Programming for Mr-Reves (Reden virtual engineering system) in C#

Internship at Reden in Hengelo The Netherlands,
Period: 14 November 2016 - 1 February 2017

Jinhua Dekker s1183508

University of Twente
CTW, ME
Design Production Management(DPM)

Supervisors:
External supervisor: Niels Nijenmanting
UT supervisor: Wessel Wits

Date: March 6, 2017

UNIVERSITY OF TWENTE.

Chapter 1

Preface

Thanks to Reden for providing the internship spot and treating me kindly. I enjoyed the team outing very much and felt welcome at work to try things out. Final thanks to Niels Nijenmanting for supervising me and for having me make something that will be used.

Chapter 2

Summary

The complete process of bringing a new product to the market is called new product development, and for that well-defined requirements for the product to be made are needed. MrReves (Reden virtual engineering system) was built as a tool to help during new product development and can be used to explore possible solutions within the specified requirement boundaries. To attract more people to use MrReves and to provide contracted clients like FrieslandCampina with all the tools they need it is necessary to keep improving MrReves, adding new features and adjusting old ones. During this internship C# was learned and new features were added like colour coding for the parameters/properties and a radar chart, and old features were improved like adding a window that displays multiple dot plots instead of a single dot plot. These features and improvements were then implemented in MrReves and will become available to the users once updates are released.

Contents

Chapter 3

Introduction

New product development

New product development (NPD) entails the complete process of bringing a new product to the market[1]. During NPD well-defined requirements are needed, these have to be determined by costumer demands and market research. Usually the details are vague guestimates or only entail the requirements of the most essential things with a direct impact on the products perceived market or business need. Furthermore, no standard convention is used for finding the optimal solution and most companies have their own methods.

MrReves

MrReves (Reden virtual engineering system) was built to offer a standard method for NPD and to provide the "enabled engineer with the ability to virtually test their design"[2]. The tools provided in MrReves are to help analyse and visualise data in such a way that choosing the optimal solution both design and engineering wise is done faster compared to for example a spreadsheet. Still to keep improving the process even more the program needs to keep changing by adding new features and adjusting old ones.

Diamond

One of the companies that uses a tailored version of MrReves is FrieslandCampina. Their tailored version is called Diamond and it is used to find the optimum profit for their business by "depending on market demand, prices, available product and available capacity"[3]. By using Diamond the solution with the highest profit is found after optimization. Most work done during this internship was to provide the tools and options that FrieslandCampina requires in MrReves to improve the solution finding process.

Report

This report will not discuss programming details for most of the work went into writing the code itself. In this case such details are not useful to document so the focus is on the general process and the functionalities that were added. To still give a general idea of the programmed features and the work spend on them many figures are included. The structure of the next chapter detailing the work during the internship starts with an introduction to a 'topic' that has been worked on and some additional information when needed. After the introduction the rough lines and the design process during each 'topic' are described with figures to support the descriptions. At the end some recommendations are suggested and the conclusion shows the end result of the programming work.

Chapter 4

Activities report

4.1 Learning C#

At the start of the internship C# needed to be learned, this was done with a crash course of 2.5 days using pluralsight. Pluralsight is a website that provides several video courses on software development programs amongst others. Two course were followed one about the “C# essentials” and another one about “best practices for C# programming” . After that a small exercise in object oriented programming (OOP) followed.

4.1.1 Object oriented programming

Object oriented programming (OOP) is a way to program using ‘object’ [4]. These objects contain ‘behaviours’ (methods) and ‘properties’ (attributes) which can be edited through the methods. Using multiple objects and letting them interact with one another is a possible way of create a computer program using OOP.

For the exercise in OOP the goal was to create a ’ball’ object that would bounce of the boundaries of the window panel. This ball object would have several attributes like colour, size, speed etc. With C# it is possible to make a derived object from a previous made object. This object can do the same things as the ‘parent’ object yet it can also be adjusted to have a different ‘ball’ shape and bounce behaviour. Several objects were created, some decreased speed with each bounce and others changed colour depending on the speed of the ‘ball’. After this assignment was considered finished by the supervisor the week ended with a small graphical adjustment of an already existing tool within MrReves.

4.1.2 Sankey Panel

The final assignment of the first week was a very small assignment to get started with the already existing code of MrReves. The arrow in the sankey panel had a solid colour which looked plain and needed a visual tweak. The goal was to figure out if it was possible to get a gradient in the colour. The adjustment was finished quickly since C# contained a function that would overlay a colour gradient in a specified drawn object/shape. This concluded the first week.

