Optimization of building energy consumption

University of Southern Denmark

Software Engineering

Almir Mehanovic, Exam nr. XXXX
Emil Sebastian Rømer, Exam nr. 324790

01 June 2015

Abstract

This study discus the application of statistical methods and machine learning in the field of buildings energy consumption. The project is developed in association the municipality of Odense, and the focus have been on creating value by solving existing problems.

The product is a server system and a website application which works a platform for obtaining information regarding the energy consumption in public buildings in the municipality of Odense. In addition, the product implements functionality enabling the user to maintain the system and data. The system is written in R, html, JavaScript and Java.

This paper will show, that using statistical methods and machine learning it is possible to optimize the workflow of the employees in the municipality of Odense's Energy and Maintenance department and improve the decisions made.

This report is a technical report, however the project have been conducted using a user centered iterative process model and therefore it contains a section of user evaluation.

Foreword

This paper is the result of 6th semester Bachelor project in Software Engineering at University of Southern Denmark in the period 2/2/2015 – 1/6-2015.

The group wants to give a special thanks all involved supervisors and staff at the municipality of Odense:

- Kim Allan Kristensen, Head of the department of Energy and Maintenance
- Mikkel Baun Kjærgaard, Associate Professor, PhD (Supervisor)
- Sanja Lazarova-Molnar, Associate Professor, PhD (Supervisor)

Further thank to the municipality of Odense for participating and for making the data available. Lastly, a thank to Lone Borgersen, Associate Professor, Educational coordinator Software Engineering, for coordination, planning and organization the bachelor.

Contents

Reading guide

The paper is written to be read in chronological order, and in some cases more examples are explained in the appendix, if that is the case it will be mentioned in the text. In this paper OM have been used in short for Odense municipality. Next, the product have been named BEOVulf -Building Efficiency Optimizer and Visualizer. This name will be used throughout the report.

Chapter 1

Introduction

The developed world's dependency on fossil fuels to power growth and prosperity is becoming an increasingly unsustainable and unfeasible situation. The world will eventually run out of fossil fuels, and the dependency on it means that vital infrastructure is depending on politically unstable regions. What is perhaps even more concerning is the damage our environment suffers from our burning of fossil fuels. Energy efficiency is a major step away from the dependency on fossil fuels, and towards a society powered by economically and socially sustainable energy.

In March 2012 a new Energy Agreement was reached in Denmark, with goals to have approximately 50% of electricity consumption supplied by wind power, and more than 35% of final energy consumption supplied from renewable energy sources in 2020. The final goal is to have 100% renewable energy in the energy and transport sectors by 2050[1].

Buildings in Denmark have over the last decades become more energy-efficient as a result of advances in engineering and the ongoing strengthening of requirements for new buildings. However, around 40% of the total energy consumption is used in buildings making them the largest contributor to energy usage in Denmark[2]. Therefore retrofitting buildings with energy efficient technology is important, if Denmark is to fulfill the energy agreement and reach it's goals. The field of Energy Informatics has recently attracted much attention as large energy efficiency improvements remains to be realized, by utilizing the field to better understand, predict and optimize energy consumption efficiency in buildings[3]. This field is a cornerstone for realizing the visions of smart buildings and smart cities. In southern Denmark, the Municipality of Odense (henceforth referred to as OM) has come one step closer to these visions, by putting data regarding energy consumption of approximately 550 public buildings on the

[1] Energy Efficiency Policies and Measures in Denmark 2012; Page 3.

[2] http://www.kebmin.dk/klima-energi-bygningspolitik/dansk-klima-energi-bygningspolitik/byggeriet-danmark/bygningers

[3] Energy Informatics -DOI 10.1007/s12599-013-0304-2

roadmap to becoming Open Data[4]. This data contains hourly measurements of raw consumption for electricity, heating and water in a building. The availability of such data is a crucial step, but the raw data is of little use, unless it is transformed into useful information that can be represented in an intuitive manner to various stakeholders.

This project has worked with OM to find potential use cases of the consumption data, and explore what kind of useful information can be extracted from the available data. The aim was to develop a functioning prototype website capable of highlighting various trends, correlations, odd events, odd buildings etc. The scope for this project was not limited to researching and theoretically proving the applicability of various data processing methods, but included understanding and applying state of the art statistical methods to solve real problems for OM. Thus the challenges faced by this project were also those of a software development project, in which a software system suiting the clients' need had to be developed and delivered. The project utilized an iterative development approach, to identify needs and problems related to building energy consumption in OM, design and evaluate prototypes and develop the final software solution.

The problems OM is facing and which the developed software attempts to solve, are rooted in issues relevant to the entire energy management domain. Those issues include how individuals and organizations can gain a better understanding of their energy consumption, and be motivated to save energy. How to identify odd consumption patterns and get a quicker response time to leakages. How to benchmark buildings to guide investments in renovations to yield optimal results. How to optimally support the workflows of building energy managers, while minimizing operation and maintenance efforts of such a system. How to more precisely predict energy consumption. Thus anyone working with those or related issues might be interested in the results and findings of this project.

[4]http://odensedataplatform.dk/

Chapter 2

Background

Odense Municipality (OM) is the third greatest in Denmark with an size of 304 km^2 and a population of nearly 200.000 people. In order to reach the goals of the national environmental policy, OM has created the directive "Environmentally friendly construction - requirements and recommendations", which as the directive states will: "concretise the environmental policy and objectives, so that future building projects in the municipality contributes to the up-filling of the overall vision of Odense as Denmark's most sustainable city."[1]. This directive sets energy efficiency requirements for newly constructed buildings and also for renovated ones.

OM owns approx 850 buildings, which are used for different purposes like schooling, administration, care centers and cultural institutions. OM is responsible for paying suppliers of water, heat and electricity for the consumption in these buildings. This is done on basis of billing meters which measure the consumption, and are installed in each building. Depending on the size of the building, there might be a number of internal meters (so called distribution meters), which measure the consumption in different parts of the building, but are not used as billing meters.

Each public building has an appointed energy manager who has the responsibility of performing the manual tasks involved in reading and checking the meters. The energy manager logs reading internally in a control book or scheme and reports them on the last weekday of the month through a web portal. If reading deviate more than +/- 10 % from the budget forecasts, the manager must submit possible causes for this. As the document "Energy efficiency in the municipality of Odense" states, both types of meters must be often checked to avoid energy leakages. In buildings less than 1.000 m^2, meters must be checked at least once a week and for larger buildings the frequency is at least once a day. OM has started installing meters which automatically report their readings on an hourly basis. Those are not yet installed in all buildings, but OM has plans to do so. Depending on the concrete building and setup there is a delay from

[1] Miljørigtigt byggeri – krav og anbefalinger

around 1 to 7 days before readings from the automatic meters are visible in OM's energy management system 'Energy Key'[2].

2.1 Current limitations

Through interviews with the head of OM's Energy and Maintenance department, the current practises in the domain of energy and building management have been investigated. This analysis revealed a number of areas in which improvements could be made. Before 2013 each building's energy consumption was monitored on site by appropriate staff, and energy bills were paid from the budgets of the organization residing in the building. However building managers on site had no real tools or means of monitoring their consumption, making it difficult to identify high usage or leakages. This function has now been centralized in the department of Energy and Maintenance which owns the buildings and is responsible for monitoring and paying for energy consumption, as well as renovation and maintenance of the buildings. The organizations residing in the buildings will then just rent the building for a fixed price. This centralization should give OM a better control and overview of the municipality's energy consumption, but a number of new issues have arised. The task has scaled with the number of buildings that now needs to be monitored, and the problem analysis showed that OM's Energy and Maintenance department does not have the needed staff nor the proper tools to complete it effectively. The problem analysis resulted in the identification of 5 main issues, which has been the focus points of this project.

System and data maintenance

Maintaining accurate data about a building's size, age, type, associated meters etc. can seem like a seemingly small effort, but has scaled to become an overwhelming task when considering the amount of buildings. If the primary data is imprecise, erroneous or inconsistent, the value of data analysis and anything that depends on it will decline. OM's current energy management system Energy Key has no way of telling the user when certain data has last been verified and by whom, nor is there any easy way for users to update incorrect data entries. This makes it difficult to work with the data, because the users are often unsure about the validity of it, and it causes redundant work for multiple users that need to both verify the same data. There are currently no procedures for how often a building's consumption should be checked and validated, nor does Energy Key support such continuous maintenance tasks in any way. Additionally OM struggles with poor data quality and many missing values regarding consumption figures, which according them is because the energy suppliers' systems are simply not built for sharing data with clients.

