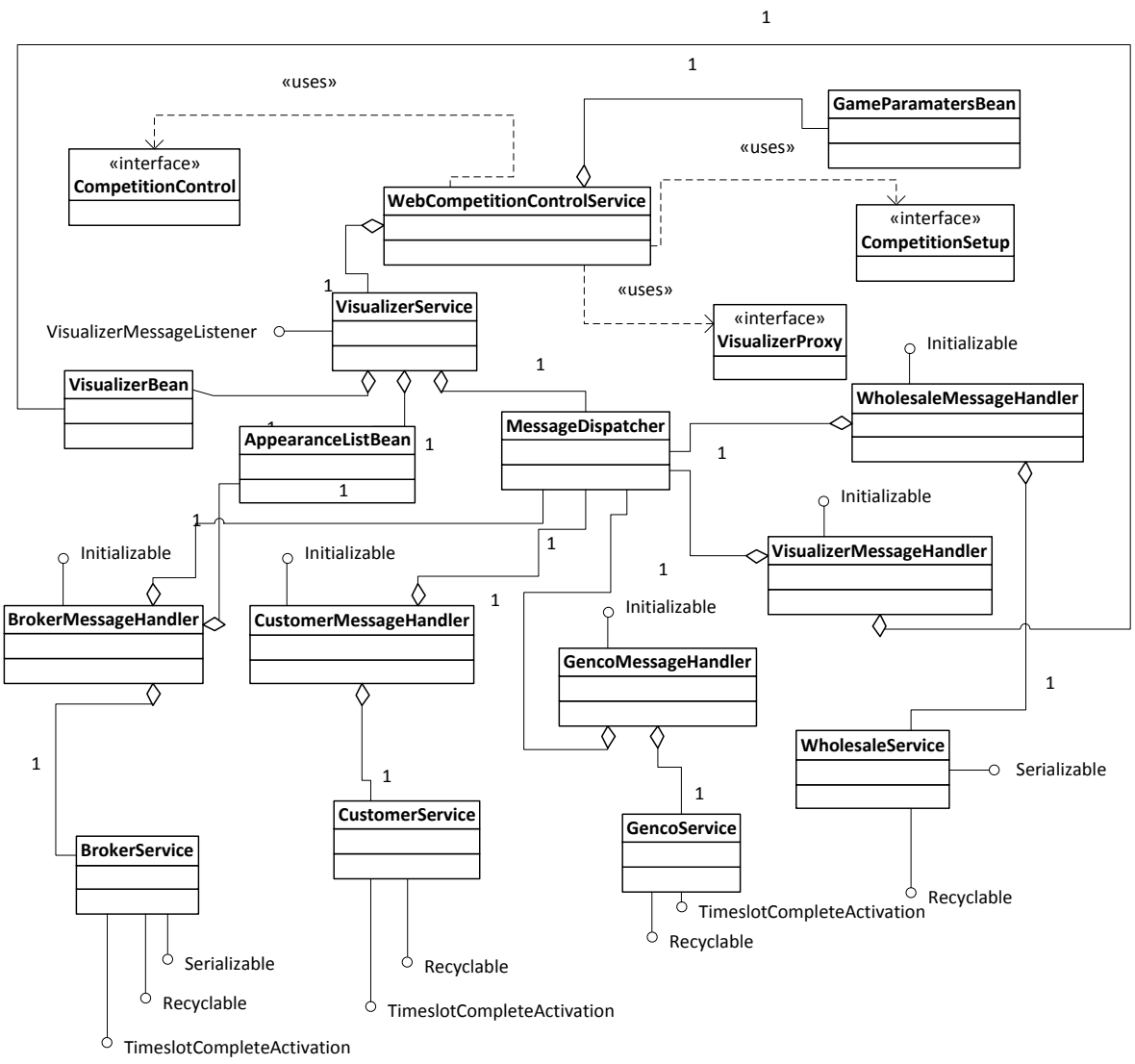

Figure 12: Class diagram – the core architecture overview

Competition management component

Competition management component provides access to the simulator through use of the front-end user interface. This component consists of the following main artefacts:

- `WebCompetitionControlService` is an object that provides methods for managing the simulator from the front-end module. Simulator management is possible through the use of `CompetitionControl` and `CompetitionSetup` interfaces. Prior to competition start, the

`WebCompetitionControlService` object will set up Middleware component with `VisualizerProxy` interface.

- `GameParametersBean` is an object used for storing competition configuration options. The list of brokers and the server configuration filename are some of possible configuration options held in `GameParametersBean` object.

Middleware component

Middleware component is responsible for registering Visualizer with the simulator and receiving messages. It also dispatches received messages to message handling component. This component consists of the following main artefacts:

- `VisualizerService` is an object that provides registration of the Visualizer by implementing `VisualizerMessageListener` interface. It is responsible for message receiving and calling `MessageDispatcher` object for message dispatching. It is used by competition management component to register the Visualizer back-end with the simulator. This will also trigger reset of the Visualizer's state.
- `MessageDispatcher` is an object responsible for routing messages from `VisualizerService` object to message handling component.

Message handling component

Message handling component extracts the valuable information from received message and calls methods from model component in order to update Visualizer's state. Each class within this component implements `Initializable` interface. By implementing this interface, objects from message handling component are able to register for an arbitrary set of message types. Registration is achieved by calling `MessageDispatcher` object's method from middleware component. This component consists of the following main artefacts:

- `BrokerMessageHandler` is an object that registers for broker-related messages. Its primary goal is to extract valuable broker information from

received message (i.e. cash position, tariff specification, tariff transaction, distribution transaction and balancing transaction) and update a broker model. At the competition start, `BrokerMessageHandler` object will assign an appearance for each broker using `AppearanceListBean` object.

- `CustomerMessageHandler` is an object that registers for customer-related messages. Its primary goal is to extract valuable information about a customer from received message (i.e. tariff transaction and customer's history usage) and update a customer's model.
- `GencoMessageHandler` is an object that registers for Genco-related messages. Its primary goal is to extract valuable information about the Genco from received message (i.e. cash position, market transaction and market position) and update a Genco model.
- `VisualizerMessageHandler` is an object that registers for competition-related messages. Its primary goal is to extract valuable competition information from received message (i.e. current timeslot, weather report and distribution report) and update a competition model.
- `WholesaleMessageHandler` is an object that registers for wholesale-related messages. Its primary goal is to extract valuable broker information from received message (i.e. order, order book and cleared trade) and update a wholesale market model.

Model component

Model component is the set of domain objects that describe Visualizer's state. These objects serve as a primary source of data for the front-end module's visualization process. Furthermore, objects from model component can be grouped in the following groups i.e. models:

- *Broker model* represents a group of objects that contain information about competing brokers. A simplified architecture overview of the broker model is given in Figure 13. `BrokerService` is an object that holds a collection of `BrokerModel` objects. The exact size of that collection depends on number of

brokers participating in the competition. The `BrokerService` object acts as a mediator between `BrokerMessageHandler` object from message handling component and `BrokerModel` object. Such object will implement two interfaces: `TimeslotCompleteActivation` and `Recyclable`. The first interface allows the `BrokerService` to be called by message handling component after each timeslot has completed, while the second one indicates that the `BrokerService` object will be recycled, i.e. it will clear the list of brokers from the previous competition. The `BrokerModel` object represents a broker. Such object implements the following three interfaces: `VisualBroker`, `DisplayableBroker` and `TimeslotModelUpdate`. While the first two interfaces describe a list of methods needed for visualization purposes, the `TimeslotModelUpdate` interface indicates that the `BrokerModel` object will be updated after the end of each timeslot. Broker's portfolio records are kept within multiple `CustomerModel` objects, one for each customer model in the competition. Broker's graph data can be found in the `BrokersJSON` object, while broker's aggregate day results are kept in the `DayState` object.

- *Competition model* contains objects for recording general information about the competition, including: current timeslot, current day, current week, weather report. It also contains an animation model for the day overview.
- *Customer model* contains objects for recording information about customers, including: cash outflow, cash inflow, produced energy, consumed energy and historical net usage data.
- *Genco model* contains objects for recording information about generation companies or Gencos. Since the Gencos only participate in the wholesale market, the Genco model will record Genco's wholesale market performance and financial performance.
- *Wholesale model* contains objects for recording information about wholesale market interactions. This include a list of available markets, list of market clearings for a particular market, list of bids and asks before the market clearing, list of bids and asks after the market clearing and information about total traded quantities.

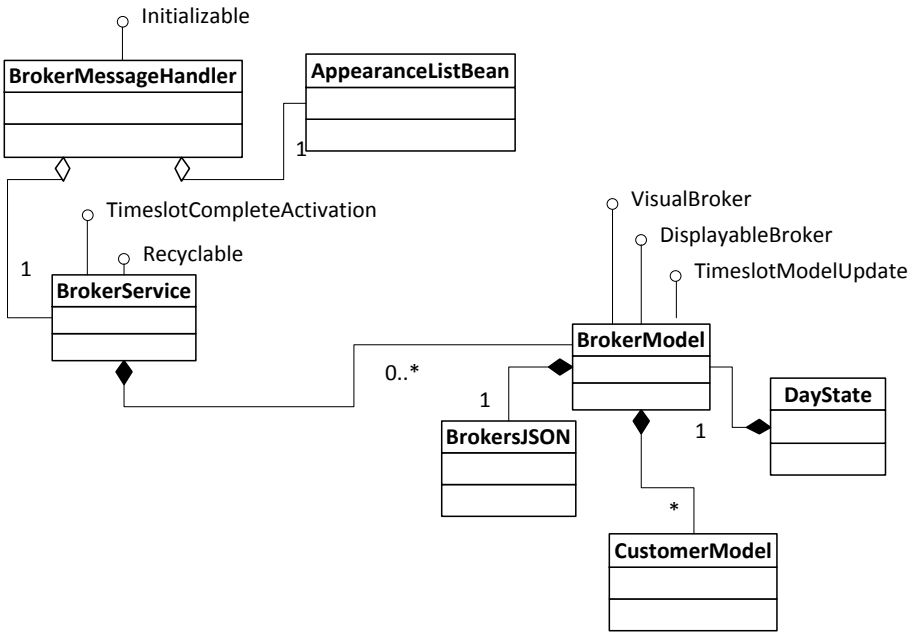

Figure 13: Class diagram – architecture overview of the broker model

4.3.2. Visualizer's state update mechanism

Visualizer's state is determined by current values from the model component. It is used by the front-end module to retrieve the needed data and generate visualization in form of a web-page. One of the main characteristics of Visualizer's state is that it directly reflects the real-time status of the competition. In order to support the real-time data representation, it is very important to develop an efficient mechanism for Visualizer's state update.

Such mechanism should be able to quickly handle a message received from the simulator and update the Visualizer's state. Since the back-end module receives a large number of messages, the per-message state update might not be a reasonable option: creating new immutable objects after each message is received can have a significant negative impact on the Visualizer's performance and can lead to a lower-quality service. An additional reason for not utilizing a per-message state update is the user perspective: the end-user is generally not interested in an instant state update. Instead, the user is more likely to require a complete competition overview with a reasonable update period. This assumption has led to the solution for the Visualizer's state update mechanism, shown in

Figure 14. A specific message type (`TimeslotComplete`) is used to trigger Visualizer's state updating. This message is typically received every five seconds from the simulator, although the interval can vary depending on the competition setup.

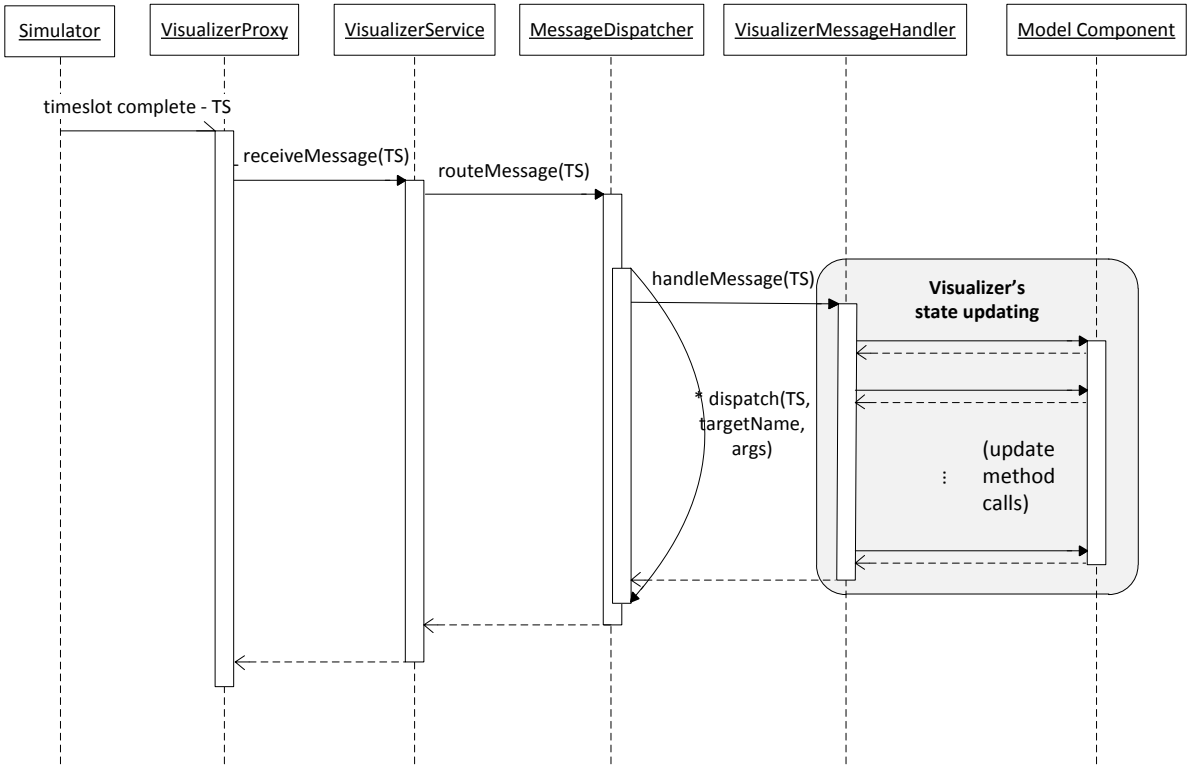

Figure 14: Sequence diagram – message handling and Visualizer's state updating

First, the simulator sends the `TimeslotComplete` message to `VizualizerProxy` object. `VizualizerProxy` object forwards the message to `VisualizerService` with a `receiveMessage(TS)` method call after which `VisualizerService` calls the `MessageDispatcher`'s `routeMessage(TS)` method. `MessageDispatcher` will now iterate over handler objects that were registered for a `TimeslotComplete` message. In an example from Figure 14, such handler object is `VisualizerMessageHandler`. The final step in this mechanism is an actual update of the Visualizer's state using update method calls for objects from the model component.

This procedure helps to maintain a sufficiently good user experience in terms of the updated display of information, while maintaining performance requirements within satisfactory boundaries.

4.4. Front-end module

The front-end module of the Visualizer is a module that contains the presentation logic for visualization of the Power TAC platform. Its main purpose is to retrieve necessary data from the back-end module and use that data to create rich and intuitive representation of the Power TAC competition. In addition to rich visualization, the front end module also provides an administration interface for competition management. The Visualizer competition control replaces the manual set up of the simulator that uses the command line interface.

The front-end module was developed using JSF-related and standard Web technologies. Software artefacts from this module are web pages. The basic skeleton of a page is built using *Facelet* template facility, provided by JSF. Implementation of the master layout provided consistent look for all the web pages. To provide rich visual design, the *Primefaces* library was used. One of the widgets from the *Primefaces* that were used for page navigation is the dock element. In order to support the analysis of the competition, various charts that describe the work of individual Power TAC entities were implemented. Visualizer's charts are provided by *jqPlot* library and they include line charts, pie charts and bubble charts. Dynamic display of content was achieved using a periodic page update mechanism.

To meet the need for a high-level overview, an animation in the *Game overview* page was implemented. The game overview page does not provide a detailed insight into the competition data; instead, it is able to demonstrate the basic interactions between entities in the competition.

Pages that show the competition content are based around the Power TAC entities. Entities that have separate pages are:

- brokers,
- customers,
- generation companies, and
- wholesale market.

In order to support the ability to administer the competition over the Visualizer, the *Competition control* page was developed.