4.2 Solution explorer

The second week the real work started. One of the main ‘utilities’ in MrReves is the solution explorer which is a chart that contains data points like a dot plot. These data points are depicted usually as circles/dot and are plotted along a simple x and y scale. In MrReves each dot represents a possible solution set that fit the mathematical model and predefined constraints and requirement. All the solutions combined form a dot cloud of solutions. Along the x and y axis it is possible to compare two parameters for all solution. It is also possible to use multiple solution explorers with different y (and x) properties to compare. This is a good way to examine

the possible solutions that fit a model and to see how the properties correlate and fit together. In addition, when using boundaries it is possible to see if the preferred output parameter values are within range.

4.2.1 colour coding

As mentioned before in the introduction FrieslandCampina uses a business solution of Mr-Reves which gives them a tailored version. Within FrieslandCampina they use something called colour coding to visually link different properties together. A visualization of this can be seen in figure 4.1. Take for example the component casein: How does this affect the abstract property appearance, taste and nutrition of a product? Each of those properties can be linked to other more concrete measurable properties. To find the influence of component casein on the abstract property it is important to know the influence on the concrete properties. By giving properties that are related the same colour coding this relation can be seen without much effort, for this reason colour coding was implemented.

Figure 4.1: Colour coding usage by FrieslandCampina.

4.2.2 Programming

Aside from the colour coding that had to be added other adjustments needed to be made. The first adjustment that was needed concerned the option of having multiple solution explorers open for comparing parameters. Currently the main solution explorer (the parent) only had a button to click which would create one copy (a child) of itself and place it into a new window. This process would need to be repeated in case of needing more than 2 plots. After opening the required amount of additional solution explorers it was still necessary to arrange the multiple windows on the screen which was a hassle. The goal was to have one window that would auto arrange the opened child plots. The next feature that needed to be added after having one window for all child plots was the ability to change the x-axis of all child plots at the same time. This would be useful if the y values of the child plots would cycle through the properties while the x property would be the same for all plots. This way the influence of property x on (all) other properties could be seen.

Multiple solution explorers

Starting with the window containing multiple solution explorers, the first thing to do was create a form, a representation of a graphical user interface window[5]. The form needed to be divided into parts using a grid that would be filled with solution explorers once it was opened. This form would be shown as soon as the button for adding a solution explorer was clicked. The new plots iterate through the y-properties so each plot shows a different comparison. To give

the user more freedom in deciding the amount of plots and how they are ordered it is possible to adjust the vertical and horizontal amount of plots. The window will automatically fill all empty grid boxes with new plots. To adjust the amount of plots in vertical and in horizontal directions two selection boxes and corresponding simple icons to indicate the meaning of each box were added as can be seen in figure 4.2.

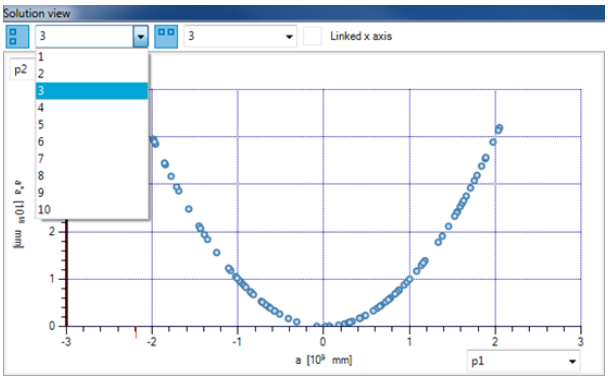


Figure 4.2: Adjusting the solution explorers window.

Link x axis

Now the multiple solution explorers window was finished the ability to change the x-axis of all child plots at the same time could be added. This was done by a having a check-box for linking the x-axis as can be seen in figure 4.3. When the link x-axis box is checked and an x-axis property is changed all other child plots are notified and change their x-axis property to the same property.

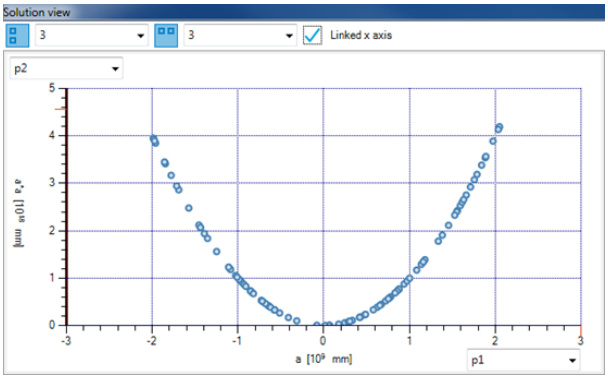


Figure 4.3: Link the x axis in the solution explorers window.