[2]http://www.emtnordic.com/da/energykey-energistyringsprogram

Visualization of buildings and consumptions

Energy Key allows users to view electricity, water and heat consumption data for individual meters, but the possibilities to visualize and represent data in meaningful formats seems limited. The user can view diagrams depicting consumption over time, and histograms that compare a building's reported consumption with the budget. Other than this, the system does not summarize important key features and energy statistics for a given building. It does not analyze the data to provide new insights nor does it provide any tools for users to do so. The lack of meaningful information and visualization of consumption patterns makes it hard to understand a building's energy usage. This makes it more difficult to motivate users and building managers into making better decisions that lead to energy savings, because they do not know how their behaviour influences the consumption. Furthermore it is not possible to easily see what kind of motivation works, and for how long the effects of energy saving campaigns lasts.

Fault detection, diagnosis and resolution

Energy Key has an automatic fault detection feature which can raise an alarm when consumption goes above or below a given percentage of what is budgeted. OM's Energy and Maintenance department has tried experimenting with this system for a single building, and the results were that 37 false alarms were received in 2012 and 34 in 2013. When considering all buildings the amount of false alarms becomes overwhelming. The large amount of false alarms combined with the fact that tasks involved in diagnosis and resolution of faults are not supported in Energy Key, has lead to the alarms being completely ignored. Better visualization of buildings and consumptions would allow users to more easily diagnose what caused the alarm. Even if a user can verify that an alarm is valid the fault might be neglected, as there no standard procedures to follow in order to resolve the fault, nor is there any personnel assigned as responsible. Energy Key provides no support for diagnosis and resolution of faults, as an alarm can not be associated with a set of tasks assigned to relevant users. It is not possible to track status and progress of its resolution and users can't make comments to coordinate efforts between them. Another type of fault which is currently not detected by the system is when meters suddenly go offline and stop reporting data.

Budgeting

Currently energy consumption and budgets for each building are estimated by looking at the same building's consumption in previous years. Estimations have monthly granularities, meaning that the energy consumption estimate for a given month depends on the consumption reported the same month previous years. This approach does not allow one to properly predict consumption under

changing circumstances (people usage, environment, building changes etc.). A better model for a building's energy consumption would allow for better predictions and more exact budgets. When information about a building is changed in Energy Key, the previous information is lost making it is difficult to use the previous consumption patterns to plan future budgets as there is no historic data to show under which conditions the previous consumption was produced. Because of the issues with maintaining data in Energy Key, information like how many people are using a building is stored and maintained in excel files, meaning budgets are created only on basis of the very basic building features stored in the system.

Selecting buildings for renovation

In 2015 OM's Energy and Maintenance department has budgeted 200 million dkk to renovation of public buildings. As the directive Environmentally friendly construction - requirements and recommendations[3] says all renovations must be done with the most energy efficient solutions. The department now faces the task of finding the set of buildings for which renovations would bring down energy usage the most. Currently a taskforce is screening through the available data manually, and more or less takes educated guesses about which buildings are inefficient and can be optimized. This is a difficult task as there is a large amount of buildings and no visualization of the data in a meaningful manner resulting in consumption pr. square meter being the only parameter considered in this manual process. The current system is unable to perform any sophisticated benchmarking or ranking of buildings, making it nearly impossible to verify that the most energy inefficient buildings have been found. However, identifying energy inefficient buildings is only the starting point, as a vast range of economical and political parameters also have to been considered before a building is selected for renovation.

[3]See Appendix 01 -Environmentally friendly construction

Chapter 3

Requirements

Based on the initial problem analysis and identification of OM's current limitations, a quick low fidelity horizontal prototype system was created[1]. This prototype was evaluated through an interview with a representative from OM's Energy and Maintenance department[2]. Based on this interview the problem analysis was deepened and a final backlog of requirements was for the system to be developed was specified. The full requirements list can be found in Appendix 05 -Requirement Specification, while this section will give an outline of how the requirements solve the identified issues.

3.1 System and data maintenance

Any client of a new software system would require that the servicing efforts are minimal, but the problem analysis uncovered that for OM this is of critical importance. OM's Energy and Maintenance department has had a large increase in workload without a proportional increase in resources or personnel. This means that the department simply does not have the time or resources to maintain a new system if the required servicing effort is too high. A number of steps to prevent this have been taken, resulting in formal requirements which can be measured and evaluated.

Firstly, in long term, the new system should replace OM's existing system, but changes in an organization's processes, workflow and technology comes with the expense of time and money. In order to minimize the expenses, a gradual adoption of the new system will be in focus. Requirement R18, ensures data consistency in a such adoption phase where multiple systems work on the same data. This ensures further, that maintenance in an adoption phase will remain the same as in the existing system.

Secondly, to minimize the effort needed to maintain large amounts of building information (such as type, age size and so forth), requirement F2-06a and F2-

[1] See Appendix 04 -Website Alpha Prototype.zip
[2] See Appendix 02 -Meeting with Kim Allan 29-4-15

06b make sure it is possible to easily update and validate these general building informations. The effort associated with maintaining the data is minimized by informing the user when data was last validated and if it is time to re-validate. Thus the user needs not spend time investigating validity of data every time it is needed. Also as opposed to the current system it will be possible to easily update data when a building is viewed anyway, and then offering the user the choice to validate it now. This solution ensures that even building managers for the different buildings will participate in keeping the information correct. This is additionally supported with requirement F15 and F17, which will allow the assignment of maintenance tasks to the building managers thus distributing the maintenance effort.

3.2 Visualization of buildings and consumptions

Requirements F2 and F9 together with the respective sub requirements state that it should be possible to view and compare information about buildings and their energy consumption. The requirements formulate a number of concrete aspects of the energy consumption that must be visualized to the user. In contrast to OM's existing system this will give users better information and support better decision making. Requirement F2-02 gives concrete examples of a number of energy features that must be extracted and visualized from the raw consumption numbers. Those are for instance peak loads, standby usage and daily average consumptions.

Besides pure consumption figures, the visualization of consumption patterns can further enhance a manager's understanding of the buildings consumption. The consumption patterns of a building may be viewed in different levels of granularity, like for instance patterns on a micro scale (dayly) and patterns on a macro scale (yearly). Depending on the user's task, he might be interested in patterns on different scales. The three requirements F2-03a to F2-03c describe how consumption patterns on a micro scale is to be visualized through so called daily consumption profiles. Requirement F2-04 requires that a buildings seasonal variation in consumption is visualized which might be considered a macro scale pattern.

From looking only at the consumption of a single building without a context or reference point, it can be hard to identify faults or even verify the that everything is as it should be. Therefore requirement F9 and its sub requirements aim to give the user a way to visualize a building in comparison with others, to have a reference point when evaluating a building[3].

[3]Requirement F9 and comparing buildings is further discussed in the Selecting of buildings for renovation section

3.3 Fault detection, diagnosis and resolution

Requirement F3, addresses the fault detection, and here the aim is to build a fault detection system which is more accurate and allows for diagnosing the feature which caused the flaw.

Requirements F3-02, F3-03 and F12 (handling of alarms) will try to make sure alarms are not ignored or forgotten by establishing a process/workflow for how they should be handled. The system will provide an easy overview of alarms, and requirements regarding task management (F15-17) will contribute to the creation of a "standard" procedure of handling alarms.

3.4 Budgeting

Requirement F14 states that the system is should implement functionality regarding creating and maintaining a budget. The system must support the user in creating precise budgets, by being able to accurately forecast the expected consumption of a building as stated in requirement F13.

3.5 Selecting buildings for renovation

Selecting buildings for renovation is a decision process with many political and economical parameters which would be impossible for one system to fully implement. However the system will be able to support the process by giving users the ability to quickly identify potential candidates. This is done by implementing requirement F11, which will make the system come up with various rankings of buildings, making the worst ranked buildings potential candidates for renovation. It will be considered to map the benchmarks of buildings to the national energy labeling scheme[4].

3.6 Other requirements

The environment in which the application is to be used, spawns quality requirements for the systems portability and scalability which needs to be meet in order to develop a successful application.

The users from OM's Energy and Maintenance department work both from their offices and from various locations when visiting buildings to inspect. This need is met by requirement XX, which addresses the portability of the system and states the application must work on a tablet.