4.4.1. Game overview

The Power TAC simulation design is composed of numerous entities that mutually interact. This makes the competition quite complex. This can be an obstacle for people who are not within the Power TAC community and can ultimately cause the low-level of popularity of the platform. To tackle these issues, the Visualizer provides a high-level game overview for casual users. Such overview does not have a detailed insight into the competition data; instead, it is able to demonstrate the basic interactions between entities in the competition. To create a high-level game overview, the front-end module needs to retrieve a reduced set of information available from the back-end module and create the appropriate visualization. The solution for the game overview is implemented in the form of animation. Designing of animation was done using XHTML and CSS, while the actual simulation movements were achieved with the help of JavaScript and jQuery library. Game overview is provided in the *Game Overview* page (Figure 15) of the Visualizer.

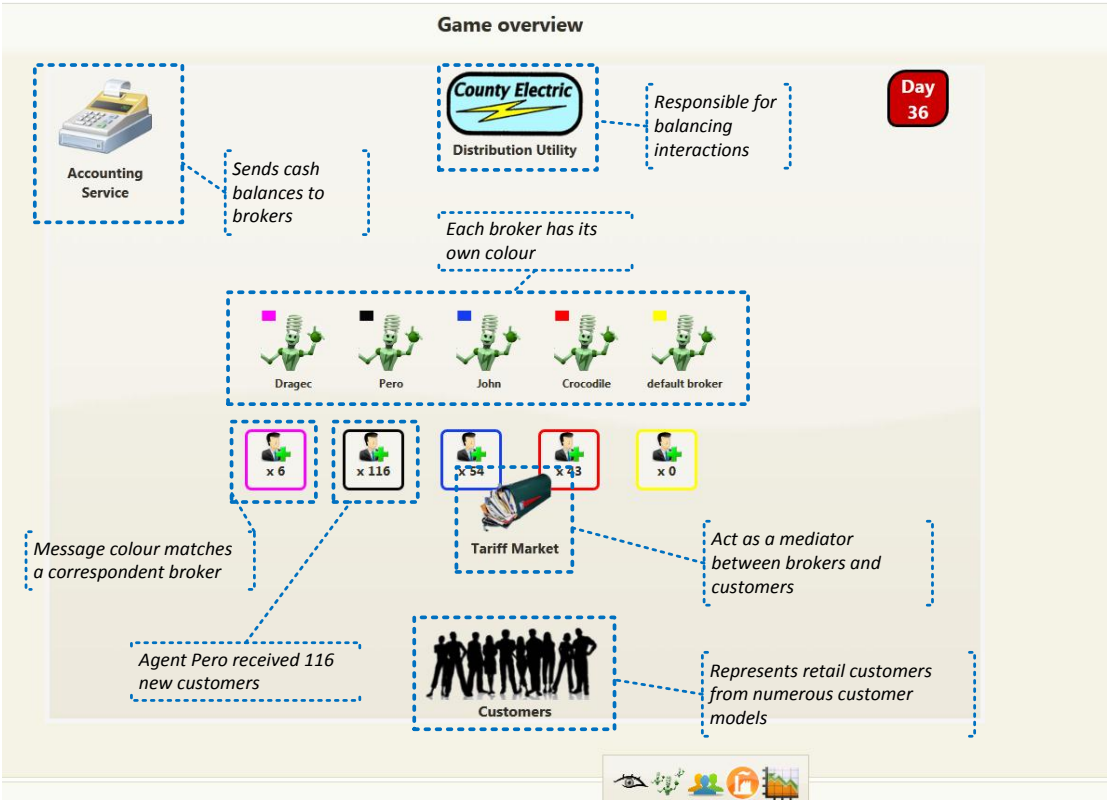

Figure 15: Game overview is represented with animation

Brokers who are currently competing in the game are set in the middle. Each broker is represented with one colour. The colour links the broker with broker's interactions. An example of interaction is a moveable message that contains information about number of new customer that subscribed to broker. This message travels from *Customers* over the *Tariff market*, to the *receiving* broker. Animation currently supports the following visualization of per-day information:

- number of new customers that subscribed to broker,
- number of customers that withdrew from broker,
- number of new tariffs sent by broker,
- balancing fee from distribution utility to broker, and
- broker's financial status.

The main problem for the game overview visualization is the game speed: typical competition configuration specifies a duration timeslot of five seconds. Within a timeslot, competitive brokers interact with numerous entities, such as tariff market, customers and distribution utility. It is obvious that it takes more than five seconds to efficiently display just the important interactions from a timeslot. To counteract this problem, a day view model is built within the back-end module. This model aggregates the timeslot data to a day data. The front-end module uses the day view model to prepare the animation. A sequential display of the activities for building an animation can be seen in Figure 16. First, the front-end module contacts the back-end module to retrieve the day overview data. Once the front-end receives the day overview data, it is necessary to dynamically generate needed XHTML elements. The front-end will create XTML elements for entities and interactions among them. Interactions between entities are shown as moveable messages. An example of a moveable message is the information about balancing report: it travels from the distribution utility to the receiving broker. Once the elements are generated, it is necessary to generate an animation scenario. Generation of scenarios involves determining the trajectory for each moveable element. The last step before launching animation is to set timeout for each movable element, in order to have a desired sequence of movements.

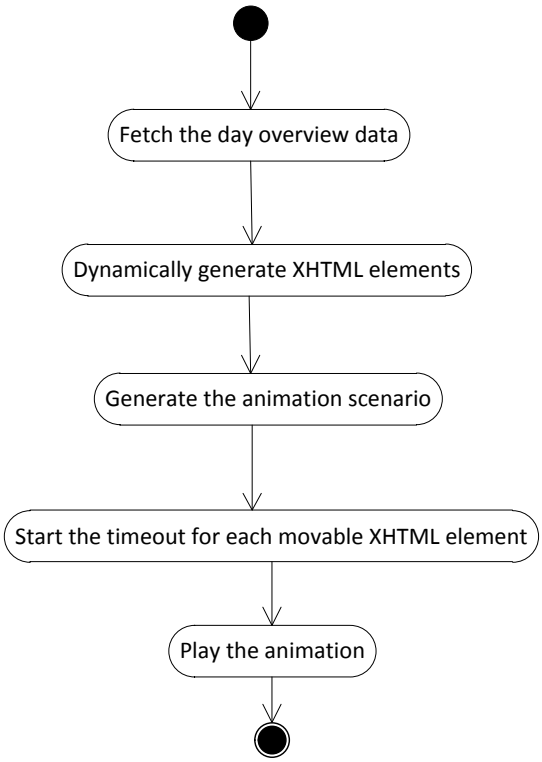

Figure 16: Activity diagram – the game overview animation

4.4.2. Brokers

Broker, represented by a trading agent, is the main element of the Power TAC competition scenario. In each timeslot, a broker goes through a three-step process: trading in the wholesale market, portfolio management and balancing. More details about this process can be found in chapter 2.2.1.

A trading agent that competes in the Power TAC competition is developed by a research team. Before they submit broker to a final competition, research teams goes through iterative process that includes coding and evaluation of a broker. For evaluation part, research teams are allowed to use the generated logs for finding useful information. While logs can provide a detailed insight into the competition's message flow, they are highly inconvenient for a frequent use.

To help research teams with the evaluation process, the Visualizer front-end module provides an extensive representation of brokers with *Brokers* page (Figure 17). Using tab widget, user is able to pick an individual broker to display. User is also able to watch both the individual info and aggregate info from all the brokers at the same time.

Aggregate info is placed on the right side of the page, next to the main content, and it is part of a master page layout. This allows users to have the crucial information about brokers on every other page. The following elements are shown in the aggregate info (Figure 18):

- *data table* showing current rankings of competing brokers,
- *line chart* showing cash balance time series for every broker, and
- *pie chart* showing current market share for competing brokers.

Data table shows relative positions of competing brokers. Each broker is represented by a row with the following info:

- *an icon*,
- *cash position*,
- *energy balance*, and
- *number of broker's customers*.

Line chart shows cash positions of each broker for all timeslots. It is useful for studying financial behaviour of brokers.

Pie chart displays market shares of each broker. Market share is determined by broker's number of customers.

Individual info contains information about broker's current cash balance, energy balance and subscription count. Beneath this, user can choose the content from the following three sections: finance section, energy balance section and portfolio section.

Finance section

Design of the finance section is shown in Figure 17. For a selected broker, the front-end Visualizer module will display line charts that show broker's evolution of finance status. User can use checkbox widgets to select an arbitrary set of charts, including:

- *daily values chart*,
- *all values chart values*, or
- *current day values chart*.

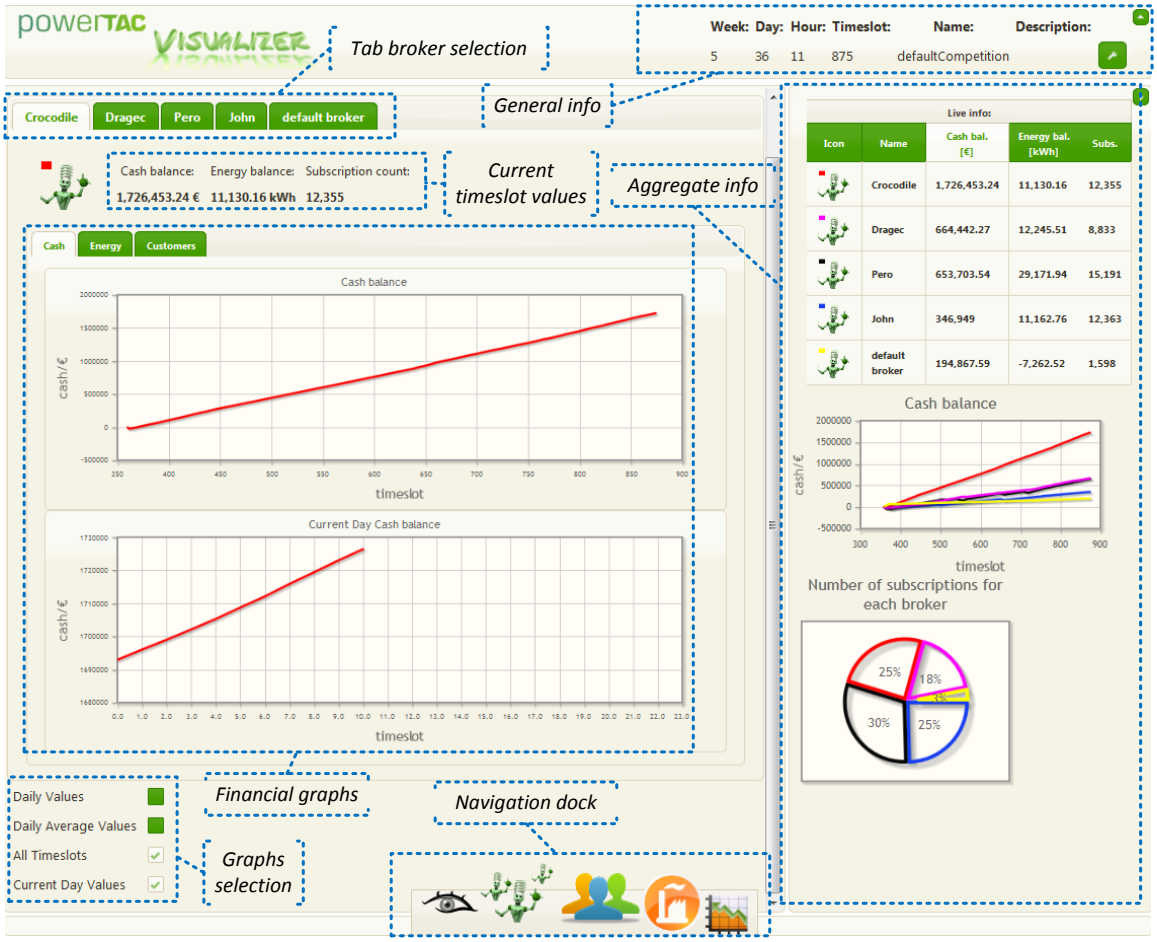

Figure 17: Broker's financial status

Energy balance section

Design of the energy balance section is shown in Figure 18. It has the similar look as the finance section. For a selected broker, the front-end Visualizer module will display line charts that show broker's energy balance. Since the broker's promising strategy is to keep energy imbalance as low as possible, this section can provide useful insights about how an individual broker handles the balancing process. User can customize the view to get an arbitrary set of charts, including:

- *all timeslots values chart*,
- *current day values chart*, and
- *daily average values*.

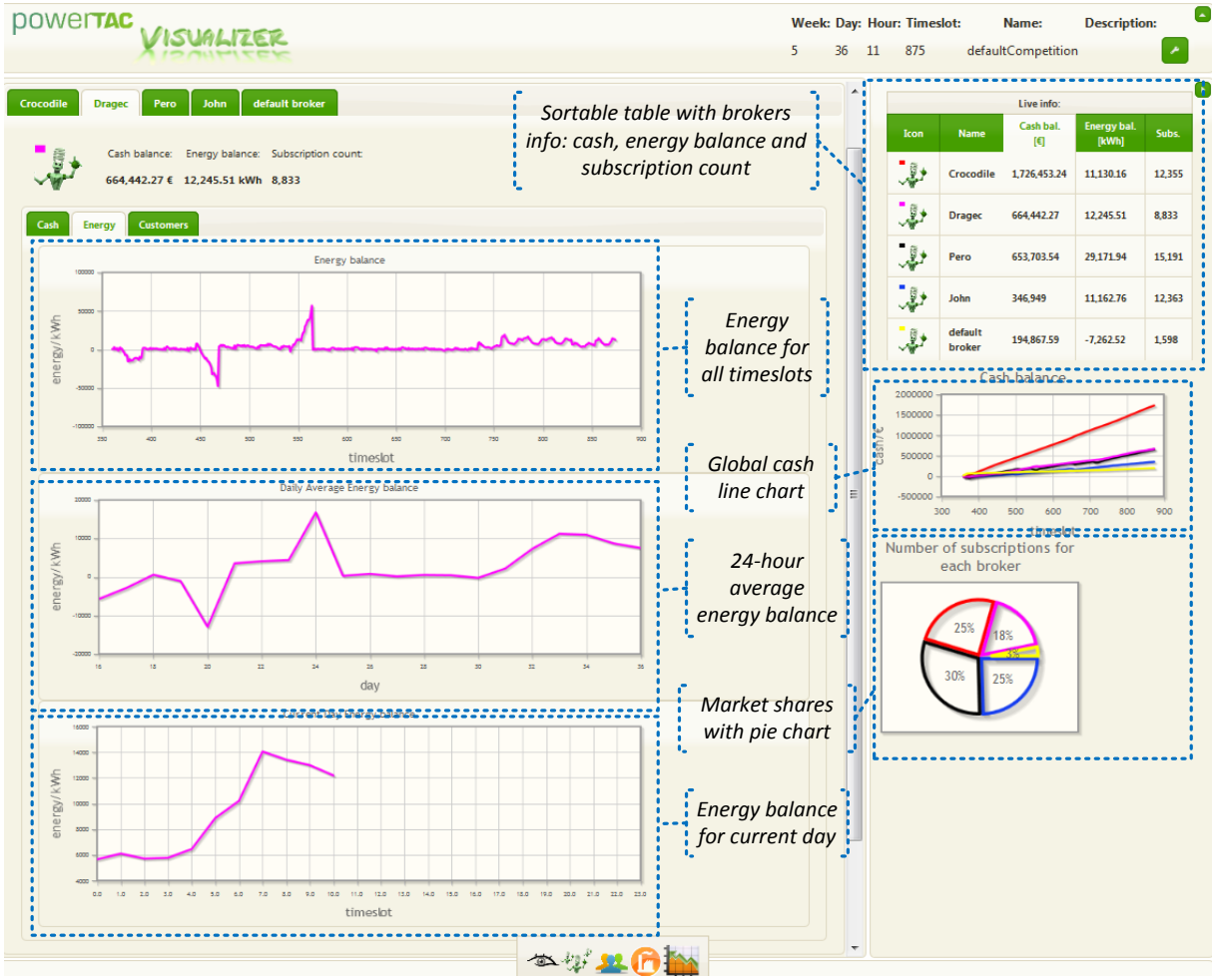

Figure 18: Broker's energy balance

Portfolio section

Design of the portfolio section is shown in Figure 19. Broker's portfolio is represented by both bubble chart and data table. Each entry in a bubble chart denotes one customer model that represents a population subscribed to an observed broker. It displays the current status in three dimensions:

- *total cash* (x-axis),
- *total energy* (y-axis), and
- *population size* (radius).

