Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Utilization of business intelligence principles for energy data processing

Jakub Srna

Supervisor: Ing. Jan Šulc
May 2017

Acknowledgements

The offer of this work would like to acknowledged the support of PhD. Jan Široký and Ing. Marek Kulvejt for contributing their vast experience and knowledge as well as Ing. Jan Šulc supervising this work without which this work would have never seen the light of day.

Declaration

I hereby confirm that my thesis entitled "Utilization of business intelligence principles for energy data processing" is the result of my own work. I did not receive any help or support from commercial consultants. All sources and materials applied are listed and specified int he thesis.
Furthermore, I confirm that this thesis has not yet been submitted as part of another examination process neither in identical nor in similar form.

In Prague, 23. May 2017

Abstract

The central goal of this work will be the development of a software tool which will be used for data analysis by Energocentrum PLUS, s. r. o., within the context of the newly emerging field of Energy intelligence. Energy Intelligence describes using big data analysis and data scraping methods to streamline the performance of controlled energy systems, such as heating or air conditioning.
This work will also introduce two custom built structures named "Modulo Carpet" and "Scatter Plot", both were inspired are the result of extensive research into the field of Business Intelligence, which results and process are detailed as well.
The research was conducted in order to obtain a greater understanding of the concepts, ideas, and logic which form the foundation of a good BI (business intelligence) tool, in order to improve the functionality of our custom tool.
The field of Business Intelligence was picked due to the field's long-term development of data analysis tools and their diversity. The top three tools of the various tools covered in our research were compared to an average one in order to enlighten the reader in regards to each tool's strengths, weaknesses, and general functionality and effectivity.
Following the research is an in-depth review and examination of the customized analytical tool being developed, as well as various tests, conducted on actual company data, in order to discover flaws, faults, and deficiencies early, so as to have sufficient time to fix each of them and assure the software remains effective in the future. The work will then be summarized along with evaluated results from field tests of the developed custom application.

Keywords: Energy intelligence, Business intelligence, development, analytical tool, software architecture, software development, Big data, data analysis

Supervisor: Ing. Jan Šulc
Oddělení průmyslové automatizace,
Fakulta Elektrotechnická,
Karlovo náměstí 13,
12000 Praha 2

Abstrakt

Cílem této práce je vývoj softwaru, který v budoucnosti bude sloužit pro analýzu dat společnosti Energocentrum PLUS, s. r. o., v rámci nově vznikajícího konceptu Energy intelligence. Tento koncept seskupuje analýzy energetických dat za účelem zefektivnění navržených řídicích systémů. Po představení analytických struktur Modulo Carpet a Scatter plot následují výsledky výzkumu provedeného v oblasti Business intelligence, který byl proveden za účelem získání podkladových materiálů pro vývoj vlastní aplikace. Oblast Business intelligence byla vybrána z důvodu dlouhodobého vývoje analytických nástrojů. Tři nejlepší z nich byly porovnány s jedním průměrným za účelem srovnání poskytovaných funkcionalit a jejich implementace.
Po výzkumu následuje popis vývoje analytického nástroje, který je v praxi průběžně testován na datech společnosti za účelem odhalení nedostatků, které by v budoucnu mohly způsobit, že se daný software stane nedostačujícím. V závěru práce je umístěno shrnutí obsahující získané výsledky pomocí vyvinuté aplikace společně s jejich zhodnocením.

Klíčová slova: Energy intelligence, Business intelligence, vývoj, analytický nástroj, Softwarová architektura, Big data, analýza dat, vývoj softwaru

Překlad názvu: Využití business intelligence principů pro zpracování energetických dat

Contents

Figures

Tables

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Srna** Jméno: **Jakub** Osobní číslo: **420203**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Využití business intelligence principů pro zpracování energetických dat

Název bakalářské práce anglicky:

Utilization of business intelligence principles for energy data processing

Pokyny pro vypracování:

1. Seznamte se s principy tzv. business intelligence nástrojů (konkrétně programy Tableau, PowerBI a Qlik) a navrhněte možnosti využití business intelligence principů pro zpracování energetických dat.
2. Navrhněte architekturu aplikace typu server-klient pro výše zmíněné použití.
3. Navrženou aplikaci implementujte.
4. Demonstrujte možnosti vyvinuté aplikace s využitím reálných energetických dat.

Seznam doporučené literatury:

[1] Business Intelligence, Jak využít bohatství ve vašich datech, David Slánský, Jan Pour a Ota Novotný
[2] Handbook of web based energy information and control systems Autor: Capehart, B. L., Middelkoop, Timothy
[3] Data science for business Autor: Provost, Foster, 1964-, Fawcett, Tom

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Šulc, UCCEB Buštěhrad

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **17.02.2017** Termín odevzdání bakalářské práce: **26.05.2017**

Platnost zadání bakalářské práce: **30.09.2018**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Chapter 1

Introduction

Nowadays, many businesses around the world own and operate a variety of tools to analyze their respective customer data and draw conclusions from it. These conclusions grant the businesses the ability to personalize their products and adjust their various services to each of their consumer's specific wants and interests and to maximize profits. The desire to personalize services and maximize profits are common across businesses from all around the world. However, what should a company do if it wishes to provide such a specific service, that the support structure for it, such as analytical tools, is poorly built or does not even exist yet?

Energocentrum Plus s.r.o. currently owns such a poorly developed tool, which is an analytical tool which serves to analyze heating data gathered from the many buildings to which they provide services and for which they design intelligent heating systems.

Unfortunately, as the potential of the company continues to grow, the demands from the current tool overwhelmed its capabilities.

This thesis describes the development of a new analytical tool, which later on will serve as key software for data analysis, and is meant to be used by Energocentrum Plus s.r.o. company in the future.

Therefore, in this thesis work, we will attempt to design a tool to provide Energocentrum Plus s.r.o. employees with sufficiently robust and flexible software, so that they may achieve better results and advance the company's interests in the field. In addition, the software must be structured in a user-friendly fashion as to allow high editability in the future should the company need to modify the tool to meet its changing needs.

The beginning of this work will introduce the various data structures and graphs which are currently used by the company, followed by the definition of the concept called "Energy intelligence".

The sections following the previous definitions would cover an extensive review of the various currently available tools in the field of Business intelligence in order to provide a deeper understanding of the requirements and capabilities of tools similar to the one designed in this work.

After designing the general architecture of our tool, we will discuss, dissect and explain the various parts described in chapter three in order to provide sufficient insight and shed light on the complexities of the designed tool.

The description of the designed tool is followed by a summary and evaluation of the software's capabilities in chapter four, where the future possibilities and direction of the developed application are discussed.
This document will then come to a close with an extensive summary of the work.

Chapter 2

Comparison of Business Intelligence tools

For the last twenty years, companies in almost every industry invested time and money towards improving their ability to collect data about their customers and to exploit the collected data to gain a competitive advantage against each other. As a result of significant progress in the field of computer science and the growth in computational power, the volume of collected data surpassed the capability of teams of statisticians employed to extract information and knowledge from large datasets by manual analysis. As the problem grew beyond the reach of human manual labor, it inspired the development of algorithms to scan and interact with multiple databases, to enable deeper and more thorough analysis than previously possible. These algorithms formed the basis of several computer programs which we now call Business Intelligence tools (or "BI tools" for short).
The concept of extracting useful information from massive datasets, often called Data Mining, attracted many companies such as Oracle or Microsoft to invest their time, money and human resources to develop their own BI tools, driven by the promise of high rewards from renting out these tools. In the first part of bachelor thesis, we will discuss four tools called Tableau, Power BI, MicroStrategy, and Qlik. According to the Gartner diagram depicted in figure 2.1, Tableau, Power BI and Qlik are currently the market leaders in BI tools in 2017 which provokes questions about what makes these tools unique. To assure a proper comparison a basis in the form of MicroStrategy tool is introduced. Comparison of their performance is made in six different categories:

- Intuitiveness of control - are these programs user-friendly?
- Ability to work with large datasets - are these programs fast?
- Built-in functions - are these programs robust with many tools?
- Writing own functions - is it easy to write scripts in these programs?
- Availability of information materials and tutorials - is it easy answer questions which rise during the development process?
- Ability to achieve the set goal - does the result meets the predefined standarts?

Figure 2.1: Gartner magic quadrant for Business intelligence 2017 [7]

2.1 Definition of term Energy intelligence

After gaining a brief insight into what Business intelligence means, now we proceed to define the concept of "Energy intelligence", a concept which was formed to cover a previously undiscovered field of analysis.
Energy intelligence is nowadays used by companies aiming to design better intelligent building systems, to cover the use of data analysis to reveal faults or malfunctions of a smart building's internal systems which waste energy. To prevent such a situation, a system of sensors is put in place to measure different parameters such as temperature, humidity, or power use of the heat pump, store the results in a database, and later retrieve the results for analysis. The current leaders in the field of Energy intelligence are detailed in the following table:

Company name	Tool	Headquarter
Bulding IQ	Building IQ 5i Platform	USA, Austrálie
C3 IoT	C3 Enterprise Energy Management Platform	USA
Cimetrics	Energy Kiosk and Displays, Analytika	USA
Copper Tree Analyst	Kaizen	Kanada
Ecova	Continuous Building Optimization	USA
Energy Print	Energy Print	USA
EnerNOC	Energy Intelligence Software	USA
Ezenics	Ezenics	USA
KGS Buildings	Clockworks	USA
Retroficiency	Retroficiency Dashboard	USA
SkyFoundry	Sky Spark	USA
Wegowise	WegoPremium	USA

Figure 2.2: Companies analyzing energy data

In this work we use two main data structures to reveal patterns in measured data. The first structure is a table called "Modulo Carpet" which axes are formed by two different time intervals extracted from the timestamp of a sample, such as hours and weekdays as depicted below. The sampled data is always subjected to only one aggregation function at a time.

AVG Actual power	Hour																							
Weekday	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	308.68	305.18	305.55	304.86	325.99	339.17	373.91	423.08	476.82	518.46	527.81	519.32	506.85	510.97	505.94	504.40	490.66	473.90	436.04	400.67	378.87	342.01	326.67	322.49
2	317.55	315.80	314.46	314.51	334.19	346.11	378.97	434.39	488.98	540.25	552.28	544.87	530.95	533.89	526.69	518.65	497.12	479.56	442.72	403.60	377.30	343.28	328.54	323.21
3	319.23	316.00	315.30	315.33	333.50	346.53	377.04	437.98	496.40	542.28	552.60	540.23	526.45	522.97	514.69	509.19	495.06	474.65	442.39	405.14	376.88	343.26	328.27	323.09
4	318.59	315.58	315.73	315.58	334.78	346.18	376.62	428.30	475.30	518.10	525.22	513.94	504.07	498.40	491.58	486.26	474.90	463.49	428.29	398.57	379.26	347.17	328.89	322.54
5	318.77	314.89	313.84	314.37	333.02	344.45	369.51	417.78	459.66	487.40	488.93	469.87	450.98	433.64	419.51	399.62	386.74	378.51	366.20	357.04	349.38	328.05	317.16	312.50
6	308.45	303.67	302.81	302.32	302.79	301.95	303.83	302.75	297.32	287.68	281.30	277.59	276.06	280.62	285.19	293.35	301.40	310.41	312.31	312.08	309.74	308.75	306.91	305.68
7	304.06	302.20	299.79	299.01	298.97	298.45	297.49	296.50	290.96	283.11	275.79	272.97	272.77	276.43	279.51	291.21	302.07	309.02	313.93	314.34	312.31	311.33	310.49	309.13

Figure 2.3: Example of modulo carpet

The "Modulo Carpet" mentioned above illustrates a real situation in which we average the power used by a heat pump at CTU building and present the results for easier analysis. Thanks to the intuitiveness and simplicity of the aforementioned structure, it is easy to gain an extensive overview of the entirety of the examined period, and in most cases detect and prevent the various malfunctions and issues which may rise from an extended use of the heat pump or from wrong initial settings.
The "Modulo Carpet" structure is also capable of simplifying massive amounts of data, for example summarizing years worth of data and categorizing it according to the days of the week or month in which it was measured. For time intervals which are longer than one month, the graph becomes unintelligible and devoid of useful information. It is therefor recommended, for maximal clarity, to use the structure to present data in one of the preset formats of hours, days, weeks, or month. As we can see the heating system is set well because the largest values of the power are found in days and hours during which people are present at school.
The second data structure is named "Scatter plot", also known as a scatter diagram. Similarly to the aforementioned Modulo Carpet, the Scatter plot uses two variables to separate data samples - in the case of this work variables such as average outdoor temperatures, Instantaneous power, etc. In case that the samples are meant to be colored differently, a third variable will

be included. In contrast to "Modulo Carpet", each axis can use a different aggregation function. An example of such a scatter plot is introduced below.