This project has focused on the municipality of Odense, but other municipalities in Denmark could potentially have similar requirements for an energy management system. Based on this, there is a requirement that the application is scalable. Another scenario is that the system is expanded to not only public

[4]http://www.ens.dk/forbrug-besparelser/byggeriets-energiforbrug/energimaerkning

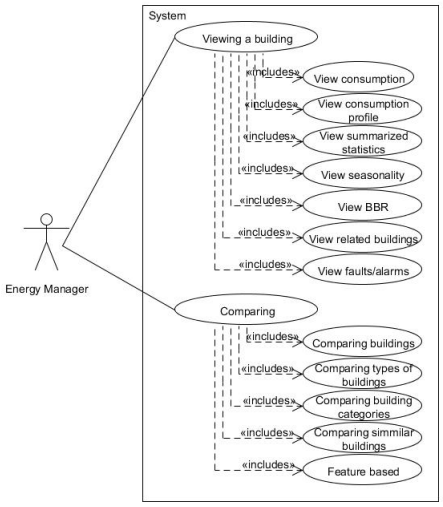
buildings. In that case the number of buildings handled would be an order of magnitude more, and the system should be able to handle this[5].

3.7 Prioritization of requirements

All requirements have been prioritized using the MoSCoW-prioritization technique. The prioritizations do not reflect the wishes of the client, but are based on what is within this project's scope and boundaries. For instance a task management system is out of scope for this project and therefore requirements F16 and F17 have been prioritized as $WON'T$ have. If a finished product was to be made, these requirements are essential for the user and should therefore be implemented.

3.8 Overall system description

Based on the different requirements and on the prioritization of these the following use case diagram have been developed. The uses cases have been organized into two groups. This have been done to generalize functionality somewhat similar into groupings and to link different functionality which is supports the user in a bigger use case together. The use case displays all the use cases estimated can be developed in this project. In relation to this, one should note, that the use cases included in this diagram does only extent to the functionality directly linked to the end user. As an example the requirement for integrating the system with the existing system have not been included, even though it will be implemented.



[5]See requirement Q19

Chapter 4

Related Work and Methods

Within recent years numerous research papers have emerged showing successful applications of various methods to real word problems like detecting faults, benchmarking, classifying, predicting or recognizing patterns in the energy consumption of buildings. Most of the applied methods rely on theories rooted in the fields of statistics, machine learning and data mining to analyze and interpret the data. To successfully apply such methods to real word problems one must be familiar with the nature of the data, and have working knowledge of the methods being applied. This section will explore some of these methods and the current literature that applies them to various problems related to energy management.

4.1 Decomposing time series

The most visible pattern that emerge from time series data of building energy consumption is the daily rise and fall in consumption, which loops every 24 hours following the daily cycles of building users. Also a weekly cycle can be noticed, as different days of the week tend to have different consumption profiles. If looking at several years of data, the yearly cycle following the seasonal changes in environmental conditions can be seen. These cycles form the basis for the components that can be found when decomposing a time serie, and which could be used to reconstruct the original by combining the components. For instance considering the weekly cycles and yearly seasonal change, it would be possible to make a decomposition into a Cyclical Component, and a Seasonal Component. Additionally [REF wiki] suggests that the decomposition produces a Trend Component and an Irregular Component. The Trend Component reflects the long term progression irrespective of any cycles and seasonality, while the Irregular Component is the noise in the original time serie which could not be explained with the other components. Figure XX shows an example of a decomposition:

The example illustrates how decomposing time series of energy consumption can give more insight into the nature of the consumption. We can see that the consumption is generally going up XX each XX, and that consumption is XX higher in the winter than in the summer. Also decomposition is relevant when comparing consumption between different days, because it might be beneficial to first remove the seasonality component. There are however many ways to adjust for seasonality. For instance [3] suggests subtracting from each day the minimum consumption for that day, while [5] transforms the features of one day by determining the difference between it and a the features of a 1-week average of surrounding days.

The different components of a decomposition can serve as different units of analysis, and are therefore important to consider when extracting features from the data. Whatever forecasting consumption load, detecting abnormal consumption or something else is desired, most approaches agree upon that the first step is to extract interesting features from the data, which can be used in an analysis. This might be raw consumption figures (daily average, daily max/min), ratios between features (consumption night/day), temporal properties (time of daily max, first time above 1kW) or statistical properties (variance, correlation between features). [REF beckel 2012] brings a list of potentially interesting features when considering energy consumption data, while [7] suggests a range of interesting building properties to consider.

4.2 Cluster analysis

Cluster analysis is widely used within data mining and machine learning, to find underlying structures and patterns in data. [5] proposes a pattern recognition algorithm for determining days of the week based on the raw consumption data, and explains how this information can be used for both forecasting consumption load and detecting abnormal consumption. By creating a cluster for each day of the week and then identifying similar clusters based on various features of daily consumptions, it is possible to come up with a number of 'day types' with distinct consumption profiles. The type of day and consumption profile can then be used as input to forecasting algorithms, or as a model of normal consumption from which a large deviation can be considered an anomaly. [3] uses a very similar approach.

There are many different techniques for clustering, and even more algorithms implementing them. Some popular ones are the the centroid-based K-means algorithm and the density- based DBSCAN. The K-means algorithm fixes the number of clusters to k (a parameter given to the algorithm). The algorithm will assign each data point in the dataset to the nearest cluster center, and then seek to find the optimal positions for the k cluster centers, such that the squared distances from each object to the cluster centers is minimal. K-medoids clustering is very similar to k-means, except that while the cluster is represented by a centroid in k-means, it is represented by the data point closest to the center of the cluster in k-medoids. A drawback of this approach is that K-means and K-

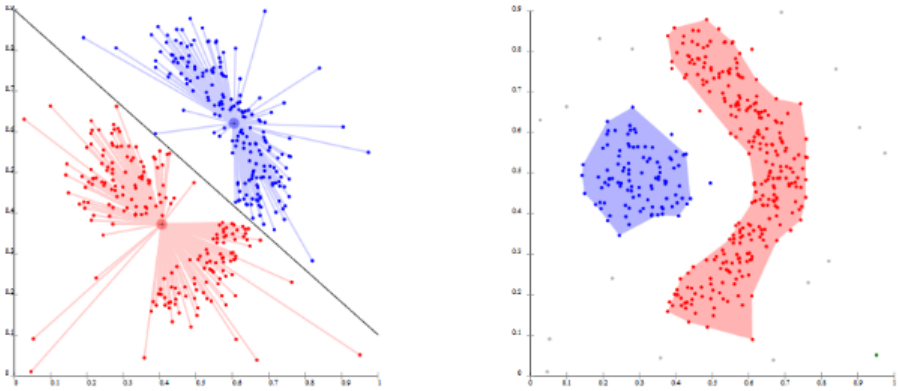


Figure 4.1: *Note these pictures have been taken from Wiki -Cluster analysis*

medoids clustering does not work well with non convex clusters, which might be present if data point features do not follow normal distributions. This problem is avoided in DBSCAN which creates clusters by connecting points which have a specified minimum number of other points within a specified radius. Based on this density criterion clusters will of following arbitrary shapes can be formed. Figure 4,1 illustrates how cluster analysis with K-means and DBSCAN can result in very different clusters for the same dataset. DBSCAN additionally has the ability to detect outliers, which would be the data points residing in low density areas not reachable with the density criterion. [4] compares the use of DBSCAN alone with using K-means followed by the Generalized Extreme Studentized Deviate (GESD)[1] outlier detection method, to find outliers and detect faults in building energy consumption. The results however seem somewhat inconclusive.

[1] and [7] demonstrate the use of SOMs (Self Organizing Maps) to perform cluster analysis of building energy consumers. [1] finds clusters of energy consumers by training a SOM using raw energy consumption data, while [7] uses building information as basis for the clustering. Perhaps as important as the clustering method is the features chosen to use as input for the clustering. Too many features might introduce too much noise, so it might be beneficial with some dimensionality reduction before performing cluster analysis. Using too few features however might not properly separate the clusters. A dimensionality reduction can be achieved with a principal component analysis. By using only the principal components that explain the majority of variation, one would have found the best low-dimensional representation of the variation in a multivariate data set. [3] argues that a better approach to use when classification is the purpose is CVA (canonical variate analysis), known as linear discriminant analysis (LDA). The purpose of LDA is to find the linear combinations of the

[1] See section Outlier Detection

original variables that gives the best possible separation between the groups. Thus LDA is used with day of week as the grouping variable, to project the original data into new axes which maximize separation between different days[2].

4.3 Detecting faults

Detecting abnormal energy consumption seems always to follow the basic steps of building some kind of model of normal consumption and then detecting faults with an appropriate outlier detection method [2], [3] and [4]. Another approach would be to use supervised learning to train a classifier that can distinguish faulty consumption from normal. However this would require a proper training set with already identified and labeled errors for each building. The lack of such datasets and the fact that faulty consumption is a fairly rare event, makes the outlier detection approach the favoured one. Apart from using outlier detection methods to detect abnormal consumption, it is used as a preprocessing method to remove data points which are so unusual that they could indicate a measurement error.