Total cash is the total cash balance for a particular customer model. Negative values implies that broker earned money on customer model, while positive values implies that broker lost money on customer model. Thus, customers that are towards left side of

the chart are more profitable than ones on the right. Similar observation goes for a total energy, which is a total energy produced (positive values) or consumed (negative values) by a particular customer model. The producers are placed in the upper part of the chart, above the zero. Since consumers have negative values, they will be placed in the bottom part of the chart, under the zero. Users can also use the data table to review the actual numbers represented by a bubble chart.

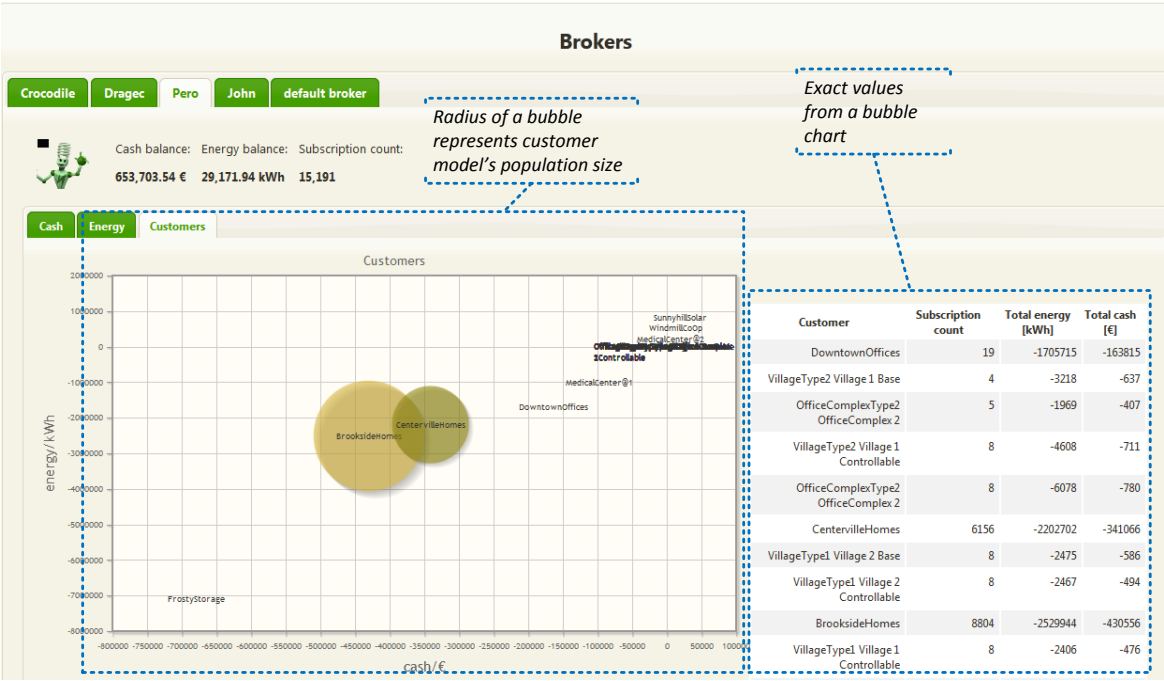

Figure 19: Broker's portfolio

4.4.3. Customers

Retail customers are essential entities connected to the grid. They can be producers, consumers or prosumers. In the real life, such customers can be households, office complexes, small wind turbines, electric vehicles or similar. In Power TAC, these customer types are modelled into customers models. Each customer model has different behaviour, including different consumption pattern, the way customer model evaluates tariff and responses to a dynamic price change.

To address the need for representation of customer models, the *Customers* page was developed. Navigation between customer models is enabled through the use of tab widgets. Each customer model has info about:

- *population size*,
- *number of active subscriptions*, and
- *power types*.

List of power types is visualized in form of icons. The Visualizer front-end visualizes three generic power types with icons. Generic power types are:

- *production type* (*positive thunderbolt icon*, shown in Figure 21),
- *consumption type* (*negative thunderbolt icon*, shown in Figure 20) and
- *storage type* (*battery icon*).

Beneath this info, user can select the content from the following sections:

- *finance section*,
- *energy section*, and
- *historical consumption data section*.

Finance section

Design of the customer's finance section is shown in Figure 20. For a selected customer model, the front-end Visualizer module will display a current timeslot values and line chart that shows customer model's evolution of finance status. Line chart consists of three series:

- *outflow series*,
- *inflow series*, and
- *total series*.

Outflow series shows the amount of money a particular customer model paid from all brokers. On the contrary, inflow series shows the amount of money a particular model received from all brokers. An aggregate finance situation of a customer model is given with total series.

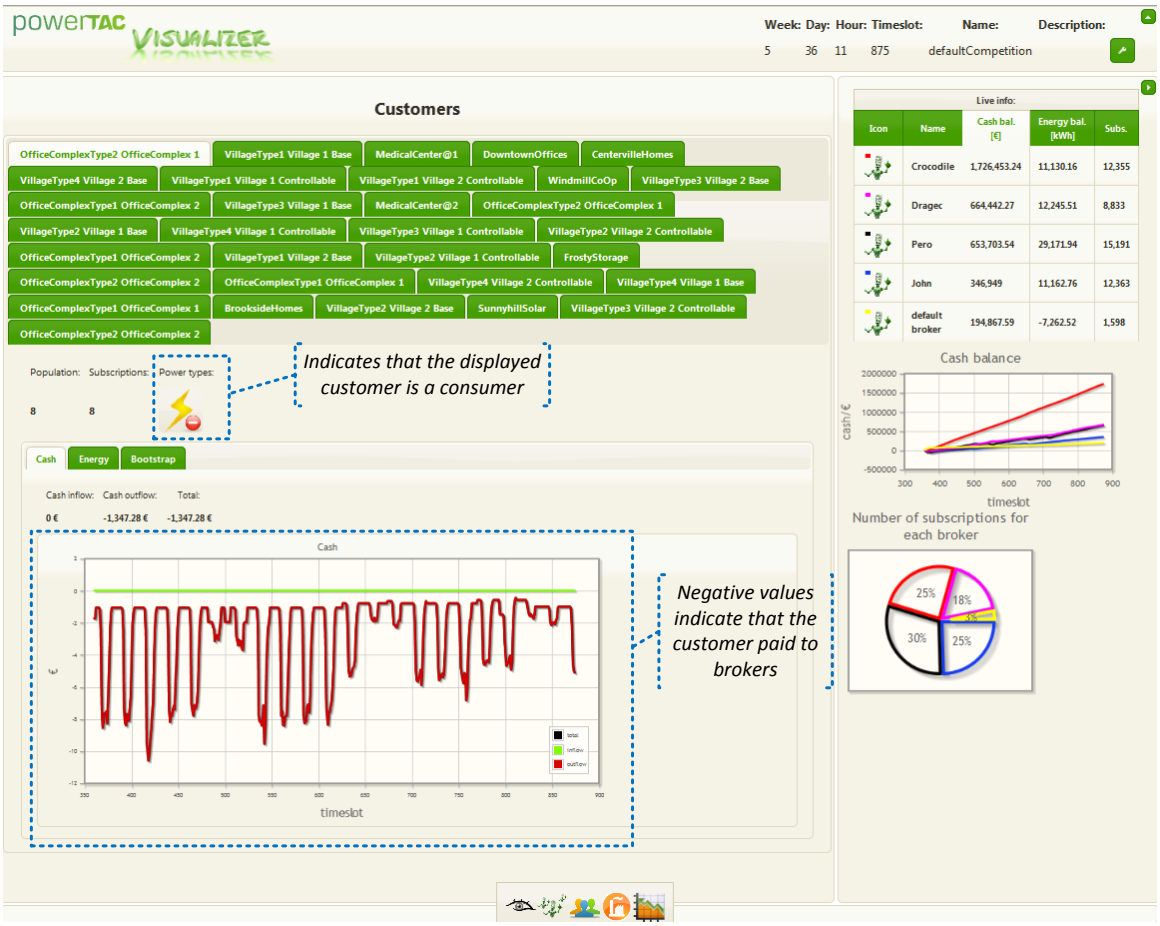

Figure 20: Customer's financial status

Energy section

Design of the customer's finance section is shown in Figure 21. For a selected customer model, the front-end Visualizer module will display a current timeslot values and line chart that shows customer model's evolution of energy usage. Line chart consists of three series:

- *production series*,

- *consumption series*, and
- *total series*.

Production series shows the amount of energy a particular customer model produced for all brokers. On the contrary, consumption series shows the amount of energy a particular model consumed from all brokers. An aggregate energy situation of a customer model is given with total series.

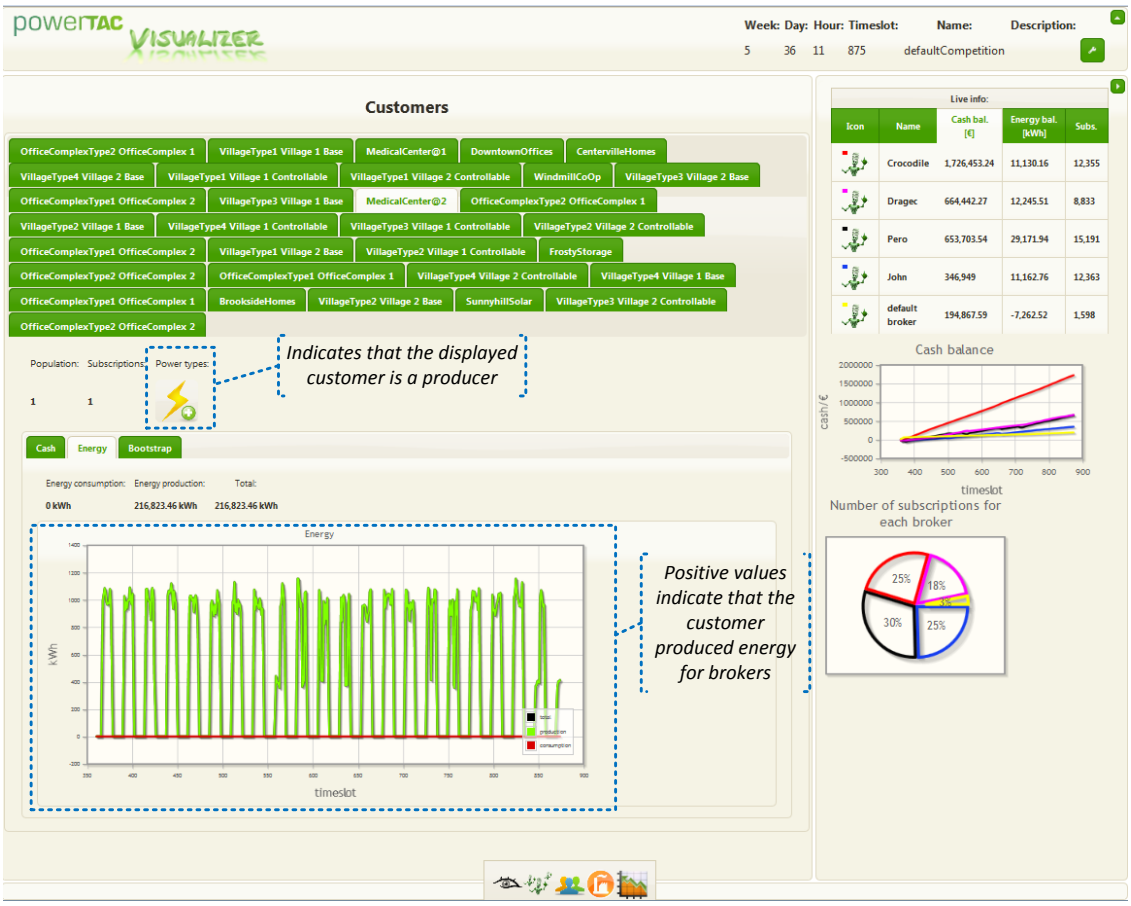

Figure 21: Customer's energy

Historical consumption data section

Design of the customer's historical consumption data section is shown in Figure 22. For a selected customer model, the front-end Visualizer module will display a line chart that shows consumption and production data for each customer model for the 14 days

preceding the start of the simulation, under the terms of the default tariffs provided by the distribution utility.

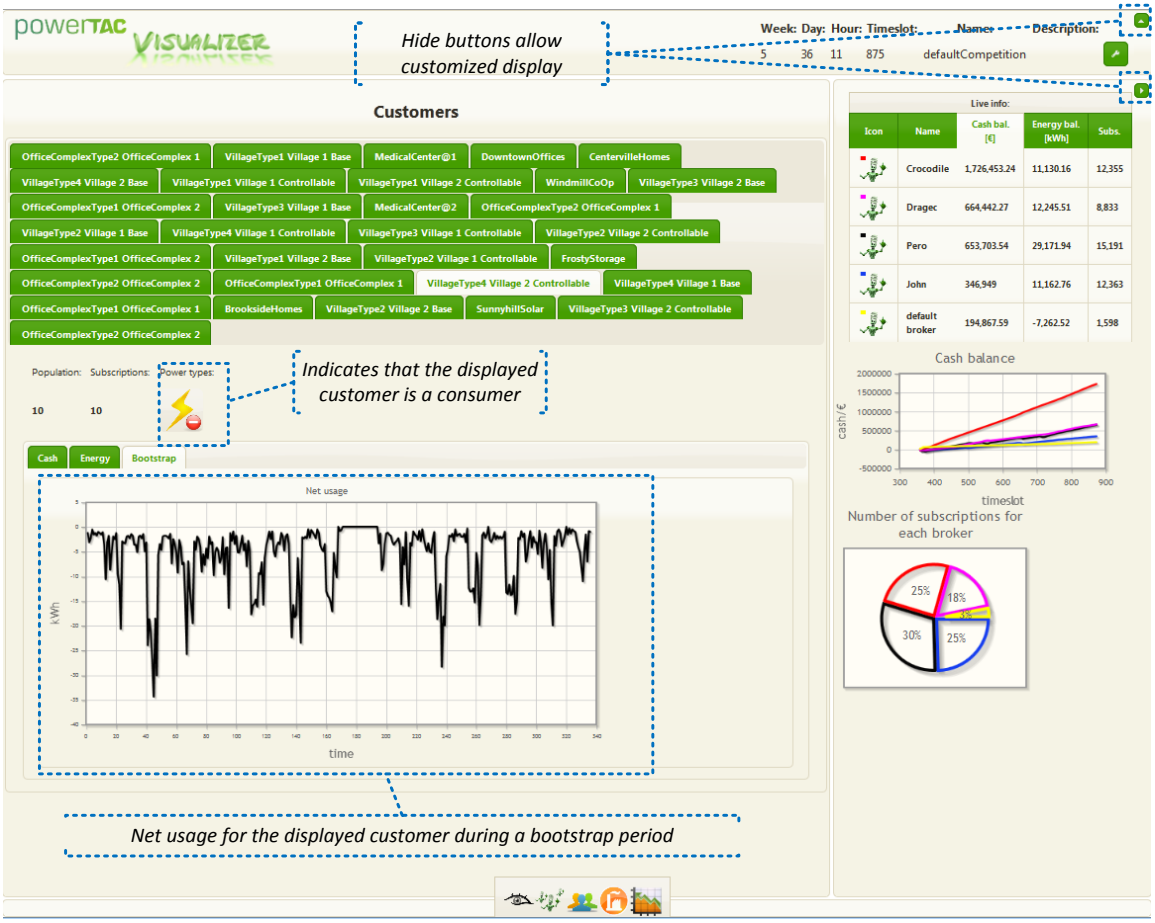

Figure 22: Customer's consumption pattern from a bootstrap period

4.4.4. Generation companies

The generation companies, or simply Gencos, are entities that only participate in the wholesale market. A large industry plant from smart-grid deployment (Figure 3) is the real-life example of such entity. To provide users possibility to get an overview of Gencos' effect on the competition, the *Gencos* page (Figure 23) was developed.

Navigation between Gencos is enabled with the use of tab-widget. The content for each Genco consists of:

- *current financial status*,

- *financial line chart*, and
- *wholesale performance data table*.

The wholesale performance data table provides a display of aggregate results in trading for a particular Genco. Each row of the data table is defined by:

- *timeslot for which trades were made*,
- *total financial balance*, and
- *total energy traded*.