Colour

The final adjustment to the solution explorer that needed to be done was the colour coding possibility that FrieslandCampina would like to use. The biggest challenge was to find a compromise between making the colour coding noticeable without looking unprofessional or being distracting. Various test figures were made to compare as can be seen in figures 4.4. After discarding most options as being too distracting and looking unprofessional the colour coding in 4.4f was chosen (while 4.4g was 'kept' as a reconsideration for in the future).

Sort

After adding the colours it was considered useful to have parameters with the same colour as close to each other as possible. The plots needed to be sorted on colour and on the amount of each colours present in the window when the 'sort panels' button was clicked. The sorting

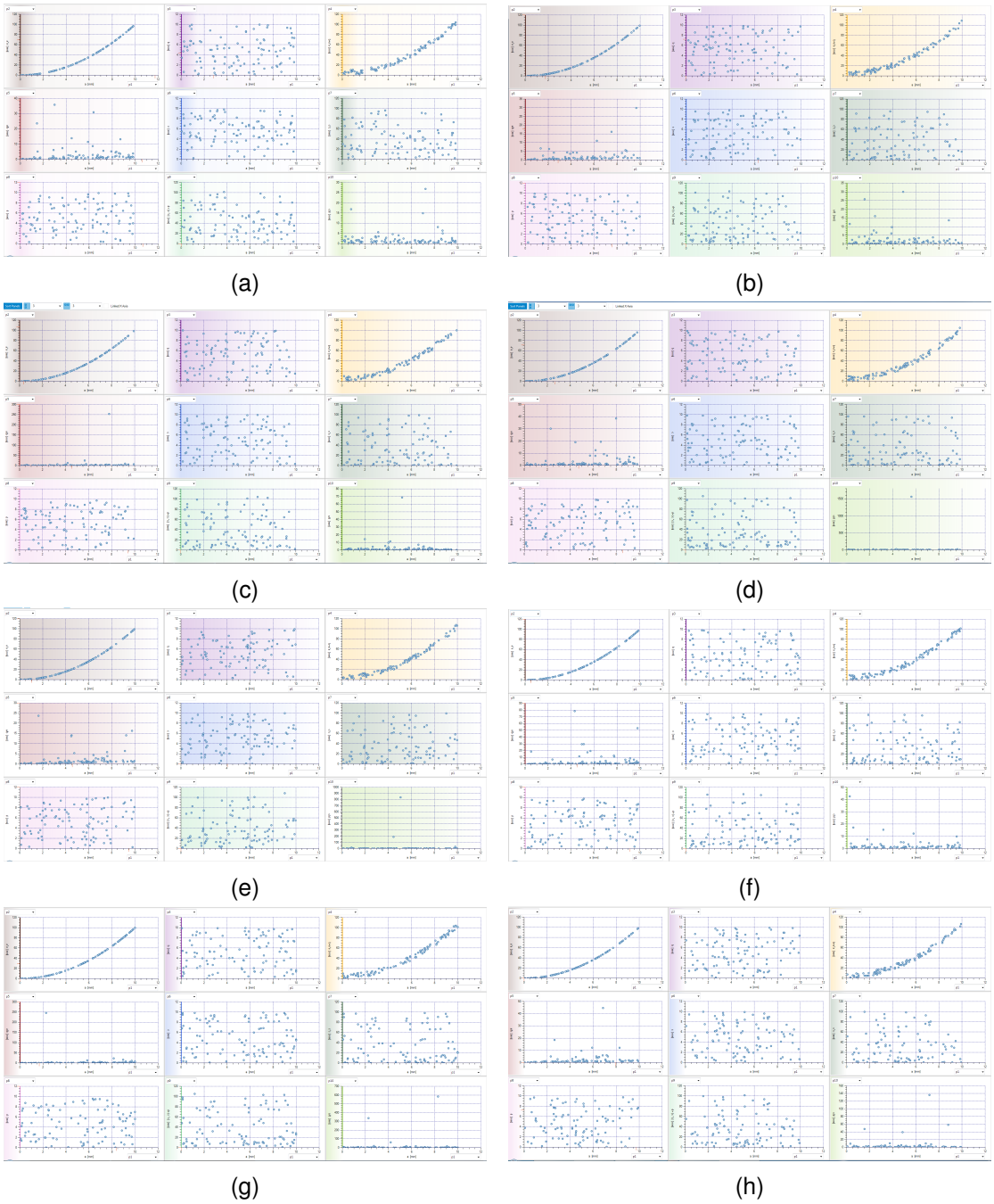


Figure 4.4: Possible combinations/ways of colour coding the solution explorers.

on the amount of each colour is necessary so each colour uses the least amount of rows. The colour with the largest representation would be mainly on the first row(s) depending on the columns selected and the colours with the smallest representation would be at the bottom.