An outlier can be defined as data which appear to be inconsistent with the rest of the data set, like for instance a numerical value which is significantly distant from the rest of the data set. There is no mathematically rigid method for defining an observation as an outlier, and many methods to detect outliers both in univariate and multivariate data sets have been proposed. If the data is approximately normally distributed univariate outliers can easily be identified with a hypothesis test calculating the p-value which is the probability obtaining a result equal to or more extreme than what was actually observed. [3], [4] and [5] all make use of the GESD generalized extreme studentized deviate [REF måske?] univariate outlier detection method. This method works by doing Grubbs' test which is defined for the hypothesis:

H_0: There are no outliers in the data set

H_a: There is at least one outlier in the data set

If there is an outlier it is removed and the test is performed again, until a specified upper limit is reached or there are no more outliers in the data set. Besides an upper limit on the number of outliers an significance level alpha must be specified which would translate into the probability of getting a false positive.

In [3] the GESD procedure is applied to each feature of an observation to detect an outlier. However as figure 4,2 illustrates, an outlier in a multidimensional dataset is not always an univariate outlier. Considering the x and y values independently does not obviously reveal the outlier flagged in the first graph. To deal with multivariate outliers one needs a proper method capable of

[2] A more in depth explanation of PCA and LDA can be found in: http://little-book-of-r-for-multivariate-analysis.readthedocs.org/en/latest/src/multivariateanalysis.html

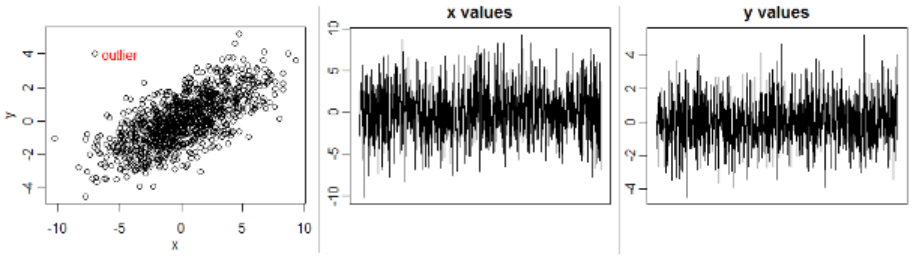


Figure 4.2: Multivariate outliers

considering the correlation between variables when detecting them. One such method is PCOut [6]. PCOut is an advanced algorithm which utilize inherent properties of principal components decomposition for outlier detection in high dimensional data. It does not require that the data come from any particular distribution, and is computationally fast.

Chapter 5

Design and Implementation

5.1 Development approach

The system was developed using an user centered iterative approach inspired by the development model of Rogers, see figure 5,1. This approach puts weight on prototyping and interactions with the client to both identify needs and evaluate prototypes. Thus 3 meetings with a representatives from OM was conducted during the two development iterations.

5.2 Overall architecture

In order to develop a system that fulfils both the functional and nonfunctional requirements, it was decided to develop a web-application in form of a website. Making the system a work as a website will make it highly portable as most client systems have browsers. The system thus becomes a distributed system with a very thin client. The thin client distributed pattern has low complexity as only the presentation layer needs to be distributed, while processing and

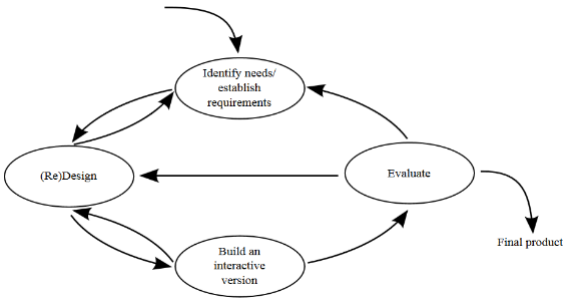


Figure 5.1: Rogers development process

data is centralized at the server. This ensures data consistency, and allows for a simple and efficient solution. The deployment diagram in figure 5,2 shows the different components of the system and how they are connected. The diagram is split in three parts by the dashed borders. Part 1 includes the gathering, formatting and storing of primary building and consumption data to a central database. Meters in OM's buildings measure the consumption of heat, water and electricity. These informations are depending on the building's setup, sent through a number of intermediary steps before they reach OM's current energy management system EnergyKey. Since EnergyKey does not provide any API to access the stored data, the only option was to set it up to export data to an FTP-server. This FTP server is located on an Amazon ec2 instance along with other of BEOVulf's components. Once EnergyKey uploads data to the FTP-server it will be formatted as csv files.A Java program will start parsing each csv file as soon as it is uploaded and store the data to a central PostgreSQL[1] database. More details on this whole process can be found in section XX,XX and XX. Once a batch of new data has been stored in the database the Java program will launch Part 2 of the system.

Part 2 consists of a number of R[2] scripts, which will read the raw meter data from the database and perform a number of operations to convert the data into the kind of information that can fulfill the requirements. This includes extracting various features from the consumption, calculating consumption profiles, detecting faults and leakages, comparing buildings and more. Section XX and XX describes the details of this process. The results are stored back to the database, where they can be accessed by Part 3 of the system.

Part 3 is the website that users will see, which is hosted in an Amazon s3 bucket[3]. The website is powered purely by html and javascript and might be considered static[4]. In order to connect the website with the central database, Dreamfactory has been used. It is a framework for automatically creating a RESTful web service which allows front-end applications to easily connect to the backend. Dreamfactory[5] exposes the database via a REST API, which uses JSON to transfer data to and from the database. The communication with the database is further simplified by making the website use the Javascript client library provided by Dreamfactory. The design and implementation of the website will be discussed in sections XX, XX and XX.

5.3 Integration with Energy Key

The integration with Energy Key was done in four main steps. First, a number of exports from Energy Key were made to gather data about buildings their attributes and the which meters are installed in them. Then in step two a year

[1] http://www.postgresql.org/
[2] http://www.r-project.org/
[3] http://aws.amazon.com/s3/
[4] http://en.wikipedia.org/wiki/Static_web_page
[5] https://www.dreamfactory.com/

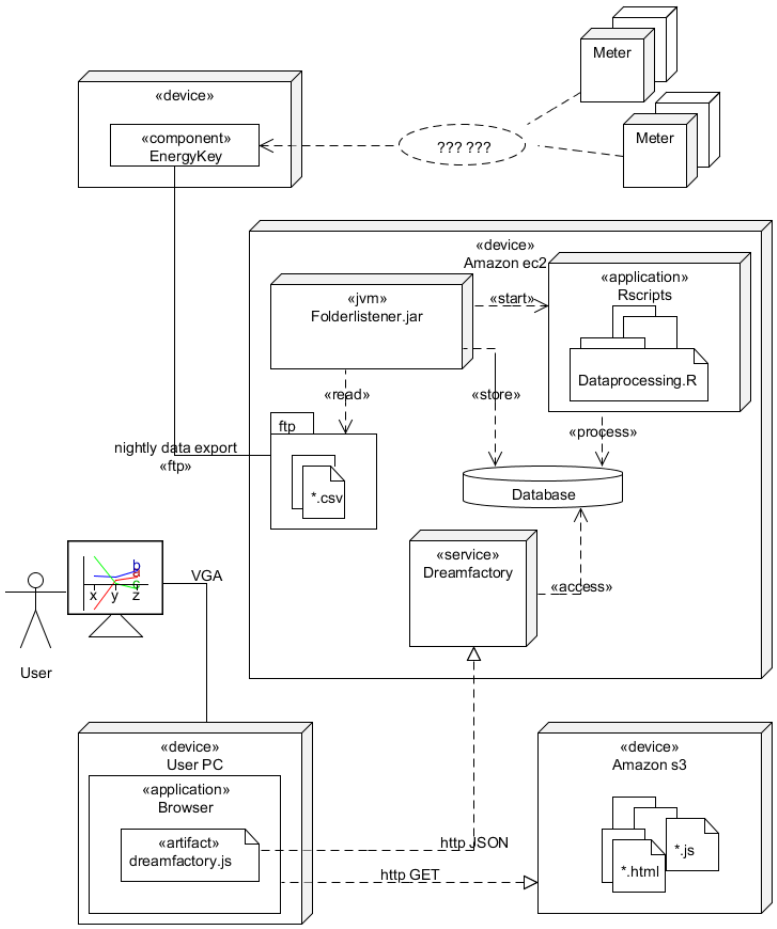


Figure 5.2: System deployment diagram

worth of measurements from the meters were exported. In step three the data was cleaned to sort out inconsistencies. And finally in step four a continues flow of data from Energy Key to BEOVulf was set up.