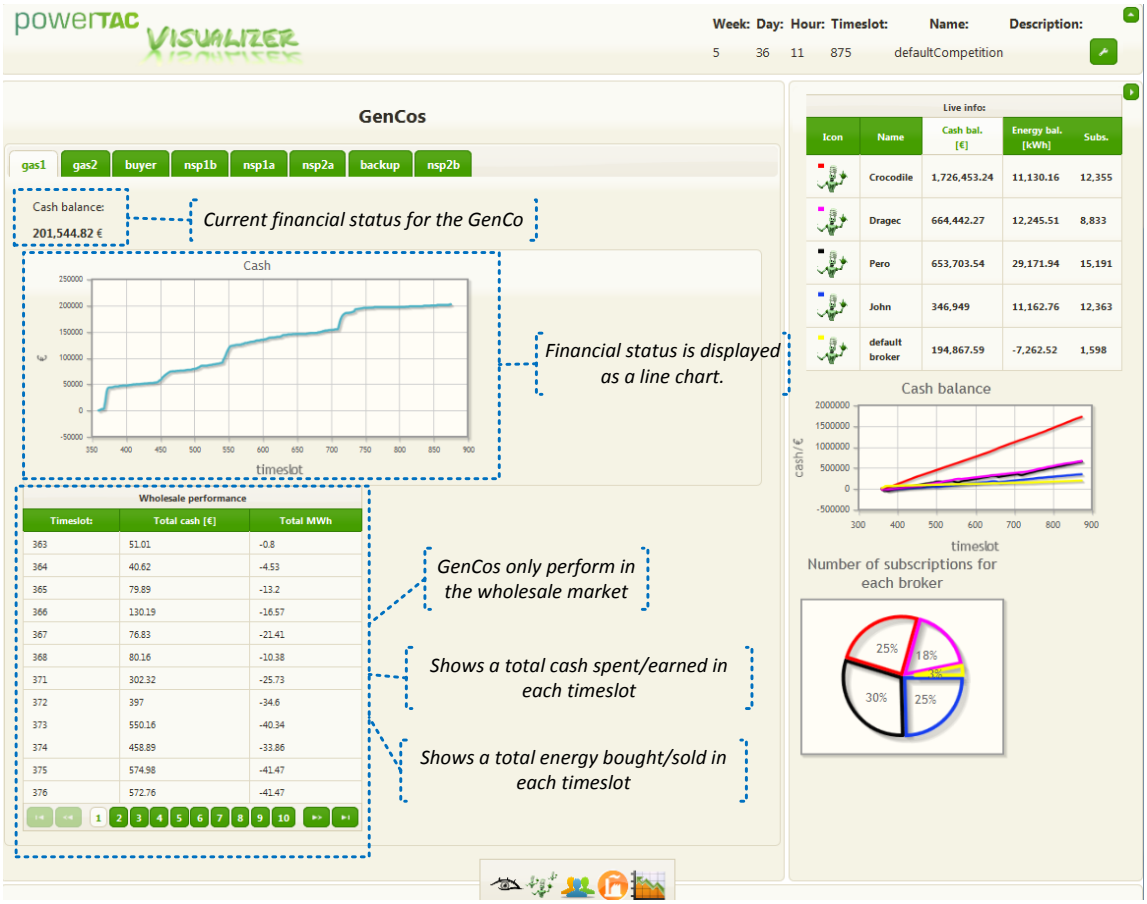

Figure 23: Financial status and wholesale performance for generation companies

4.4.5. Wholesale market

The wholesale market is used by brokers and Gencos to buy and sell quantities of energy for future delivery, up to 24 timeslots in the future. In Power TAC, this market works as a periodic double auction, clearing once every timeslot.

To address the need for wholesale market visualization, the *Wholesale* page was developed (Figure 24). It provides an overview of a market clearing. The navigation between market clearings is provided with clickable tables on left side of the page. Since there can be multiple clearings for each timeslot, the two-table system were used. First table lists all the delivery timeslots, including a total traded quantity for that timeslot. Each row can expand with an inner table that contains all the clearings for a delivery timeslot specified in the outer table. Clearing process is visualized with two line charts:

- *before clearing graph*, and
- *after clearing graph*.

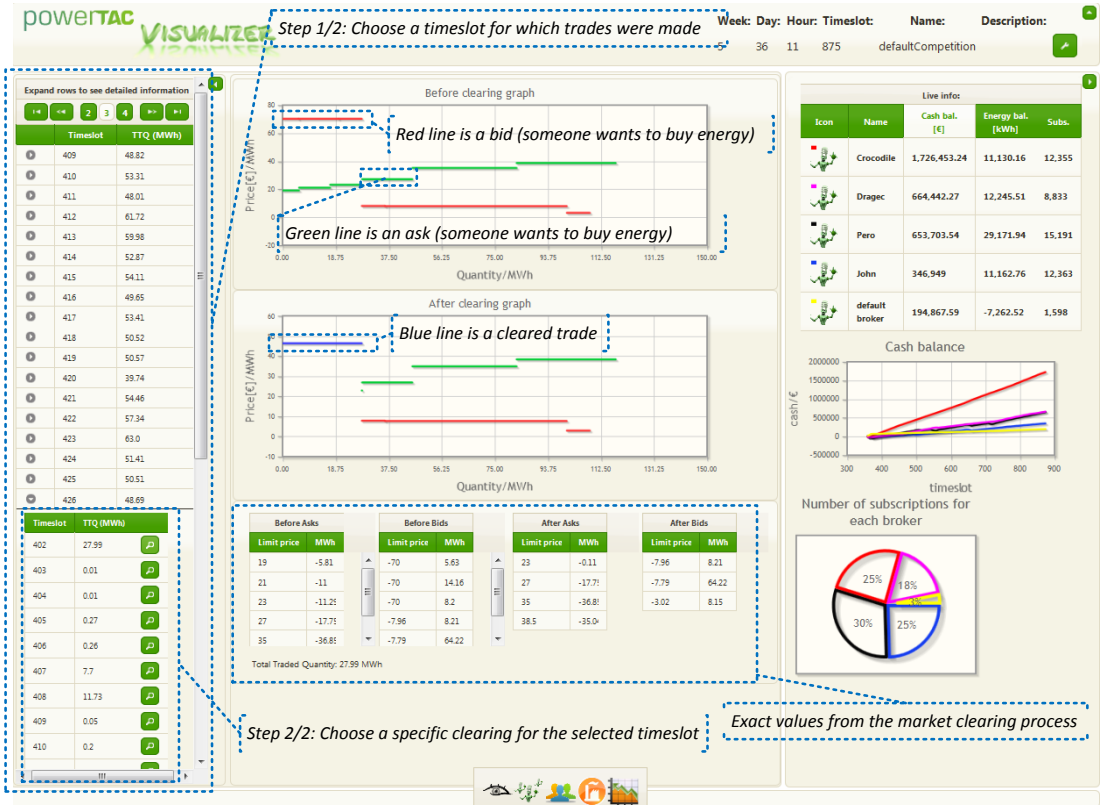

Figure 24: Wholesale market

The *before clearing graph* contains all the bids and asks gathered for a specific timeslot clearing. The result of the clearing is displayed with *after clearing graph*. Each order, visualized by a horizontal line, is represented by energy quantity and price. Red line visualizes a bid order, green line visualizes an ask order and black line visualizes a market order. If there was a cleared trade in a particular market clearing, an appropriate message will be displayed at the bottom. Both graphs have associated tables with the exact values from the market clearing process.

4.4.6. Competition control

In addition to rich visualization, the front-end module also provides an administration interface for competition control. This allows an easy configuration of the simulator according to user preferences. The Visualizer competition control replaces the manual set up of the simulator that uses the command line interface.

Design of the *Competition control* page is shown in Figure 25. At the top there is a link to the visualization part of the front-end module. Below the link, there is a space for displaying status messages. Depending on the inputs received from the user, the Visualizer displays the message about the success of the user action and the status of the game. In Figure 25, the front-end module reports that the game has successfully started and is in running status. The game configuration parameters are entered using text-input forms and data table. It is possible to configure and run two game modes:

- *bootstrap mode*, and
- *sim mode*.

The *bootstrap mode* is a game mode that runs without competing brokers. It is used to generate data that serves as input for the sim mode. To configure this mode it is necessary to enter a filename for the output file that will contain the result of the bootstrap mode game.

The *sim mode* is a game mode that runs an actual simulation with competing brokers. This mode has the following parameters:

- *input bootstrap data*,
- *jms url*, and
- *list of authorized brokers*.

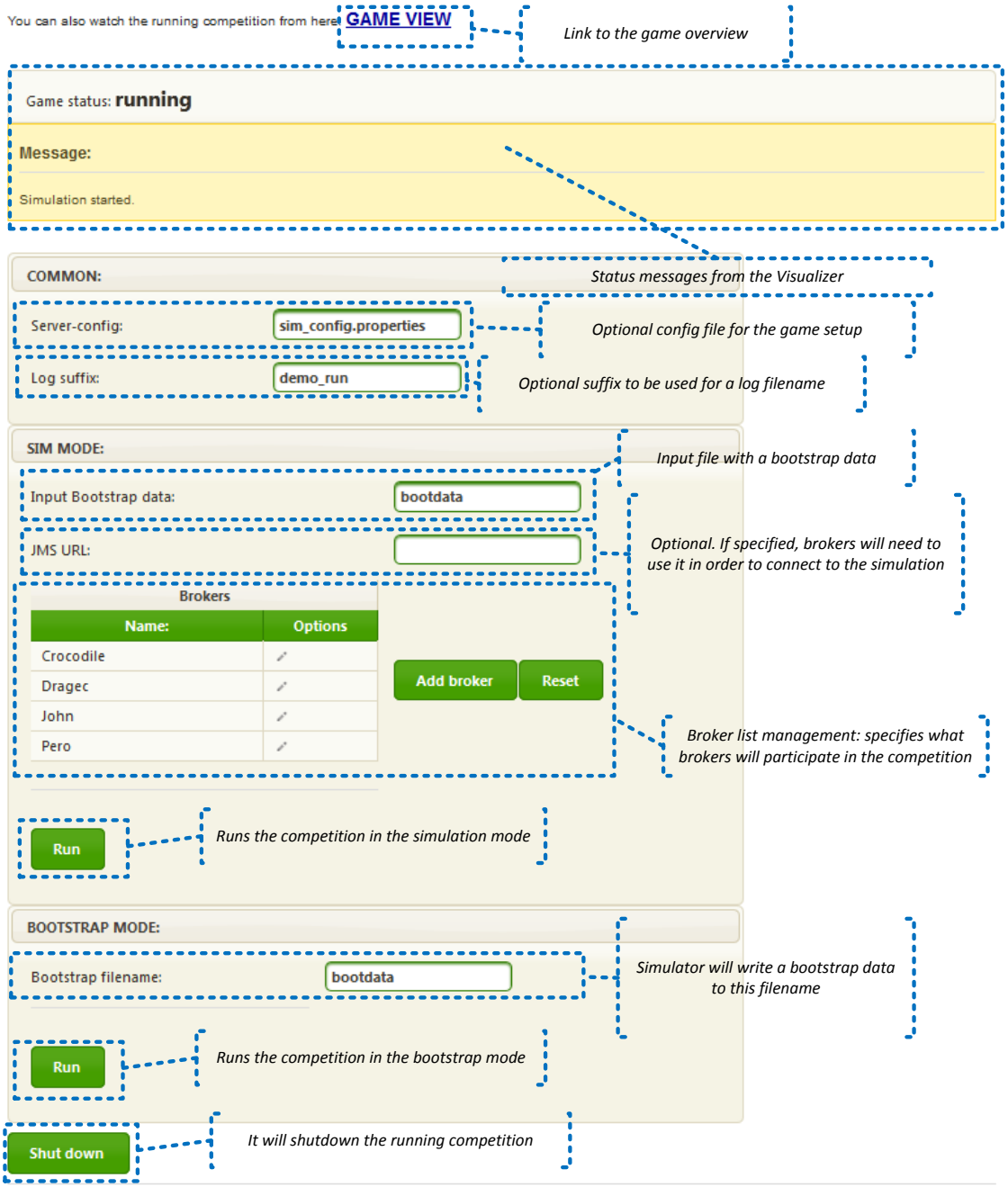

Figure 25: Admin interface – competition control

Input bootstrap data requires the name of the file that was generated in the bootstrap mode. *Jms url* is an optional parameter that will override standard URL of the message broker used by brokers. *List of authorized brokers* is the list of brokers that will participate in the game. Broker names are specified using data table.

Both modes use two common optional configuration options:

- *server-config*, and
- *log suffix*.

Server-config file has properties that override the standard server configuration. Log suffix option gives the root name for the log files. Once the required parameters are set, the game can be started with the *Run* button.

4.4.7. Periodic partial page update mechanism

One of the features front-end module has is a dynamic visualization. This allows automated update of the page content, including tables and graphs. Dynamic visualization is achieved with the period partial page update mechanism that takes use of AJAX technology. It updates only the relevant parts of a page to minimize the unnecessary network traffic load.

The sequential diagram that shows the mechanism is given in Figure 26. After the initial page load, the observer interacts with the Visualizer front-end module. The interaction lasts for five seconds, after which the front-end module issues a periodic AJAX request to the back-end module. The first task for the back-end module is to analyse parameters from a received request. When the first task is finished, the back-end module will be aware of the content it needs to provide for the front-end module. The back-end module fetches the needed data from component model, prepares an AJAX response and sends it to the front-end module. The final step in this mechanism is for the front-end module to do a partial update. This includes destroying the previous content to create space for the new data. Finally, after the front-end module cleaned the previous content, it will fill appropriate components with the new data from an AJAX response. User will now have an automatically updated page without the use of complete page refresh. This sequence is repeated in loop to provide an up-to-date visual data representation.

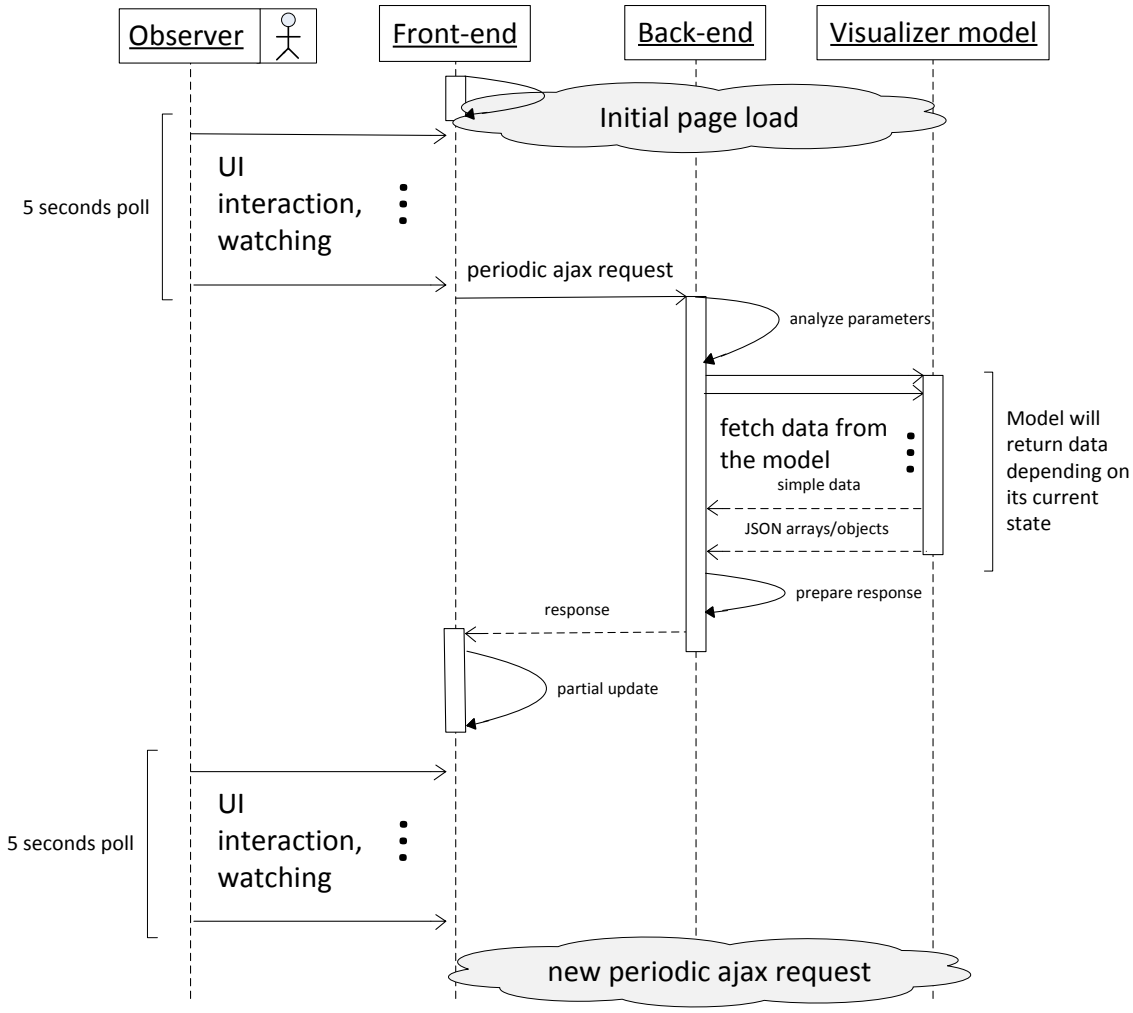

Figure 26: Sequence diagram – shows the partial page update

5. CrocodileAgent 2012

The CrocodileAgent 2012 [1] is an intelligent agent University of Zagreb team developed to participate in the Power TAC 2012. It has a modular design which is frequently encountered in TAC environment [26, 27]. Independent modules interact with each other during the competition. The CrocodileAgent 2012 is organized in the following modules as shown in Figure 27:

- Tariff Manager,
- Market Manager, and
- Forecast Manager.

