University of Zurich[UZH]

Master Thesis: Informatics and Sustainability Research Group,
Department of Informatics, University of Zurich

Date of submission: 19.12.2013

Simulating an Adaptive Heating and Cooling System

Samuele Zoppi

Zurich, Switzerland

08-702-664

samuele.zoppi@uzh.ch

Supervisor: Prof. Dr. Lorenz Hilty

Co-Supervisor: Nikolaus Bornhöft

Acknowledgments

I am very grateful to Prof. Dr. Lorenz Hilty and to Nikolaus Bornhöft, who gave me the opportunity to work on this very interesting topic. I really appreciated their constant help and all the meetings we had. It would be a pleasure for me to work on some other project together.

I would also like to thank my friend Frida Juldaschewa for the support she gave me throughout this thesis and my family for always being there when I needed.

Abstract

In this thesis the model of a combined heating and cooling system is developed in order to assess potential energy and costs savings. The system has been modeled using the discrete event simulation paradigm, while the model has been implemented with the help of the Desmo-J (Discrete Event Simulation Modeling in Java) framework. The model flexibility allows to simulate very different scenarios, since a different parametrization can be applied to the heating and cooling system configuration, the control strategy used, the pricing model chosen as well as to the energy demands. Different experiments investigating the potential savings of using different price models and energy capacities of the system storages have been executed. The results show that the largest savings are reached by increasing the energy capacity of the waste heat storage and by the use of a spot market based pricing model.

Zusammenfassung

In dieser Arbeit wird ein Modell eines kombinierten Heiz-und Kühlsystems entwickelt, um mögliche Energie- und Kostensparpotenziale zu untersuchen. Um das System zu modellieren wurde der ereignisorientierte Simulationsmodellierungsstil verwendet. Das resultierende Modell wurde mit Hilfe vom Desmo-J (Discrete Event Simulation Modeling in Java) Framework implementiert. Die Flexibilität des Modelles ermöglicht es, sehr unterschiedliche Szenarien zu simulieren, da die Parametrisierung bezüglich der Heiz-und Kühlsystem-Konfiguration, der verwendeten Kontrollstrategie, des gewählten Preismodelles als auch des Energiebedarfs variiert werden kann. Unterschiedliche Szenarien wurden durchgespielt, um die potentielle Ersparnis durch den Einsatz von verschiedenen Preismodellen und Energiekapazitäten der Systemspeicher zu untersuchen. Die Resultate zeigen, dass die grösste Ersparnis durch die Erhöhung der Energiekapazität des Abwärmespeichers und durch den Einsatz eines spotmarkt-bezogenen Preismodelles erzielt werden.

Contents

1. Introduction

In office buildings, heating and cooling systems are at the same time one of the major energy consumers [9] and the systems with the greatest energy saving potential by optimizing their control strategy [3, 5]. A study quantifying the saving potential for an existing office has been presented in [2, 6, 9].

This thesis models an adaptive heating and cooling system in order to assess the potential impact of any possible electricity pricing model and system setting on the electricity costs as well as on the total energy costs for any given energy demand. The model is built using the discrete event simulation paradigm. Different configurations of the original model and assumptions about pricing models and energy demands can be adopted as model parameters.

Section 2 of this document presents the models of the heating and cooling system, the energy demand and supply as well as the control strategy. Within that section an algorithm generating synthetic time series for energy demand and supply has also been developed. Section 3 describes the implementation of the model as a running computer simulation. Section 4 verifies the implementation and validates the model by comparing it with the one in [9] of Rasathurai. While our model has been successfully verified, the comparison with the one of Rasathurai highlighted major discrepancies between the models, but also significant inconsistencies in the model of Rasathurai. In a second part of Section 4 the impact of the accumulator's energy capacity and of different pricing models on the electricity costs and on the total energy costs is investigated in different experiments. Section 5 concludes the thesis with the main findings and offers future works. In the appendix, a user guide explains how to create new experiments and investigate new scenarios with the implemented model.

2. Modeling

All the models presented in this section are an abstraction of the corresponding real world systems, which are simplified in order to allow a clear but still precise representation of a working heating and cooling system. For this reason Section 2.1 presents the simplifications and the assumptions undertaken. After this, we present three models: the model of a heating and cooling system, the model of the energy demand and supply and a model describing the control strategy for the heating and cooling system. The models are described with the use of several parameters, which are assigned to specific values in each experiment.

2.1. Simplifications and Assumptions

In a real world heating and cooling system there is a continuous stream of energy flows between all the components. In our model, in order to simplify this, we decided to map the continuous energy flows to discrete events, where the energy flows change in their intensity. This simplification was also taken for the incoming energy demands, mapping them to a list of discrete events of changes in the demands' intensity.

Across this thesis it is often referred to the term "cold energy". This term does not exist in thermodynamics; what we mean with "cold energy" is the heat energy which is extracted from a given substance in order to cool it [9]. This term is needed in order to simplify the description of cold demands quantities and the fulfilling of such demands.

2.2. Heating and Cooling System

The heating and cooling systems of modern office buildings integrate the production, storage and management of heating and cooling power as well as the supply of domestic hot water. The demand for cooling power is determined by various systems such as cooling ceilings, air handlers and no-frost refrigerators, while the demand for heating power is generated by underflow heatings, air heaters or other kinds of heating systems [9].

The components of a heating and cooling system can be classified in five main categories: the producers of heating and cooling power, the accumulators, the consumers, the energy flows and the controller.

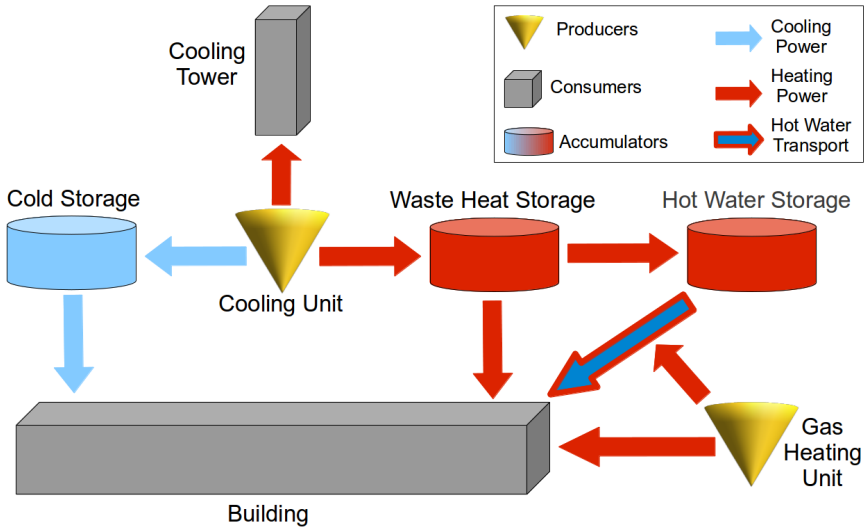

Figure 2.1.: Components of a heating and cooling system

Category	Component	Ingoing Flows	Outgoing Flows	Parameters
Producers	Cooling Unit	-	cooling power, heating power	-min/max power [kW] -coefficient of performance
	Gas Heating Unit	-	heating power	-
Accumulators	Cold Storage	cooling power	cooling power	-water capacity [l] -min/max temperature [°C] -max unload rate [kW]
	Waste Heat Storage	heating power	heating power	-water capacity [l] -min/max temperature [°C] -max unload rate [kW]
	Hot Water Storage	heating power	hot water	-water capacity [l] -min/max temperature [°C] -max unload rate [kW]
Consumers	Building	cooling power, heating power, hot water	-	-
	Cooling Tower	heating power	-	-

Table 2.1.: Characteristics of the components of a heating and cooling system

Figure 2.1 illustrates the components of a heating and cooling system. The gray cuboids represent the heat and the cold energy consumers, which are the building and the cooling tower. The yellow cones are the producers: the cooling unit and the gas heating unit. The accumulators are drawn with a cylinder shape and are used to model the cold, the waste heat and the hot water storages. Finally, the energy flows are represented using arrows. The only component which is not in the figure is the controller. The arrangement of the system regarding its components (the type, the number and the connections between them) is fixed, the only things which can be set in a different way are the parameters that each component offers.

Table 2.1 states the characteristics of each component of the model. The third and the fourth column contain the ingoing respectively outgoing flows for a given component, the last column reports all the parameters, which are going to be set in different ways depending on the experiment. In the following subsections the functionalities and integration of each component are described systematically.

2.2.1. Producers

In a heating and cooling system the producer components are responsible for the production of the heating and cooling power necessary to fulfill the demands.

- **Cooling Unit**: The cooling unit (also called heat pump) transfers, with the use of electricity, the heat from the cold storage to the waste heat storage, cooling the water contained in the cold storage and heating the water of the waste heat storage. The coefficient of performance of a cooling unit describes the ratio between input power and output power and is explained in details here below. If the temperature of the water in the waste heat storage already reached its maximum, the waste heat is transferred to the cooling tower to be dissipated. The cooling unit can work on different power levels within a minimum and a maximum range, which are going to be parameters for the modeling of a cooling unit [2].
- **Gas Heating Unit**: The gas heating unit is used to heat the building and/or the water coming from the hot water storage, in case the waste heat of the cooling unit is not sufficient to do this.

Coefficient of Performance *(COP):*

The coefficient of performance COP of a cooling unit is given by the ratio of output power P_{out} and input power P_{in} [8]:

$$COP = \frac{P_{out}}{P_{in}}$$

The output power is equal to the waste heat H, which can also be expressed as the sum of input power and the produced cold power C [9]:

$$P_{out} = H = P_{in} + C \Rightarrow COP = \frac{H}{P_{in}} = \frac{P_{in}+C}{P_{in}}$$

Thus, the waste heat energy produced for a given time interval t is:

$$H_t = COP * P_{in} * t$$

and the cold energy produced is:

$$C_t = (COP - 1) * P_{in} * t$$

The technical specifications of a cooling unit specify P_{in}, which is actually the nominal power input, and C, which is the nominal cooling capacity of the unit. With these two variables we are able to infer the COP.

Example: a cooling unit that has a nominal power input P_{in} of 200 kW and a nominal cooling capacity C of 600 kW has a COP of:

$$COP = \frac{200\,kW+600\,kW}{200\,kW} = 4$$

therefore, letting the cooling unit work for 6 minutes we have a produced waste heat energy of:

$$H_t = 4 * 200\,kW * 0.1\,h = 80\,kWh$$

and a production of cold energy equal to:

$$C_t = (4 - 1) * 200\,kW * 0.1\,h = 60\,kWh$$

2.2.2. Accumulators

The accumulators act as energy storages. Water is the medium used in order to store energy. Thus, accumulators are water tanks containing a constant volume of water in a constant temperature range, which are parameters that can be set differently for each accumulator. A maximum unload rate is also set for each accumulator as an additional parameter. The energy capacity of an accumulator is given by the water mass contained, the temperature range of the water and the specific heat capacity of water.