Step 1 exporting buildings and meters

The following reports were exported from Energy Key and used to map buildings and meters:

- Meter list[6] (Energy Key → Konfiguration → Målere → Generer rapport)
- Measurement Reading Report[7] (Energy Key → Aflæsninger → Visning → Generer rapport)

From these reports it was possible to generate a list of all buildings along with a number of attributes such as id, address, name, type (ex. '*cultural institution*') and subtype (ex. '*library*'). For each building it was possible to attribute a number of meters which measured either water, electricity or heat. For each meter it was possible to get a installation number, meter number and whatever it was a socalled *billing meter* or *distribution meter*. This attribute is important, because the readings from a *distribution meter* can not be classified as part of the consumption.

Step 2 - Exporting a year of measurements

The initial export of measurement data was limited to go one year back, as the data quality declined rapidly when going further back. The exported data was in csv files, one for each meter. The files were thoroughly analysed to determine the structure of the data. Then a Java program was created which could read through the files and insert the measurements in the central database, following the structure described in section XX. The parser made sure that data was consistent, it translated between different units, and if some data did not fit in the program stopped with a warning. The error was examined manually and it was decided what to do with it. Then the parser was changed, the database wiped, and a re-run initiated. This last step was repeated until the parser successfully parsed all 3003 csv files with more than 5.5 million measurements.

Step 3 - Detecting errors and cleaning data

When exporting building data from Energy Key there were unfortunately many buildings with missing attributes, which could not be automatically gathered from anywhere since automated access to the BBR registry is prohibited[8].

[6] Translated from: Målerliste
[7] Translated from: Aflæsnings Rapport
[8] http://bbr.dk/fatibbr

-	Heat	Water	Electricity
Total meters	1282	849	709
Billing meters in %	45%	67%	97%

Figure 5.3: Meters and distribution of type

Buildings where no type and subtype existed were removed along with all meters belonging to them, while other missing attributes were tolerated. The final amount of buildings in the database was 405. Meters which did not have both installation and meter number were removed, along with meters that had no reported data in the whole exported period. The cleaning reduced the number of meters 33% from 4293 to 2840 meters. Table 5,3 shows the final amount of meters stored in the central database:

Upon further analysis it was detected that the non automatic meters often had large numbers of overlapping or missing intervals. They were not removed as use cases for them might be found, but data from the non automatic meters was simply too unreliable to be used for advanced calculations and analysis of the building consumption. Therefore a view was created in the database which could filter out non automatic meters, by looking at whatever they have hourly readings or not.

Step 4 - Continuous meter data collection

After a year's worth of measurements was initially loaded into the database, a daily export job was created in Energy Key. This export job would automatically export csv files containing the daily measurements to BEOVulf's FTP-server located on the Amazon ec2 instance. The Java program used to parse the initial batch of measurement data was extended to be able to listen for changes in the folder used by the FTP-server. Once a csv file is fully uploaded to the folder, the java program parses it, puts the measurements contained in it in the database and then archives the csv file. Thus observations from buildings' meters would come into the database on a daily basis.

5.4 The central database

This section will describe the design of the central database in two parts. The first part is concerned with how the raw data coming from Energy Key is stored, and the second part is about the processed data.

Raw data

Figure 5,4 shows an UML diagram of the part of the database, where raw consumption and building data is stored. The main criteria for this design was

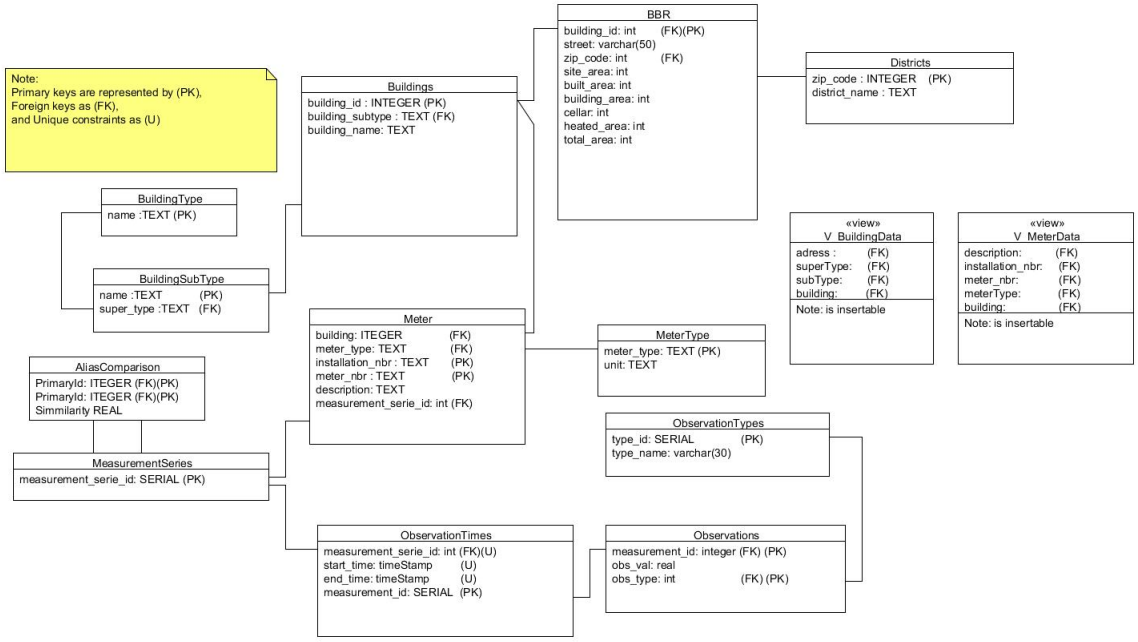


Figure 5.4: Database diagram

to be able to store all the data retrieved from Energy Key efficiently without losing any information. The database structure was modeled in a way which is generic enough to capture all kinds of observations without needing a table for each observation. This ensures that whatever data comes from Energy Key could be stored, and the design greatly reduces the number of tables needed. It makes sure that the database complies with the first three normal forms of databases, and the structure makes it possible to treat all measurements the same way when processing data.

Buildings and BBR data

The database contains a table called buildings. Each building has a building_id, type and subtype which is taken directly from Energy Key. The BBR-table represents the BBR-data collected for each building. This data contains variables like address, building area, site area, cellar size etc.

Meters and time series

When analysing the data, it was found that some buildings had the same physical meter appearing under two or more different installation- and meter num-

bers, and causing duplicated measurements. The solution was to introduce the *MeasurementSeries* table which represents data from one real physical meter and then map multiple meters from the Meter table to the same measurement serie if they in fact measure the same consumption. Additionally the *AliasComparison* table was created which through a number of triggers and procedures can calculate the similarity between time series of different meters. This enables the database to identify identical meters and dynamically remap meters and series eliminating duplicate time series on a building.

Observations

The table Observations, contains all the measured values from all meters. Each record in Observations points to a record in *ObservationType* indicating its type, and to a record in *ObservationTimes* indicating the timeslot in which the consumption was measured. It is possible for many observations thus to point to the same time slot. For example the following observation types can be recorded for a certain time slot when data from a heat meter arrives: in- and outgoing temperature, energy, flow and more.

5.5 Processed data

Figure 5.5 is a diagram showing the part of the database which stores data that is the result of analyzing and processing the original raw data. The main goal of this design was to make the data as quick to access as possible in the way that it was going to be queried. As an example, all time series data in these tables are stored as arrays with no indication of time nor date, besides a single date in the settings table indicating the start time for all time series. This is because arrays are the most compact representation most compact representation of time series data when transferred in the JSON format used by the website. To further optimize data transfer speeds, all numbers are rounded to a single decimal only. All tables and views which names are denoted with *W_** are used by the website to retrieve information. The *W_Setings* table, holds configurable settings for both the website and other parts of the system.

5.6 Processing data

Processing and analyzing the raw meter data is done by a number of R scripts. The raw consumption data will be loaded from the database into an R session which will do various statistical calculations in order to get useful information from the raw meter data. The R session is launched once a day after new data has arrived, and runs until it has finished processing it. However, some operations are only performed on a weekly basis. The system is designed such that it remains usable while this batch job is running.