The final result of the solution explorer adjustments after some tweaks can be seen in figure 4.5

4.3 Radar chart

The next assignment was to make a radar chart (RC) that could be implemented in Diamond. A RC is a useful tool to use during new product development and an effective way to compare multiple solutions with each other. [6] This is possible due to the a circular shaped of the chart with the centre being the start for each axis leg. Each axis in the chart can display a different parameter and a line in the chart represents a solution, which is a set of the different parameter

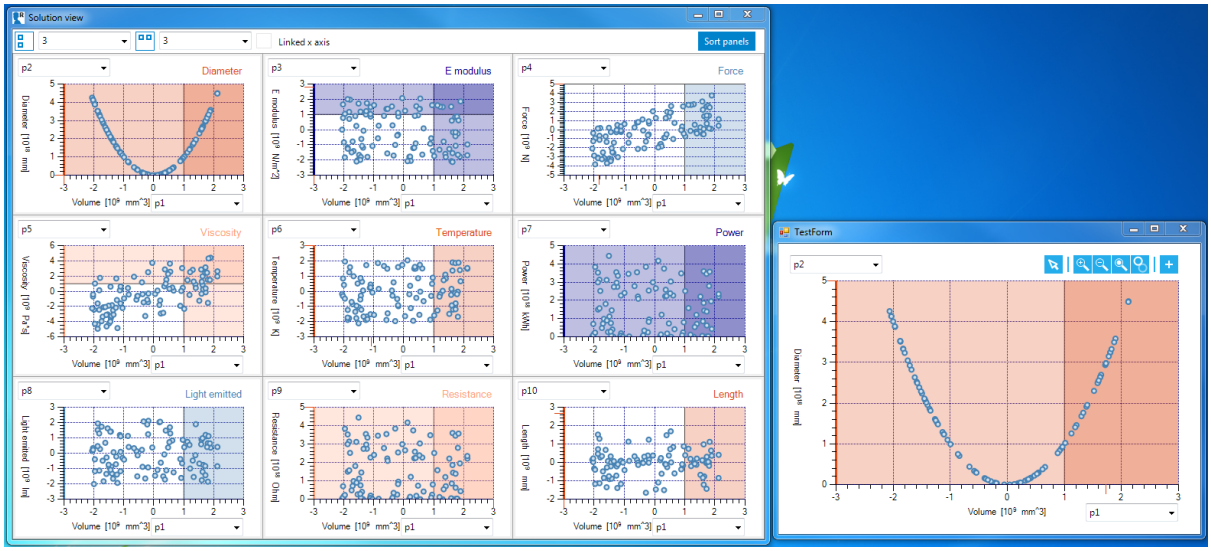


Figure 4.5: The resulting solution explorer after adjustments.

values. Also due to the parameters being in one chart it is possible to quickly see any solution with parameter values that are outliers. In addition, property differences between products are shown on one axis line which makes it easy to compare.

4.3.1 Implementation in MrReves.

During the created of the RC is was envisioned to become an addition to the solution explorer. Each dot in the solution explorer is a solution with different parameter values. Those can be draw in the RC as a solution line. The solution explorer can be used to filter preliminary data and the RC can be used to further inspection and compare the remaining solutions. In the end the RC was implemented in the dev(eloper) version of Diamond in a window called ‘elemental details’.

4.3.2 Programming

Before starting with coding the RC some research was done on the possible looks of circular charts and additional functionalities that could be added. This gave some useful idea and inspiration for later in the creation of the RC.