Example: if an accumulator contains $1000\,kg$ of water, with a maximum temperature of $50\,°C$ and a minimum temperature of $40\,°C$ and given the

specific heat capacity of water of $4.187\,kJ/\,(kg*^{\circ}\,C)$, then the energy capacity of this accumulator is given by:

$[1000\,kg*(50\,^{\circ}C-40\,^{\circ}C)]*[4.187\,kJ/\,(kg*^{\circ}\,C)]\,/\,[3600\,kJ/kWh]\cong 11.63\,kWh$

An accumulator is said to be "empty" of its heating capacity when it comes to its given minimum temperature, similarly an accumulator is said to be "empty" of its cooling capacity when it gets to its given maximum temperature [2]. In the model we find three different kinds of accumulators:

- **Cold Storage**: The cold storage contains cold water refrigerated by the cooling unit. The cooling unit is responsible to keep the water within the given temperature range. The temperature range chosen affects the temperatures of the water contained in the waste heat and hot water storages, since they just collect the waste heat. The only consumer of cooling power is the building.
- **Waste Heat Storage**: The waste heat storage collects the waste heat deriving from the cold storage and the cooling unit. The waste heat can be used to heat the building or be transferred to the hot water storage.
- **Hot Water Storage**: The hot water storage stores hot water, which is going to be used as domestic hot water. The water is heated by the waste heat storage.

2.2.3. Consumers

There are two consumers of heating and cooling power as well as hot water:

- **Building**: The building shapes the demand for cooling and heating power as well as hot water. The demand, as already mentioned, is created by various systems such as cooling ceilings, air handlers, no-frost refrigerators, underflow heatings, air heaters and other kinds of heating and cooling systems. The daily demands can vary very much depending on the season, the week day, the volume and usage of the building. Section 2.3 analyzes the demand and supply of energy.
- **Cooling Tower**: The cooling tower is a special type of consumer, since it does not have any influence on the demands. The cooling tower consumes the excess of waste heat dissipating it, which happens when the waste heat storage and the hot water storage already achieved their full capacity.

2.2.4. Energy Flows

The energy flows in the model are divided into three categories. There are flows of cooling and heating power implemented through heat exchangers as well as hot water flows [2]. The input flows of electricity and gas are not explicitly modeled, but in the experiments we will keep track of the quantity of electricity and gas used.

2.2.5. Controller

The controller is a piece of software, which manages the cooling unit, the gas heating unit and the energy flows through a set of rules. The set of rules used by the controller can be seen as a management strategy. The input variables for the controller are the current demand and supply of energy, the current state of the system including all the temperatures of the accumulators and all the constant values given by the system such as the efficiency rate and power capacity of the cooling unit and other components [2]. Section 2.5 presents the control strategy model.

2.3. Energy Demand and Supply

In this section we describe the demand and supply of energy. The demand is shaped by the consumers needs for cooling and heating power as well as domestic hot water. The supply of energy is given by electricity and gas, which are converted into cooling or heating power by the cooling unit and the gas heating unit. Since the gas prices are assumed to be constant, our focus in this section lies on the supply of electricity for the cooling unit. In particular, we analyze how the system can adapt to pricing schemes with hourly variation in the price.

2.3.1. Energy Demand

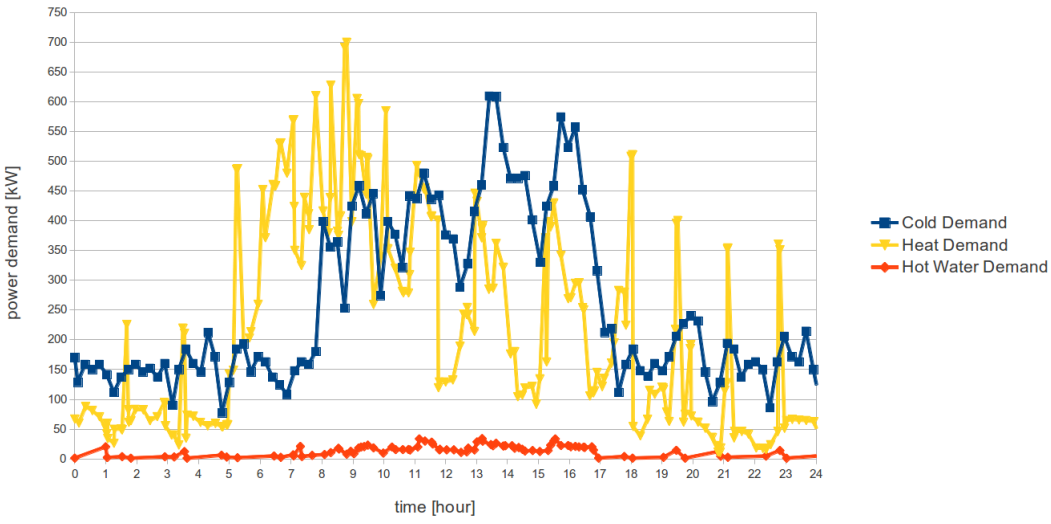

Figure 2.2.: Example of the power demand changes during a day, data from [9]

The power demand at a given point in the time is expressed in kW, the energy demand for a given period of time is given by the product of the power demand and the period length.

Example: a power demand of 10 kW for a time period of 3 hours corresponds to an energy demand during that period of: $10\,kW * 3\,h = 30\,kWh$

Figure 2.2 gives an example of the changes in power demand during the day. On the x-axis the time in hours is stated, on the y-axis the power demand in kW is reported. The demand time series for cold power, heat power and hot water are stated with different colors and shapes. All the time series are subject to variations, which can appear in intervals of a few minutes. Depending on the season, the week day and the weather conditions, the energy demand can vary. From Figure 2.2, for instance, we can notice a peak in the cold demand at 2 p.m., while the heat demand peak is achieved at 9 a.m..

2.3.2. Energy Supply

The supply of energy is composed by electricity and gas. Our focus lies on the price at which energy is delivered. Gas supply is delivered at a constant price. The electricity prices can derive from different pricing models. For instance we can have pricing models with fix rates and others with flexible rates. We want to present one of each: a fix rate pricing model we will call the standard pricing model and a flexible rate pricing model called spot market pricing model.

- **Standard pricing model:** The standard pricing model is very simple and consists of a peak and an off-peak rate. The peak rate is applied from Monday to Saturday from 6 a.m to 10 p.m., for the rest of the time the off-peak rate is applied (i.e. Sunday is entirely an off-peak day).

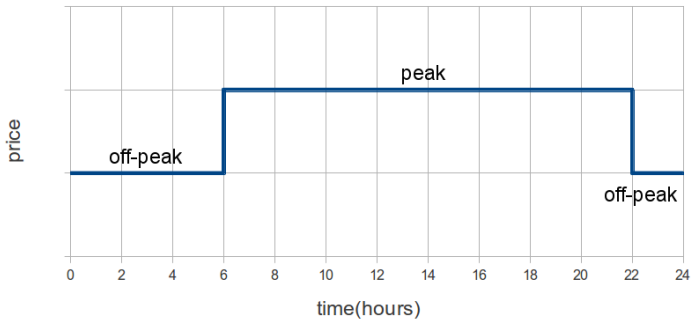

Figure 2.3.: Peak and off-peak time between Monday and Saturday

- **Spot market pricing model:** The spot market pricing model is built by taking the hourly price of electricity at the local energy exchange and adding a profit margin rate of the local electricity distributor. Figure 2.4 shows the spot market price for the Swiss Electricity Index (SWISSIX) negotiated at the European Energy Exchange of Leipzig in Germany for a given day. The price varies during the day achieving a maximum price twice as big as the minimum price. Given the hypothetical condition of a free electricity market, it is rational also for the end consumer to buy electricity when the price is low, store it in form of heating and cooling power, and use it later in the day. In Figure 2.4 the red horizontal line in the graph represents the average price during the peak hours and the black line shows the average price for the entire day. Every different week-day shows a different pattern and also the price spread between the minimum and maximum price varies a lot. Figure 2.5 indeed shows how the peak and the total average prices change from day to day during an entire week. The first day in the graph, the 29.7., was a Monday.

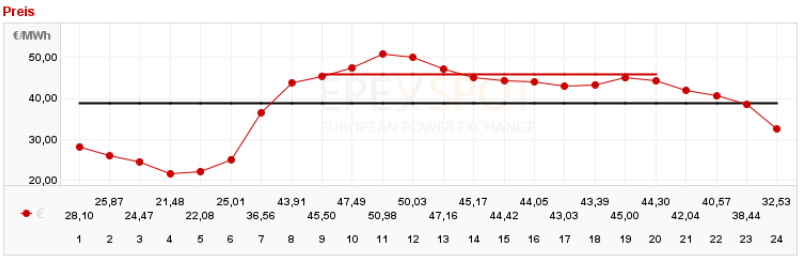

Figure 2.4.: Example of the electricity price during a day for the Swiss Electricity Index (SWISSIX)[1]

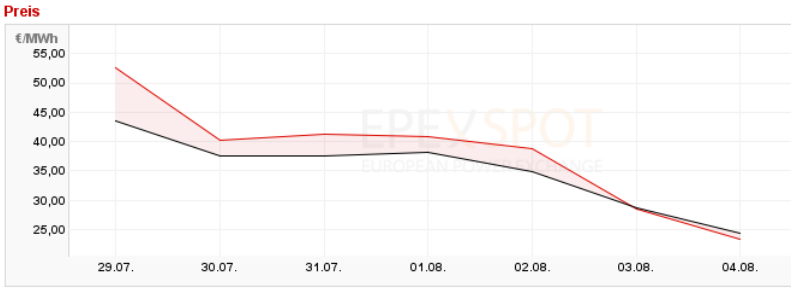

Figure 2.5.: Example of the average electricity price during a week for the Swiss Electricity Index (SWISSIX)[1]

[1] www.eex.com

2.4. Synthetic Time Series Generation

This section presents a method to generate synthetic time series of energy demand and supply based on historical time series. Synthetic data allows to explore the model with a richer set of possible energy demand levels and supply prices. The synthetically generated time series should allow to abstract from historical data, but still keep some properties of them.