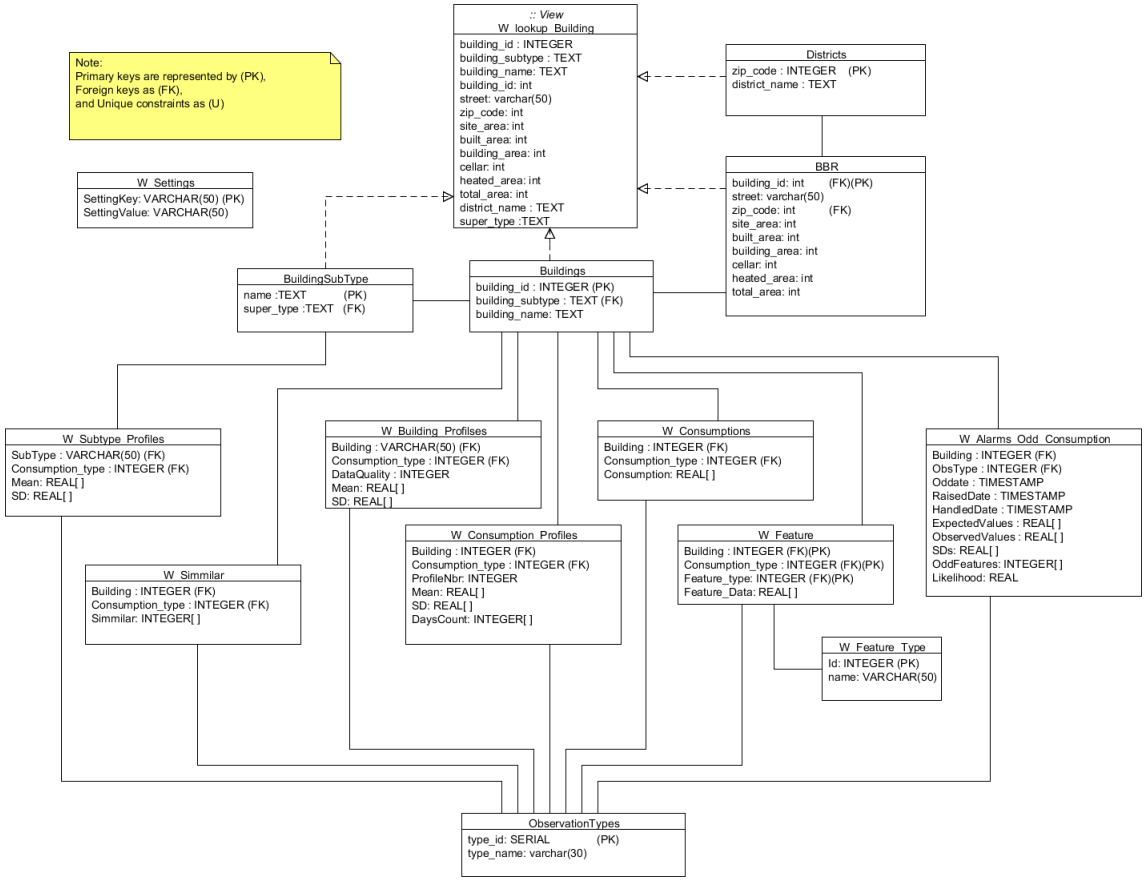


Figure 5.5: Database diagram, -Website

Loading data

Once the main R script has been launched it will start by loading observations and related data from the tables containing raw data from Energy Key. Only observations which have come in since last time the process ran will be loaded. For each building in the system, the below operations are performed to get consistent time series for each supported consumption type:

- Only observations of types water (m^3), electricity (kwH), and heat energy (GJ) are loaded as those are the only ones currently supported by the website.
- Only observations originating from *billing* meters are loaded because the readings from a *distribution* meter can not be added to the total consumption.
- Observations which do not represent a one hour timeslot of consumption are removed because they do not originate from the automatic meters and are as explained earlier too inconsistent to deal with.
- Observations across different *billing* meters which are of the same type and measure the same time slot are summed together in order to get a time serie representing the total consumption within each consumption type.
- A number of projections are done in order to get all the new raw observations and previously processed observations from the *W_consumptions* table merged into a single vector of values representing the consumption each hour since a specified epoch date which is loaded from the *W_settings*.
- A linear interpolation is performed in order to sort out missing values. The max gap for the linear interpolation is 5. This means if there is no data for more than 5 consecutive hours in a row the values will just be left as missing values.

Now once the loading of data has finished the system holds for each building and consumption type vectors of equal length, representing the hourly consumption since a specified epoch date until now. At each index i will be the amount consumed between i hours after *epoch* until $i + 1$ hours after epoch. This vector is stored as an array, along with the building id and consumption type into the *W_consumptions* table where it can be accessed by the website.

Extracting features

Once the raw consumption is loaded as vectors, the system will convert them to a matrices with 24 columns, one for each hour of the day beginning with 00:00 at the first index and ending with 23:00 at the last as illustrated by figure 5.6. Each row will thus correspond to a feature vector for a given day, where the 24

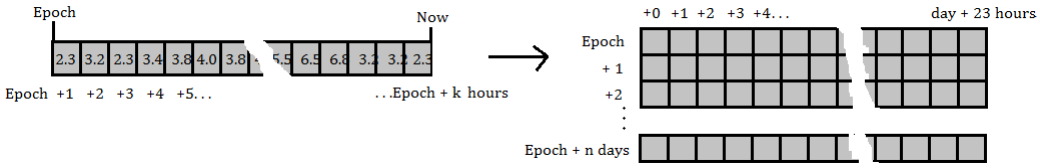


Figure 5.6: Illustration of feature extraction

Feature	Description	Feature number
raw_consumption	The raw hourly consumption figures during the day	1-24
mean	Average hourly consumption across entire day	25
peak	Maximum consumption observed for one hour	26
min	Minimum consumption observed for one hour	27
night	Average hourly consumption between 00:00 and 03:00	28
morning	Average hourly consumption between 06:00 and 12:00	29
afternoon	Average hourly consumption between 12:00 and 18:00	30
evening	Average hourly consumption between 18:00 and 00:00	31
mean/peak	The ratio between features mean and peak	32
min/mean	The ratio between features min and mean	33
night/day	The ratio between features night and day	34
time_of_peak	At which hour does the daily peak occur	35
hours_above_mean	For how many hours is the consumption above the daily mean	36
no_of_peaks	How many peaks occur during the day	37
day_of_week	Categorical features indicating day of week	38
date	The date for when these measurements were observed	39

Figure 5.7: Table of different features index

features are the hourly measured consumption during that day. Now for each day extra features are extracted and appended as extra columns to the original matrix. Table 5.7 shows a full list of the features that are extracted from the raw consumption figures of each day. Each feature is easily calculated in R by applying various arithmetic functions to each row of the matrix. Each feature is then appended as an extra column to that matrix, resulting in a final matrix with 39 columns. Columns 1 to 24 contain the raw consumption measures, while the extracted features occupy the other columns. Now when looking at an entire column alone, it can be considered a time serie itself with daily resolution. For instance column 25 will be a time serie of the average hourly consumption for each day since the *epoch* setting. Column 26 will be the peak consumption recorded for each day, and column 28 the average hourly consumption at night. Thus each column is considered a time serie in itself and is stored as an array in the *W_Feature* table along with ids for the building and consumption type (*water, electricity or heat*) that it describes. Thus the website can access this

$$\mu = \frac{\mu_1 \cdot n_1 + \mu_2 \cdot n_2 + \cdots + \mu_k \cdot n_k}{n_1 + n_2 + \cdots + n_k}$$

$$S_w^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2 + \cdots + (n_k - 1)s_k^2}{n_1 + n_2 + \cdots + n_k - k}$$

Figure 5.8: Subtype Profiles equation

table to fetch time series showing the progression of various features for the buildings.

5.7 Creating consumption profiles

To extract further information from the data a number of consumption profiles are calculated. The consumption profiles can be viewed mainly as two parallel vectors where one is the mean values of certain features, while the other contains the standard deviations for these means. Consumption profiles are generated on 3 different levels, and the procedures to generate them are only run once a week rather than daily.

Building profiles

A building profile is generated for each building and consumption type if there is enough data to do so. These profiles will contain the means and standard deviations of features 25 to 37. Thus a building profile summarizes data about a building's consumption patterns by calculating means and standard deviations for the various feature time series described in previous section. Thus a building profile would for instance contain data about the overall averages and standard deviations for a buildings hourly consumption, night consumption, day vs. night consumption ratio etc. The building profiles are then stored in the *W_Building_Profiles* table, ready for the website to access.

Subtype profiles

The subtype profiles are created by grouping building profiles which belong to buildings of the same subtype (eg. schools, offices, nursing homes etc). Combining the means (my) and standard deviations s would normally be done with the formulas in figure 5.8, where n is the number of observations in each sample, which would be the number of days.