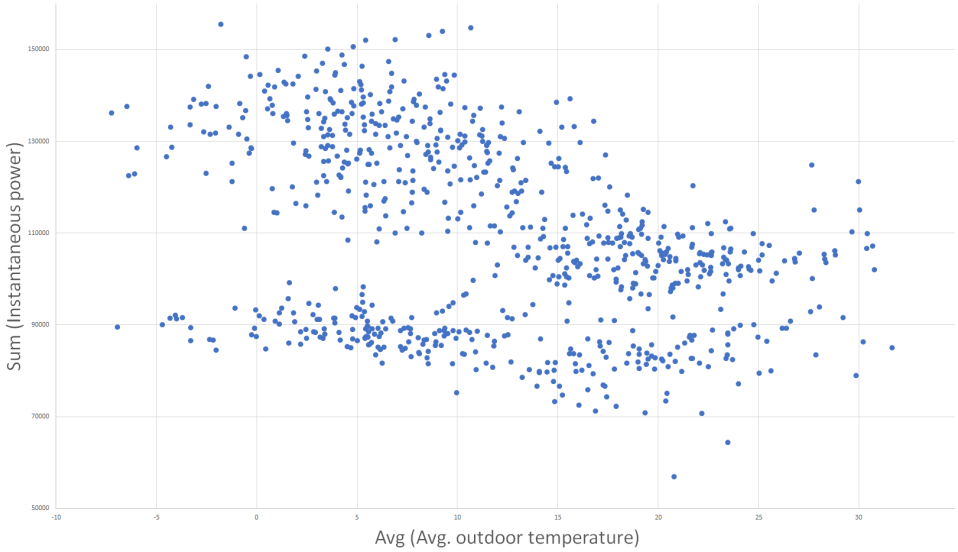


Figure 2.4: Example of scatter plot

After introducing these two key components necessary for the analysis of the field of Energy intelligence, we will proceed to survey the Business intelligence tools mentioned at the beginning of this chapter and compare their performance. It is important to note that many of the currently available tools are programmed and optimized for different types of analysis, so there is no simple way to compare and contrast them. All tools together are examined in very specific field of analysis.
At the end of each review of the separate Business tools, a Modulo Carpet and a Scatter plot are shown to illustrate the capabilities of that particular tool and provide a screenshot of the whole program environment.

2.2 Tableau

Tableau was established in January 2013 in the US by three Standford university professors: Pat Hanrahan, Christian Chabot, and Chris Stolte.
The current version of Tableau, version 10.1, offers the option to import data from a variety of file types, such as CSV, JSON, Statistical file or even an excel sheet. Also, there is no need to download vast datasets from a database, because Tableau enables a direct connection to, and manipulation of, the database itself.

User friendliness

Since the start of its development process, the Tableau team shared a common vision of making data understandable to ordinary people.

This, in turn, resulted in Tableau itself being very intuitive and user-friendly. One of the many advantages of this tool is the common use of the "drag and drop" approach used throughout the program. For example, importing data from files from one of the formats mentioned above can be done by simply dragging and dropping the file into the program window. Nowadays such functionality may be considered elementary and even expected, but no other software included in this research had this feature at the time of revision.
Is Tableau ready for 4K displays? Extensive examination assures us that it is. All menus and graphs were perfectly readable, despite minor scaling problems such as missing bottoms of letters in the various dialog windows.

Large datasets

Tableau's ability to analyze vast amounts of data in reasonable time makes it a powerful tool and the current market leader in its field.
In 2003 Tableau made a breakthrough with the invention of a query language called VizQL. According to the official company web page, VizQL "...is a patented query language that translates your actions into a database query and then expresses the response graphically." The fundamental innovation of Tableau was its "...ability to do an ad-hoc analysis of millions of rows of data in seconds with Tableau's Data Engine". As the number of samples increase, the speed difference between Tableau and the other tools grows rapidly.

Built-in functions

A wide range of built-in functions, such as "WEEKDAY", save a lot of time for the average user. Moreover, these functionalities are easy to use and easy to access by two clicks at most. With the exception of Power BI, all tools contain built-in functions to complement work with dates, such as the aforementioned "WEEKDAY" function, or "MONTH", which extracts the required information from a timestamp. The difference is that Tableau's built-in functions are easily accessed trough a drop down menu while other tools usually store the definition of a function as a variable, which needlessly fills and clutters the workspace, impairs workflow, and wastes time, or has a just small list of basic functions with the rest being created by the user.

User-defined functions

Tableau's "Calculated fields", or user-defined functions, are conducive for filtering displayed data, providing new insights to the Dashboard, and provide almost limitless possibilities to explore the data stored in the Dashboard. All functions can be divided into seven elementary groups according to their use:

- Logical Functions
- Numerical Functions
- Date Functions
- String Functions
- Type Conversion
- Aggregate Functions
- User Functions

Materials

In addition to its main support, Tableau provides a free series of tutorial videos on their company's web page, which is also easily accessible by links from Tableau's start window. In case the tutorial videos are not sufficiently helpful and more insight into how to use the software is required, the company publishes many articles on their web page to provide as robust and diverse a support as possible. Moreover, Tableau has an enormous user community which expands on the tutorials made by the company through homemade videos posted on youtube or by opening new topics on the company's official discussion forum.

Test result

From both figures below we can see that the results obtained thanks to Tableau software are very similar to the exemplary figures in section 2.1.

Weekday ..	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
pondělí	308,7	305,2	305,6	304,9	326,0	339,2	373,9	423,1	476,8	518,5	527,8	519,3	506,9	511,0	505,9	504,4	490,7	473,9	436,0	400,7	378,9	342,0	326,7	322,5
úterý	317,5	315,8	314,5	314,5	334,2	346,1	379,0	434,4	489,0	540,3	552,3	544,9	530,9	533,9	526,7	518,7	497,1	479,6	442,7	403,6	377,3	343,3	328,5	323,2
středa	319,2	316,0	315,3	315,3	333,5	346,5	377,0	438,0	496,4	542,3	552,6	540,2	526,5	523,0	514,7	509,2	495,1	474,7	442,4	405,1	376,9	343,3	328,3	323,1
čtvrtek	318,6	315,6	315,7	315,6	334,8	346,2	376,6	428,3	475,3	518,1	525,2	513,9	504,1	498,4	491,6	486,3	474,9	463,5	428,3	398,6	379,3	347,2	328,9	322,5
pátek	318,8	314,9	313,8	314,4	333,0	344,5	369,5	417,8	459,7	487,4	488,9	469,9	451,0	433,6	419,5	399,6	386,7	378,5	366,2	357,0	349,4	328,1	317,2	312,5
sobota	308,4	303,7	302,8	302,3	302,8	301,9	303,8	302,7	297,3	287,7	281,3	277,6	276,1	280,6	285,2	293,3	301,4	310,4	312,3	312,1	309,7	308,7	306,9	305,7
neděle	304,1	302,2	299,8	299,0	299,0	298,5	297,5	296,5	291,0	283,1	275,8	273,0	272,8	276,4	279,5	291,2	302,1	309,0	313,9	314,3	312,3	311,3	310,5	309,1

Figure 2.5: Modulo Carpet - Tableau

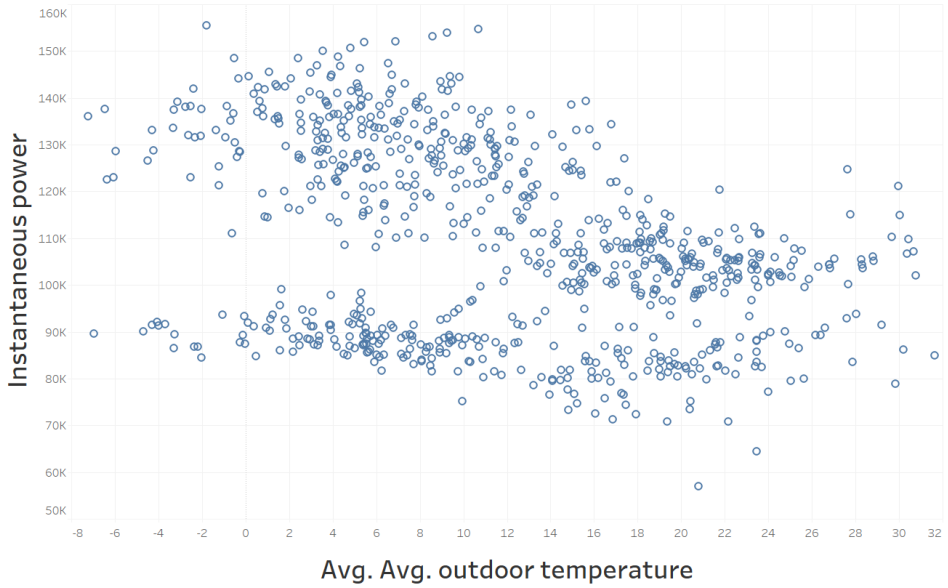


Figure 2.6: Scatter plot - Tableau

Conclusion

Tableau was really pleasant to work with and it definitely is the current leader among the business intelligence tools thanks to its user-friendliness, its use of its personal query language or the wide range of built-in functions which reduce the time needed to achieve the set goal.
In comparison to the other software tools presented in this research, the only competitive disadvantage is that Tableau is the only one which does not provide its basic version for free.

2.3 Power BI

Power BI is a business analytics tools developed by Microsoft in 2013. Functions Data from online services promises to be a rich source of information because many other online services are currently being developed with similar data capacities, such as GitHub or Azure Enterprise.

User friendliness

The Power BI tool is pretty similar visually to other software developed by Microsoft such as Word, Excel or PowerPoint, allowing an effortless transition from other software and a comfortable learning experience. Also, tool placement in the program window is familiar and convenient, which results in an intuitive experience without any unnecessary searching.
When it comes to visual adaptation to 4k screens, Power BI is the most compatible BI tool included in this research. There were no scaling issues, and everything was perfectly readable. Even letters in dialogue windows were

complete without any trimming.
However, the developers would be wise to consider improving the scaling of graphs or charts to fit into the program window in next version of the program. For example, When analyzing large tables, there is the possibility of missing a part of the graph due to it being hidden behind the side menu. Such issues can be solved manually by changing letter sizes, but even after these changes, the graph may still be covered and not fully available.

Large datasets

During research, no special query language used by Microsoft PowerBI was found. Regardless, data was processed quickly even in the case of large datasets. When compared to Tableau, it achieved almost the same data processing time. On the other hand, when compared to MicroStrategy or Qlik Sense, the difference in time required to process data was quite noticeable.

Built-in functions

Power BI's main disadvantage, compared to the other examined tools, is that it lacks implemented functions such as WEEKDAY or HOUR in the basic, unmodified program. Functions such as HOUR, MINUTE and SECOND can be computed quite easily in the same way as in Excel - however, WEEKDAY takes quite some time when written in name form rather than in numbers form due to Power BI not differentiating WEEK from its usual tendency to organize lists alphabetically. To summarize, working with Power BI is not as easy as working with Tableau, but thanks to Power BI's internal scripting engine, once one learns to write one's owns scripts Power BI offers the same flexibility as Tableau does.