For a given set of time series two parameters are given:

- the mean time series, which is the time series generated aggregating all the time series in the set
- the variance of the time series in the set

Algorithmus 2.1 Synthetic Time Series Generator

input : time series t, series variance $varS$, points variance $varP$, percent p

1. $generatedSeries$; //the random generated time series
2. $normalDistS$; //the normal distribution that shifts the entire series up or down
3. $normalDistP$; //the normal distribution that shifts every single point up or down
4. $normalDistS$ = **new** $normalDist(0, varS)$**;** //initialization with $\mu = 0,\ \sigma^2 = varS$
5. $normalDistP$ = **new** $normalDist(0, varP)$; //initialization with $\mu = 0,\ \sigma^2 = varP$
6. $seriesShift$= $normalDistS.sample$; //The shifting value for the entire series
7. **for** $i = 1, ..., t.length$ **do**
8. $\quad generatedSeries\,[i] = t\,[i] * p + seriesShift + normalDistP.sample$;
9. $\quad$ **if** $(generatedSeries\,[i] < 0)$
10. $\quad\quad generatedSeries\,[i] = 0$;
11. $\quad$ **end if**
12. **end for**
13. **return** $generatedSeries$;

Using these two parameters Algorithm 2.1 generates synthetic time series. Starting from a time series, which can be seen as the mean time series, the algorithm generates a new time series in two steps:

1. The entire time series is shifted by a random value, which is generated using a normal distribution. The normal distribution has mean value equal to zero and a variance which is given as input.
2. Every single point in the time series is shifted by a random value, which is generated using a normal distribution. The normal distribution has mean value equal to zero and a variance which is given as input.

With the first step the algorithm generates a time series which shows the same variance as the input one. With the second step the algorithm aims to change the shape of the time series by bringing in a certain additional randomness regarding the values of the single points of the time series. Lines 1 to 3 of the algorithm declare the variables necessary to store the generated series and the normal distribution generators. Lines 4 and 5 initialize the normal distributions with a mean equal to zero and the variances as given in the input. Finally the for-loop of lines 7 to 12 generates a new time series making sure that every value is bigger than zero.

In order to additionally steer the generation of new time series we introduced an input variable p which is a percentage rate between zero and any positive number. Multiplying the input time series with the percentage rate we aim to simulate days that have generally high, respectively low energy demand or electricity prices.

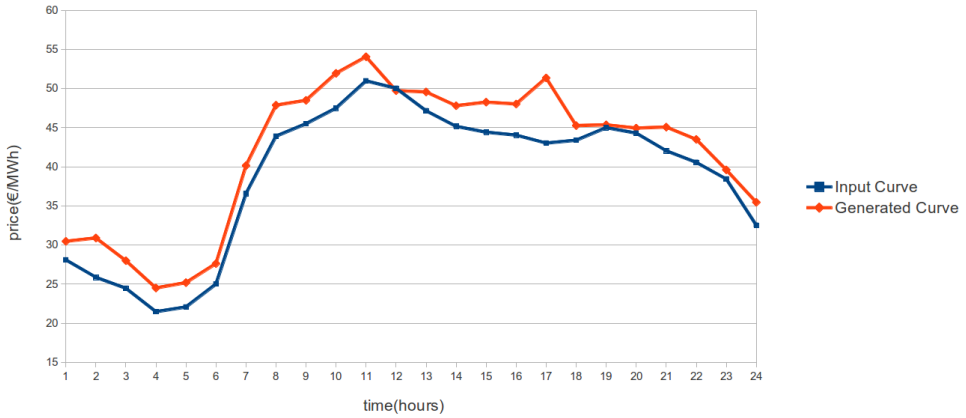

Figure 2.6.: Example of generated time series of the price of electricity

The input values are crucial in order to generate statistically valid data in terms of mean value and variance. As we saw in the sections about energy supply and demand it is very important to distinguish between different seasons and days of the week. For instance, if we want to generate synthetic data for a summer Monday, then we should take all the historical data about Mondays in summer, aggregate them, and give the aggregated time series as input. To aggregate data means, that for each timestamp of the time series an average value among all the available data is taken. The value of the variance used for the normal distribution that shifts the entire time series should be taken as the mean variance given by the distances of the aggregated time series and the historical time series. The variance for the normal distribution that shifts the single points should be chosen as one third of the one used for previous normal distribution.

Figure 2.6 reports an example of a generated time series in red. This was generated using the time series in blue and setting the variance for the shifting of the entire time series to 5 and the variance for the single points shifting to 1.7.

2.5. Control Strategy

The control strategy of a heating and cooling system defines the execution of two major tasks. On the one hand there is the fulfilling of the cold, heat and hot water demand. On the other hand there is the reload of the various accumulators depending on their energy state and on the energy demand and supply.

2.5.1. Fulfilling the Demand

The demands for cold, heat and hot water are fulfilled using different components of the system. In the following subsections we are going to describe the process of fulfilling each demand.

2.5.1.1. Cold Demand

The cold energy demand is fulfilled by the cold storage. If the cold storage does not contain enough energy in order to meet a given demand, then the cooling unit starts working, loading the cold storage with the missing cooling energy. The waste heat given by the loading process of the cold storage is forwarded to the waste heat storage or the cooling towers in case the waste heat storage is already full.

Example: if we have a cold energy demand of 51 kWh during 30 minutes and the cold storage has a capacity of 30 kWh cold energy, the cooling unit is going to produce the remaining 21 kWh. Having a cooling unit with $COP = 4$, the input power for the cooling unit in order to produce the additional 21 kWh cold energy during 30 minutes is:

$$P_{in} = \frac{C_t}{(COP-1)*t} = \frac{21\,kWh}{(4-1)0.5h} = 14\,kW$$

Thus the waste heat energy produced is:

$$H_t = COP * P_{in} * t = 4 * 14\,kW * 0.5h = 28\,kWh$$

2.5.1.2. Heat Demand

The heat energy demand is fulfilled by the waste heat storage and/or the gas heating unit in case the waste heat storage does not contain enough energy in order to meet the incoming demand.

Example: if we have a heat demand of 33 kWh for 15 minutes and the waste heat storage only has a capacity of 12 kWh, then the gas heating unit will supply the remaining 21 kWh. The waste heat storage output power will be set to $\frac{12kWh}{0.25h} = 48kW$.

2.5.1.3. Hot Water Demand

In order to meet the hot water demand the system uses the hot water storage. If the energy present in the hot water storage is too low in order to meet a given demand, then the gas heating unit will supply the remaining energy exactly like in the example for the heat demand.

2.5.2. Loading the Accumulators

The loading of the three accumulators of the system is done differently according to their function. In the following subsections we are going to explain in detail the loading process of each accumulator.

2.5.2.1. Cold Storage

The cold storage is loaded by the cooling unit. The loading process is started in two cases:

- An incoming cold demand cannot be satisfied by the capacity of the cold storage. In this case the cold storage is loaded only with the energy necessary to fulfill the incoming demand and not more.
- The control strategy states that the cold storage has to be reloaded.

For the second case we developed two possible strategies in order to decide when and how much to reload the cold storage. The two strategies are the following:

- **Strategy 1:** The first strategy relies on historical data. Historical data of cold demand have to be entered in the system. Based on them the algorithm makes an approximation of when the cold storage will be empty; after this the time spot with the lowest price, before the cold storage will be empty, is given and the cold storage is reloaded during this time frame. If no historical data exist, then they can be synthetically generated using the algorithm presented in Section 2.4.

- **Strategy 2:** Using the second strategy a loading process is started once the cold storage energy capacity drops a given percentage level of its maximal capacity. The loading process can be shifted for a given maximal amount of hours choosing the time spot where the price is the cheapest. Thus, for this strategy, there are two parameters to be set: the energy capacity percentage level after which the cold storage is going to be reloaded and the maximal amount of hours that this process can be shifted.

2.5.2.2. Waste Heat Storage

The waste heat storage is loaded with the waste heat of the cooling unit. The cooling unit is never turned deliberately on in order to produce waste heat for the waste heat storage.

2.5.2.3. Hot Water Storage

The hot water storage is loaded by the waste heat storage (see Figure 2.1). A loading process is started once the energy capacity of the hot water storage has dropped a given percentage level of its maximal capacity, which is a parameter for the hot water storage.

3. Implementation

This section describes how our model has been implemented as a running application. At first the methodology is illustrated, after this the actual implementation is presented in detail.

3.1. Methodology

Since all the data used in our model are referring to discrete changes of the system, we chose the discrete simulation paradigm as the method to implement and simulate our system. In a discrete simulation it is assumed that [7]: "all system state changes are mapped onto discrete events and assuming that nothing relevant happens between, i.e. that states remain constant and no computations are needed during such intervals". Thus, the discrete simulation paradigm suites very well to simulate our system.

3.2. Implementation as a Running Application

In order to implement our model as a running application we were looking for a tool enabling us to implement a discrete simulation and letting us define the components of our system. In this range of tools we found Desmo-J (Discrete Event Simulation Modeling in Java) [1] as the tool meeting our requirements as well as being very user friendly.

Desmo-J supports the implementation of process-oriented, event-oriented as well as a mix of the two modeling styles [1]. Since most of the events in our system can be mapped to a single entity of the system, it has been convenient to choose the process oriented paradigm . Here is a definition of process-oriented modeling style:

"A process-based model design takes an object-oriented perspective, identifying the relevant entities, their properties and behaviors. All activities "owned" by an entity are grouped into a process, which can then be viewed as that entity's "life cycle". Model time passes during such active entities phases; i.e. whenever a time delay is encountered. Lifecyles are described from the modeled entities' own perspectives. This includes all relevant activities, the sequence in which they occur, and their relationship to other entities in the model." [2]