However, this approach is not used since we are not interested in the combined mean and standard deviation for each day across different buildings, but rather we will calculate new means and standard deviations using the means found in the building profiles as input. The subtype profiles are then saved in table *W_Subtype_Profiles*.

Daytype profiles

When it comes to energy and water consumption most buildings will have at least two different day types; workdays and weekends. On each type of day the consumption will follow a distinct pattern. For each building the distinct day types are identified, and a daytype profile modelling the consumption pattern is created. The day type profiles are created in four steps which include preprocessing, LDA, clustering and outlier detection. The matrix containing features for each day since epoch as explained in section *Extracting features*, is the input for these processing steps.

Preprocessing: Two adjustments are done in order to make similar days easier to detect. First the seasonality is removed by subtracting the minimum consumption for each day. Second, because all observation times are measured in GMT time, observation values measured during winter time are shifted one hour later. Otherwise the feature vectors of winter time days and summer time days would become out of sync, as people's routines would be one hour earlier during winter time.

Outlier detection: Using the PCout multivariate outlier detection algorithm[9] extreme outliers are removed from the dataset. The algorithm is applied to features 25, 26 and 27 (mean, peak and min), as at least one of them will be affected directly by an extremely high or low observed value at any given time during the day. It is important to remove outliers before performing LDA, because it is not a robust statistical method, and can potentially be highly affected by outliers.

LDA: Because the dataset is large, it is important that the optimal number of day types to use for each building, and the best features to use for the grouping are selected automatically. Based on the knowledge that similar consumption patterns mostly occur on the same day of the week, a linear discriminant analysis (LDA) was performed to find the linear combinations of features 25 to 37 that give the best separation between the classes. The maximum number of useful discriminant functions is the minimum of G-1 and p where G is the number of groups and p the number of features. Thus the LDA resulted in 6 different linear combinations of the features that could best separate the days of the week, and thus also different types of days as those will most often be on the same day of the week. Using the linear discriminant functions from the LDA, each day's features 25 to 37 were reduced into 6 features also called canonical variates (CV's).

Clustering Based on the CVs of each day a k-medoids clustering was performed. K-medoids was chosen over k-means because it performed well, and the implementation pamk in the R package fpc automatically detects the

[9] See Related work and Outliers

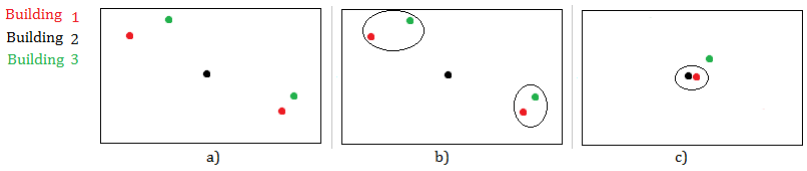


Figure 5.9: Building similarity

optimal number of clusters using the optimum average silhouette width which is a measure of how well the data is clustered [8].

Once similar days have been clustered, a day type profile is created for each cluster. This is done by finding means and standard deviations for features 1 to 37 and putting them in two parallel arrays. These arrays along with the building id and consumption type would be stored in the W_Consumption_Profiles table. An array of length 7 is also stored, which indicates the occurrences of each weekday in this pattern. Additionally a weight for each profile is computed as $(\frac{total\_days}{days\_in\_cluster_n})$. For instance if 200 days were initially used for the clustering and clusters A and B were created with 50 and 150 days respectively, then two day type profiles would be created with weights 0.25 and 0.75.

5.8 Comparing buildings

To identify buildings with similar consumption patterns the daytype profiles are used rather than the building profiles. The system uses an approach similar to the K Nearest Neighbor method to find similar daytype profiles. Figure 5.9 illustrates with a fictitious example how using the daytype profiles is different from using the building profiles to compare buildings. In figure 5.9 a) 5 daytype profiles belonging to 3 buildings are shown in a two dimensional space (each profile is assumed to be of equal weight here). Building 1 and 3 have two day type profiles each which are located pairwise near each other, while Building 2 only has 1 day type profile. Now using proximity as a measure of similarity we can see in b) that if we compare based on the daytype profiles Building 1 and 3 would be similar to each other. If we average the day type profiles belonging to the same building we would have essentially the building profiles. Figure 5.9 c) shows that if those were used for comparing that now Building 1 and 2 would appear similar. This demonstrates that the two approaches render different results, and that by using the building profiles two buildings with very different consumption patterns can appear to be similar because of the averaging of features. Thus the comparison is done based on all the daytype profiles for each building.

First a distance matrix between all day type profiles is computed using euclidean distances. The same canonical variates used for clustering are used to compute the distances. The smaller the distance the more similar the profiles

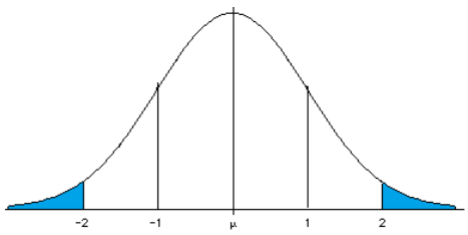


Figure 5.10: Two tailed cutoff on normal distribution

will be. However as day type profiles only partly represent the full consumption pattern of their buildings, both each row and each column is divided by the weight of the profile at that row or column. This means that each measure of distance in the matrix is adjusted by the weights of both the two profiles being compared. For each building the distances across all its profiles to other profiles are summed. Then picking the profiles with lowest distance measure will be for the buildings that are most similar. The 30 profiles with lowest distance are picked out and the ids of the buildings behind those profiles are stored in an array which is saved to the *W_Similar* table from where the website can access the results.

5.9 Detecting faults

The detection of faults is based on the assumption that day days of similar types have variables that are each normally distributed. This assumption has been verified using a QQ-plot. The daytype profiles contain means and standard deviations of all variables for each type of day. With this information the following hypothesis test can be constructed for any daytype profile type and a given day

- H_0:*day* is of *type*
- H_A:*day* is not of this *type*

A two tailed hypothesis test is then performed using Z-scores to get the p value of this hypothesis test. First the Z score of each variable i is computed with:

$$Z_i = \frac{x_i - \mu_i}{\sigma_i}$$

where x_i is the value of variable i for day, μ_i and σ_i is the mean and standard deviation of variable i from the daytype profile. With the Z score the p value π for variable i can be calculated for the two tails as illustrated in figure 5.10: To get the total p value p_{total} across all variables the following equation is used:

$$p_{total} = \prod_{i=1}^{N} p_i$$

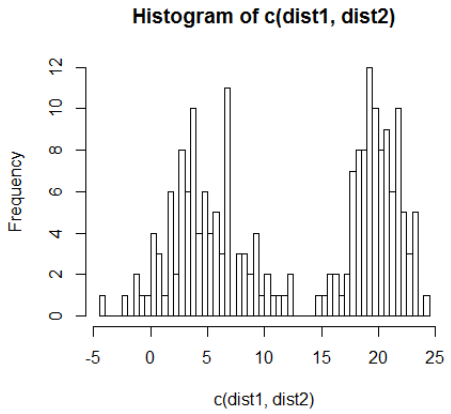


Figure 5.11: Two different normal distributions

where N is the total number of variables.

The significance level alpha indicates when to reject H_0 and say that a given day does not conform to a certain day type profile. A low alpha would decrease the chance of false alarms but also decrease the chance that true anomalies are detected (ie. the chance of making type 2 errors increase). On the other hand a high alpha value increases the chance of detecting anomalies, but potentially increases the number of false alarms (ie. type 1 errors). H_0 is rejected when following equation evaluates to true:

$$p_{total} < \alpha^N$$

where N is the total number of features.

If all H_0 is rejected for all day type profiles the tested day is considered an anomaly, and the variable with lowest p value is flagged as a probable cause. The reason why a day is checked against every daytype profile individually rather than against a single building profile, is because the hypothesis test assumes that the means and standard deviations of variables come from a normal distribution. This would not be the case if a building's consumption comes from two or more distinct day types that each have normally distributed variables with different means and standard deviations. This is illustrated in figure 5.11, where two different normal distributions are plotted together, showing that the result is not normally distributed.

The features to use to detect anomalies and the α value can easily be configured by users through the website as the settings are stored in the *W_Settings* table. This will allow users to tweak the power of the test. The standard configuration is to use α 0.5 and two features, the average consumption across the entire day and the consumption at night. The first feature will detect an anomaly if the total consumption goes up or down for any reason, while the

night consumption feature would be particularly sensitive if standby consumption of a building changes. When an anomaly is detected an alarm is created and stored in the *W_Alarm_Odd_Consumption* table, where it can be accessed by the website.