Basics

The RC was built from scratch and consists of the following items:

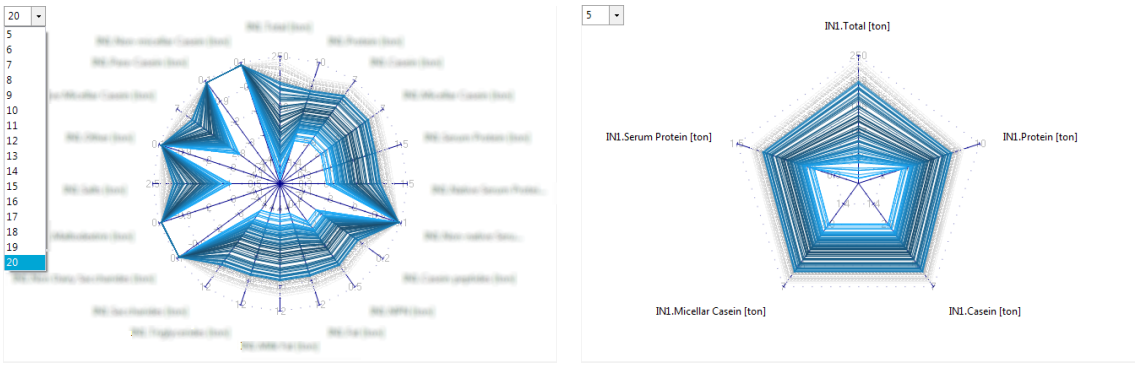
- An axis class that creates an axis object.
- A line class that creates a line object.
- An item class that stores and processes the input data.
- A control (a type of form) that handles the drawings and user end of the chart.

The control also contains the method to set a list of solutions or to add a list of solutions when another list has been set before during the session. This data is then processed to determine:

- The amount of axes displayed at the start.
- The min and max grid values along the axes.
- The position of the solutions on the axes, each set of data in the list is a line in the chart.

To view more or less parameters in the chart it is possible to change the axis amount through the selection box seen in figure 4.6. The minimum amount of axes is set at 5 axes since a chart with less than 5 axes cannot be considered a RC. The maximum amount of axes is partially determined by the maximum axes amount the user sets in the setting of the RC control and the length of each solution line set. If the solution line sets are smaller than the maximum axes amount then the axes cannot be increased beyond the solution line set count. If the maximum axes amount is the smaller one, then this value will prevent the chart from becoming larger.

To give a clean look to the chart the drop down boxes to change the properties along the axes are hidden 'under' the property names along the axes. Hovering over the property name will highlight it and show a hand icon to indicate it is click able (figure 4.7a). When clicking a property name the name will be hidden and the drop down box will be unhidden. Now it is possible to change the property on the axis (figure 4.7b). Once the user click somewhere else or clicks another property name the drop down box is hidden again. Before the property name is unhidden the new alignment of it along the axis is recalculated.



(a) (b)

Figure 4.6: Change the axes amount.

(a) (b)

Figure 4.7: Change the property that is displayed on the axis.

The RC also has several settings that are the same as the solution explorer to create cohesion, one is the toolbar. With large amounts of data it is necessary to be able to select a line or multiple lines. This selection can be kept (while excluding the rest), excluded or included (if currently excluded). Excluded lines can then be hidden so the remaining lines can be observed better. These actions can be done by clicking a toolbar item (figure 4.8a).

When a selection box is made by dragging the mouse over the chart only the inclusion of the solution point on the axis determine if the line is selected. This was done in case of many lines going criss-cross through each other which would make it difficult to precisely select a

group of lines.

Furthermore, just as with the solution explorer it is possible to display a panel with detailed information on the values in the RC on an axis. Depending on the settings selected this can be done by clicking or hovering over the value on the axis figure 4.8b).

(a) Radar chart toolbar.

(b) Radar chart dot data panel.

Figure 4.8: Radar chart additional features.

Colours

With the basic functionality of the RC working it was time to start thinking and experimenting with colour coding the parameters. During the initial research of the RC two possible options where found that could be used for colour coding. The first option (figure 4.9) did not do the trick. The colour did not attract attention since the opaqueness of the colour could not be too high or it would interfere with the line and axis visibility, furthermore it did not look 'clean'. The second and chosen option for the colour coding can be seen in figure 4.10. By placing the colours perpendicular to the corresponding axis and placing it below the axis property name it is identifiable with that axis. In addition, by placing the colours outside the chart the opacity of the drawn colour could be made solid without interfering with the readability of thing drawn within the chart outline. For the RC sorting the colours did not seem like a priority or a necessary thing to do. This is because the RC is more compact compared to multiple dot plots window, making it easier to see which parameters have the same colour.

Grid values

The next thing to do was to improve the chart, first the grid values needed to be improved. Though to do that the type of input data needed to be considered. Would the data be scores, percentage, monetary values or values of physical properties? With scores and percentage all axes would need to have the same value and the values of the input data would need to be normalized to put on the axes. For monetary values or values of physical properties the real values would make more sense in most cases to spot the outliers. Both options should be available when using the RC. Another thing that needed to be considered when changing the grid values was how to keep them readable without looking messy or being in the way? The following decisions were made for the absolute and the relative values.