[2] http://desmoj.sourceforge.net/tutorial/processes/design0.html

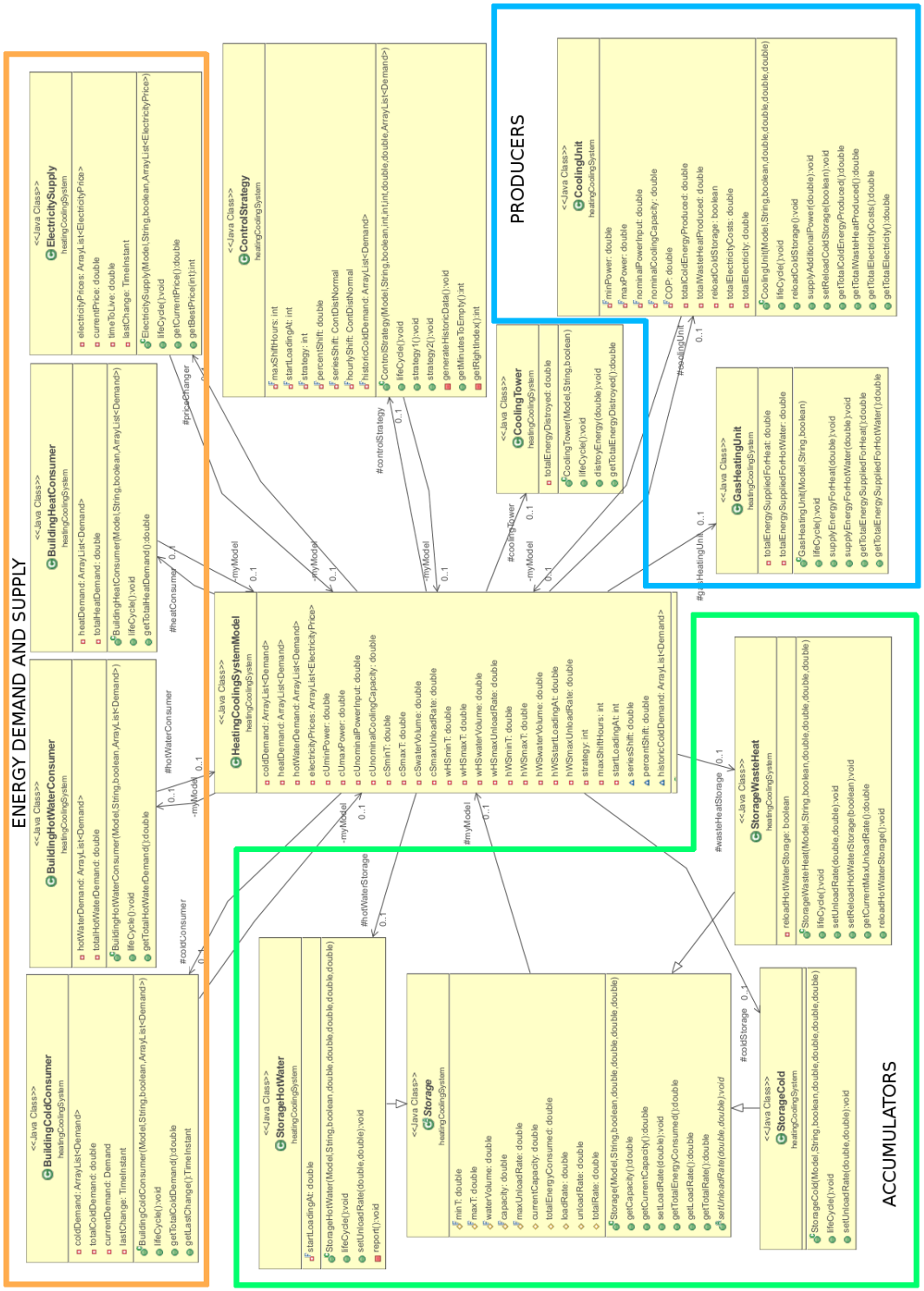

Figure 3.1.: UML diagram of the implemented model

As the definition of process-oriented modeling style states, for each entity of the system a method called *lifeCycle()* has to be implemented. The *lifeCycle()* method describes the behavior of the implemented entity during the simulation. All the lifeCycle() methods for all the implemented classes are going to be analyzed in the following subsections.

Figure 3.1 reports the UML diagram of the implemented entities as Java classes and their attributes and methods as well as the relationships between them. The HeatingCoolingSystemModel class is the central class of the model, which instantiates all the other classes and maintains a reference to them. In return, all the instantiated classes save a reference to the HeatingCoolingSystemModel class, in order to be able to access all other classes in the model.

Three categories of classes are grouped with different colors: the classes modeling the **energy demand and supply** in orange, the classes representing the **accumulators** in green and the classes describing the **producers** in blue.

3.2.1. Accumulators

All the accumulators, the waste heat storage, the cold storage and the hot water storage inherit from the abstract class *Storage*. This abstract class stores all the parameters for the accumulators presented in Table 2.1 plus some variables keeping track of the current capacity of the storage, the total energy consumed, the current load and unload rate as well as the total rate which is the load rate minus the unload rate.

3.2.1.1. Cold Storage

The central method of each class is the *lifeCycle()* method, which contains a loop that runs in intervals of one minute. An interval of one minute has been chosen, since all the changes in the system are mapped to discrete events with a maximum frequency of one minute between each others. Algorithm B.1 in the appendix reports this method for the cold storage. Each minute of the simulation time the algorithm checks the total rate (the difference between load and unload rate) of the storage. In case the rate is bigger or equal to zero nothing happens beside raising the total energy consumed. When the total rate is negative however, it is checked if the storage still has enough energy in order to fulfill the incoming demand or if it has to ask the cooling unit for an additional load. Finally, the current capacity of the storage is updated.

3.2.1.2. Waste Heat Storage

Algorithm B.2 contains the *lifeCycle()* method for the waste heat storage. As for the cold storage the algorithm checks if the total rate (the difference

between load and unload rate) of the waste heat storage is positive or negative. In case the total rate is bigger or equal to zero the total energy consumed as well as the current capacity of the storage are updated; in case the total rate leads to a fully loaded storage then the load in excess is forwarded to the cooling tower. In a second step it is checked whether the hot water storage needs to be reloaded or not; if so, the maximum possible load rate is applied to the hot water storage.

In case the total rate is negative the algorithm checks if there is enough energy to fulfill the incoming demand; if not, some additional power is requested from the gas heating unit. If instead the current energy level is enough to fulfill the incoming heating demand, it is checked if the hot water storage needs to be reloaded.

3.2.1.3. Hot Water Storage

As for the two other storages, also the *lifeCycle()* method of the hot water storages distinguishes between a positive and a negative total rate. In case the energy level is too low to satisfy the incoming demand then an additional power supply from the gas heating unit is requested. As an additional action, at the end of each loop, it is checked if the capacity of the hot water storage has dropped a given level; if so, a reload request to the waste heat storage is sent (see Algorithm B.3).

3.2.2. Producers

3.2.2.1. Cooling Unit

The cooling unit class contains all the parameters described in Table 2.1 as well as four additional variables keeping track of the total cold energy produced, the total waste heat produced, the total electricity used and the total costs for the used electricity.

The *lifeCycle()* method of the cooling unit controls every minutes if the cold storage needs to be reloaded (which is a decision taken by the *ControlStrategy* class). If this is the case, the cold storage is reloaded at its maximum possible rate and the waste heat resulting from this process is forwarded to the waste heat storage. When the cold storage cannot be loaded further more the reloading process is stopped (see Algorithm B.4).

3.2.2.2. Gas Heating Unit

The gas heating unit has an empty *lifeCycle()* method, since the only job of the gas heating class is to keep track of the gas supplied for pure heating purposes and to heat domestic hot water.

3.2.3. Energy Demand and Supply

In order to better support the management of energy demand and supply data, two new data types have been created: the *Demand* and the *ElectricityPrice* data types. The raw data of Table 3.1 are read, transformed into Java-objects of the given data type and stored into an array list of *Demands* respectively *ElectricityPrices*. In the first column of the raw demands data we can find the time duration in minutes for the incoming power demand, in the second column the power demand in kW is stated and in the third column the resulting energy in kWh is listed.

(a) Energy demand

duration [minutes]	**power [kW]**	**energy [kWh]**
60	1.1415	1.1415
3	20.142	1.0071
29	2.2104	1.0684

(b) Electricity price

duration [minutes]	**price [CHF/MWh]**
60	126.054
32	107.964
44	101.826

Table 3.1.: Raw data of energy demand and supply

The raw data of the demands can be of either cold, heat or hot water, but the data structure is the same for all of them. For the raw data containing the electricity prices we can read the time duration in minutes from the first column and the actual price in swiss francs per MWh from the second column. After the array lists are created they are forwarded as parameters to the constructors of the energy demand and supply classes, which are described below.

3.2.3.1. Energy Demand

After the raw data of the demands are transformed into array lists of Java-objects, an object at the time is removed from the array list and the data is read. Algorithm B.5 gives an example with the reading of cold energy demands: a new cold demand is stored into the *currentDemand* object; after this the new cold demand is forwarded to the cold storage; finally the total cold energy demand is updated and the while-loop holds a break of the duration of the demand just read before reading a new one.

3.2.3.2. Energy Supply

Similarly as for the energy demands, also on the energy supply side a new electricity price is read and then as much time is waited (before reading a new one) as the current price lasts (See Algorithm B.6).

3.2.4. Control Strategy

The control strategy class controls the loading of the cold storage. For this task we implemented two different strategies as described in Section 2.5.2.1.

3.2.4.1. Strategy 1

Algorithm B.7 illustrates the first strategy (the one relying on historical data). At first the algorithm, if needed, generates historical data from an input time series using the method described in Section 2.4 and implemented in Algorithm B.8. In a second step the algorithm of Strategy 1, using the historical data, estimates the minutes remaining before the water storage will be empty. As a third step the algorithm sets when to reload the cold storage, looking for the best price during the time interval before the cold storage will be empty. The entire process is repeated every minute in a loop.

3.2.4.2. Strategy 2

The second strategy is implemented as in Algorithm B.9. As described in Section 2.5.2.1 this strategy is based on two parameters. Every minute the algorithm controls that the current energy capacity of the cold storage has dropped a given energy capacity level (first parameter); if this is the case then the best time frame in order to reload the cold storage is chosen from a given time interval (second parameter) and the cold storage is reloaded then. The best time frame corresponds to the cheapest price for electricity.

4. Simulation and Results Evaluation

This chapter has two goals. The first one is to verify our implementation and to validate our model using different experiments as described in Section 4.1. Section 4.2 investigates the impact of the energy capacity of the accumulators and of different pricing models on the electricity costs as well as on the total costs; two pricing models are used and only one or few parameters are changed at once. This section is intended to demonstrate the flexibility of our model and its easy configuration capacity that enables the investigation of a large number of different scenarios.

4.1. Implementation Verification and Model Validation

In a first step we verify our implementation looking if it corresponds with the model we wanted to implement. Section 4.1.1 shows the results of the verification of the implementation. In a second step our model is validated comparing it with the one of [9] as reported in Section 4.1.2.

4.1.1. Verification of the Implementation

The implementation is verified by setting the system to a given configuration (Section 4.1.1.1). Using such configuration, four different experiments under a given control strategy (Section 4.1.1.3) and different energy demand and supply time series (Section 4.1.1.2) are run. Finally, it is verified that the output values of the simulation coincide with the expected output values (Section 4.1.1.4).

4.1.1.1. Heating and Cooling System Configuration

Figure 4.1 shows the setting of the heating and cooling system used to verify the implementation.