Writing own scripts

Writing one's own scripts is painless and effortless in Power BI thanks to its similarity to Excel. Most of Excel's native functions are present in Power BI as well. The only drawback is that due to the lack of built-in functions there are plenty of new, unofficially designed functions in the workspace, making it a slightly confusing

Materials

In the matter of tutorial and study material availability, Power BI is on a level of its own. Everything that users need can be found on the Power BI web page for free in the form of video guides, articles, discussion forums, and webinars. Each of these provides new insights into this tool which is considered as the second best BI tool available on the market.

Test result

Microsoft's Power BI passed our tests marvelously, producing a result analogous to that of Tableau.

DAY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1. Mon	308,68	305,18	305,55	304,86	325,99	339,17	373,91	423,08	476,82	518,46	527,81	519,32	506,85	510,97	505,94	504,40	490,66	473,90	436,04	400,67	378,87	342,01	326,67	322,49
2. Tue	317,55	315,80	314,46	314,51	334,19	346,11	378,97	434,39	488,98	540,25	552,28	544,87	530,95	533,89	526,69	518,65	497,12	479,56	442,72	403,60	377,30	343,28	328,54	323,21
3. Wed	319,23	316,00	315,30	315,33	333,50	346,53	377,04	437,98	496,40	542,28	552,60	540,23	526,45	522,97	514,69	509,19	495,06	474,65	442,39	405,14	376,88	343,26	328,27	323,09
4. Thu	318,59	315,58	315,73	315,58	334,78	346,18	376,62	428,30	475,30	518,10	525,22	513,94	504,07	498,40	491,58	486,26	474,90	463,49	428,29	398,57	379,26	347,17	328,89	322,54
5. Fri	318,77	314,89	313,84	314,37	333,02	344,45	369,51	417,78	459,66	487,40	488,93	469,87	450,98	433,64	419,51	399,62	386,74	378,51	366,20	357,04	349,38	328,05	317,16	312,50
6. Sat	308,45	303,67	302,81	302,32	302,79	301,95	303,83	302,75	297,32	287,68	281,30	277,59	276,06	280,62	285,19	293,35	301,40	310,41	312,31	312,08	309,74	308,75	306,91	305,68
7. Sun	304,06	302,20	299,79	299,01	298,97	298,45	297,49	296,50	290,96	283,11	275,79	272,97	272,77	276,43	279,51	291,21	302,07	309,02	313,93	314,34	312,31	311,33	310,49	309,13

Figure 2.7: Modulo carpet - Microsoft PowerBI

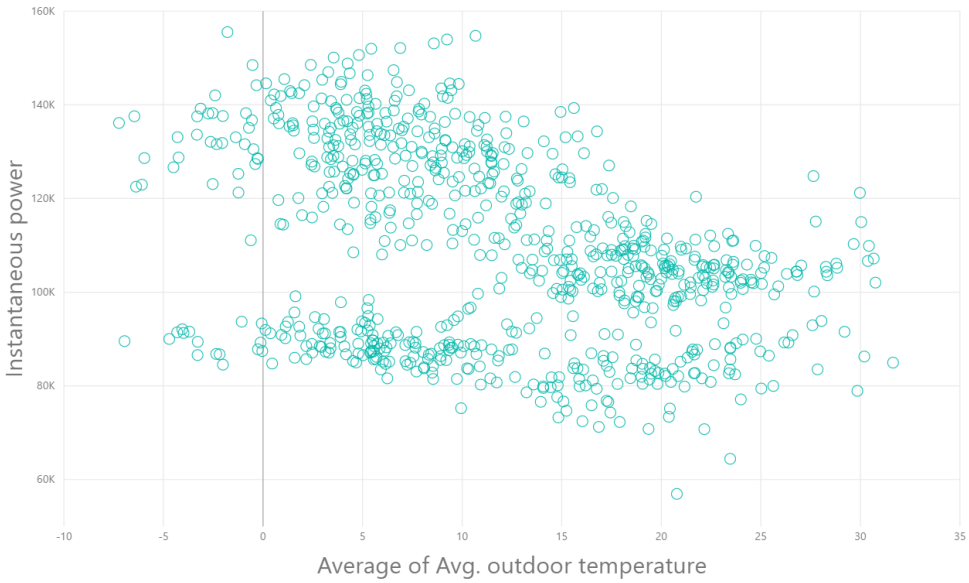


Figure 2.8: Scatter plot - Microsoft PowerBI

Conclusion

To summarize, Microsoft's Power BI is a useful and powerful tool for analyzing data in reasonable time. Unlike Tableau it does not have its own query language, and some of its features are not quite as elegant, but Power BI provides its software version for free. Moreover, Power BI has an equivalent mobile version which is free as well and is very useful in case of limited access to a computer.

2.4 Qlik Sense

Inspiration, Imagination, and Innovation are the three main drivers of the company behind "QlikView", a product for data viewing, and "Qlik Sense", which allows full manipulation of data - both of which are business intelligence tools created by Qlik. Qlik, the company behind the development of Qlik Sense, was established in 1993 in Sweden, from where they expanded to their current company headquarters which is now in Pennsylvania, USA.

User friendliness

Upon startup, Qlik Sense seems very intuitive and user-friendly. The first problem to present itself is the lack of support for 4k displays - everything in the program window seems to be oversized or too small, which did not help orientation around the program. Several components which could normally be accessed by clicking on them would occasionally disappear, and some were hidden for a time, or that by chance that specific part of the screen was flashing for no apparent reason.
As in Power BI described in section 2.3, fitting a whole table into the program window was troublesome. In Qlik Sense there is no option to change the size of letters, which means that viewing a whole table is not at all possible. On the other hand, such a problem can be solved by the vertical and horizontal slider.

Large datasets

Qlik Sense is far behind Tableau and Power BI when tackling large datasets. In the case of the Scatter plot described in section 2.1, it took almost 15 seconds to get the results from the dataset containing about 220,000 samples compared to Tableau's runtime of fewer than 0.5 seconds.

Built-in functions

There are not enough built-in functions which can be easily found on the menu. Nearly the only functions which are available are SUM, COUNT, AVG, MIN and MAX - all the other functions have to be written by the user.

Writing own functions

Writing your own functions in Qlik Sense is very intuitive and user-friendly. Most of the functions are actually written in the same format as in Excel, which is very helpful. The main advantage of Qlik when compared to Power BI is its function "WEEKDAY". This function has to be written by the user as a script, but it has the letter-based format at the correct order of days in the week, unlike Power BI's alphabetical order. Writing new functions in Qlik is very simple and intuitive - a click of a button brings up a new window where a whole function may be written. Another button applies that function

to your command window. Qlik Sense lacks the ability to store the definition of functions - which on the one hand is good, due to limited space in the program window, but on the other hand, especially when one definition of the function is meant to be used in more than one place, copying it over and over again is quite unpleasant.

Materials

There are three main sources of information about how to use Qlik Sense. The first is directly from the company's web page, where a lot of information is offered in the form of videos, virtual instructor-led training or a so called "Continuous Classroom". The second is to attend Instructor-Led Training which can be held publicly in public classrooms or privately in a company facility.
Last is the Qlik official youtube channel where Qlik updates information and tutorials about new features presented in the latest version of software.

Test results

When attempting to determine whether or not Qlik Sense achieves its goal, the conditional formatting for Modulo Carpet is glaringly missing - a substantial drawback in this research.
Scatter Plot leaves more room for optimism as it looks the same as it should.

	Hour(Date)										
WeekDay	0	1	2	3	4	5	6	7	8	9	10
po	414866	410158	410665	409738	438134	455838	502540	568617	640841	696815	709378
út	422976	420648	418855	418928	445143	461015	504788	578608	651323	719615	735642
st	425209	420913	419980	420022	444220	461574	502219	583388	661204	722311	736066
čt	427867	424143	424335	424134	449947	465272	506174	575633	638803	696328	705893
pá	428432	423212	421800	422516	447578	462941	496619	561495	617783	655065	657125
so	414556	408131	406971	406323	406948	405819	408350	406891	399592	386640	378061
ne	408656	406157	402915	401875	401810	401118	399826	398498	391055	380497	370665

Figure 2.9: Modulo carpet - Qlik Sense

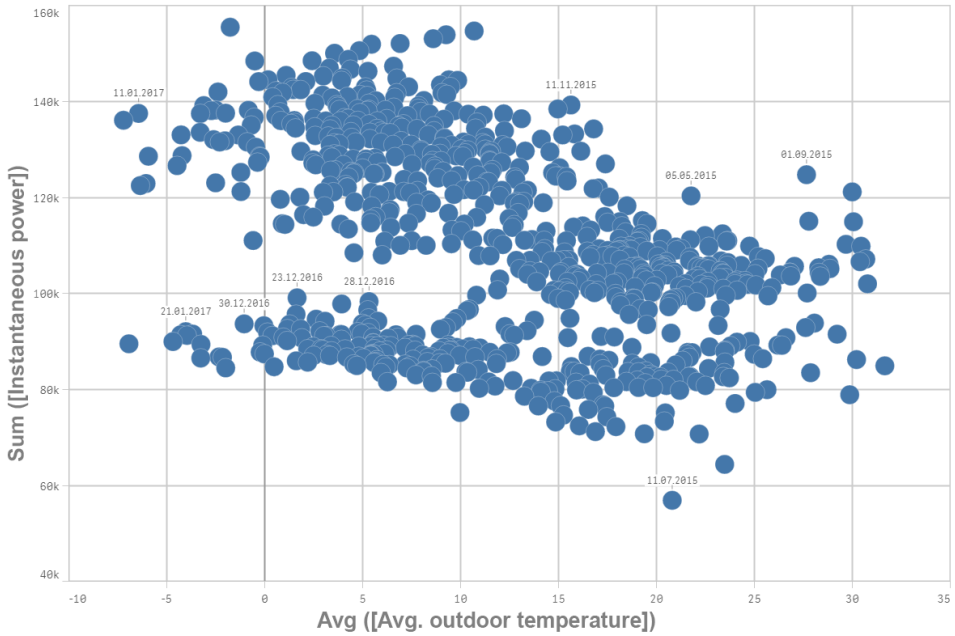


Figure 2.10: Scatter plot - Qlik Sense

Conclusion

To conclude, Qlik Sense is a very intuitive program with not too many built-in functions, placing it at a slight disadvantage when compared to Power BI or Tableau. Despite the rather negative overtones of the review, Qlik has several notable advantages, such as the ability of the user to earn an official "certificate of knowledge" of using the programs developed by Qlik, or provision of Qlik's basic version for free.

2.5 MicroStrategy

MicroStrategy desktop is an analytics platform developed by the American company Microstrategy which was established on November 1989.
As a business intelligence software, it supports a wide range of file formats, from files such as JSON, Excel, CSV, and Text, to even SAS. Similarly to every other BI software presented in this research, MicroStrategy enables the user to import data from a wide range of supported database formats such as SQL or IBM. Like Microsoft's Power BI, Microstrategy stands out due to its ability to extract data from a variety of different sources such as online services like Facebook and Dropbox, as well as its support for other BI tools formats such as SAP or BO.

User friendliness

The program itself looks bland and unadorned, without any special visual effects or a particularly pleasing design. This becomes particularly obvious when a user with a high-resolution 4k display uses the program - MicroStrategy is not even remotely optimized for these screens. Everything, including menus, variables, pushbuttons is so painfully small that working with such a program is often very unpleasant.

Large datasets

MicroStrategy's main issue is easily discernible from the way that it works with large datasets. When tackling large datasets, a dialog window appears with a warning message that dataset is too big and will be processed slowly. Such a warning did not appear in any other software included in this research.

Built-in functions

The level of built-in functions of Microstrategy lies between Tableau and other tools presented in this research. The currently available tool stores its functions in the form of variables called "Attributes". These attributes include functions for maintaining and supplementing work with dates. On the other hand, MicroStrategy desktop suffers from a lack of built-in functions to work and adjust the appearance of objects such as tables. Qlik Sense described in section 2.3 displays a similar problem.

Writing own scripts

Writing one's own functions in MicroStrategy is quite the same as in Qlik Sense, with the exception of Microstrategy's attribute-centric approach. Every used function has to have a pointer reference to the data it works with, which translates into a lot more attributes than desired stored in the workspace. The author found working in such an environment unpleasant.