Each module is divided into a service that runs the logic of the module and a repository that tracks all changes during the game, providing storage for relevant data that is used in each timeslot to maximize broker's profit.

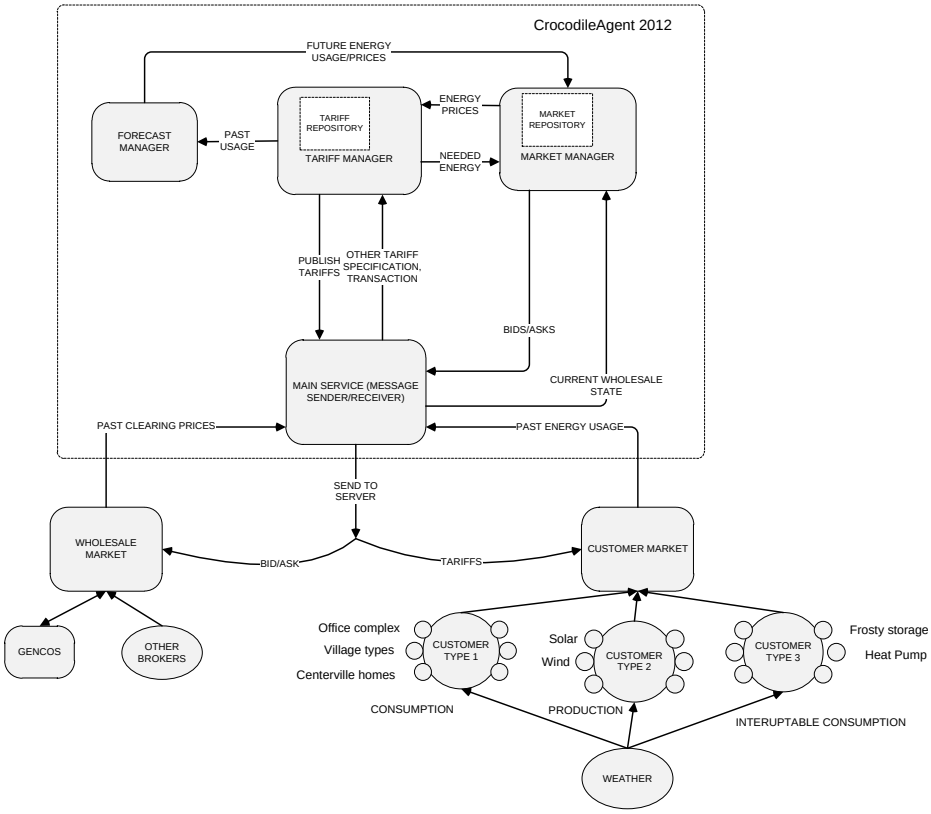

Figure 27: The CrocodileAgent's model – a high-level overview of broker's modules and their interactions with simulator components.

The Power TAC broker (e.g., CrocodileAgent) communicates with the *customer market* and the *wholesale market* receiving information about energy consumption/production, weather forecast and clearing prices. That data is used as an input to the *forecasting module* which enables the broker to identify different behaviour models for each of the customer types available in the simulation.

Energy balance greatly depends not only on the produced energy that the broker received in the *customer market*, but also on the broker's bidding strategy in the *wholesale market*. Therefore, it is of utmost importance to predict energy consumption for 24 timeslots (maximal number of timeslot in future in which the broker can place bids/asks) ahead so the broker can bid/ask for the exact amount of energy predicted.

5.1. Broker activities

The focus of the CrocodileAgent 2012 design is in developing broker's logic for participating in the customer market and the wholesale market. Theoretical background for this goal is given in the following two sections.

5.1.1. Broker activity in the customer market

In the customer market broker offers tariffs which can then be subscribed by different customers. Profitable tariff design and specification are crucial for generating profit. Also, the process of tariff creation has to include customer tariff evaluation which is calculated as the expected cost (or gain) during the lifetime[23] of the contract with the customer. Customers choose tariffs depending on the rationality parameter λ, ranging from the random selection ($\lambda = 0$) to the selection of tariff that best suits customer needs ($\lambda = \infty$).

Individual behaviour of different customer types can be described as a function of multiple variables presented as data available to the broker. The CrocodileAgent 2012 models energy usage for all customer types (i.e., $C1$ – consumption, $C2$ – production, $C3$ – interruptible) as:

[23] Lifetime of the contract represents the time of duration for which the contract is signed (e.g., one day, one week, one month).

$$energyUsage(C_1) = f(\delta, \vartheta) \tag{4}$$

$$energyUsage(C_2) = f(\delta, \vartheta) \tag{5}$$

$$energyUsage(C_3) = f(\delta, \vartheta, \sigma) \tag{6}$$

where δ is the weather information, ϑ is the information about the current TAC hour and σ is the percentage of controllable capacity offered by particular customers.

Figure 28 shows the accumulated energy consumption of typical household customers $(C1, C3)$ that are subscribed to one of the broker's tariffs. Consumption is periodic on a one-day basis, ranging from low consumption during the night, medium consumption during the morning and high consumption in the late afternoon.

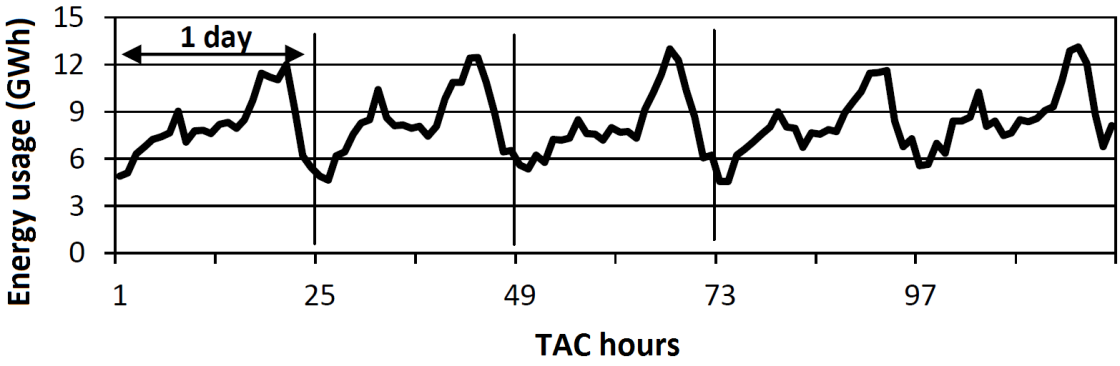

Figure 28: Consumption curve for the CentervilleHomes customer type $(C1)$

Figure 29 shows the energy consumption of a typical office complex that consists of numerous offices. As expected, the highest consumption is during working hours (08:00 – 16:00), while during the rest of the day the consumption decreases to the minimal consumption needed to maintain operation of the complex. Periodicity is manifested on a weekly basis (in addition to daily periodicity), where high consumption is during working days (Monday-Friday) and low consumption during weekends. Daily periodicity is more often seen with house customers that have similar behaviour during every day.

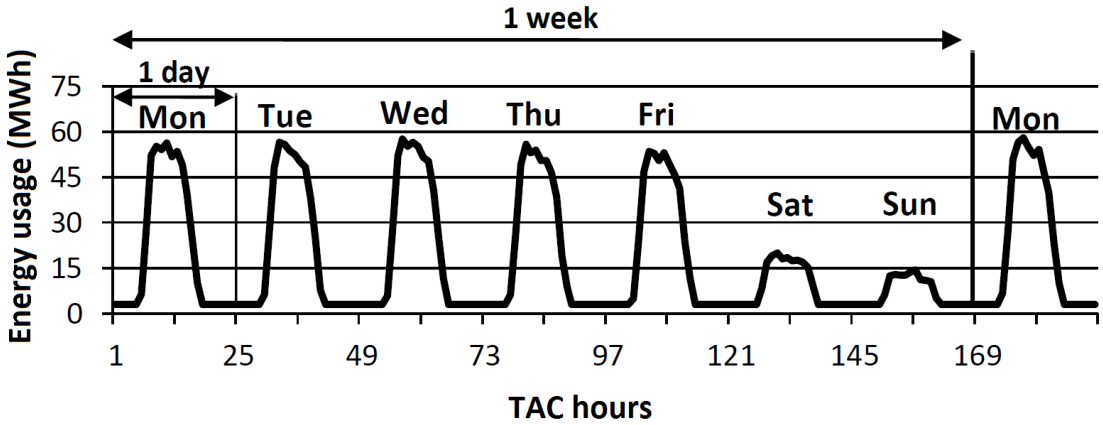

Figure 29: Consumption curve for the OfficeComplex customer type ($C1$)

On the other hand, Figure 30 shows energy production from windmills which depends on the wind speed and wind direction. Due to the variation of wind speed and wind direction, they do not exhibit steady periodicity as it can be seen in other customer types.

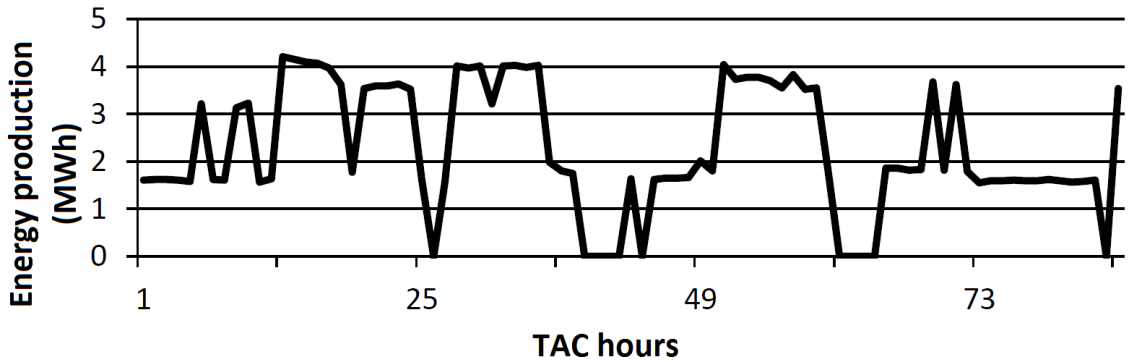

Figure 30: Production curve for the WindmillCoOp type ($C2$)

5.1.2. Broker activity in the wholesale market

In the wholesale market brokers buy and sell energy at a negotiated price known as the "market clearing price", which is a function of the following:

$$price(T_i) = f(\overrightarrow{\phi_i}, \overrightarrow{\chi_i}, [\psi_i]) \tag{7}$$

where T_i is the serial number of a TAC hour for which the trading window is open, $\overrightarrow{\phi_i}$ is the vector of bid prices, $\overrightarrow{\chi_i}$ is the vector of ask prices and $[\psi_i]$ is the matrix of energy on the bid and ask per offered contract for each i. The wholesale market is a periodic double auction which clears every TAC hour. Broker can send orders for 24 TAC-hours ahead

(i.e., day-ahead market). Orders that have positive energy and negative price values represent bids, whereas orders that contain negative energy and positive price values are considered asks. Every order must contain the broker's identity, TAC hour for which the broker is submitting the order, quantity of electric energy in megawatt-hours (MWh) and the maximum price the broker is willing to pay in euros per megawatt-hour (€/MWh). At the end of each TAC hour, supply and demand curves are constructed for each of the enabled TAC hours in order to determine market clearing prices. The clearing price is the price at which the latter curves cross or, in case they do not intersect, mean price between the highest ask and the lowest bid price. Figure 31 shows the curve of last clearing prices for TAC hours $[1-48]$. As can be seen from Figure 31, the clearing price function contains peaks at certain hours and is not periodic due to the stochastic nature of the bidding process.

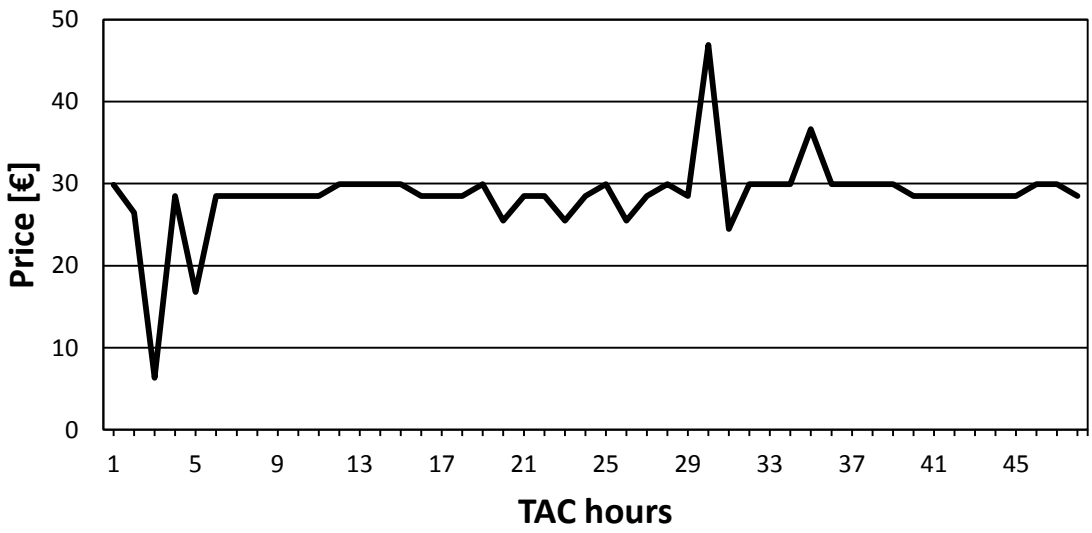

Figure 31: Clearing price curve on the wholesale market

5.2. Agent implementation

As described earlier, the CrocodileAgent 2012 consists of several independent modules that communicate with each other. In the following subsections functionalities of every module are described.

5.2.1. Holt-Winters method for predicting energy usage and prices

If CrocodileAgent's is to successfully balance its portfolio and minimize all expenditures caused by energy imbalance at the end of every timeslot, forecasting of energy usage for all customers is needed. Bids/asks on the wholesale market are submitted every hour for the upcoming 24 hours. Therefore, the Market Manager uses data which consists of past energy usage to predict the energy usage for the upcoming hours to optimize its bids/asks.

Predicted energy usage/production is calculated by the triple exponential smoothing technique well known as the Holt-Winters method [28, 29]. Calculation of the predicted usage is based on the collected data of customers' past energy usage stored in the *Tariff Manager* repository. The predicted values, which are results of the technique, can be described as:

$$predictedValues = f(\alpha, \beta, \gamma, x, period, m) \tag{8}$$

where α, β, γ are parameters used to calculate the triple exponential smoothing of past energy usage values consisted in the parameter x while m is the number of values to be predicted. It is also necessary to determine periodicity ($period$) for each set of values that we use to predict future energy usage, e.g., *Centerville Homes* have seasonality of 24 hours (Figure 28) contrary to *OfficeComplex* that has the seasonality of 168 TAC hours (one week converted into number of hours in it). The *WindProduction* and the *SolarProduction* (Figure 30) are not the subject of energy usage/production prediction because they do not have periodicity in their energy production.

Parameters α, β and $\gamma \in [0.0, 1.0]$ are sequentially used in every iteration of the prediction technique. Since we do not know which combination of those parameters returns the best results for predicted energy usage, an additional module called the *Training Module* is developed to obtain those parameter values.