Relative values

For the relative values it was chosen to only have a maximum of 4 axes that would display the values (figure 4.10a). Since all axes have the same value it is not necessary to see the values on each axis. Having the values only on the top most axis is common although when the chart becomes bigger it might be inconvenient to check the first axis for the grid value. Due to positioning and alignment difficulties when placing the values next to diagonal axes it was

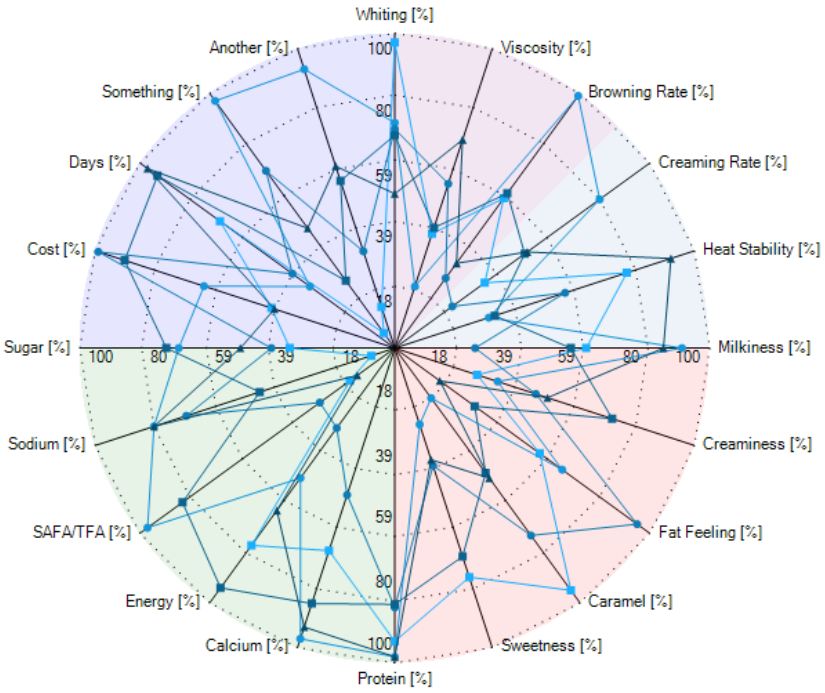
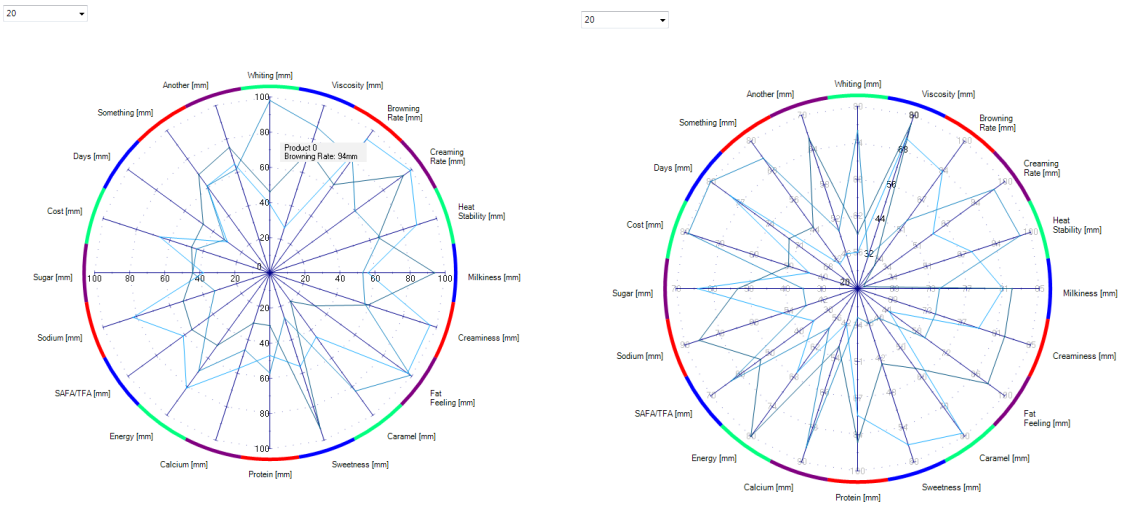


Figure 4.9: First try at colour coding the radar chart.

chosen to only have the values next to the vertical axes and below the horizontal axes, which also gave it a cleaner appearance.