Materials

Unlike most companies in the field, Microstrategy provides an in depth personal instruction in specialized training centers.
In case the customer is not able to visit company training center, said customer still has the possibility to watch numerous training videos on the official Youtube channel of the company.

Test results

MicroStrategy desktop exhibits the same issues as Qlik Sense does. There is no conditional formatting available for pivot tables which form the Modulo Carpet structure.
Furthermore, when the graph set is shrunk to a smaller size, the axes captions

stay in the same position they occupied previous to the decrease in size. This is the reason why the only Scatter plot presented in this research without any captions is the one generated by Microstrategy.

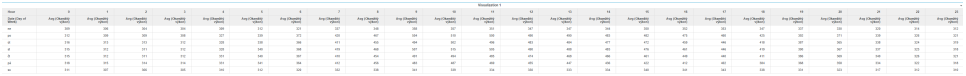


Figure 2.11: Modulo carpet - MicroStrategy

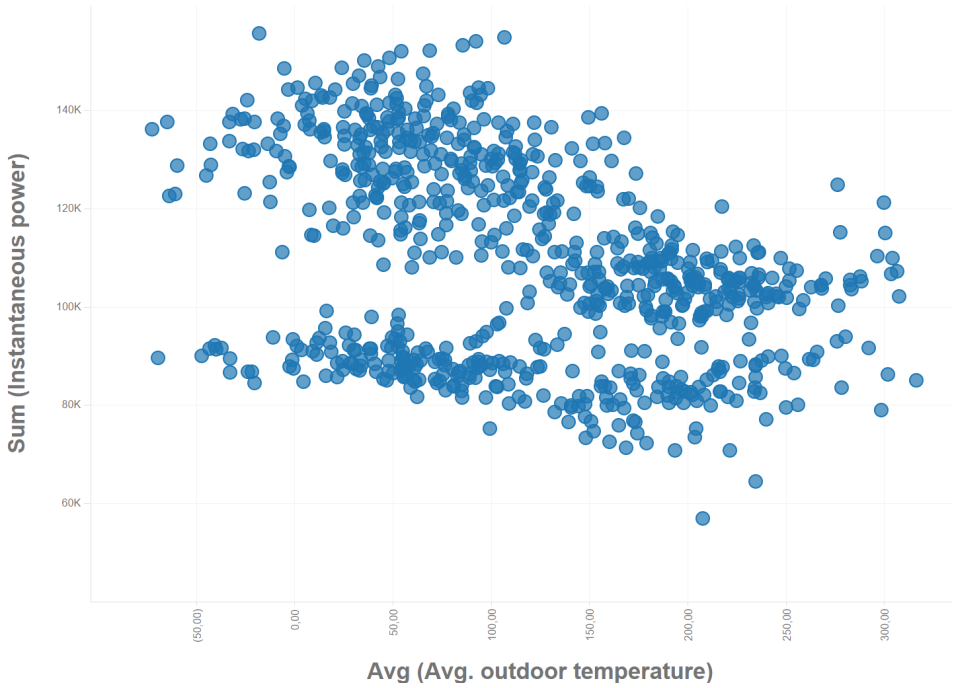


Figure 2.12: Scatter plot - MicroStrategy

Conclusion

MicroStrategy displays a host of issues, and generally disappoint. It's scaling issues make the program unusable on 4k displays, and while a number the built-in functions is bigger than in Power BI or Qlik sense, the binding of every function to one group of data makes the workspace slightly overpopulated. Another downside of the program is the rapid decrease of computational speed when large datasets are introduced. No other BI software examined in this research exhibited such issues.
Microstrategy provides its basic desktop version for free and also has other versions available such as a mobile or a server one.

2.6 Chapter summary

In this chapter, we introduced the newly-minted concept of Energy intelligence. The term Energy intelligence is used to tie together several different

methodologies in order to create a model or framework of thought used to detect and analyze patterns in energy systems, thus enabling improvement of existing systems by detecting and eliminating malfunctions or bad settings of the examined systems.
In order to improve and modify the existing systems, it is essential to have proper insight into the gathered data. The chapter provides a full introduction into the "Modulo Carpet" and "Scatter Plot" structures currently used as key instruments by Energocentrum PLUS, s. r. o., to gather the necessary data and comprehensively analyze it.
The rest of the chapter covers the extensive research conducted in the field of Business Intelligence tools, which analyze business data and provide insight into possible solutions and methods to improve productivity and effectivity of the company's services. This research was conducted in order to gather the information and inspiration necessary to design a better tool, which would provide in the future superior Energy Intelligence analysis. The aforementioned tool would be presently incorporated into the framework of tools and systems currently used by Energocentrum PLUS, s. r. o.

Chapter 3

Design of the query language

Based on our review of the various BI tools currently available in the market, which we have covered in the previous chapter, we may now proceed and attempt to implement our own application. Such software will be later converted into an API in order for Energocentrum PLUS, s. r. o., to make full use of it, based on the experience and knowledge gathered during our aforementioned research and analysis.
The main inspiration for building a new tool from scratch was to provide better-tailored service for Energocentrum PLUS, s. r. o., and equip the company with the necessary tools to better implement their existing methodologies to the newly developing field of Energy intelligence. Since Energy intelligence is a relatively new and yet-to-be fully explored field, companies intending to make full use of its capabilities in this field would need to develop its own tools and software to provide satisfactory service based on a specific analysis. In order to fit said company needs and provide optimal, practical service in real-world situations, such an application would need to meet four main requirements:

- The capability of extracting information from a Sample Timestamp (Day of the week, Hour), where "Sample" may be defined as a pair consisting of a timestamp and its respective value.
- The possibility of obtaining data from a wide range of data sources - such as, for example, CSV, Excel, or database files.
- Implementation of variety of mathematical operations such as Average, Sum, or Median
- The ability to process user-defined mathematical operations, such as thresholding or data filtering.

The software which was developed for this work was designed with these four main requirements in mind, and if at some point during the development process some of these requirements were violated or not optimally met, the development process was stopped and reformed in order to provide a better solution that would meet the aforementioned requirements and satisfy them. Another consideration made while writing the application, which is not

mentioned in the four requirements and yet is common for all commercial-collaborative applications, was to maintain the code structure, transparency, and editability. Since the tool written for this article was supposed to be used by an actual company on a variety of real-world cases, it was of paramount concern, even if not officially stated.
In the following section, we will discuss how the tool was built and implemented. Furthermore, we will review the structure of the program itself, and compare its capabilities with those of Tableu, which was chosen thanks to its various capabilities as an inspirational for the development of the tool.

3.1 Software design and implementation

3.1.1 Code structure

The entirety of the aforementioned program is separated into specific nodes, each of which has its own unique functionality.
The code has been segmented into nodes in order to allow for greater transparency of its structure, as well as allow for increased expandability - optionally expand the application to handle additional sources of data or new mathematical operations, such as integrals, for example. The code's general structure is detailed in the next page.

In the two following sections, each of the main classes, structs, and enums would be explained and discussed in detail, accompanied by examples and snippets of the real code in order to provide sufficient understanding of each part.
The section named "Functional nodes" describes the purpose of each of the classes included in the computational tree structure of the whole program depicted in the figure on the next page.
In the section titled "Structs and data classes", we will discuss the way that data is stored and structured across the application, and introduce at the beginning of each subsection a figure with the names of the newly introduced structures - classes, structs, enums, etc.

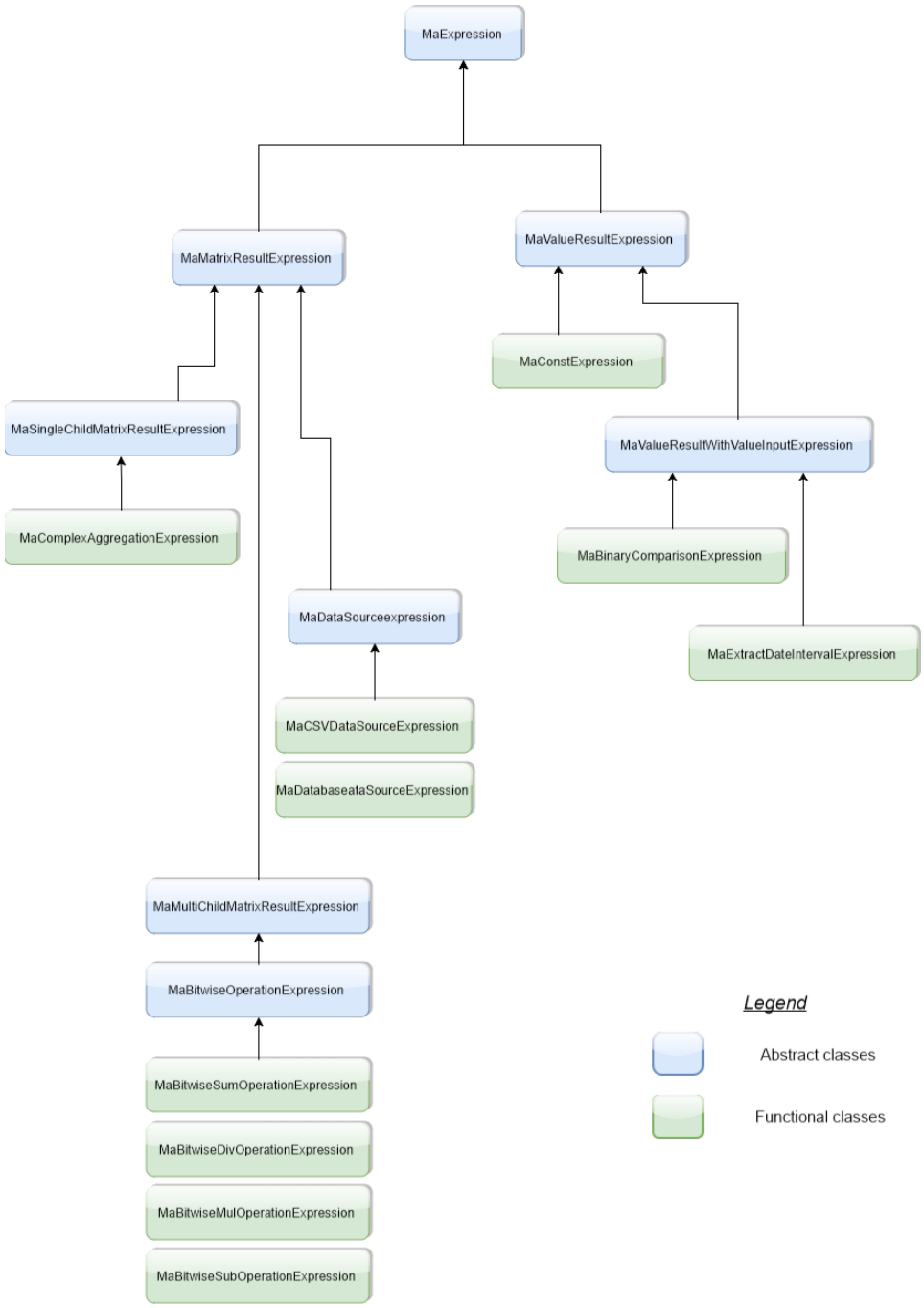


Figure 3.1: Computational nodes hierarchy

3.2 Process calculation principles

This section will introduce the main principles and logic behind the operational process which manages the computational nodes, which are elaborated in the next section. As the basic process of all calculations is similar, it can be explained in general with no need for a specific example from each of the processes.