The *Training Module* uses different combinations of α, β and γ parameters ranging from (0.0, 0.0, 0.0) to (1.0, 1.0, 1.0) and instead of just x values uses $x + m$ number of values stored as past energy usage/production. Additional m values are needed so the predicted values that are calculated from the x set of values can be compared to the actual data (m values). The Training Module then calculates the normalized root-mean-square

error (NRMSE) between the predicted m values and the actual values. Parameters α, β and γ that minimized the NRMSE are used in the prediction requested from the Market Manager. This evaluation is done every time before the prediction is called upon in order to obtain more precise results.

In the Figure 32 and Figure 33 we can see the relations between the predicted energy and the consumed energy for two customer types described in the previous section. The predicted values are very close to the actual values and hence are used to buy specific amounts of energy in the wholesale market. The mean error is approximately 10% in the prediction of energy usage for households, and 5% for offices. The reason of the small error is the typical consumption of offices which does not change very much during different days of the week. Consumption of homes varies due to current population in one house and usage of appliances.

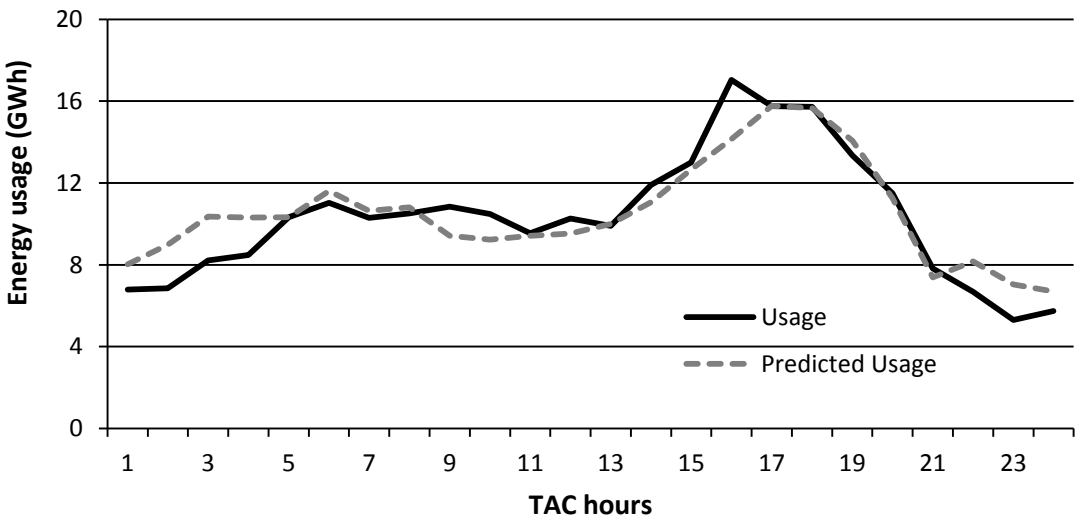

Figure 32: CentervilleHomes prediction for 24 hours

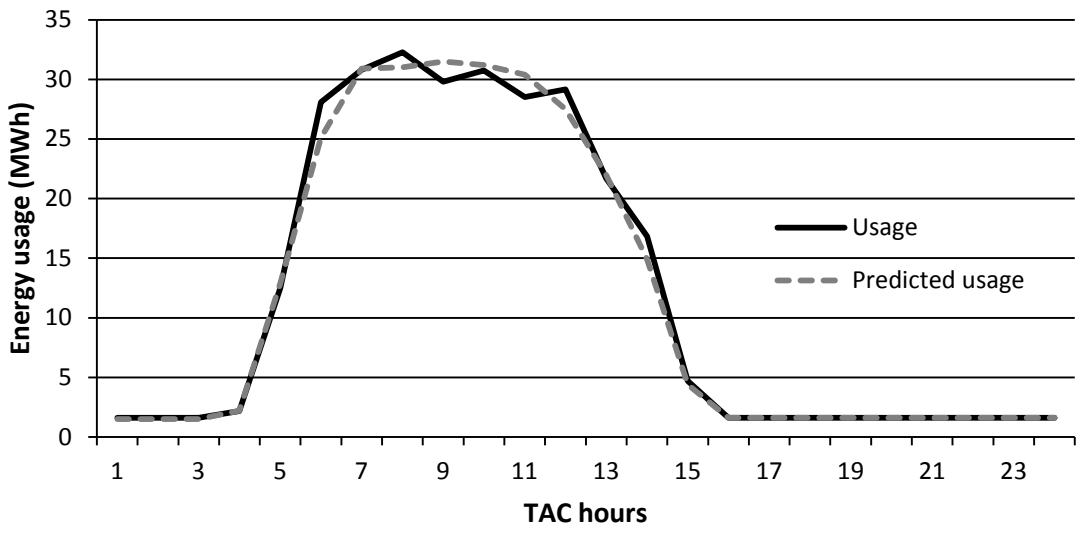

Figure 33: Comparison for load and its forecast for OfficeComplex

5.2.2. Tariff creation and evaluation mechanisms

Tariff Manager constantly tracks and analyses energy and cash flow and evaluates the received data to create *Tariff Specifications*. The Tariff Manager is very active when it comes to publishing new tariff specifications. Therefore it frequently sends tariff specifications to the customer market which it presumes will secure a reasonably good profit. Tariffs that are not profitable enough are being replaced (superseded) with tariffs that have better utility, in terms of CrocodileAgent 2012. Tariffs that show the worst or no utility (e.g. tariffs with the smallest revenue) over time are revoked completely from the game and saved in the repository so that the broker can repeatedly check and compare tariffs and create new, better ones. Tariff utility can be presented as:

$$tariffUtility = f(cC, fPk, tR, profit) \tag{9}$$

where *cC* is the number of customers currently subscribed to the tariff, *fPk* is the fixed price per kWh, *tR* is the tariff revenue calculated by summarizing tariff transactions for the specific tariff received every hour in the game and finally *profit* is the brokers profit margin. The utility of each tariff is not calculated frequently (i.e. every day) and there is a certain probability (parameter in the broker's properties which can be changed during different competitions) that the broker will calculate the utility of each tariff. The probability in the CrocodileAgent is set to 15%. Also, utility is not calculated in every TAC hour because it takes some time for the value of the utility to change. Therefore, the CrocodileAgent uses the aforementioned probability to decide if the broker will calculate

the utility. If the utility is calculated, revocation and superseding of unsuitable tariffs is immediately initiated.

On the other hand, tariff creation is initiated in each timeslot depending on the number of brokers' tariffs currently active on the market. If the number of tariffs is within the specified range, $numberOfTarrifs \in [0, N]$[24], new tariffs are created for every major customer type (i.e., consumers, producers and prosumers). If the number of tariffs is greater than the specified value, there is a certain probability p (i.e., 15%) that tariffs will be created anyway.

After the creation of the tariff specification, *Tariff Manager* implements different features to the tariff depending on the current broker's status and past tariff revenues.

Main price (€/kWh) which is a part of every rate in a tariff is calculated from past cleared prices in the wholesale market. Broker calculates the arithmetic mean price for the specific period and adds margin to it. The main price is altered for each rate depending on the rate type and purpose (i.e. time of day, day in a week, etc.). As an example, offices have tariffs with rates that have lower prices in working hours and high prices for non-working hours. It is expected for the office to subscribe to those types of tariff rates because they suit their needs. Another example is that the price of energy varies during the week (e.g. price of the energy rises from Monday to Wednesday, and from then decreases).

CrocodileAgent 2012 has four types of tariffs with different rates and required parameters based on the game time:

- fixed rate during all TAC hours,
- three-part rates during TAC hours (00:00 – 06:00, 06:00 – 18:00, 18:00 – 24:00),
- three-part rates with fixed rate during the week (different prices for rates that depend on the current weekday), and

[24] The parameter N depends on the competition and number of brokers in the game, e.g., if the number of brokers in the game is 8, N is 50.

- flat-rate model, i.e., tariff with fixed rate during the whole duration of the tariff, but with specific, fast-profit generating parameters (i.e., periodic payment[25]) to be deployed in critical situations.

According to tests that were carried out on the Power TAC 2012 platform, the most lucrative tariff model proved to be the flat-rate model, shown in Figure 34. The graph displays financial progress of the flat-rate model by presenting the amount of money tariff model earned from one customer for each timeslot. A negative starting peak suggests that the CrocodileAgent gives certain amount of money (i.e., 240 euros) given by parameter $newCustomerBonus$ to a new customer. After the initial loss, the tariff will now have a fixed per timeslot revenue provided by a periodic payment parameter $pPayment$ (i.e., 120 euros). This particular tariff design, along with its frequent spawning, proved to be the main cause why customers were lured to the CrocodileAgent's energy services.

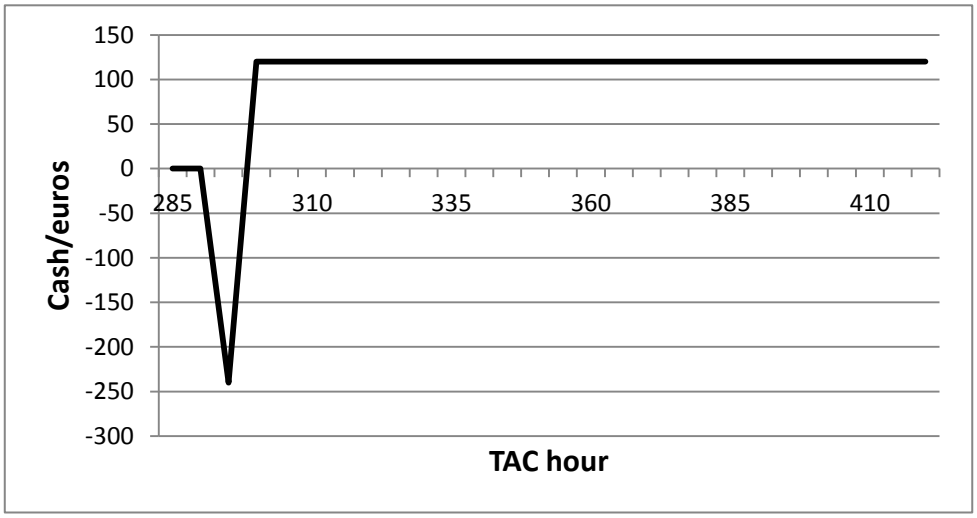

Figure 34: Hourly revenue per unit for the flat-rate tariff model

5.2.3. Wholesale bidding strategy

Market Manager obtains predicted energy usage and using data from the past market clearings and the Holt-Winters method predicts clearing prices for 24 TAC hours ahead. The input parameters are the same as in the energy load prediction in (8), but in this

[25] Periodic payment is the amount of money that each of the subscribed customers pay in each timeslot to the broker.

case x describes a set of the past clearing prices which will be used for predictions. In the case of predicted energy surplus, the broker submits an ask, and in the case of predicted shortage it submits a bid. Figure 35 shows the results of the Holt-Winters prediction on the wholesale market. The inability to predict price spikes causes inaccurate predictions.

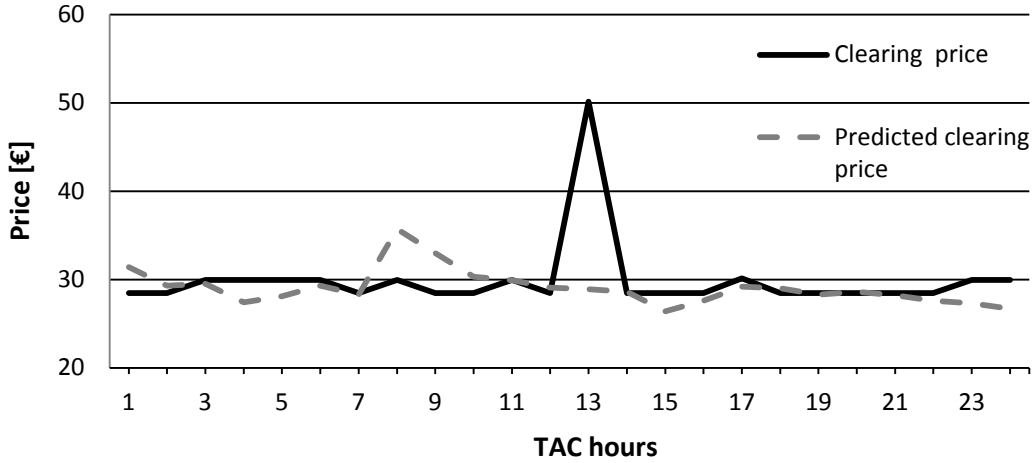

Figure 35: Predicted and actual clearing prices on one hour ahead market

Market Manager retrieves the information about the amount of energy it needs to buy or sell for each enabled TAC hour. The broker then computes the price (€/MWh) depending on the number of tries left, e.g., if the current TAC hour is T_i and the broker wants to bid for the TAC hour T_{i+10}, it has 10 tries remaining to submit orders for that particular TAC hour. As the respective TAC hour approaches, the broker increases the price it is willing to pay in order to buy the needed amount. If there is only one try left, the broker offers a predefined maximum price, which is equal to balancing cost set in the game configuration file. The algorithm for creating bids/asks is shown in Algorithm 1.

```
for each enabled TAChour
{
getAmountNeeded();
submitOrder(TAChour, amount)
}
submitOrder(TAChour, amount){
if amount > 0 {
price = computePrice(TAChour,
amount);
createOrder();
sendOrder();
}
}
computePrice(TAChour, amount)
{
calculateRemainingTries();
if remainingTries > 0
{
price =
previousPrice*2/remainingTries;
}
else{
price = maximumPrice;
}
```

Algorithm 1: Creating bids and asks for the wholesale market

5.3. Technical description

More technical details about CrocodileAgent's design are given in this section. Software architecture is described using UML class diagram. Agentware that was used for this project is explained. Two of the CrocodileAgent's technical features are also described. Finally, description of CrocodileAgent's lifecycle is given in form of the UML state diagram.

5.3.1. Software architecture

Software architecture is described as a UML class diagram shown in Figure 36. Programming artefacts are grouped in three groups:

- services,
- repositories,
- and Plain Old Java Objects.

Each of these is described in the following subsections.

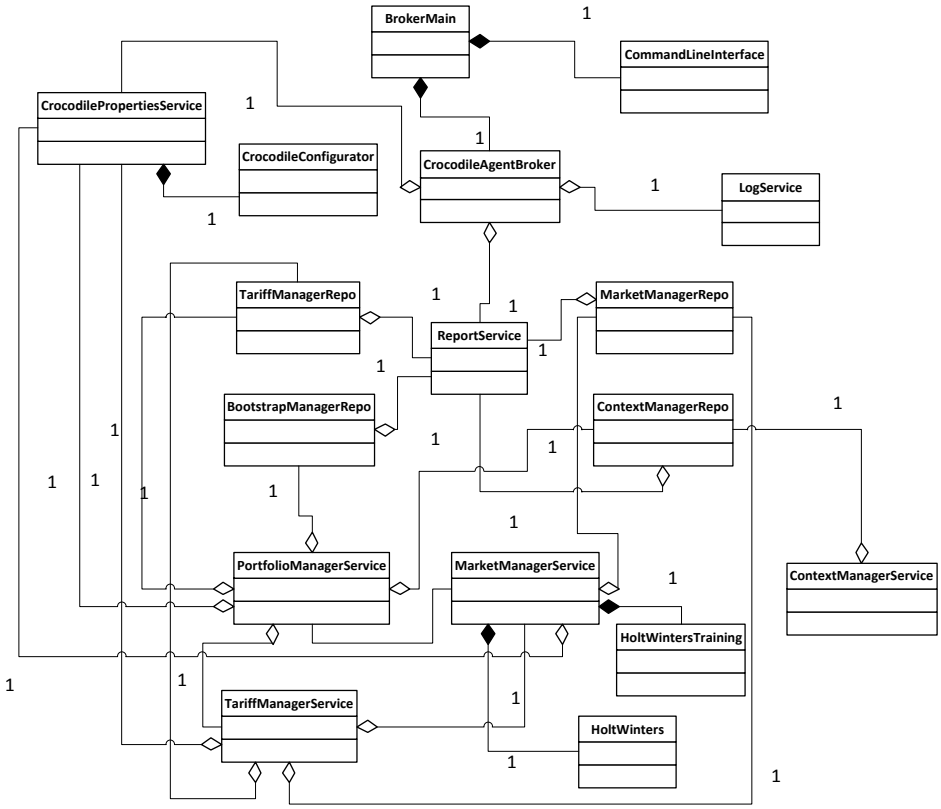

Figure 36: Class diagram – CrocodileAgent's software architecture

Services

Services are implemented as Java classes annotated with `@Service` annotation provided by the Spring framework. Thanks to this feature, a developer is able to establish dependencies between programming artefacts in an automatic and simple way. In the CrocodileAgent 2012, services contain agent's logic for competition as well as all the underlying system logic (e.g., communication with server, configuration of an agent and generation of reports). The list of the main services, as well as their brief description, is given below:

- `CrocodileAgentBroker` is an extended version of the top-level controller `PowerTacBroker` for an agent. Other than inherited functionalities, this service is also responsible for initialization of logging and reporting. Upon competition end, `CrocodileAgentBroker` will trigger the generation of a report in form of the *Excel* spreadsheet.
- `CrocodilePropertiesService` is a modified version of the `ServerPropertiesService` class that can be found in Power TAC project bundle (i.e., `server-main` project). The idea behind this service is to allow agent to use a similar configuration scheme used for configuring Power TAC projects. The retrieval of configuration parameters, as well as properties setup for each module is done by `CrocodileConfigurator` object, contained inside `CrocodilePropertiesService`.
- `LogService` is used for setting up loggers for the agent. This service is beneficial for identifying possible errors in agent's implementation as well as evaluating agent's behaviour in post-competition analysis.
- `ReportService` is a key enabler for generating reports in form of the *Excel* spreadsheet. It is used by repositories that have some information to report.
- `PortfolioManagerService` encapsulates agent's logic that deals with portfolio management responsibilities (e.g., recording customers energy usage and choosing tariff task for `TariffManagerSerivce`). It is also responsible for sending tariff-related messages generated by `TariffManagerService`.