5.10 Design of the website

This section will describe the design of the website. It will clarify which perspectives have been in focus, what technologies have been used. Lastly will it describe the different elements of the website.

The use case diagram from the Requirements section gives an overview of the requirements. Further does it show a grouping of the requirement into two main concepts: *Viewing a building* and *Comparing*. In relation to user interface design Steve Krug, argues that clearly defined areas and hierarchies aids the users learnability of a system and supports from getting lost [9]. Based on this, these identified concepts become important for the organization user interface, as these concepts can serve as groupings. Since the website should provide the user with a considerable amount of information, ordering and displaying it in a simple way becomes important.

In addition, each of the use cases relates to different purposes when using the system. In order to keep it simple and support the user in quickly gaining access to the information needed to perform a certain task, keeping each sub use case visually apart from each other may ease the navigation. Furthermore, by keeping them apart is it possible differ their appearance and thereby making them more memorable for the user and minimizing the risk of a user confusing/mixing up information presented.

Viewing a building

The one thing all use cases of viewing a building have in common is that they, to a certain extend only display information, and require little or no interaction from the user. Based on the findings from the previous section and with this in mind, it has been decided create and display each use case in separated bordered panels that clearly encapsulate and separates them.

Comparing buildings

Comparing buildings, in contrast to viewing a single building, requires a higher level of interaction from the user. Further, the first four use cases can be generalized into a concept of *what to compare*, and the last can be viewed as *how to compare*. This generalization, enhance the flexibility of the system, since it allows buildings, types and categories to be treated as one. Doing this, will allow comparison between buildings and types, categories and types or categories and buildings with a uniform GUI. With a uniformed GUI for the four types of

content to compare, the number of steps the user have to learn is minimized, making it easier to learn, but also easier to remember. Based on this it is now possible to define three areas of the page:

- The content to compare.
- The features on which the user compare content.
- The visualization of the comparison.

In order to clarify the placement of these areas Jeffrey Venn extension to Keith Instone theory of the fundamental visibility of a system have been used [9]. Jeffrey Venn extension associates Keith Instones three question (*Where am i?, What is here? and Where can i go?*) with three areas on a page. In relation to this the content to compare have been associated with *Where can i go?*, to encourage the user to explore the data. Since the *Where am i?*, continuously throughout the website is have been allocated for navigation and information about where on the website the user is, the features to compare have been chosen to be put into main content area along with the visualization of the comparison. These two sub areas of the main content area, have been clearly divided, but still placed close together to insure the user catches the relation between them.

Flow of website

Based on the concepts and their hierarchy, discovered in previous section, the flow of the website was developed see figure 5.12. The page Building and Compare are directly causes of the groupings into hierarchies discovered, whereas the alarm pages originates from requirement *F3-02*[10]. The homepage have been created as an control and navigation page. In addition, the requirement specification expresses a need for user policies and restricted access to some features, like validation of data and handling of alarms. In order to accommodate these requirements the login and signup page have also been added.

5.11 Implementation of website

Template panels and visual components

As mentioned previously, as a design choice, it have been chosen to visibly keep different functionality clearly apart. From a system design perspective, standardizing these functionalities into components gives the software system have many advantages. Firstly, the system becomes more flexible and enhance the possibilities for code reuse. Further, by standardizing how components are created lowers complexity and easen development at a later state in the software

[10]See requirement specification.

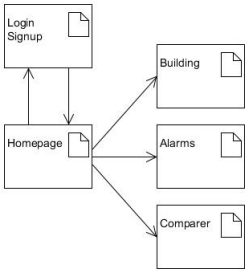


Figure 5.12: Flowchart of the website

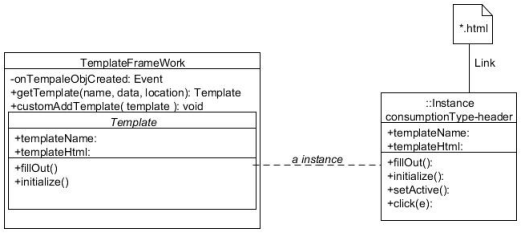


Figure 5.13: Template framework

life cycle. In addition, by standardizing reusable components the system becomes more easily changeable. Lastly by creating a "toolbox" with standard components, the view and display of information is more easy to customize to the user, if for example different users should have different views of the information. Further does it allow for generalizing functionality for collapsing, closing and reorganizing different elements on a given page.

For these reasons it have been chosen to design and implement a simple template framework (working as a factory pattern), where a controlling module contains a list of templates and how to instantiate them. Furthermore does the template framework contain a class which all templates extends. As seen in figure 5.13, an instance of the Template object contains a link or references to a HTML part. When an instance is created this HTML part is firstly cloned, and then filled out by the instance object with the provided data. The instance object, besides how to fill out the HTML, also contains all functionality regarding behavior and events. Lastly, to reduce coupling and make each component independent of other components all intercommunication between components is intended to work through events.

Besides this template framework, to ensure a rapid development of the prototype a number of libraries and technologies have been used.

```
4   //load feature for building
5   window.df.apis.postgres.getRecords(
6                 {'table_name':'w_feature',
7                  'filter': 'building = ' + buildingId + ' AND '+
8                            'consumption_type = ' + window.activeConsumption+' AND ' +
9                            'feature_type IN(' + String(activeFeatures) + ')',
10                 'fields':'feature_type, feature_data, consumption_type'
11                }, initializeGraphOnLoad);
12
13
```

Figure 5.14: Code snippet of how to retrieve data

Technologies

Firstly, as some of the most commonly used, JQuery[11] and Bootstrap[12] have been integrated. JQuery have been used to lessen the code needed to be written and to more easily handle events and animations. Further the Bootstrap framework for styling and in order to make the application more responsive and lean towards a mobile first development to ensure portability. In order to create visualizations of the data the D3[13] (Data Driven Documents) library have been used. Lastly for backend and client communication Dream Factory[14] have been used. This open source platform, does with little configuration, create a RESTful service and API to database. Thereby enabling the front end application to get and set data from the database through the API. After a session have been established, through a login (either by an anonymous or by a registered user profile) data can be retrieved the client SDK.

The code snippet in figure 5.14 shows how the Comparing buildings page gets data for the charts. The *getRecords* method takes two parameters: an object containing information of what to retrieve and where to retrieve it, and a callback-function. The object-parameter has numerous configuration parameters. Mainly, in the implementation of the web-site, only three is used (*table_name, fields and filter*), which specifies the content of a basic SQL select-statement.

For further description of the implementation of the website see Appendix 06 -Implementation of the Website. This appendix discuss the implementation of the requirements on the website, the structure of the HTML and the relation between the different HTML files and JavaScripts files.

[11] https://jquery.com
[12] http://getbootstrap.com
[13] http://d3js.org/
[14] https://www.dreamfactory.com/

Chapter 6

Test and evaluation

Chapter 7

Conclusion

Bibliography

[1] Christian Beckel,Leyna Sadamori and Silvia Santini, "*Towards Automatic Classification of Private Households -Using Electricity Consumption Data*", Buildsys'12, November 6, 2012, Toronto.

[2] Joern Ploennigs, Bei Chen, Anika Schumann and Niall Brady, "*Exploiting Generalized Additive Models for Diagnosing Abnormal Energy Use in Buildings*", Buildsys'13, November 13-14-2013, Roma, Italy.

[3] Xiaoli Li, Chris P. Bowers, and Thorsten Schnier, "*Classification of Energy Consumption in Buildings With Outlier Detection*", IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, Vol. 57, No. 11, November, 2010.

[4] Imran Khana, Alfonso Capozzolia, Stefano Paolo Corgnatia and Tania Cerquitellib, "*Fault Detection Analysis of Building Energy Consumption -Using Data Mining Techniques*", Elsevier Ltd, 2013.

[5] John E. Seem, "*Pattern recognition algorithm for determining days of the week with similar energy consumption profiles*", Elsevier B.V, 2004.

[6] Peter Filzmosera Ricardo Maronnab, MarkWernerc, "*Outlier identification in high dimensions*", Elsevier B.V., 2007

[7]

[8] Peter J. Rousseeuw (1987). "*Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis*". Computational and Applied Mathematics 20: 53–65

[9] Yvonne Rogers, Helen Sharp and Jennifer Preece, "*Interaction Design, -beyound human-computer interaction*", Wiley, 3rd Edition, 2012

Appendix A

Appendix awesome

This is a nice appendix thing

Appendix B

Appendix not so awesome

This is a nice appendix thing