On each accumulator the water volume contained, its temperature range and the maximum unload rate are stated. On the cooling unit the minimal and maximal working powers are reported. The coefficient of performance of the cooling unit is given by the formula of Section 2.2.1. The nominal power input for the cooling unit is 609 kW and the nominal cooling capacity is 2706 kW, therefore we have a coefficient of performance of:

$$COP = \frac{609kW + 2706\,kW}{609\,kW} = 5.443349754$$

The hot water storage has been chosen to be reloaded after it comes to less than 50% of its energy level.

This configuration is inspired from the configuration of the office building of IBM Switzerland in Zürich-Altstetten as described in [9]. Given this configuration the next section presents the energy demand and supply data used to verify the implementation.

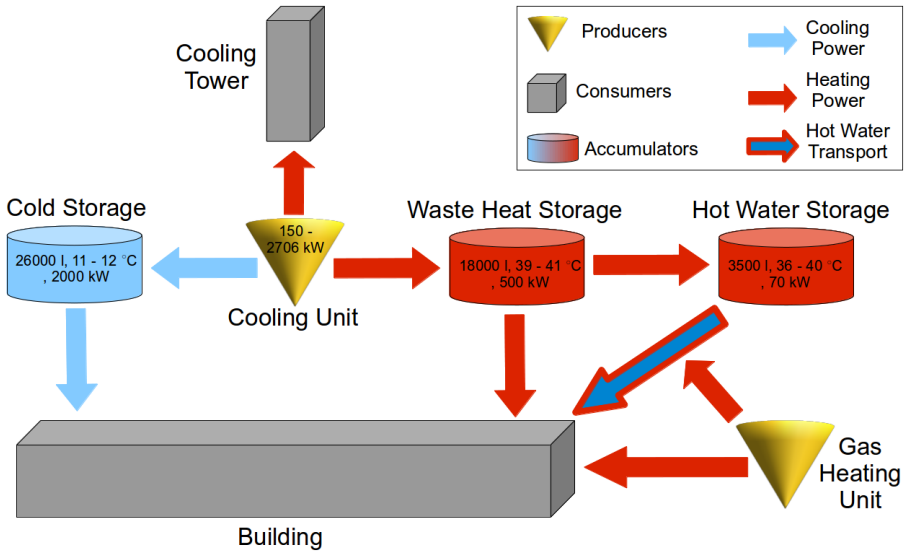

Figure 4.1.: Configuration of the system to verify the implementation

4.1.1.2. Energy Demand and Supply Data

On the demand side we use two different data sets. The data sets are taken from [9] and refer to the demands for heat, cold and domestic hot water in the months of December 2011 and June 2012 derived from a real heating and cooling system. The total energy demands are reported in Table 4.1. The heat and the domestic hot water energy demands are high during the month of December since in Switzerland it is winter. The month of June is characterized by a high cold energy demand and low heat and domestic hot water demands.

	Heat	Cold	Domestic Hot Water
December	175018.39	143863	5244.98
June	42275	306421	2044

Table 4.1.: Total energy demands in kWh

On the energy supply side we investigate two different pricing models (also taken from [9]): the **fix prices** pricing model, that is the one actually in use for the city of Zurich and the **spot market prices** pricing model, that instead is the construction of a potential future pricing model. Here is a detailed description of the two pricing models:

- **Fix prices**: For the fix prices pricing model a peak and an off-peak rate is applied depending on the week day and the time. A peak rate of 162 CHF/MWh is applied from Monday to Saturday from 6 a.m to 10 p.m., for the rest of the time an off-peak rate of 86.4 CHF/MWh is applied. Therefore the average price of such model is 130.8 CHF/MWh.

- **Spot market prices**: The spot market prices for our experiment are based on the SWISSX index of the energy exchange of Leipzig in Germany. A conversion in CHF is done using an exchange rate of 1.2 CHF/EUR as fixed by the Swiss National Bank in the summer of 2011 as the minimum exchange rate between the Swiss Franc and the Euro. Moreover, a profit margin for the local energy distributor is added to build the final spot prices for the consumer. Such a profit margin is calculated by dividing the average fix price applied by the local distributor by the average price on the market. In our case we have an average spot market price of 71.047 CHF/MWh in December and 47.171 CHF/MWh in June, which gives a yearly average spot market price of about 59.109 CHF/MWh; dividing 130.8 (the average fix price) by 59.109 we obtain a profit margin for the local distributor of about 2.21 times the market price. Thus a total conversion rate of 1.2*2.21= 2.652 has to be applied to the spot market prices of the energy exchange of Leipzig [4, 9]. Therefore the average spot market prices for the end-consumers are 157.014 CHF/MWh in December and 104.248 CHF/MWh in June.

For each month (December and June) both pricing model are investigated. Thus we have a total of four different experiments.

4.1.1.3. Control Strategy

The control strategy chosen for all the experiments is reported in Section 2.5 and more specifically, regarding the strategy for the loading of the cold storage, we chose to use Strategy 1 as described in the corresponding Section 2.5.2.1. This means that historical data are used to steer the loading of the cold storage. In order to keep the system as simple as possible we decided to assume that the historical data state with no error the future energy demands.

4.1.1.4. Results

In order to verify the implementation of our model we are going to look at four crucial output values: the cold energy produced, the electricity consumed, the waste heat produced and the electricity costs. For each of these output values a value, deriving from a right implementation of the model, is calculated , we will further refer to such a value as the "expected value". The expected values will be then compared with the simulation values. The expected values are calculated as following:

- **Cold energy produced**: follows the total cold energy demand as reported in Table 4.1
- **Electricity consumed**: is equal to the cold energy produced divided by (COP-1) as explained in Section 2.2.1
- **Waste heat produced**: is, as reported in Section 2.2.1, equal to the electricity consumed times the COP
- **Electricity costs**: are calculated taking the electricity consumed multiplied by the average prices reported in the previous section

Table 4.2 confronts the expected and the simulated output values. The **cold energy produced** shows simulation values extremely close to the expected ones; the small differences are given by the fact that the cold storage remains partially full at the end of the experiments. In fact, the differences do not go over 30.24 kWh which is the maximum energy capacity of the cold water storage. The same situation and arguments also apply for the **electricity consumed** and the **waste heat produced**.

		December		**June**	
		Fix Prices	**Spot Market Prices**	**Fix Prices**	**Spot Market Prices**
Cold Energy Produced in kWh	**Expected**	143863	143863	306421	306421
	Simulated	143888	143887	306432	306432
Electricity Consumed in kWh	**Expected**	32377	32377	68962	68962
	Simulated	32383	32383	68964	68964
Waste Heat Produced in kWh	**Expected**	176239	176239	375383	375383
	Simulated	176270	176270	375396	375395
Electricity Costs in CHF	**Expected**	4235	5084	9020	7189
	Simulated	4479	5289	9542	8164

Table 4.2.: Expected results and simulated output values

The expected **electricity costs** are smaller than the simulation output values. This derives from the fact that taking the average prices to estimate the electricity costs assumes that the electricity is bought uniformly during the course of the day. Instead, we actually have the peaks in the demand coinciding with the peaks in the prices. Thus, the differences between expected and simulated electricity costs are not to be considered as an error in the model implementation.

Given these results we can assume that our implementation coincides with the model we wanted to implement. Thus, the implemented model can be used for further analysis.

4.1.2. Comparison with Another Model

In order to validate our model, we compare it with the one implemented by [9]. In the work of Rasathurai three different scenarios are investigated. All experiments are run with energy demands time series derived from original data, which are the same as the ones described in the implementation verification section (Section 4.1.1.2). Using these fixed assumptions, the rest of the setting for each of the three scenarios is varied as following:

- **Experiment 1 (current strategies):** This experiment uses the fix pricing model as described in Section 4.1.1.2. The heating and cooling system configuration is the same as reported in Section 4.1.1.1. The only difference between this experiment and the one for testing the implementation is given by the control strategy. The control strategy in this case imposes that the cold storage is reloaded every time its energy capacity drops under 75% of the maximum capacity.
- **Experiment 2 (new control strategies):** The second experiment examines the model under the assumption of a different control strategy. It extends the temperature range of the cold storage moving the minimal temperature to 5 degrees Celsius instead of 11 and bringing the maximal temperature of the hot water storage to 41 degrees Celsius instead of 40. Furthermore, this experiment applies a different logic to determine when the cold storage is reloaded. The control strategy states that the cold storage is already reloaded when its energy content drops under 95% of the maximum. In addition, the loading process can be shifted 6 hours in December and 12 hours in June, if better prices are available in these time frames.
- **Experiment 3 (dynamic electricity pricing):** The last experiment has exactly the same setting as the second one except for two things. The first is that the reloading process can be shifted 12 hours in December and 24 in June instead of 6 and 12 hours. The second difference is the use of the spot market prices pricing model as described in Section 4.1.1.2, with the difference that the profit margin of the local energy

distributor is set to 1.5 instead of 2.21. There is apparently no reason for this choice; indeed the profit margin in [9] is calculated the same way as in this thesis, however in the simulation a profit margin of 1.5 is used. Thus, in order to achieve a fair comparison, we will also set the profit margin of the local energy distributor to 1.5.

		December		June	
		Rasathurai	**This thesis**	**Rasathurai**	**This thesis**
Exp. 1: current strategies	**Consumption [kWh]**	45789	32378	101485	68962
	Costs [CHF]	24530	4483	53372	9543
Exp. 2: new control strategies	**Consumption [kWh]**	37054	32395	76979	68962
	Costs [CHF]	22056	4457	44147	9539
Exp. 3: dynamic electricity pricing	**Consumption [kWh]**	31770	32395	76138	68962
	Costs [CHF]	21937	3593	36786	5543

Table 4.3.: Comparison with Rasathurai

The model of Rasathurai and the one of this thesis are compared by the quantity of electricity used and the costs for it. Table 4.3 reports these two output values for the simulation with both models.

The differences in the output values are large. The consumptions of electricity in the "current strategies" and "new control strategies" experiments are much bigger with Rasathurai's model than with ours, reaching 47% more consume in June of the first experiment. In the "dynamic electricity pricing" experiment, using our model, we have a higher electricity consumption in December, but not in June. The electricity costs are always much bigger in Rasathurai's results, constantly deviating with at least factor four higher costs.