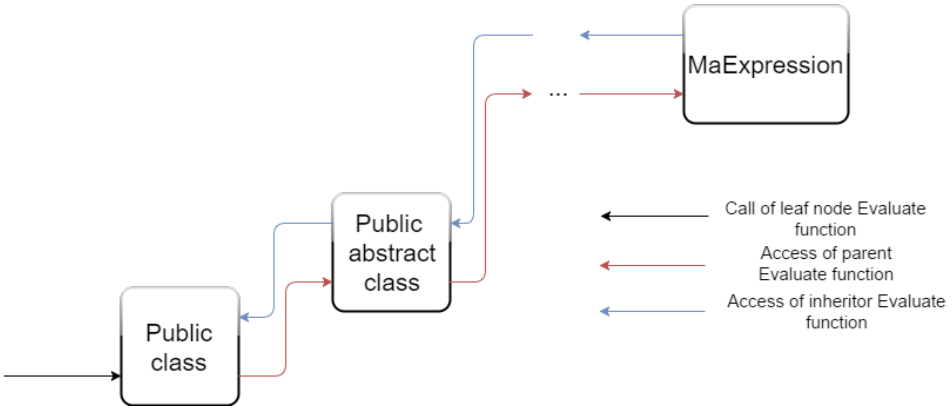


Figure 3.2: Process calculation principles flowchart

Each and every calculation process starts by calling the function "Evaluate" of the leaf node in a tree. This function call, depicted in the figure above, is shown as a black arrow connecting the "Public class" block, which can express any leaf node in our structured tree. When we recall the structure of the whole node tree depicted in figure 3.1, we may see that each of the leaf nodes implements a specific function which is a combination of conditions and parameters imposed by the previously activated parent classes.
Once the "Evaluate" function call is made, the same function call is made in each of the parent classes as well, leading to a chain of "Evaluate" calls all the way up to the closest available node to the root of the tree called "MaExpression". In figure 3.2 in this section, this chain of events is expressed by a series of red arrows, each of which connects a pair of child-parent classes. The reason for this regressive chain of function calls is to ascertain the existence and fulfilling of each of the initial conditions stated by each parent class.
Once all the function calls acting on the parent classes are done, the process reorients itself and proceeds down to the child classes, depicted in the above figure as a set of blue arrows.
Once the lowest class in the structure is accessed, and the "Evaluate" procedure chain is finished, the entire calculation is finished, and the required data is prepared for another process, such as saving said data to a file.

3.3 Computational nodes

In this section, we will discuss the issue of the computational tree structure. The goal of this section is to propose such a structure that would satisfy the various different requirements on function nesting. As an example, we would use the data providers output, data provider being a class which retrieves data from a CSV file, and use said output as an input for the resampler node. Each of following subsubsections describe one of the nodes in the computational tree hierarchy. For clarity, at each headline, the type of the class, which represents the computational node, is presented.
Next, we will define the following three terms:

- "Expression" - a general term for any computational node in this work which accepts, transforms and prepares data for the other computational nodes or for the applications' output.
- "Parent" - A class that is used as the basis for inheritance is called a superclass or base class. [13]
- "Child" - A class that inherits from a superclass is called a subclass or derived class. The terms parent class and child class are also acceptable terms to use respectively. A child inherits visible properties and methods from its parent while adding additional properties and methods of its own.[13]

MaExpression (abstract class)

"MaExpression" is an abstract class at the root of the application's object tree. The main reasons for placing MaExpression at the root of the tree are the high combinational requirements resulting from the various demands of real usage and extensive requirements on a variability.
MaExpression serves as a framework for all other classes.
This class introduces an abstract procedure named "Evaluate", which is common for all inheriting classes, with each of them redefining it according to the need of its specific usage and purpose. As an input parameter to this function, we utilize an object of class "EvaluationContext" which is discussed later in this work.
MaExpression has two descendants, both would be discussed shortly.

MaMatrixResultExpression (abstract class)

Similarly to the MaExpression class, the class named "MaMatrixResultExpression" is defined as an abstract class. Unlike MaExpression, this class contains a parameter in which the succeeding classes stores the result of their calculations in form of a MaMatrix object.
As we recall the previous discussion about the calculation principle, MaMatrixResultExpression is the top class which all its succeeding classes access.

Once its "Evaluate" function is finished, the process of evaluation starts accessing and calculating the chain of child classes "Evaluate" functions on the way to the leaf expression.

MaSingleChildMatrixResultExpression (abstract class)

"MaSingleChildMatrixResultExpression" is an abstract class which adds new requirements to the input data in the form of a single matrix, which is processed later by its child.

MaComplexAggregationExpression (regular class)

Forms a very complex tool to create queries for data and their processing. This class was inspired by the Tableau way of data processing, which we observed in the research we introduced in this work. We designed this part of our program similarly to Tableaus in order to assure that our application would be capable of manipulating the same range of functions with a single matrix of data as Tableau does.
As this class is considered to be a key junction in this work, we would proceed to a more detailed description of it.

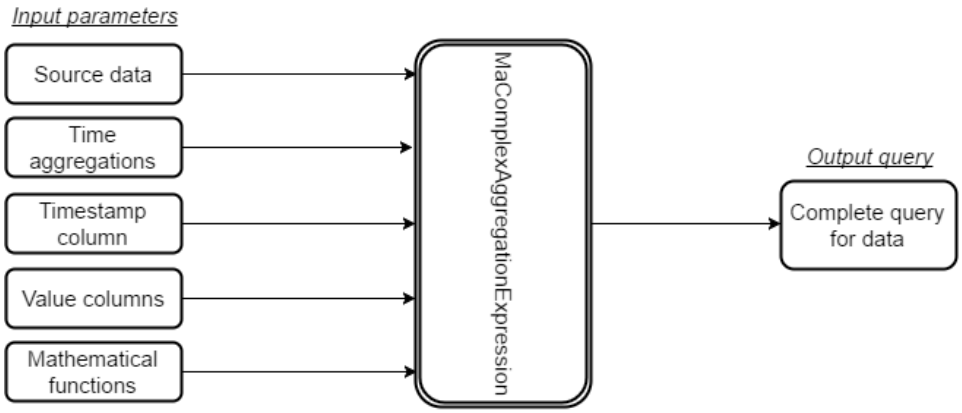


Figure 3.3: MaComplexAggregationExpression structure

There are five input parameters that together form all the information we need to obtain data results which are similar to Tableaus software. These input parameters are:

1. *Data source*

Object of class MaMatrixResultExpression which, in this context, is considered as a data source for operations and aggregations. It is important to remember that the MaMatrixResultExpression class is an abstract class, meaning that it can represent any of the data providing classes, such as CSV or database.

2. *Time aggregations*
Time aggregations we require to be applied on data in a single array. Each of these time aggregations can be of a discrete or continuous type, both of which are described in detail later in this work.

3. *Timestamp column*
Column of input data where timestamps are stored. Thanks to this parameter, after calling the "Evaluate" function of this class, we know where to look for the timestamps of input data which we want to aggregate in a way defined by the second parameter.

4. *Value columns*
Provides information about which columns of source data contains actual sample values and which columns we would like to use as input for mathematical operations such as average or sum.

5. *Mathematical functions*
Array of mathematical operations we would like to use.
To uniquely determine on what data we would like to perform which given mathematical operation, we need to specify it by the number of the column in the source data and by the identifier of the function itself.

MaMultiChildMatrixResultExpression (abstract class)

In the two previous subsections, we sufficiently covered the part of the analysis where we used only one matrix as an input and only a single column for one mathematical operation. Unfortunately, the wide range of analysis where we need to use more than just one data column is still uncovered. To satisfy our requirements in this field of multi-column analysis we introduce "MaMultiChildMatrixResultExpression".
"MaMultiChildMatrixResultExpression" is an abstract class designed for working with more than just one column of data (or datasets), on which we would like to perform mathematical operations such as bitwise sum or subtraction.

MaBitwiseOperationExpression (abstract class)

"MaBitwiseOperationExpression" may be considered as an interface between certain parts of the application and a set of bitwise mathematical operations. Our current requirements from each bitwise function, such as sum or multiplication, is that it will be calculated on a single matrix, but only on certain rows in it. The principle of bitwise operations is depicted in the following figure with the bitwise sum operation as an example.

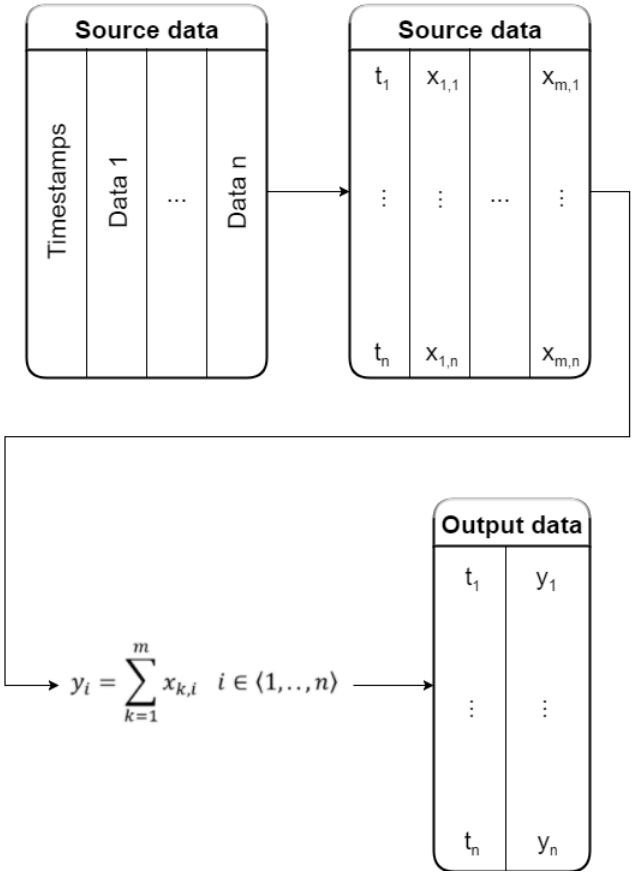


Figure 3.4: Bitwise sum operation principle

The "sum" operation is implemented as an example. Implementation of other operations is just matter of coding.

MaBitwiseSumOperationExpression (regular class)

MaBitwiseSumOperationExpression is an example of how the bitwise operations are coded.
As can be seen in the previous figure (3.4), we can expect to obtain an output matrix which would have the same number of rows as the source matrix and two columns - the first of which stores the timestamps and the second contains the calculated values.
In order to greater flexibility of the tool we had created only the function Sum, to allow future users of the tool to build and customize it as they see fit.

MaDataSourceExpression (abstract class)

"MaDataSourceExpression" is a generic data source abstraction which represents a node that retrieves data from external sources such as CSV, XLSX, Mervis DB, SQL and more.

MaDataSourceExpression, being an abstract class, stores the starting and ending timestamps between which all inherited data processing classes operate.

MaCSVDataSourceExpression (regular class)

"MaCSVDataSourceExpression" is a specific data providing class, implementing sourcing data from CSV files. The process's data retrieval principle is depicted in the following figure.