- `TariffManagerService` has several responsibilities. First, it is used to process all the tariff-related messages. This allows agent to track status of a tariff as well as its utility. Second, it forecasts customers' energy usage. Information about forecasted energy usage is then used by `MarketManagerService` for assembly of a wholesale order. Third, it is responsible for generating tariff-related messages (i.e., new tariff specifications and tariff revoke).
- `MarketManagerService` encapsulates agent's logic that interacts with the wholesale market. It is responsible for handling wholesale-related messages as well as assembling wholesale orders to resolve energy imbalance.
- `ContextManagerService` is responsible for handling reports (i.e., broker's cash balance, bank transactions and distribution reports) and context-related messages (i.e., competition info and competition properties). Upon receiving competition properties, this service will trigger all services that need to adapt their behaviour based on competition parameters (i.e., services that implement `CompetitionConfigurationAdaptable` interfaces).

Repositories

Similar to services, repositories are also implemented as Java classes. However, unlike services, repositories are annotated with `@Repository` annotation provided by the Spring framework. This feature provides exactly the same benefit as `@Service` annotation described in the Services section above. In the CrocodileAgent 2012, repositories are used for storing received messages and agent-generated data. There are two main purposes for repositories. First, there are used by agent to store/retrieve the data for using during the competition. Second, upon the end of competition, there are used to generate reports for post-competition human analysis. The list of the repositories, as well as their brief description, is given below:

- `TariffManagerRepo` is used by both `TariffManagerService` and `PortfolioManagerService` for customer market activity. It keeps track of broker's and competing tariff specifications, broker's tariff transactions, customer energy usage from current timeslot, customer energy usage history and customer usage data from bootstrap period.

- `MarketManagerRepo` is used by `MarketManagerService` for storing and retrieving wholesale-related data. `TariffManagerService` also uses this repository to calculate arithmetic wholesale clearing price from the specified interval. It keeps track of broker's balancing transactions, cleared trades and order books from the wholesale market, broker's distribution transactions issued by the DU, wholesale data from bootstrap period, broker's evolution of market positions, broker's market transactions, weather reports and weather forecasts.

- `BootstrapManagerRepo` is used by `PortfolioManagerService` to store the data related to a bootstrap period. It keeps track of customer energy usage from bootstrap period as well as weather reports from bootstrap period. The main purpose of this repository is to transform the data into user-readable information in form of an *Excel* sheet.

- `ContextManagerRepo` is used by `PortfolioManagerService` to extract information about competition parameters. Also, this repository is used by `ContextManagerService` to store cash-related and context-related messages. It keeps track of broker's bank transactions, broker's cash positions, broker's per-timeslot revenue, competition information and distribution reports issued by the DU.

Notable Plain Old Java Objects

In addition to services and repositories there are many other objects that are part of agent's design. They are called Plain Old Java Objects (POJOs) because they are not bound by any restriction other those forced by the Java Language Specification. Three of the notable POJOs contained in the CrocodileAgent are listed below:

- `CommandLineInterface` is an object that provides simplified management of input parameters from a terminal. Thanks to *Simple JOpt*[26] library, contained in `CommandLineInterface`, developers are able to build an arbitrary list of input parameters for an agent in a simple and consistent way.

---

[26] **Simple JOpt** library is a Java library for parsing command line options (http://pholser.github.com/jopt-simple/).

- `HoltWinters` is an object that implements the Holt-Winters triple exponential smoothing for time series analysis (i.e., forecasting customers energy usage and forecasting wholesale clearing price). The source code used in this class is originally implemented by Nishant Chandra[27].

- `HoltWintersTraining` is an object that uses `HoltWinters` object to run the Holt-Winters algorithm. The purpose of this object is to determine values for parameters used in Holt-Winters method by minimizing NRMSE.

5.3.2. Agentware

Agentware is template software used by broker developers as a starting point in broker development process. It is provided by Power TAC organizers in the form of the Maven[28] project called *sample-broker* project[29]. The *sample-broker* project is accompanied with the *common* project, which contains all domain classes common for all the Power TAC projects.

Agentware core services

In order to free developers from implementing core functionalities (e.g., communication with server, maintenance of broker's internal clock and tournament login) and put more emphasis on the development of broker's intelligent behaviour, a standardized set of core services are incorporated within the *agentware*. A brief description of each core services is given below:

- `BrokerMessageReceiver` is service responsible for receiving JMS messages for the broker. It converts incoming XML message to an appropriate message type forming an object which is routed to `MessageDispatcher`.

- `BrokerPropertiesService` is used for broker's properties setup (e.g., broker's username and JMS URL). The inability of using this service for

---

[27] Github repository for Holt-Winters implementation by Nishant Chandra is available on https://github.com/nchandra/ExponentialSmoothing/.

[28] **Apache Maven** is a software project management and comprehension tool (http://maven.apache.org/).

[29] Agentware is hosted on the Power TAC Github repository: https://github.com/powertac/sample-broker.

configuring services not located in `org.powertac.samplebroker` package is the reason for existence of the `CrocodilePropertiesService` service.

- `BrokerTournamentService` is service responsible managing initial interaction between the broker and the tournament scheduler.
- `JmsManagementService` acts as an enabler of message-driven communication between the broker and the simulator. The underlying technology chosen for implementing this type of communication is JMS technology.
- `MessageDispatcher` routes incoming messages to broker components, and outgoing messages to server. Components must register for specific message type and implement a `handleMessage(msg)` method that takes the specified type as its single argument.
- `PowerTacBroker` is the top-level service for the broker. It sets up the other components, maintains the clock and is responsible for termination of the broker.

Per timeslot activation mechanism

Besides core services, *agentware* also gives guidelines for broker development and an example of handling all message types that broker can receive from a simulator. Furthermore, *agentware* is equipped with per timeslot activation mechanism, shown in Figure 37.

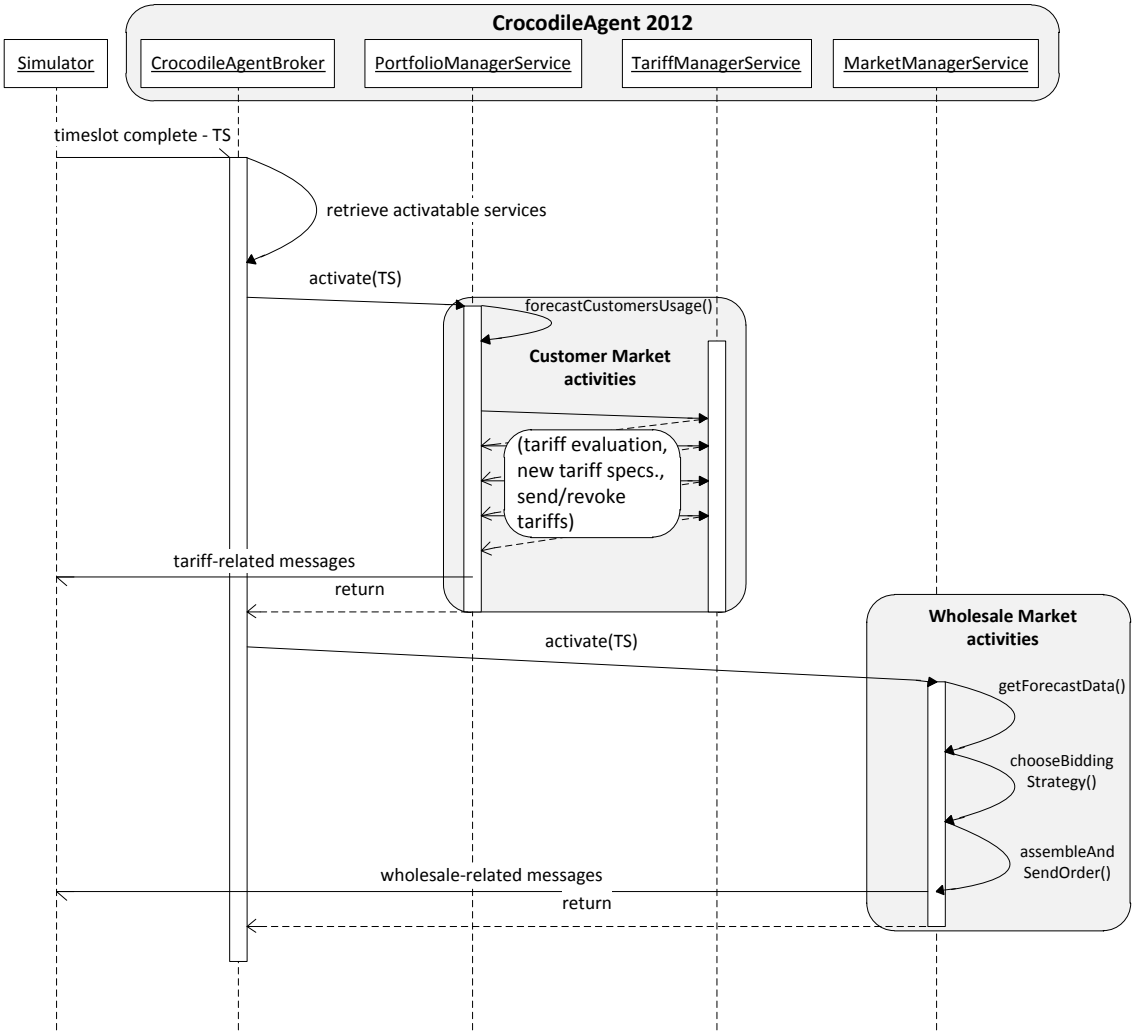

Figure 37: Sequence diagram – Activation of per timeslot activities

The per timeslot activation mechanism is triggered upon receiving `TimeslotComplete` message. After that, `CrocodileAgentBroker` service will attempt to retrieve *activatable* services. *Activatable* service is a service whose `activate(TS)` method needs to be called each timeslot and it needs to implement `Activatable` interface. From Figure 37, one can see that the CrocodileAgent has two such services: `PortfolioManagerService` and `MarketManagerService`. Retrieved services are then sequentially called for activation. It is important to note that there is no specific order for activation of *activatable* services. Consequently, broker developers should be careful not to establish close coupling between *activatable* services for the sake of possible undesired behaviour.

5.3.3. Technical features

Apart from implemented broker's logic needed for trading in wholesale and customer market, the CrocodileAgent 2012 is characterized by two technical features: configurable and reportable.

Configurable features of CrocodileAgent 2012

Power TAC broker is an autonomous agent whose aim is to acquire as much profit as possible by aggregating supply and demand of energy for customers while minimizing the energy imbalance. The term autonomous implies that agent is operating on an owner's behalf but without any interference of that ownership entity. The only communication between the CrocodileAgent and owner (i.e., developer) is indirect during the initialization process. Based on owner's preferences, the owner is able to fine-tune the agent's behaviour without making any changes to the source code. Configuration of the CrocodileAgent is achieved using an input configuration file, which is loaded by `CrocodileAgentProperties` service before the start of the competition. This allows any CrocodileAgent's component variables to be initialized according to a configuration file.

```
43 #
44 #
45 # MarketManagerService
46 crocodileagent.services.marketManagerService.shouldAdaptToCompetitionConfiguration = false
47 crocodileagent.services.marketManagerService.buyLimitPriceMax = -1.0
48 crocodileagent.services.marketManagerService.buyLimitPriceMin = -38
49 crocodileagent.services.marketManagerService.sellLimitPriceMax = 38
50 crocodileagent.services.marketManagerService.sellLimitPriceMin = 0.5
51 crocodileagent.services.marketManagerService.minMWh = 0.001
52 crocodileagent.services.marketManagerService.buyLastRemainingTryPrice = -50
53 crocodileagent.services.marketManagerService.sellLastRemainingTryPrice = 50
54 #
55 #
56 #
```

Figure 38: An excerpt from a configuration file

Example of `MarketManagerService` configuration is shown in Figure 38. According to this configuration, the CrocodileAgent will not adapt the wholesale bidding strategy to competition configuration once it is received from competition (line 46). The next two parameters define price interval for a bid order: the least expensive bid will be priced at one euro per MWh (line 47) while the most expensive bid will be priced at 38 euros per MWh (line 48). Similar parameters are available for defining price interval of an ask order: the less profitable ask is set at 0.5 euros per MWh (line 50) while the most

profitable ask is set at 38 euros per MWh (line 49). The `minMWh` parameter defines the energy limit for which a wholesale order will not be made (e.g., 0.001 MWh, line 51). Finally, last two parameters describe the last time price for bid and ask, each set at 50 euros per MWh (lines 52 and 53).

Reportable features of CrocodileAgent 2012

In addition to standard logging, the CrocodileAgent introduces the new way of storing information. Upon competition completion, each of agent's components (mainly repositories) that have `Report` interface implemented will be called to create and fill a sheet for the Excel spreadsheet file. The generated report is then used to observe broker's per timeslot performance progress and behaviour, allowing developers to enhance their learning curve about the Power TAC and the broker itself. The CrocodileAgent currently has 16 Excel sheets that include the data about:

- Per timeslot weather reports;
- Per timeslot customer energy usage for each customer type;
- Competition configuration;
- Extensive per timeslot broker's financial indicators (e.g., cash positions, cash earned and bank transactions);
- Per timeslot total consumption and production;
- Per timeslot wholesale market cleared trades;
- Broker's per timeslot balancing and distribution transactions;
- Broker's per timeslot wholesale performance;
- Revenue for each broker's tariff; and
- Tariff specifications of competing brokers.

An example of the sheet output is shown in Figure 39. An observer is able to analyse per timeslot cash position (i.e., total cash balance), cash earned during a timeslot and per timeslot bank transaction. Also, the CrocodileAgent reports total consumption and total production in kWh across all customers in a given timeslot. These are absolute (positive) values.

	A	B	C	D	E	F
1	Timeslot	Cash Position	Cash Earned	Total consumption	Total production	Bank Transaction
2	360	0	0	31956,82	0	0
3	361	-6.800,3	-6.800,3	31.942,0	1.984,0	
4	362	2.396.049,3	2.402.849,7	34.151,5	2.015,6	
5	363	2.409.711,5	13.662,1	52.085,5	1.759,2	
6	364	2.422.582,9	12.871,5	56.147,3	1.788,3	
7	365	2.432.747,6	10.164,6	61.924,3	1.853,9	
8	366	2.443.786,8	11.039,2	48.546,8	4.154,4	
9	367	2.457.624,5	13.837,7	62.483,2	6.445,4	
10	368	2.734.783,3	277.158,8	64.080,1	6.784,3	
11	369	3.070.946,2	336.162,9	55.998,6	7.055,2	
12	370	3.398.778,1	327.831,9	53.455,9	6.421,6	
13	371	3.717.982,0	319.203,9	52.569,3	6.192,6	
14	372	4.042.368,8	324.386,9	52.299,4	4.321,6	
15	373	4.343.213,1	300.844,2	55.476,3	4.197,2	
16	374	4.753.925,2	410.712,1	55.816,5	3.978,2	
17	375	5.373.543,8	619.618,6	57.288,5	4.968,6	
18	376	5.934.892,5	561.348,6	52.512,9	4.593,1	
19	377	6.408.458,2	473.565,8	56.665,6	5.597,6	
20	378	6.876.083,7	467.625,5	56.372,9	4.947,7	
21	379	7.327.858,5	451.774,8	53.793,3	4.428,8	
22	380	7.744.676,6	416.818,1	49.731,2	3.474,2	
23	381	8.095.361,4	350.684,9	42.769,0	1.960,5	
24	382	8.404.398,4	309.037,0	39.026,2	1.973,7	
25	383	8.673.504,1	269.105,7	35.422,4	1.956,3	

Figure 39: An excerpt from the *Excel* report file

5.3.4. CrocodileAgent's lifecycle

In this section the CrocodileAgent's lifecycle is described using the UML state diagram shown in Figure 40.