These disparities cannot be explained by invoking some little rounding or precision discrepancy, thus the definition of the models itself has to play a role. For instance, regarding the consumptions of electricity for a given month in Rasathurai, it is not possible that the consumptions change so much from one experiment to the other, since the cold demand, as well as the efficiency of the cooling unit, is always the same. The only plausible changes are in the range of the capacity of the cold storage, which can be empty or full at the end of an experiment. But since the cold storage has a maximal capacity of 30.24 kWh in the first experiment and of 211 kWh in the second and third experiment, the differences cannot be explained with it. Furthermore, the costs resulting from the electricity consumption in Rasathurai lead to an average electricity price of 530 CHF/MWh in the first experiment, 584 CHF/MWh in the second one and 586 CHF/MWh in the third one. These average prices are completely

incompatible with the given average price of 130.8 CHF/MWh for the fix prices pricing model used in the first two experiments. The third experiment makes use of the spot market prices pricing model with a profit margin of 1.5 for the local distributor, which leads to an average price of 88.64 CHF/MWh, that in turn is much smaller than the average price of 586 CHF/MWh of the third experiment of Rasathurai.

The comparison with the model of Rasathurai has not led to the expected results and we could not assess the validity of our model. However, we were able to catch some significant inconsistencies in the model proposed by [9], so that our model is not invalidated by this comparison.

4.2. Impact of the Accumulators' Energy Capacity and of Different Pricing Models

In this section we analyze the impact of the energy capacity of the accumulators as well as the use of different pricing models on the electricity costs and on the total energy costs which include the electricity costs plus the costs for natural gas. The price for natural gas has been taken as 0.1 CHF per kWh. This is approximately the price for natural gas offered by the natural gas provider of the city of Zurich (Erdgas Zürich[3]) during 2011 and 2012.

In order to vary the energy capacity of the accumulators we chose to change their water volume capacity. More water capacity means more energy capacity as shown in Section 2.2.2.

Multiplier	**Cold Storage (26000 liters)**	**Waste Heat Storage (18000 liters)**
0.1	2600	1800
1	26000	18000
5	130000	90000
10	260000	180000
20	520000	360000
30	780000	540000
40	1040000	720000
50	1300000	900000
60	1560000	1080000
70	1820000	1260000
80	2080000	1440000
90	2340000	1620000
100	2600000	1800000
200	5200000	3600000

Table 4.4.: Water capacity of the accumulators during the experiments

[3] http://www.erdgaszuerich.ch

All experiments are run using exactly the same configuration as the one presented in Section 4.1.1 regarding the heating and cooling system parametrization, the control strategy as well as the energy demand and supply time series, whereby only the water capacity of the accumulators is going to vary. This means that, as in the implementation verification section, each experiment will be run using four different settings and thus, four different sets of output values will be available. The different settings are given by the use of two different pricing models and two different data sets regarding the energy demands as described in Section 4.1.1.

Changes in the water volume capacity of the accumulators are made in three distinct experiments. In a first experiment we vary the water volume capacity of the cold storage only. In a second experiment we vary the water volume of the waste heat storage alone. And in the third experiment we vary the water volume capacity of both the cold and the waste heat storages at the same time. The modifications of the water volume capacity are made taking the initial setting of 26000 liters for the cold storage and 18000 liters for the waste heat storage and multiplying them with a factor called multiplier that goes from 0.1 to 200. Table 4.4 shows how the water volume capacity of the accumulators changes through the experiments: the initial values times the multiplier gives the water volume capacity in liters.

4.2.1. Cold Storage Only

In this first experiment the cold storage is set to contain an increasing water volume that goes from 2600 to 5.2 million of liters as stated in Table 4.4.

Figure 4.2 reports two graphs. On the left side of the figure the results of the experiment that is run with the December data set for the energy demands are stated. On the right side the output values of the same experiment using the data set of June are reported. On the x-axis the multiplier factor is stated, while on the y-axis the costs in CHF can be read. In each graph four lines are drawn. In light and dark blue the costs using the fix prices pricing model are stated. The light blue line reports the electricity costs, while the dark blue line states the total energy costs. On the light and dark red lines the costs using the spot market prices pricing model can be read. The light red line is used for the electricity costs, while the dark red line is intended for the total energy costs.

One of the goals of this thesis is to assess the impact of the electricity pricing models on the electricity costs as well as on the total energy costs. In order to do this we have to compare the two light lines with each other and the two dark lines together. Comparing the light lines shows the impact of the pricing models on the electricity costs. With the December data set the electricity costs are lower using the fix prices pricing model, with a constant costs saving

rate of 15% on the spot market prices. In June instead, the electricity costs can be minimized with the use of the spot market prices, which constantly save 15% of the costs in comparison to the fix prices. This makes sense, since the distributor profit margin of 2.21 for the spot market prices calculated in Section 4.1.1.2 is a yearly average. Thus, depending on the season, you can be better off with the fix prices or with the spot market prices. In our case, the spot market prices are high in December and low in June compared to the fix prices.

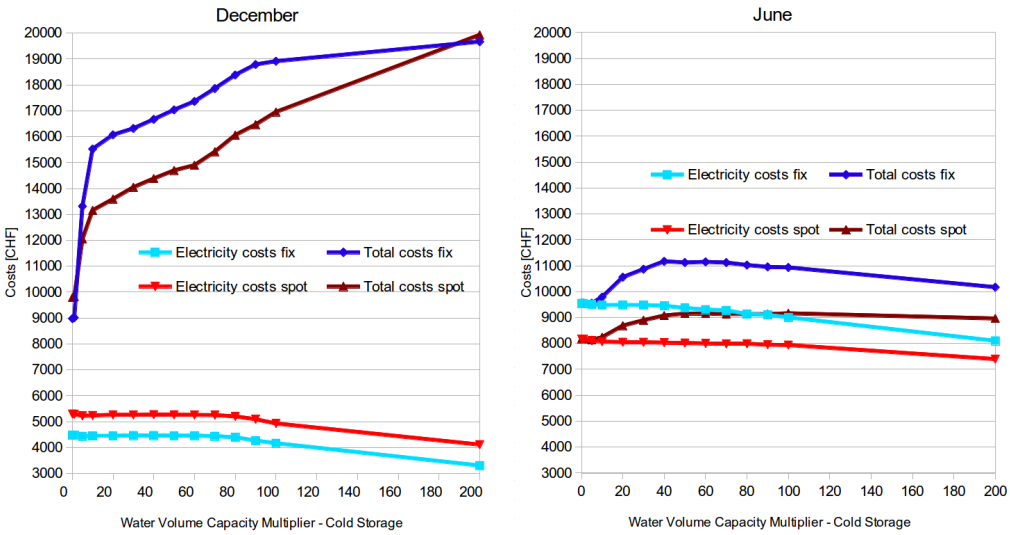

Figure 4.2.: Costs evolution with increasing water volume capacity of the cold storage

However, in absolute values, the saving potential for the electricity costs in June is twice as much as the one in December with 1400 CHF against 700 CHF. This is given by the fact that the cold energy demand in June is much higher than the one in December. Thus, all in all, the spot market prices pricing model is to be preferred to the fix prices pricing model. Regarding the impact of the pricing models on the total costs we can say that for the month of December the total costs using the spot prices pricing model show better results for almost the entire experiment. In June instead, the two pricing models show the same impact on the total costs.

The impact of the water volume capacity on the electricity costs is to be considered as low, since highly decreasing electricity costs are observed starting just at a greater water volume capacity than 2 million of liters (a multiplier of 80 in the graphs), which is in reality already an unimplementable value for a cold storage. On the side of the total energy costs however, a negative

impact of the increasing water volume capacity is observed. This effect is created by the fact that more and more electricity can and is bought on single relatively cheap price time spots, triggering two consequences at the same time: an increased quantity of waste heat destroyed by the cooling tower, since the waste heat storage cannot store large quantities of waste heat, and, as a second effect, an increased future consume of gas, since part of the waste heat has been destroyed. We can say that the heating section of the heating and cooling system suffers from the repeated production of big quantities of waste heat in a short period of time, preferring instead a linear production of waste heat over time.

In general, from this experiment we can observe that almost no saving potential is given by buying electricity in the off-peak time spots. There are two connected reasons for that: on the one hand the peaks in the cold energy demand correspond to the peaks in the prices, so, in order to save money, more electricity should be bought in advance (in the off-peak time spots), but on the other hand, in order to do this, we should be capable of storing enormous quantities of cold energy, having cold storages with a water volume capacity of millions of liters (as shown in the graph). The saving potential of using spot market prices, as well mainly derives from the average lower prices in the summer months, where also the highest demands for cold energy are registered.

4.2.2. Waste Heat Storage Only

In this experiment the water capacity of the waste heat storage is increased like we did for the cold water storage. Increasing the capacity of the waste heat storage means that more and more waste heat energy can be stored and used to fulfill the heat and the hot water energy demands instead of being dissipated. Therefore, we expect decreasing total costs and stable electricity costs, since the cold storage is set to contain a constant standard value of 26000 liters of water. The waste heat storage water content is going to vary using the same mechanism as in the previous experiment. Table 4.4 states the water capacities that the waste heat storage is going to acquire by multiplying the standard content of 18000 liters with an increasing multiplier factor.

Figure 4.3 reports the evolution of the electricity costs and the total energy costs with an increased capacity of the waste heat storage.

As in the previous experiment a constant electricity costs saving of 15% in December is observed using the fix prices, while in June the same saving rate is achieved using the spot market prices. Furthermore, in this experiment, the pricing models show the same impact on the total costs in both months.

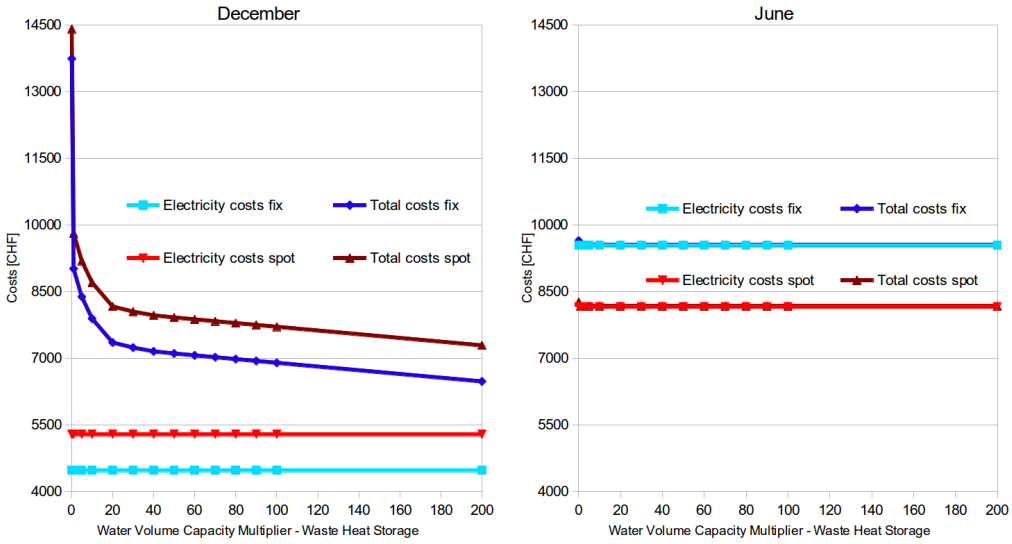

Figure 4.3.: Costs evolution with increasing water volume capacity of the waste heat storage

As expected, the increasing water capacity for the waste heat storage has no impact on the electricity costs, but does have an effect on the total costs. Indeed, the total costs decrease with the increasing capacity of storing bigger quantities of waste heat. This effect is particularly visible in December, since the demand for heat and hot water energy is very high during the winter months. In December the total costs decrease as an inverse logarithmic function, while in June the total costs almost equal the electricity costs, meaning that nearly the whole heat and hot water energy demand is fulfilled with the use of waste heat.