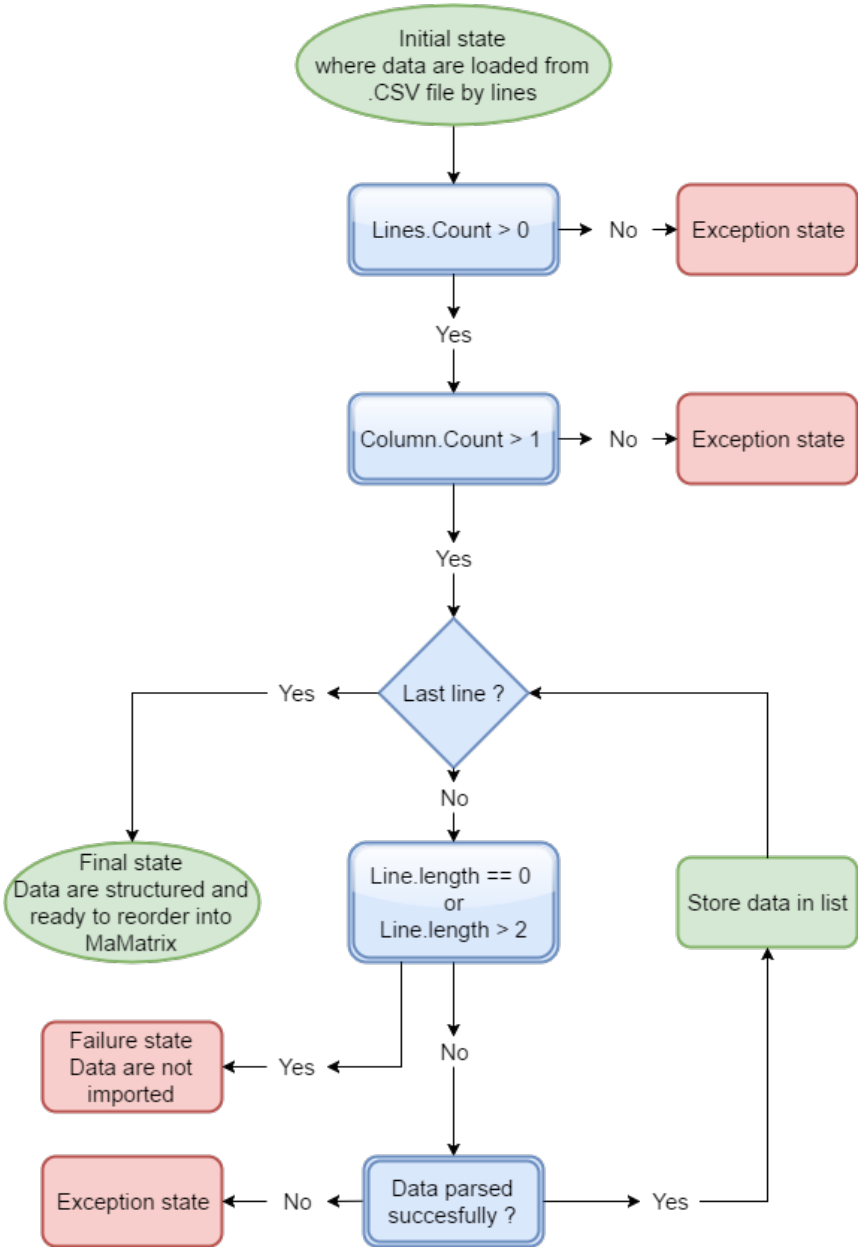


Figure 3.5: CSV import principle

In order to provider greater clarity and understanding of the issues at hand, appendix B.2 , which contains an example class, has been attached to this work.

MaValueResultExpression (abstract class)

As the name would prompt, all classes on the right-hand part of the computational tree are working with a single value, from which information is extracted and stored as a "MaValue", which discussed in detail in section "Pre-defined structs and data classes". In the following subsubsections, we will describe two main functions which operate on single values. These functions are the binary comparison of two operands, where the output is binary one or zero representing a true or false state, and the extraction of a specific interval from a DateTime stamp. These two classes are described in details in the following subsubsections.

MaValueResultWithValueInputExpression (abstract class)

This class adds additional parameters for computation to the previous class, which stands above it in the computational tree. This means that it adds additional parameters, such as input value from the previous "MaValueResult", to its child functions to which other values are compared. This class is the next step in the specification of the parameters with which the following classes are intended to work with.

MaExtractDateIntervalExpression (regular class)

This class covers work done in the field of analysis where we are required to extract and work with a specific interval, such as the hour or day of the week from the timestamp of the given sample. These values are usually used to group the data according to a given value and later on work as a timestamp for newly calculated time intervals. It is important to say that currently, it is not possible to get smaller interval than what the actual sample time is. As an example, it is not possible to get seconds from data sets which are sampled in hours.
Currently, the supported operations are:

- Year
- Month
- Day of week
- Hour
- Minute
- Second

MaBinaryComparisonExpression (regular class)

The MaBinaryComparisonExpression class was created due to requirements which rose regarding data filtering, such as when we want to obtain the subset of values from the whole data set which satisfies a given condition. Such form of the function gives us an option to actually generate our own mathematical functions in the current tool, which means there is a wider range of possible analytical operations which can be created.
Currently, the operations supported by the application are:

- Less than
- More than
- IsEqual

3.4 Pre-defined structs and data classes

In this section, we will discuss pre-defined classes, structs, and enums, which are used as parameters and properties in the computational nodes presented in section (3.3), and in which data is stored. The section is divided into three central parts describing pre-defined structs, enums, and classes in that order. In case that one element contains another and a description of the inner element is not yet presented, a brief outline is introduced to provide sufficient insight into the subject.

3.4.1 Pre-defined structs

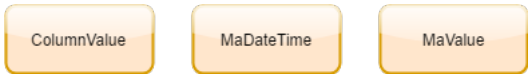


Figure 3.6: Pre-defined structs

MaValue

"MaValue" is the essential element in which data is stored and from which data is retrieved. MaValue data is stored later on in more complex structures, such as MaMatrix.
Currently, supported data types which MaValue supports are:

- Null - Element in datasheet is missing
- Double - Actual value is an element of 'double' type
- MaDateTime - Actual value is a timestamp in form of struct described in subsection 3.4.2.

Each element can represent only one of the value types listed above. The value type which is stored in MaValue is uniquely determined by an overloaded constructor which can obtain only one value from the aforementioned set of Null, Double, and MaDateTime. Once the appropriate constructor is chosen according to the input argument, the value of the pre-defined enum MaValueType is set. We will discuss the reason for introducing this enum and further details in subsection 3.4.2.

MaDateTime

'MaDateTime' is a struct which is defined in this work to enable working with the data values timestamps. This struct is in effect a superstructure to the built-in struct 'DateTime' which is working with date and time data, designed to extend the built-in struct's capabilities and to satisfy our additional requirements which are discussed in greater detail below.
Each instance of MaDateTime is composed of four main attributes which together form a very powerful tool to manage work with timestamps. These attributes are:

- $MaDateInterval$
  Pre-defined enum allowing work with time aggregations.

- $MaDateKind$
  Pre-defined enum indicating if time aggregation stored in current MaDateTime is continuous or discrete.

- $DateTimestruct$
  This struct comes as a built in part of $C\#$ and enables work with date and time. When a new instance of MaDateTime is created, DateTime parameter and a parameter named MaCalendar are set, creating a sample timestamp to be manipulated and worked upon.

- $MaCalendarclassinstance$
  This instance extends the built-in capabilities of $C\#$ by choosing the first day of the week.

Most of these attributes are discussed in greater detail in other sections of this work.

ColumnValue

"ColumnValue" is a struct defined to store data in an output format, formed by two separate parameters:

- "MaDateTime", which we discussed earlier in this section, which in this case serves to store timestamp information in any form which combination of its inner parameters MaDateKind and MaDateInterval allows.

- "MaValue", which can hold either a value or a timestamp. Such a combination of these two parameters can give us plenty of information about a given sample thanks to its indexability given by the MaDateTime property.

3.4.2 Enums

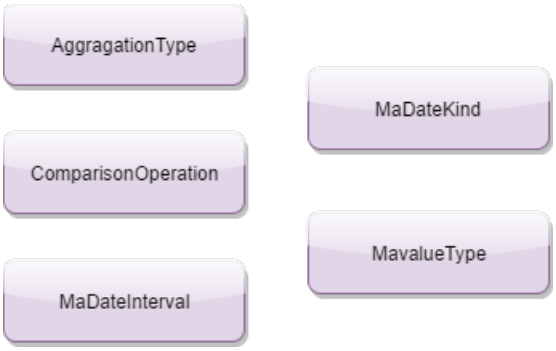


Figure 3.7: Defined enums

MaValueType

As mentioned previously in this section, MaValue Type is set during the initialization of a new instance of struct MaValue. According to an overloaded constructor, MaValueType obtains its value from the set of values Null, Double or MaDateTime. MaValueType is introduced due to the control of data consistency in MaMatrix (more about it in subsection 3.4.3) and also prevent of exception states due to access and usage of not initialized properties of the MaValue struct.

AggregationType

Is important as an indicator in class MaComplexAggregationExpression where defines which mathematical operations from set Sum, Average, Min and Max are applied to the data. It also serves as a switch argument in class MatrixColumnAggregationDefinition where the functions themselves are defined.

ComparisonOperation

is a user defined structure to cover filtering input data according to the specific threshold given by a user. Currently, specified thresholding functions are: Less than

MaDateKind

MaDateKind is the key junction in an extraction of data from the timestamps. Set of values which this enum contain is just two-element set (Continuous, Discrete) and each of them works with data in the very different way.
Discrete MaDateKind works with data in a way that DateTime class does. Except for the specified information, everything else is thrown away. So in the case of an interval such as year or month, a just particular time interval is retrieved and stored from the process of calculation. In contradiction continues MaDateKind extracts all information till set time interval. In the case of choosing as an example the month, all values which will be retrieved from the process of calculation will contain a year and month of the particular timestamp. It is important to realize, that in the case of functions week of the year, the day of the year, the day of month and day of the week there is no continuous distribution of the values. In these special cases, we obtain the same information from continuous and discrete distribution type.

MaDateInterval

MaDateInterval is an enum introduced to store the value of time aggregation of the timestamp. In the case of this work, it is useful as the indicator of which value is stored in actual MaDateTime. MaDateInterval is defined for following intervals:

- Year
- Quarter
- Month
- Week of year
- Day of year
- Day of month
- Day of week
- Hour
- Minute
- Second
- Exact timestamp

In the case of need to extract just particular discrete information such as a year from the timestamp, we would not need to create own apparatus to achieve it due to it is already part of the DateTime struct. Our need stem from the creation of semi-continuous time intervals described by the parameter MaDateKind.

3.4.3 Data classes

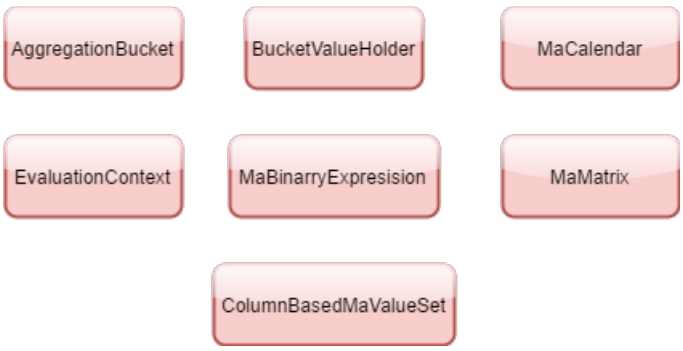


Figure 3.8: Data classes

MaMatrix

The MaMatrix class represents a table like structure where MaValues are stored. A specific data type (Null, Double, MaDateTime) is stored in each column. In case that this law is violated the whole program ends, a halt which is handled by a proper exception status. The main features of MaMatrix are:

- 'Get' column counted
- 'Get' rows counted
- 'Get' and 'Set' MaValue on a specific position
- Compare two MaMatrixes

MaCalendar

The MaCalendar class extends the possibilities of working with and manipulating functions such as day of the week. In this class, DateTime it is impossible to set the first day of the week to different day. By default, it is set to Sunday which is inconvenient for the European way of thinking.
MaCalendar is used to return day of the week on whichever day is predefined as the first day of it.

EvaluationContext

EvaluationContext is currently a placeholder for the MaCalender, meant to be extended and built upon in the future.

BucketValueHolder

BucketValueHolder is an internal class working with the data used during the calculation process. Its primary goal is to create buckets (a.k.a containers) in which the data is stored in an order based on a predefined condition.