(a) The chart and the grid values when the values are considered relative to each other.

(b) The chart and the grid values when the values are considered to be absolute.

Figure 4.10: Grid values.

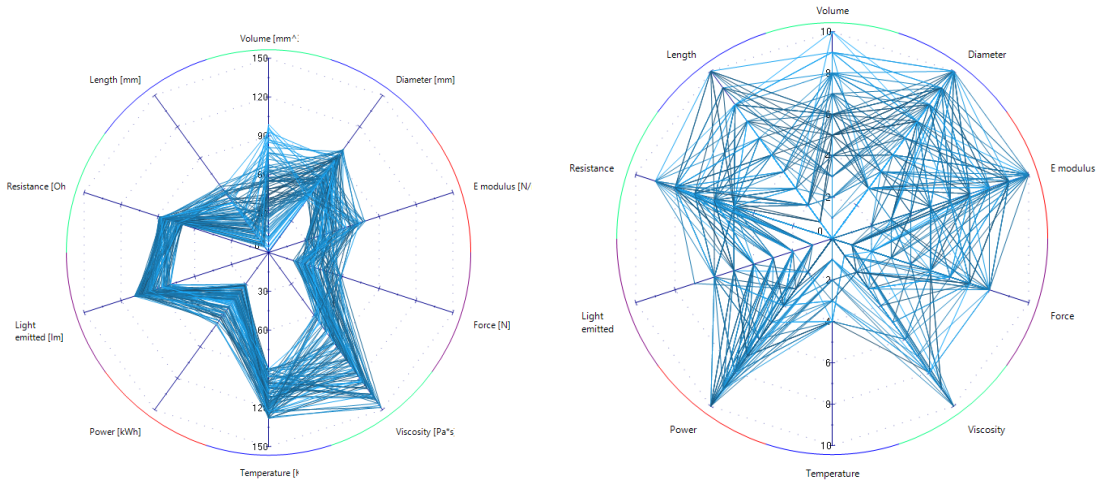
Absolute values

With the absolute values each axis would have its own axis values, which meant that each axis would need to have grid values visible. Since aligning the values next to the diagonal axes was difficult it was decided to make the axes a lighter coloured and more transparent than the grid values which would allow the grid values to be aligned on top of the axes. Nevertheless, having all the grid values in black in the chart made it look cluttered especially near the centre of the chart where the space between the axes was much smaller. This was solved by making

the axes values slightly transparent and light grey which caused them to blend in with the background while still being slightly visible. To see the values of an axis more clearly the mouse needs to be hovered over the axis to make the grid values and the centre value appear in solid black (figure 4.10b).

Relative conversion

Fast forward and the RC was finished and could be implemented into Diamond. After testing the RC in combination with the solution explorer the question arose how the relative values should be done. Did the values need to be converted or would the relative value type only mean something about the axis minimum and maximum? In the end it was decided to give the relative values an additional specification in the form of 2 different kind of relative values types. These would only be used when the data is labelled relative. The first option was to not convert the data, useful when no conversion is needed or when the advantage of having all the axes the same is needed. The second option is to have a (0-10) score. The main difference is that units are not used behind the values. The RC code itself does not implement a conversion of the data to a score, this is currently only done by the form that contains the RC in diamond. Before the data is set for the RC the data is automatically converted if the parameters have an upper and a lower boundary in the solution explorer. The closer the value is within the boundaries (, closest to the centre between the boundaries) the higher the score. Values that are outside the boundaries have a score of 0. If the lower and upper bounds are not set then no conversion is done.



(a) The values are not converted.

(b) The values are converted to a score.

Figure 4.11: The chart and the grid values when the values are considered relative to each other.

Animations

The last addition to the RC was animation in the form of axis and line animation. Animation makes the RC look more professional and makes changing axes amount and changing parameters feel more fluid and natural. Animating the RC was one of the bigger challenges since 3 different animation cases presented itself:

- Line animation without axis animation.
- Axis animation without line animation
- Axis animation with line animation

The first animation that was started with was the line animation. This line animation was for when the parameter of an axis would change so axis animation was not considered. This animation could be produced by calculating a distance vector between the current location and the next location. With each time step the solution point would move a fraction of that distance until the end time, which gave the line the illusion of moving.