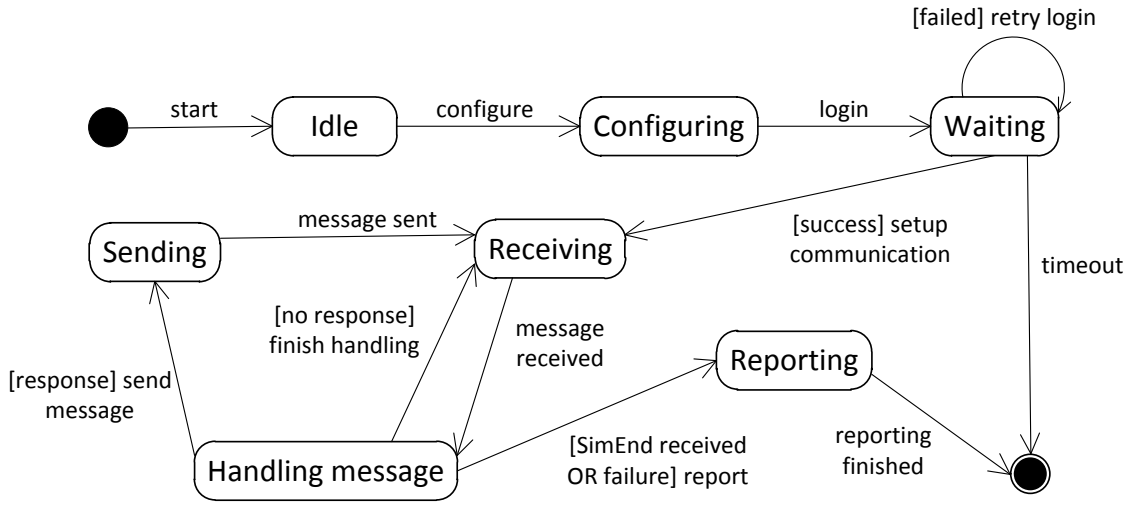

Figure 40: State diagram – CrocodileAgent's lifecycle

At the very beginning, the broker switches from starting state to *Idle* state. During this transition, the broker's context is initialized from a XML context configuration file. On *configure* event, the broker will load configuration file and will start to configure its

components, placing the broker in *Configuring* state. After the broker has been configured, the broker will try to login to the competition using the login parameters retrieved from a configuration file. From *Waiting* state, the broker is able to retry login in case of a failed login, to quit in case of a timeout or to progress to *Receiving* state in case of a successful login. After the communication between the broker and the competition is established, broker will wait for messages to arrive. For each new message, the broker switches to *Handling message* state from which the broker can handle a message without sending a response (returning to *Receiving* state) or with sending a response (switching to *Sending* state). After the message has been sent, the broker will return to *Receiving* state. The broker can reach the end state in the case of received *SimEnd* message or in the case of unexpected failure. In any case, the broker will now start to generate report, entering *Reporting* state. Upon report competition, the broker will reach ending state and finish.

5.4. Improving competition design: infinite periodic payments and the PAC-MAN syndrome

One of the goals of Power TAC is to help improve the market design of modern power systems. One of the goals of the CrocodileAgent 2012 is a similar goal but on a one level lower scale - to help improve the Power TAC competition design. In electricity markets, like in other competing games within complex systems (CGCS), agents aim to maximize their profit, looking for the weaknesses in the competing game or the complex system design. For this season of Power TAC competition the CrocodileAgent 2012 has identified two main drawbacks:

- infinite periodic payments; and
- the PAC-MAN[30] syndrome.

Although detected earlier within the competition community, their impact was shown in the finals official Power Trading Agent Competition, June 4th-8th, Valencia, Spain. It is advantageous that these two have been discovered as their discovery improves the design of complex electric market simulators.

---

[30] **PAC-MAN** is an arcade video game that was made by Namco Ltd. It was released in 1980, and became very popular in the history of games.

First, the CrocodileAgent 2012 has discovered that the customers are ignorant to the periodic payments that they have to pay to the brokers and are willing to pay infinite periodic payments. Assigning high (potentially infinite) amount to periodic payment in compare to the price of electric energy has proven to be the promising strategy for the CrocodileAgent 2012. In order to use that to the full extent, the CrocodileAgent 2012 priced its energy at 0 €/kWh in the later stage of the game, using a so-called flat-rate tariff model taking only the periodic payments from its customers.

Second, the CrocodileAgent 2012 found out that penalization of about 1000 € is not high enough to discourage it to offer similar tariffs to the customers, therefore, at certain times, it offered tariffs with random factors that define the amount of periodic payment. In order to counter other brokers from following the same strategy, besides the possibility to change the periodic payment, the CrocodileAgent 2012 could issue more tariffs. The name PAC-MAN syndrome for this tactic has been inspired by the CrocodileAgent 2012's successive spawning of the new tariffs in a stable market environment, when its market share increases to the level in which the pie diagram showing its market share starts to resemble the PAC-MAN game character, as shown in Figure 41.

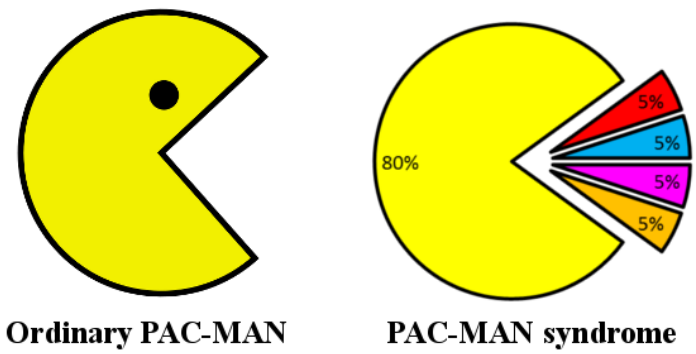

Figure 41: Ordinary PAC-MAN game character and PAC-MAN syndrome

In our opinion, the learning curve for the competition design can be improved by segmenting the one-year competition runs into more stages, e.g., three stages separated by three months each. We believe there are two reasons it makes sense to increase the learning curve of the competition design improvement for competing games in complex systems. The first, there will always be a need to find better market designs and mechanisms in CGCS. The second, improved learning curve will contribute to faster development of better designs and mechanisms which can save substantial welfare to societies, best seen in the costs of failed market mechanisms such as California energy crisis [15].

Conclusion

Electric power systems of the future (usually referenced as smart grids) will be based on digitally enabled electrical grid that will gather and distribute electrical power based on behaviour of all stakeholders (suppliers and consumers) in order to improve the efficiency, reliability, economics and sustainability of electricity services. However, such systems will need more than low-cost renewable energy sources – they will also need efficient price signals that motivate sustainable energy consumption as well as a better real-time alignment of energy demand and supply.

The Power Trading Agent Competition (TAC) is an agent-based simulation platform designed to enable exploration of the retail electricity market. The main goal of the Power TAC is to give a complete overview of the possibilities and limitations of open markets to identify good practices and policy guidelines necessary for management of smart grids. The Power TAC platform offers a great deal of flexibility by enabling to incorporate many different types of consumer or producer models, as well as simulate different scenarios.

The Power TAC is developed in the cooperation of six universities across Europe and North America, including the University of Zagreb. Responsibility of the University of Zagreb's team is to develop a visualization module for the Power TAC platform. The Power TAC Visualizer is a crucial component for the success of the whole Power TAC project because not only it enables real-time observing of Power TAC competitions, but also provides enhanced analysis of stakeholders' behaviour in the Power TAC market and as such presents a grounding for identifying good practices for smart grid management.

In early June 2012, at the International Joint Conference of Autonomous Agents and Multi-Agent Systems (AAMAS 2012) in Valencia, the Power TAC competition ran for the first time. The CrocodileAgent 2012, a software agent developed by University of Zagreb team, was one of participants in the Power TAC 2012 and was competing against several agents prepared by other international teams from industry and academia. Due to technical issues with the Power TAC server, the competition was rescheduled for September. Nevertheless, the CrocodileAgent 2012 has made a considerable impact within Power TAC community. Its progressive strategy on the tariff market (i.e., frequent

spawning of tariffs with huge periodic payments) has made Power TAC designers rethink about the game flaws from both game design and program implementation. Hopefully, these findings from University of Zagreb team will be taken into account for the next Power TAC release to develop even more credible and competitive simulation platform for power trading.

Bibliography

[1] J. Babic et al., “The CrocodileAgent 2012: Research for Efficient Agent - based Electricity Trading Mechanisms,” in *Special Session on Trading Agent Competition @ KES-AMSTA 2012*, 2012, pp. 1-13.

[2] UN, “Agenda 21: Earth Summit - The United Nations Programme of Action from Rio,” *United Nations Conference on Environment and Development UNCED or the Earth Summit Rio de Janeiro Brazil 1992*. United Nations, p. 294, 1993.

[3] Eurobserv’er, “The State of Renewable Energies in Europe - 11th EurObserv’ER Report,” 2011.

[4] European Commission, “EUROPE 2020 - A strategy for smart, sustainable and inclusive growth. 3.3.2010 COM(2010) 2020 final.” Brussels, p. 34, 2010.

[5] N. Tesla, “A new system of alternate current motors and transformers,” *Proceedings of the IEEE*, vol. 72, no. 2, pp. 165-173, 1984.

[6] REN21, “Renewables 2011 Global Status Report,” Deutsche Gesellschaft für Technische Zusammenarbeit (GTZ) GmbH, 2011.

[7] European Commission, “European Technology Platform SmartGrids Strategic Deployment Document for Europe’s Electricity Networks of the Future,” 2010.

[8] J. Kumagai, “Virtual Power Plants, Real Power,” *IEEE Spectrum*, vol. 49, no. 3, pp. 13-14, 2012.

[9] BMWi, “E-Energy - IKT-basiertes Energiesystem der Zukunft,” *Stand*, 2008.

[10] US Congress, “Energy Independence and Security Act,” *Public Law*. Authenticated US Government Information, pp. 1-311, 2007.

[11] NIST, “NIST Special Publication 1108R2 NIST Framework and Roadmap for Smart Grid Interoperability Standards , NIST Special Publication 1108R2 NIST Framework and Roadmap for Smart Grid Interoperability Standards,” *NIST Special Publication*, p. 225, 2012.

[12] S. Gottwalt, W. Ketter, C. Block, J. Collins, and C. Weinhardt, “Demand side management—A simulation of household behavior under variable prices,” *Energy Policy*, vol. 39, no. 12, pp. 8163-8174, Dec. 2011.

[13] EEGI, “Roadmap 2010-18 and Detailed Implementation Plan 2010-12,” *Technical Report*, 2010.

[14] P. Joskow and J. Tirole, “Retail Electricity Competition,” *The RAND Journal of Economics*, vol. 37, no. 4, pp. 799-815, 2004.

[15] S. Borenstein, J. B. Bushnell, and F. A. Wolak, “Measuring Market Inefficiencies in California’s Restructured Wholesale Electricity Market,” *American Economic Review*, vol. 92, no. 5, pp. 1376-1405, 2002.

[16] C. Block, J. Collins, W. Ketter, and C. Weinhardt, “A Multi-Agent Energy Trading Competition,” *Technical Report*, no. ERS–2009-054-LIS. Erasmus Research Institute of Management, Erasmus University, 2009.

[17] G. Conzelmann, G. Boyd, V. Koritarov, and T. Veselka, “Multi-agent power market simulation using EMCAS,” in *Power Engineering Society General Meeting 2005 IEEE*, 2005, pp. 2829 - 2834.

[18] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawai, and H. Matsubara, “RoboCup: A challenge problem for AI and robotics,” *RoboCup97 Robot Soccer World Cup I*, vol. 1395, pp. 1-19, 1998.

[19] M. Buehler, K. Iagnemma, and S. Singh, “The 2005 DARPA Grand Challenge,” *Springer Tracts in Advanced Robotics*, vol. 36, no. 5, pp. 1-43, 2007.

[20] J. Eriksson and S. Janson, “The Trading Agent Competition – TAC 2002,” *ERCIM News*, no. 51, pp. 57-58, 2002.

[21] J. Collins, R. Arunachalam, N. Sadeh, J. Eriksson, N. Finne, and S. Janson, “The Supply Chain Management Game for the 2006 Trading Agent Competition,” 2006.

[22] P. R. Jordan, B. Cassell, L. F. Callender, and M. P. Wellman, “Ad Auctions: The Ad Auctions Game,” *Technical Report*, 2009.

[23] J. Niu, K. Cai, S. Parsons, P. McBurney, and E. Gerding, “What the 2007 TAC Market Design Game tells us about effective auction mechanisms,” *Autonomous Agents and MultiAgent Systems*, vol. 21, no. 2, pp. 172-203, 2010.

[24] W. Ketter, J. Collins, P. P. Reddy, C. M. Flath, and M. d. Weerdt, “The Power Trading Agent Competition,” *ERIM Report Series Reference No. ERS-2011-027-LIS*, 2011.

[25] S. Oren, “Multi-product pricing for electric power,” *Energy Economics*, vol. 9, no. 2, pp. 104-114, 1987.

[26] A. Petric, V. Podobnik, and G. Jezic, “The CrocodileAgent : Analysis and comparison with other TAC SCM 2005 agents,” in *Proceedings of the AAMAS Joint International Workshop on the Trading Agent Design and Analysis and Agent Mediated Electronic Commerce (TADA/AMEC)*, 2006, pp. 202-205.

[27] I. Siranovic, T. Cavka, A. Petric, and V. Podobnik, “A Bidding Agent for Advertisement Auctions : An Overview of the CrocodileAgent 2010,” in

Proceedings of the 2011 Workshop on Trading Agent Design and Analysis, IJCAI, 2011, pp. 68-75.

[28] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, vol. 6, no. 3, pp. 324-342, 1960.

[29] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5-10, 2004.

[30] World Commission on Environment and Development, *Our Common Future*. Oxford University Press, 1987, p. 400.

Sažetak

Platforma za simuliranje trgovanja električnom energijom

Ovaj rad opisuje platformu za simuliranje trgovanja električnom energijom. Na početku je opisana motivacija za prelazak iz tradicionalnih energetskih mreža u napredne energetske mreže (engl. *smart grids*). Naglasak u ovom radu jest na informacijsko-komunikacijskom te tržišnom aspektu naprednih energetskih mreža. Simulacijska platforma Power TAC predlaže uspostavu maloprodajnog tržišta električnom energijom. Opisane su glavne komponente i scenarij natjecanja Power TAC. Posebni naglasak stavljen je na vizualizacijski modul platforme Power TAC, koji je jedan od izvornih doprinosa rada. Nadalje, opisan je i programski agent CrocodileAgent 2012 koji ima ulogu posrednika u okružju Power TAC. Rješenja opisana u radu evaluirana su na završnici natjecanja Power TAC 2012, održanog početkom lipnja 2012. godine za vrijeme konferencije *International Joint Conference of Autonomous Agents and Multi-Agent Systems* (AAMAS 2012) u Valenciji, Španjolska.

Ključne riječi: napredne energetske mreže, simulacijska platforma, programski agenti, vizualizacija tržišta električne energije, Power Trading Agent Competition

Summary

A simulation platform for power trading

This thesis describes a simulation platform for power trading. Motivation for switching from traditional grid to an advanced grid called smart grid is given. The emphasis of this thesis is in information-communication and market aspect of smart grids. The simulation platform Power TAC proposes the establishment of retail-level power markets. Special emphasis was placed on the visualization module Power TAC, which is one of the original contributions from this thesis. Furthermore, the CrocodileAgent 2012 is described, a software agent which has a role of a broker in the Power TAC environment. The solutions described in this thesis were evaluated on the Power TAC 2012 finals, held in early June 2012 at the International Joint Conference of Autonomous Agents and Multi-Agent Systems (AAMAS 2012) in Valencia, Spain.

Keywords: smart grids, simulation platform, software agents, electricity trading visualization, Power Trading Agent Competition