4.2.3. Cold and Waste Heat Storages Together

As a last experiment the water volume of both the cold and the waste heat storages vary at the same time, taking values as shown in Table 4.4. Figure 4.4 states the results.

Varying at the same time the water volume of both the cold and the waste heat storage shows nearly the same results as increasing the water volume of the waste heat storage only. The only difference are the slightly decreasing electricity costs triggered by the higher capacity of the cold storage (as already observed in the experiment with the cold storage only).

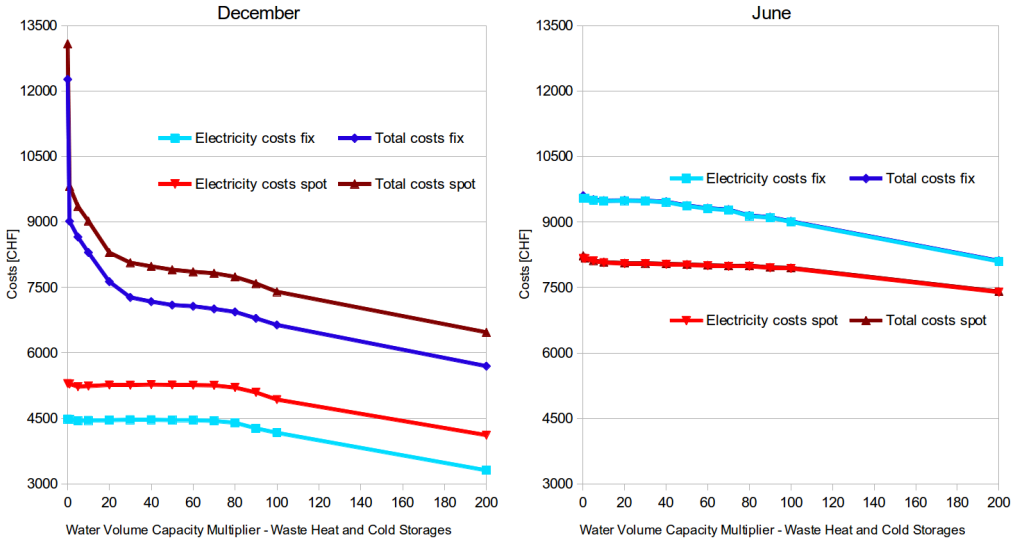

Figure 4.4.: Costs evolution with simultaneously increasing water volume capacity of the cold and waste heat storages

5. Conclusions and Future Work

The implemented heating and cooling system model has been proven to be very flexible and to permit to simulate very different scenarios thanks to its simple configuration capacity. The implementation has been successfully verified. The comparison with the model in [9] could not validate our model. However, since the model implemented by Rasathurai showed some significant inconsistencies, it is rather unlikely that our model is the cause of the discrepancies between the models.

The experiments investigating the impact of the energy capacity of the accumulators and of different pricing models on the electricity costs and on the total costs revealed very interesting results, which are summarized hereafter:

- The use of different pricing models showed that the fix prices pricing model is to be preferred in December, while the spot market prices pricing model shows better performances in June. The saving potential amounts to 15% on both the electricity and the total costs for both months using the better pricing model. But looking at the saving amounts in absolute values, the spot market prices pricing model should be preferred to the fix prices pricing model since the saving potential in June is twice as much as the one in December with 1400 CHF against 700 CHF. The savings are to be attributed to differences in the average electricity price for a given month.
- Increasing the energy capacity of the cold storage alone has a small effect on the electricity costs decreasing them, but a big impact on the total costs, increasing them. The repeated production of big quantities of waste heat in short time periods causes a raising of the total costs, since the waste heat is dissipated by the cooling tower when it cannot be stored anymore. A linear production of waste heat over time is therefore preferred in order to lower the total costs.
- Increasing the energy capacity of the waste heat storage has a big impact on the total costs, decreasing them. In December the reduction of the total costs is highly visible, in June instead just a slight decrease is observed. During this experiment the electricity costs remain constant.
- Increasing the energy capacity of the cold and the waste heat storages at the same time shows nearly the same results as increasing the energy capacity of the waste heat storage only. In conclusion we can say that the best results are given by increasing the waste heat storage energy capacity and by using the spot market prices pricing model.

Although the implemented model fulfilled the goals of this thesis, some improvements and extensions could be undertaken. For instance, it would be reasonable to introduce other control strategies, which are optimized by means of the total costs, or to add a graphical user interface in order to control and play with the system. Finally, the implemented model could be used for further experiments, investigating new scenarios.

Bibliography

[1] **Bornhöft N. A., Page B. and Schutt H. (2010)**: *Modelling of innovative Technologies for Container Terminal Yard Stacking System using an Object-Oriented Simulation Framework.* A.G. Bruzzone, et.al. (eds.), Proc. The International Workshop on Applied Modelling and Simulation, pp. 310–315.

[2] **Bornhöft N. A., Rasathurai S. and Hilty L. M. (2013)**: *Simulation der Smart-Grid-Integration eines modernen Bürogebäudes am Beispiel von IBM-Schweiz.* In: Marx-Gómez, J., Lang, C. V. and Wohlgemuth, V. (Eds.): IT-gestütztes Ressourcen- und Energiemanagement, Konferenz zu den 5. Buis Tagen, Berlin, Springer.

[3] **Erdmann L. and Hilty L. M. (2010)**: *Scenario Analysis: Exploring the Macroeconomic Impacts of Information and Communication Technologies on Greenhouse Gas Emissions.* Journal of Industrial Ecology 14(5), pp. 826-843.

[4] **Fox-Penner P. (2010)**: *SMART power. Climate Change, the Smart Grid, and the Future of Electric Utilities.* Island Press.

[5] **Hilty L. M., Arnfalk P., Erdmann L., Goodman J., Lehmann M. and Wäger P. (2006)**: *The Relevance of Information and Communication Technologies for Environmental Sustainability – A Prospective Simulation Study.* Environmental Modelling & Software 21(11), pp. 1618-1629.

[6] **Hilty L. M. and Bornhöft N. A. (2013)**: *Smart Grid Integration of an Existing Office Building: Modelling and Simulation of Adaptation Strategies.* Shaker Verlag.

[7] **Page B. and Kreutzer W. (2005)**: *The Java Simulation Handbook. Simulating Discrete Event Systems with UML and Java.* Shaker Verlag.

[8] **Petchers N. (2003)**: *Combined Heating, Cooling and Power Handbook. Technologies and Applications: an Integrated Approach to Energy Resource Optimization.* The Fairmont Press Inc.

[9] **Rasathurai S. (2012)**: *Improving on the Electricity Costs of Office Buildings by Optimal Smart Grid Integration.* Master's thesis, University of Zurich.

A. User Guide

This user guide explains how to run new experiments using the implemented heating and cooling system simulation tool.

After having imported the project into the Eclipse Integrated Development Environment[4] open the "Experiments" package and the contained class "RunExperiment.java". The class is illustrated in Algorithm A.1.

The "RunExperiment" class instantiates a new heating and cooling system model and a new experiment. The model and the experiment are then connected together and the experiment is run. In order to instantiate a new model we first need to set three things: the energy demand and supply time series (lines 5-9 in the algorithm), the parameters for the heating and cooling system (lines 12-27) and the parameters for the control strategy (lines 30-33). The energy demand and supply times series are to be given as .csv files being formatted as presented in Section 3.2.3. The parameters for the heating and cooling system are the one described in Section 2.2. The control strategy parameters are reported in Section 2.5.2.1. At first a strategy has to be picked (Strategy 1 or Strategy 2), then the parameters for the given strategy can be set as desired.

Lines 36-43 instantiate a new heating and cooling system model with the given input values. Line 46 creates a new experiment. In line 49 the model is connected to the experiment. Lines 51-53 set the parameters for the experiment. Line 53 is particularly important, since it sets after how many minutes to stop the experiment. After this, the experiment is started (line 56) running for as many minutes as set on line 53. Line 65 closes the experiment, while line 69 prints out the output values on the console.

Pay particular attention to:

- All the demands and the electricity prices time series have to be synchronized. This means that they all have to start from the same week day at the same hour and have the same length.
- The experiment's end-criterion of line 53 is very important and has to be set in accord with the length in minutes of the energy demands and supply time series.