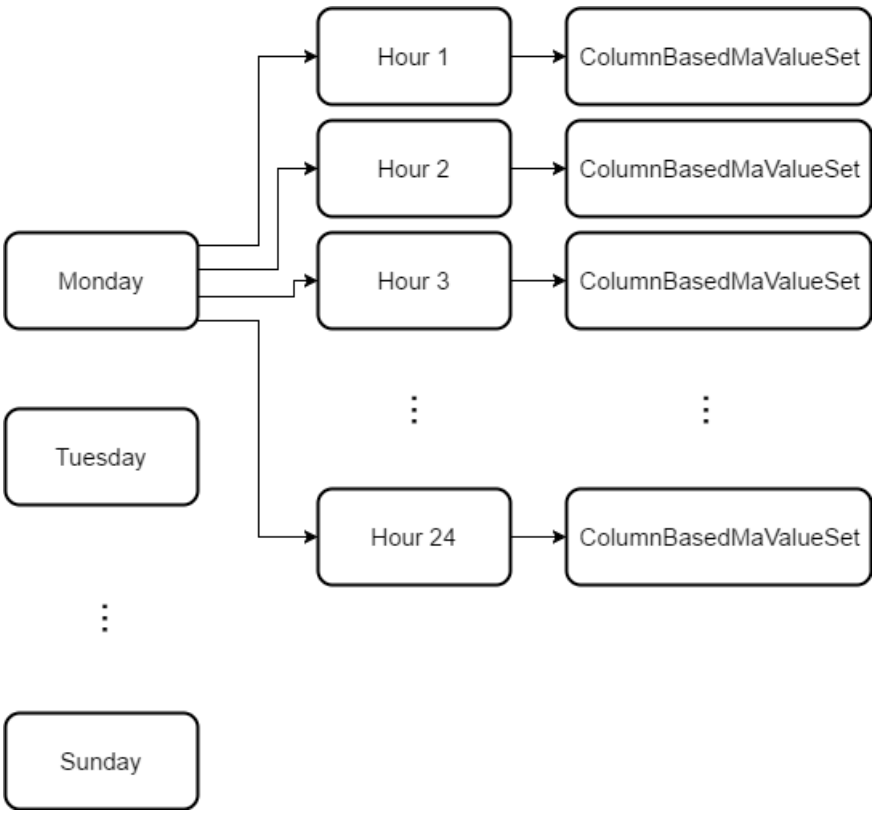


Figure 3.9: Complete BucketValueHolder for Modulo Carpet structure

As can be seen in the figure above, each so-called bucket may hold another bucket or values, not both.
When the current bucket is not in the lowest level of the structure, the bucket holds other objects of the BucketValueHolder class. When it is in the lowest level of the structure, it holds values.
To provide sufficient insight into the issue we introduce the example depicted above. We would use one of the main structures which we aim to eventually create - a structure called 'Modulo carpet', which is explained in greater detail in section 2.1.
In this structure, we split the data according to two parameters which are the hour of the timestamp and day of week extracted from it. The first level of the BucketValueHolders would be consisting of buckets, each representing one day of the week. Each of these buckets contains sub-buckets in form of hours. This method of combining all the elements in additional sets is called "Cartesian sum", which is defined as:
The Cartesian product of two sets A and B (also called the product set, set direct product, or cross product) is defined to be the set of all points (a,b) where a in A and b in B. It is denoted $A \times B$, and is called the Cartesian product since it originated in Descartes' formulation of analytic geometry. [1]
As was said before it serves to create all possible combinations of elements of multiple lists.

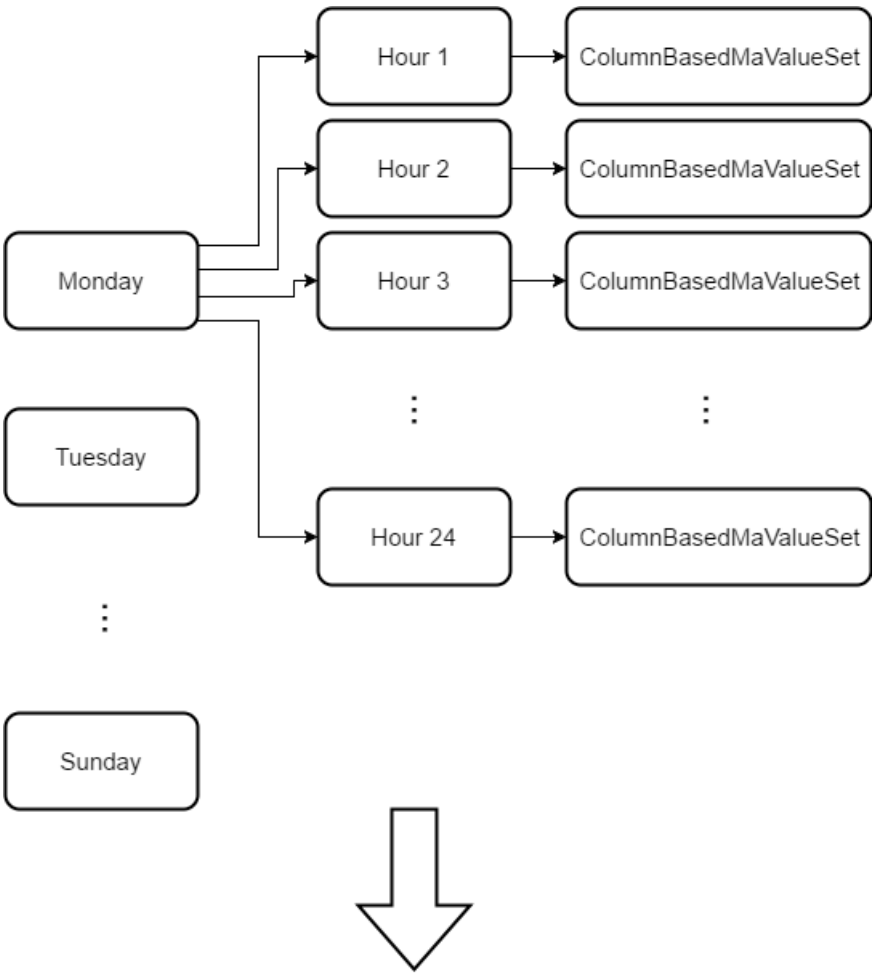
Once the basic structure is done, a simple iteration of source data is made and according to the parameters extracted from the timestamp of each sample which is then affiliated to the proper group.

ColumnBasedMaValueSet

"ColumnBasedMaValueSet" is meant to store all the values for the various aggregation functions. As the name would prompt, all the data is stored in the form of lists, where every list represents one data column. When the aggregations are primed, each data column is accessible through its column index.
If we recall the figure 3.9 Column BasedMaValueSets represents the lowest level of the structure where that data is stored in appropriate buckets.

AggregationBucket

Represents the whole structure depicted in figure 3.9, as it holds all the buckets and appropriate data points. Once the process of sorting the data into the appropriate categories is done, AggregationBucket class will contain a function to create an output matrix, where the buckets would create different categories in the output Matrix, and the output of the used function produces will be a calculated result for a given category. Such a situation is depicted in the figure on the following page.



$$Sum(ColumnBasedValueSet) = \sum_{i=0}^{n} ColumnBasedValueSet(i);\ ColumnBasedValueSet(i) \in R$$

Monday	Hour 1	Sum(ColumnBasedMaValueSet)
Monday	Hour 2	Sum(ColumnBasedMaValueSet)
Monday	Hour 3	Sum(ColumnBasedMaValueSet)
Monday	Hour 4	Sum(ColumnBasedMaValueSet)
...	...	...
Sunday	Hour 24	Sum(ColumnBasedMaValueSet)

Figure 3.10: Conversion to the MaMatrix structure

3.5 Unit testing

Unit testing is closely associated with software development of all kinds and is meant to assure that software works without failure under predefined rules, which prevents unintentional or unexpected behavior. There are generally two categories of tests which can be performed:

- Subjective tests
- Objective tests

Subjective tests are performed usually by the operator (designer, developer), who evaluates the results of the test based on their senses, preferences, and opinions. As an example, we would test the GUI of the application. It is evident that scaling problem or wrong color range used in the program can be detected simply by vision. The problem with subjective testing is that there is no straight right or wrong.
Thanks to the fact that this application is in the long-term development, GUI was not created yet and therefore there is no need for subjective tests.
The second group of tests is called objective tests. Objective tests are simple to evaluate because the result is a binary right or wrong, and no another result can exist. Simply put, this group of tests works with facts. As an example of such testing, we can introduce a test where we already know the output value which we want to obtain and compare it to the actual output of our script.
In the case of this work, we aimed at a group of objective tests due to the fact that in the future it will form the API for energy intelligence of the analytical tool. This means that there can be no miscalculations, as even the smallest miscalculation would return very wrong results.

We applied our set of objective tests to the following fields:

- Mathematical operations
- Time aggregation
- Import of data
- Storing the data in data structures

In the case of mathematical operations and time aggregations performed on data, the testing functions are relatively simple.
The right results were precalculated in a different analytical software to get a baseline result. After the process of calculation ends, the retrieved data was compared to data precalculated by Tableau. In case that both results, the retrieved and precalculated data, are equal, we may say that the tests succeeded. In all other cases, the result of the test is considered to be the failure.
In the case of the two other fields of testing, Import and storing of data, no

additional software is required.
To check whether data is correctly imported just a few samples of the whole dataset, which contains 220,000 samples, are used. The main reasons to do so are:

- Even a small dataset may reveal mistakes in the code and in the process of import

- Imported data is compared to testing data which has to be hard coded in a test function. Hard coding of the whole data set would be time and memory consuming and such an effort would not be efficient.

In the mathematical operations and time aggregation tests, we require the retrieved and testing data to be the same.
The last test is checking the data to be correctly stored in a structure MaMatrix described in section 3.4.3.
After the data importing test ends successfully we can safely assume that the correct data is provided as input to the transformation function. Similarly to the previous test we have to hard code the testing structure, which at the end of the test would be compared to the retrieved one.
The application is considered to be working properly when all the tests pass successfully.

3.6 Implementation review

To wrap up our lengthy exploration of the various BI tools currently available in the market, we introduced in section (2.1) two structures designed for Energy Intelligence data analysis - Modulo Carpet and Scatter plot. Since both of these structures are essential for the developed software to be applicable in the field, additional functions were introduced to the software in order to test the possibility of retrieving data which will enable forming these structures, as well as the software's functionality in the range of possible settings such as different time intervals or functions.
Since both structures were tested thoroughly and the data output was repeatedly found to be correct, the software can be considered applicable. As these data structures contain 128 and 782 rows respectively, describing them in full and their equivalent hard coded unit test classes is unreasonable. Therefore, a summary of the tests in the form of partial key analytical data parts is described in the following tables:

Day of week	Hour	Avg (Instantaneous power)
Monday	0	308.68
Monday	1	305.177
Monday	2	305.554
Monday	3	304.865
Monday	4	325.993
Monday	5	339.165
Monday	6	373.914
Monday	7	423.078
Monday	8	476.816
Monday	9	518.464
Monday	10	527.811
Monday	11	519.322
Monday	12	506.855
Monday	13	510.973
Monday	14	505.942
Monday	15	504.405
Monday	16	490.658
Monday	17	473.902
Monday	18	436.038
Monday	19	400.674
Monday	20	378.866
Monday	21	342.008
Monday	22	326.666
Monday	23	322.487
...	...	...
...	...	...
...	...	...
Sunday	20	312.309
Sunday	21	311.331
Sunday	22	310.491
Sunday	23	309.132

Figure 3.11: Sample input for Modulo Carpet structure

In order to provider greater clarity and understanding of how data for Modulo Carpet is being verified, a testing method is appendixed in appendix B.1.

Day of year	Avg (Avg. outdoor temperature)	Sum (Instantaneous power)
1/1/2015	3.392	88036
1/2/2015	3.908	97857
1/3/2015	3.827	91498
1/4/2015	3.895	90442
1/5/2015	3.66	139337
1/6/2015	3.41	140789
1/7/2015	1.361	142939
1/8/2015	5.23	138129
1/9/2015	8.655	126007
1/10/2015	12.497	91600
1/11/2015	5.36	92926
1/12/2015	6.715	141817
1/13/2015	10.132	131593
1/14/2015	9.273	141484
1/15/2015	5.185	142363
1/16/2015	5.601	124968
1/17/2015	5.239	96581
1/18/2015	5.28	94900
1/19/2015	4.234	136475
1/20/2015	3.32	130981
1/21/2015	3.401	135065
1/22/2015	5.299	138351
1/23/2015	2.455	127885
1/24/2015	1.58	95683
...	...	...
...	...	...
...	...	...
2/17/2017	6.107	110926
2/18/2017	6.201	88220
2/19/2017	5.514	89729
2/20/2017	6.342	85185

Figure 3.12: Sample input for Scatter plot structure

Should some analytical function be found missing or in need of an update, the code's high editability, as well as its massive extendability factor, allow for such a function to be easily and swiftly incorporated into the designed application.

3.7 Possible outcome

As previously mentioned, this work is part of the long-term development of an analytical tool which would serve as a key instrument in data analysis. The data gathered by this analytical tool would be mined and processed to verify proper setting of heating systems developed by the company. The tool may also be used to identify malfunctions and bad configurations, as well as bugs and human error, allowing the company to correct such issues and make their systems far more effective.
As the application is at a stage where it is possible for it to be transformed from an application into an API (application programming interface) to be used for Energy intelligence, we may easily imagine two possible ways to use it in real-world scenarios. Both options would succeed the current tool, which analytical capabilities are limited by the complexities of graph analysis, meaning they do not satisfy the requirements necessary for a wider range of analytical tests.
The first possible options would be to develop a GUI similarly to Tableau, which would allow the tool to be integrated in the future as a part of the company's current complex tool named Mervis. The advantages of this solution are that it is easy to design and implement, and provides greater control and an extended range of possibilities, which together result in an extremely robust analytical tool.
The main drawback, however, is that not all employees of the company are familiar with using Tableau-like analytical tools, which may lead to less than meaningful testing.
The second option for modifying the existing API will be to implement the tags system. A tag is defined as a name/value pair applied to an entity. A tag defines fact or attributes about an entity.
This options main advantage is that dividing and cataloging the data with tags into predefined groups can later be used in conjunction with a GUI to specify which graphs are available and meaningful for current data group.
The main disadvantage is that almost no employee currently working in the company possesses the knowledge required to properly tag and retrieve the tagged data.
As is evident, both solutions are not very promising and are far from optimal, due to the fact that only a minority of company employees would be able to use it. As a result, a third option was proposed - combine both solutions. Such a solution combines the wide range of possibilities of the first solution, which was the Tableau-like GUI, and limits its disadvantage by utilizing additional information gained trough the second solution - the tag system. A practical output of such an interconnected solution would be the pruning of the useless predefined graphs in the available sets, leaving the meaningful ones available and easier to notice.

3.8 Chapter summary

Similarly to chapter 2, we dove deeper into the wide variety of available business tools. However, in this chapter, we introduced the structure of our newly developed application which was inspired by knowledge gathered by our previous research conducted in chapter 2. The chapter is divided into three primary sections, corresponding to the structure of the code.
First, we defined the development goals - what would the application be able to do, and which results should it provide in the field. In addition, we incorporated one of the basic yet crucial principles of proper software architecture, which is the principle of future extendability.
We then defined and discussed the practical executory part of the program and its division into a set of specific nodes, each of which correlates in a uniquely supplemental way to the tool as a whole. This division into nodes was implemented with the goal of allowing for maximum transparency and future extendability of the software. In further service to this goal, each node is described in its unique subsection.
The third part introduced structs, enums, and classes which serve as storage devices, storing crucial operational data generated by the executory segments of the code.
An introductory figure which depicts the workflow of the various objects described in the subsection is presented at the beginning of each subsection.

Chapter 4

Conclusion

The principal goal of this work was the creation, implementation, and deployment of an analytical tool meant to be used by the company "Energocentrum PLUS, s. r. o.". The tool was meant to be used by the company to enhance its analysis capabilities in the field of Energy Intelligence - a burgeoning and still developing of using data analysis tools on data gathered from intelligent building systems in order to check for their deployed systems integrity and effectivity. At the start of this work we defined two essential data structures called "Modulo Carpet" and "Scatter plot" which are, among other graphs and tools, currently used to extract information from collected data.
After we have laid down the key structures necessary for our analysis, we sought inspiration about how to design a better tool - how should it be structured in order to be user-friendly, fast, and efficient. In order to choose the correct approach to the way the data was processed and handled, we have conducted an extensive research into the current market of Business Intelligence tools.
After concluding the aforementioned research and inferring several ideas, we have answered the original question on how to structure the tool, and have detailed several crucial parts of said structure to provide the reader with insight into its inner workings.
Since iteration testing of the software was deemed a crucial part of the development process, the final parts of this work will discuss the results of the implemented unit tests. Those unit tests verified the accuracy and precision of several of the key function's outputs to assure that the too is indeed ready for deployment and to be utilized for real analysis.
In conclusion, the Energy Intelligence application which was designed for this work was inspired by the best tools available today, designed in accordance with Gartner's Magic Quadrant[7] requirements, and was tested by iterative unit tests.
Therefore, we may conclude that this tool is ready for deployment and use in real-world case analysis.

Appendix A

Bibliography

[1] Weisstein, Eric W. "Cartesian Product." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/CartesianProduct.html

[2] Tableau software support, *Tableau desktop help*, https://onlinehelp.tableau.com/v10.2/offline/en-us/tableau_desktop_10.2.pdf, edited 2017, accessed 12 April 2017

[3] David Iseminger, *Power BI Documentation*, https://powerbi.microsoft.com/en-us/documentation/powerbi-landing-page/, 1 March 2017, accessed on 5 March 2017

[4] Qlik Sense support, *Loading and Modeling Data - Qlik Sense*, http://help.qlik.com/en-US/sense/1.1/pdf/Loading$%$20and$%$20Modeling$%$20Data.pdf, accessed on 6 March 2017

[5] Microstrategy Incorporated, *MicroStrategy Desktop User Guide*, http://www2.microstrategy.com/producthelp/10/manuals/en/AnalyticsDesktopUserGuide.pdf, created 2015, accessed on 7 March 2017

[6] Project haystack documentation, http://project-haystack.org/doc/TagModel, accessed on 24. April 2017.

[7] Jen Underwood, *EXPLORING 2017 GARTNER BI MAGIC QUADRANT RESULTS*, http://www.jenunderwood.com/2017/02/22/2017-gartner-bi-magic-quadrant-results/, 22. February 2017, accessed on 21 May 2017.

[8] Ralph Kimball, Margy Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, 3rd Edition*, July 2013

[9] O'Reilly Media, *Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking*, July 2013

[10] David Slánský, Jan Pour a Ota Novotný, *Business Intelligence, Jak využít bohatství ve vašich datech* , 2. 12. 2004

[11] Barney L. Capehart, Timothy Middelkoop, *Handbook of web based energy information and control systems*,1st Edition, 26 July 2011

[12] Roberto Melli, *Present application of artificial intelligence to energy systems.*

[13] Michelle Yaiser, *Object-oriented programming concepts: Inheritance*, `http://www.adobe.com/devnet/actionscript/learning/oop-concepts/inheritance.html`, 6 February 2012, accessed on 21 May 2017.

[14] Cesar de la Torre and David Carmona, *.NET Technology Guide for Business Applications*, November, 2013

[15] Andrew Troelsen, *C# 6.0 and the .NET 4.6 Framework*, 8 November 2015

[16] Deborah j. Rumsey *HOW TO INTERPRET A SCATTER PLOT*, `http://www.dummies.com/education/math/statistics/how-to-interpret-a-scatterplot/`, accessed on 23 May 2017

Appendix B

Code samples

B.1 Modulo Carpet test

```
[Test]
public void ScatterPlotTest()
{
        // Source files where input, output and data to
           compare are stored
        string sourcePath =
           ".\\TestData\\sample_mc_sp.csv";
        string resultPath = ".\\TestData\\scatter.csv";
        string outputPath =
           ".\\TestData\\RESULT\\SP_result.csv";

        // Load of the data from the data source file
        MaCsvDataSourceExpression sourceData = new
           MaCsvDataSourceExpression(sourcePath,
           DateTime.MinValue, DateTime.MaxValue);

        // Cration of the query for the data
           transformation
        // Parameters are in a row data source, time
           agragations, timestamp column index, data
           column indexes, mathematical operations
        MaComplexAggregationExpression complexExpr = new
           MaComplexAggregationExpression(
        sourceData,
        new BucketDefinition[] { new
           MaValueInputFromMatrixColumnBucketDefinition(0,
           new
           MaExtractDateIntervalExpression(MaDateInterval.DayOfYear,
           MaDateKind.Continuous)) },
        0,
        new int[] { 1,3 },
        new AggregationDefinition[] { new
           MatrixColumnAggregationDefinition(1,
           AggregationType.Avg),
        new MatrixColumnAggregationDefinition(3,
           AggregationType.Sum)});
```

```
        // Call of the function Evaluate in class
           MaComplexAggregationExpression with Monday
           set as the first day of week
        complexExpr.Evaluate(new
           EvaluationContext(DayOfWeek.Monday,
           CalendarWeekRule.FirstFourDayWeek));
        complexExpr.Matrix.RoundDoubles(3);
        complexExpr.Matrix.DebugPrint(Console.Out);

        // Safe data to the file defined by path
           outputPath
        SaveData(outputPath, complexExpr);
        // Test conducted to check if data are same.
        Assert.AreEqual(false, IsDifferent(resultPath,
           outputPath));
}
```

B.2 CSV data provider

```
using System;
using System.IO;
using System.Linq;
using System.Globalization;
using System.Collections.Generic;


namespace ESG.Mervis.Analytics.Expressions
{
/// <summary>
/// Reads a CSV data source with many rows. Each row
   must contain the same number of columns and each
   column must always
/// contain the same data types. First column is a
   continuous date time, other columns are "values".
/// </summary>

public class MaCsvDataSourceExpression :
   MaDataSourceExpression
{
 private static readonly char[] m_SplitChars = new
    char[] { ';' };

 private const string InvalidFormatExcText = "Invalid
    format";

 private string m_FileName;

 public MaCsvDataSourceExpression(string fileName,
    DateTime from, DateTime to)
```

```
// Call of parent class constructor
: base(from, to)
{
  m_FileName = fileName;
}

public override void Evaluate(EvaluationContext
    context)
{
CultureInfo MyCultureInfo = new CultureInfo("de-DE");

List<Tuple<DateTime, double[]>> values = new
    List<Tuple<DateTime, double[]>>();

// Import of data from csv file
// According to first line we set amount of data
    columns (total - time)
var lines = File.ReadAllLines(m_FileName).Select(a
    => a.Split(new char[] { ';' },
    StringSplitOptions.RemoveEmptyEntries)).ToList();
if (lines.Count() == 0) { throw new Exception("No
    data found in file"); }
if (lines[0].Length < 2) { throw new Exception("No
    valid data"); }

// Remove names of the columns in the CSV
lines.RemoveAt(0);
DateTime dt = DateTime.Now;

// Parsing data from the file to Date and array of
    double values
foreach (var line in lines)
{
 if (line == null || line.Length < 2) { break; }

 try { dt = DateTime.Parse(line[0],MyCultureInfo); }
 catch { throw new FormatException("Invalid Data
     format"); }

 if (dt.CompareTo(m_From) < 0) { continue; }
 if (dt.CompareTo(m_To) > 0) { break; }

 double[] data = new double[line.Length - 1];
 for (int sample = 1; sample < line.Length; sample++)
  double.TryParse(line[sample], out data[sample -
      1]);

  values.Add(new Tuple<DateTime,
      double[]>(DateTime.SpecifyKind(dt,
      DateTimeKind.Utc), data));
}
// Transform into MaMatrix structure
```

```
    FillMatrix(context, values);
    }

    private void FillMatrix(EvaluationContext context,
       List<Tuple<DateTime, double[]>> values)
    {
     if (values == null || values.Count == 0) { return; }

     int rowCount = values.Count;
     int colCount = 1 + values[0].Item2.Length; //
        Amount of columns - time column + amount of
        values.

     m_Matrix = new MaMatrix(rowCount, colCount);

     // Storage of data to the predefined MaMatrix where
        first column is always filled with timestamps.
    for (int row = 0; row < rowCount; row++)
    {
    // Date Time
     m_Matrix[row, 0] = new MaValue(new
        MaDateTime(context.Calendar,
        MaDateInterval.ExactTimestamp,
        values[row].Item1));

     for (int col = 0; col < values[row].Item2.Length;
        col++)
     {
      // Values
      m_Matrix[row, 1 + col] = new
         MaValue(values[row].Item2[col]);
     }
    }
   }
  }
 }
```

Appendix C

CD content

- PDF version of the thesis
- Code of the developed analytical tool
- Tex files with necessary content such as pictures