[4] http://www.eclipse.org

Algorithmus A.1 Run experiment class

```
public class RunExperiment {

  public static void main(String[] args) {

    /**The energy demand and supply time series*/
    ArrayList<Demand> coldDemand = ReadDataFromCSV.readDemands("src/mydata/December Kaeltebezug.csv");
    ArrayList<Demand> heatDemand = ReadDataFromCSV.readDemands("src/mydata/December Waermebedarf ohne BWW.csv");
    ArrayList<Demand> hotWaterDemand = ReadDataFromCSV.readDemands("src/mydata/December Brauchwasserbedarf.csv");
    ArrayList<ElectricityPrice> electricityPrices = ReadDataFromCSV.readElectricityPrices("src/mydata/December Spot Electricity Prices Samuele.csv");


    /**The parameters for the heating and cooling system*/
    //Parameters for the cooling unit
    double coolingUnitMinPower = 150.33; double coolingUnitMaxPower = 2706;
    double coolingUnitNominalPowerInput = 609; double coolingUnitNominalCoolingCapacity = 2706;

    //Parameters for the cold storage
    double coldStorageMinTemperature = 5; double coldStorageMaxTemperature = 12;
    double coldStorageWaterVolume = 26000; double coldStorageMaxUnloadRate = 2000;

    //Parameters for the waste heat storage
    double wasteHeatStorageMinTemperature = 39; double wasteHeatStorageMaxTemperature = 41;
    double wasteHeatStorageWaterVolume = 18000; double wasteHeatStorageMaxUnloadRate = 500;

    //Parameters for the hot water storage
    double hotWaterStorageMinTemperature = 36; double hotWaterStorageMaxTemperature = 41;
    double hotWaterStorageWaterVolume = 3500; double hotWaterStorageStartLoadingAt = 95; double hotWaterStorageMaxUnloadRate = 70;


    /**The parameter for the control strategy*/
    int strategyNumber = 2; int strategy2MaxShiftHours = 12; int strategy2StartLoadingAt = 95;
    double strategy1StandardDeviationShift = 0; double strategy1PercentShift = 100;
    ArrayList<Demand> strategy1HistoricColdDemand = ReadDataFromCSV.readDemands("src/mydata/December Kaeltebezug.csv");


    // Create the model, null as first parameter because it is the main model and has no mastermodel
    HeatingCoolingSystemModel model = new HeatingCoolingSystemModel(null, "Heating and Cooling System Model", true, true,
        coldDemand, heatDemand, hotWaterDemand, electricityPrices,
        coolingUnitMinPower, coolingUnitMaxPower, coolingUnitNominalPowerInput, coolingUnitNominalCoolingCapacity,
        coldStorageMinTemperature, coldStorageMaxTemperature, coldStorageWaterVolume, coldStorageMaxUnloadRate,
        wasteHeatStorageMinTemperature, wasteHeatStorageMaxTemperature, wasteHeatStorageWaterVolume, wasteHeatStorageMaxUnloadRate,
        hotWaterStorageMinTemperature, hotWaterStorageMaxTemperature, hotWaterStorageWaterVolume, hotWaterStorageStartLoadingAt, hotWaterStorageMaxUnloadRate,
        strategyNumber, strategy2MaxShiftHours, strategy2StartLoadingAt, strategy1StandardDeviationShift, strategy1PercentShift, strategy1HistoricColdDemand);

    // Create the experiment
    Experiment exp = new Experiment("FirstExperiment",  TimeUnit.SECONDS, TimeUnit.MINUTES, null);

    // Connect both
    model.connectToExperiment(exp);

    // Set the experiment parameters
    exp.setShowProgressBar(true);  // display a progress bar (or not)
    exp.stop(new TimeInstant(44639, TimeUnit.MINUTES));  // set end of simulation at 44639 minutes

    // Start the experiment at simulation time 0.0
    exp.start();

    // --> now the simulation is running until it reaches its end criterion
    // ...
    // ...
    // <-- afterwards, the main thread returns here


    // stop all threads still alive and close all output files
    exp.finish();


    //report the output values on the console
    model.report();
  }
}
```

B. Algorithms

B.1. Accumulators Life Cycles

Algorithmus B.1 Cold storage life cycle

```
public void lifeCycle() {

    while(true){
        //the total rate is bigger than zero
        if(totalRate >= 0){
            totalEnergyConsumed += unloadRate/60;
        }
        //the total rate is smaller than zero
        else{

            //there is enough capacity to satisfy the unload rate
            if(currentCapacity + totalRate/60 >= 0){
              totalEnergyConsumed += unloadRate/60;
            }
            //there is not enough capacity to satisfy the unload rate
            else{

              double missingPower = unloadRate - currentCapacity*60;
              //the power demand in excess is satisfied by the cooling unit
              myModel.coolingUnit.supplyAdditionalPower(missingPower);
              totalEnergyConsumed += unloadRate/60;
            }
        }
        currentCapacity += totalRate/60;
        hold(new TimeSpan(1 ,TimeUnit.MINUTES));
    }

}
```

Algorithmus B.2 Waste heat storage life cycle

```
public void lifeCycle() {

    while(true){
        myModel.hotWaterStorage.setLoadRate(0);
        //the total rate is bigger than zero
        if(totalRate >= 0){
            totalEnergyConsumed += unloadRate/60;
            currentCapacity += totalRate/60;
            if(currentCapacity > capacity){

              myModel.coolingTower.distroyEnergy(currentCapacity-capacity);
              currentCapacity = capacity;
            }
            if(reloadHotWaterStorage){
              this.reloadHotWaterStorage();
            }
        }
        //the total rate is smaller than zero
        else{

            //there is enough capacity to satisfy the unload rate
            if(currentCapacity + totalRate/60 >= 0){
              totalEnergyConsumed += unloadRate/60;
              currentCapacity += totalRate/60;
              if(reloadHotWaterStorage){
                 this.reloadHotWaterStorage();
              }
            }
            //there is not enough capacity to satisfy the unload rate
            else{

              //the energy demand in excess is satisfied by the gas heating unit
              myModel.gasHeatingUnit.supplyEnergyForHeat(-totalRate/60-currentCapacity);
              totalEnergyConsumed += (unloadRate/60)-(-totalRate/60-currentCapacity);
              currentCapacity = 0;
            }
        }
        hold(new TimeSpan(1 ,TimeUnit.MINUTES));
    }

}
```

Algorithmus B.3 Hot water storage life cycle

```
public void lifeCycle() {
    while(true){
        //the total rate is bigger than zero
        if(totalRate >= 0){
          totalEnergyConsumed += unloadRate/60;
          currentCapacity += totalRate/60;
        }
        //the total rate is smaller than zero
        else{

          //there is enough capacity to satisfy the unload rate
          if(currentCapacity + totalRate/60 >= 0){
            totalEnergyConsumed += unloadRate/60;
            currentCapacity += totalRate/60;
          }
          //there is not enough capacity to satisfy the unload rate
          else{

            //the energy demand in excess is satisfied by the gas heating unit
            myModel.gasHeatingUnit.supplyEnergyForHotWater(-totalRate/60-currentCapacity);
            totalEnergyConsumed += (unloadRate/60)-(-totalRate/60-currentCapacity);
            currentCapacity = 0;
          }
        }
        //if the current capacity drops under the limit then reload
        if(currentCapacity < (capacity*startLoadingAt/100)){

          myModel.wasteHeatStorage.setReloadHotWaterStorage(true);
        }
        hold(new TimeSpan(1 ,TimeUnit.MINUTES));
    }
}
```

B.2. Producers Life Cycles

Algorithmus B.4 Cooling unit life cycle

```
public void lifeCycle() {

    while(true){
        myModel.coldStorage.setLoadRate(0);
        myModel.wasteHeatStorage.setLoadRate(0);
        if(reloadColdStorage){

            double maxPossibleLoadRate = (myModel.coldStorage.getCapacity() -
            myModel.coldStorage.getCurrentCapacity())*60;
            if(maxPower*(COP-1) <= maxPossibleLoadRate){

              myModel.coldStorage.setLoadRate(maxPower*(COP-1));
              totalColdEnergyProduced += (maxPower*(COP-1))/60;
              myModel.wasteHeatStorage.setLoadRate(maxPower*COP);
              totalWasteHeatProduced += (maxPower*(COP))/60;
              totalElectricityCosts += (maxPower/60)*(myModel.priceChanger.getCurrentPrice()/1000);
              totalElectricity += (maxPower/60);
            }
            else if((maxPossibleLoadRate/(COP-1)) >= minPower){

              myModel.coldStorage.setLoadRate(maxPossibleLoadRate);
              totalColdEnergyProduced += (maxPossibleLoadRate)/60;
              myModel.wasteHeatStorage.setLoadRate(maxPossibleLoadRate);
              totalWasteHeatProduced += (maxPossibleLoadRate/(COP-1)*COP)/60;
              totalElectricityCosts += ((maxPossibleLoadRate/(COP-1))/60)*
              (myModel.priceChanger.getCurrentPrice()/1000);
              totalElectricity += ((maxPossibleLoadRate/(COP-1))/60);
              this.reloadColdStorage = false;
            }
            else{
              this.reloadColdStorage = false;
            }
        }
        hold(new TimeSpan(1 ,TimeUnit.MINUTES));
    }

}
```

B.3. Energy Demand and Supply Life Cycles

Algorithmus B.5 Cold energy demand life cycle

```
public void lifeCycle() {

    while(coldDemand.size() > 0){
        Demand currentDemand = coldDemand.remove(0);
        myModel.coldStorage.setUnloadRate(currentDemand.getPower(),currentDemand.getDuration());
        totalColdDemand += currentDemand.getEnergy();
        hold(new TimeSpan(currentDemand.getDuration() ,TimeUnit.MINUTES));
    }

}
```

Algorithmus B.6 Electricity supply life cycle

```
public void lifeCycle() {

    while(electricityPrices.size() > 0){

        ElectricityPrice newPrice = electricityPrices.remove(0);
        currentPrice = newPrice.getPrice();
        hold(new TimeSpan(newPrice.getDuration() ,TimeUnit.MINUTES));
    }

}
```

B.4. Control Strategy Life Cycles

Algorithmus B.7 Strategy 1

```
public void strategy1(){

    generateHistoricData();
    while(true){
        myModel.coolingUnit.setReloadColdStorage(false);
        //the approximated minutes before the system will be empty
        int minutesToEmpty = getMinutesToEmpty();
        if(minutesToEmpty != -1){
            //the exact minutes to the best price
            int minutesToBestPrice = myModel.priceChanger.getBestPrice(minutesToEmpty);
            if(minutesToBestPrice == 0){

              myModel.coolingUnit.setReloadColdStorage(true);
            }
        }
        hold(new TimeSpan(1 ,TimeUnit.MINUTES));
    }

}
```

Algorithmus B.8 Generate historic data

```
private void generateHistoricData() {

    double shiftCurve =  seriesShift.sample();
    for(int i = 0; i < historicColdDemand.size(); i++){
        double shiftHour =  hourlyShift.sample();
        double shiftedPower = (historicColdDemand.get(i).getPower()*percentShift/100)+
        shiftCurve+shiftHour;
        if(shiftedPower < 0){
            shiftedPower = 0;
        }
        double shiftedEnergy = (historicColdDemand.get(i).getDuration()/60)*shiftedPower;
        historicColdDemand.get(i).setPower(shiftedPower);
        historicColdDemand.get(i).setEnergy(shiftedEnergy);
    }

}
```

Algorithmus B.9 Strategy 2

```
public void strategy2(){

     int minutesToBestPrice = -1;
     while(true){
          if(myModel.coldStorage.getCurrentCapacity() <
          myModel.coldStorage.getCapacity()*startLoadingAt/100){
               if(minutesToBestPrice < 0){

                  minutesToBestPrice = myModel.priceChanger.getBestPrice(maxShiftHours*60);
               }
               if(minutesToBestPrice == 0){

                  myModel.coolingUnit.setReloadColdStorage(true);
               }
          }
          minutesToBestPrice --;
          hold(new TimeSpan(1 ,TimeUnit.MINUTES));
     }

}
```