After the line was animate-able the question arose if the change of axes amount should also be animated. This animation was not that difficult when the animation of the line was not considered. The axis animation behaves like opening and closing a hand fan with the first (top vertical) axis being stationary. Due to the line not being animated the solution point of the disappearing axes do not move on the axis. This gave the effect of seeing more than on solution point on the top axis near the end of the animation for a relative long time. It was decided that these solution point would not be drawn to prevent this effect. Since each line is drawn as a polygon, connecting all the point in its point list, it was not a problem to exclude those axes. All other values on the axes that only moved slightly were just handled normally by finding the data point on the new axis location each time.

Both animations could now be enabled individually though in most cases both axis and line animation would probably be enabled. For the animation during the changing of axes amount the previous mentioned method could just be used however because the lines animation was enabled the animation could be made even more ‘realistic’. Visually the most logical thing to see would be the solution point on new axis to appear out of the point of the top axis and to disappear into that point. This was the most complicated animation to program since the line had to move a distance while each solution had to stay on the moving axis during axis animation. Due to this it was not possible to use a direction vector for the line animation. In addition, when the axes had different max and min values it was important that the solution ’dot’ would know the actual drawing distance that it would need to travel instead of the distance value. In the end for each point on an added or removing axes the drawing distance to the position of the solution on the top axis was calculated. For each time step the point on the moving axis would be drawn closer to the final point location by calculating the location on the moving axis.

Changeable radar chart settings

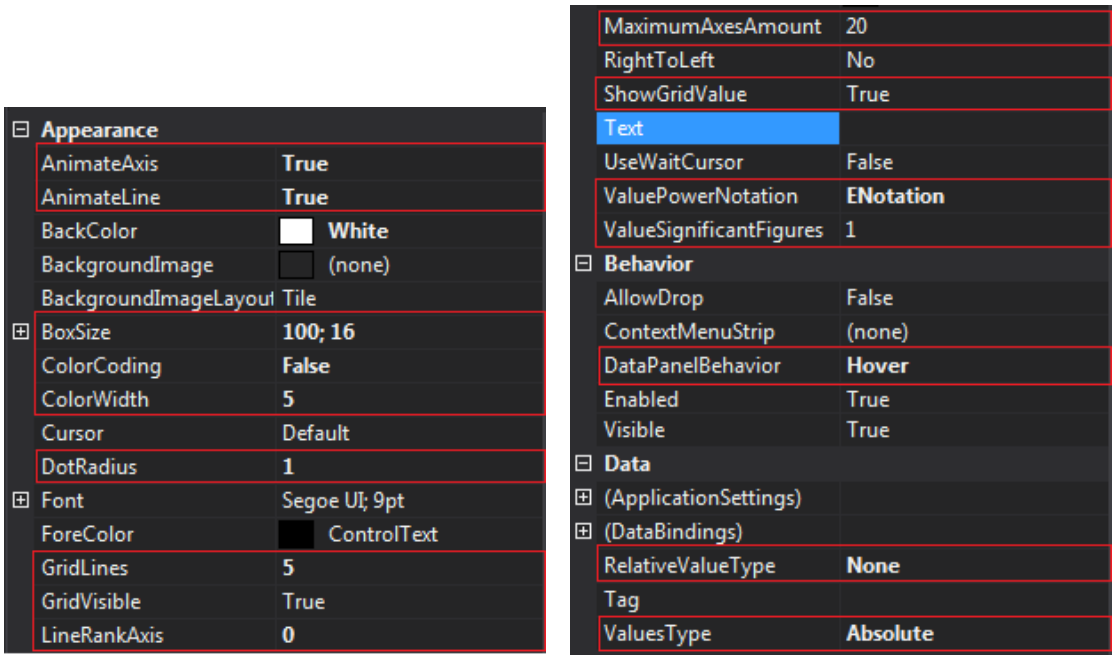


Figure 4.12: Public properties of the radar chart that can be adjusted.

The chart control has multiple changeable setting some of which have been mentioned before. Every line that is surrounded by a red box is part of added changeable settings. A summary of the settings and its purpose (when it has not been mentioned before) is mentioned below.

Settings

- AnimateAxis: Axis animation on/off
- AnimateLine: Line animation on/off
- BoxSize: The initial size of the label and selection box, this determines the chart and axes size.
- ColorCoding: Display the colour coding ring
- ColorWidth: Adjust the thickness of the colour ring
- DotRadius: The radius of the 'dots' that are drawn on the solution points on the axis. Setting the radius to 0 'hides' the 'dots'.
- GridLines: The amount of grid lines in the chart. This also determines the amount of tics on the axis and the amount of the grid values.
- GridVisible: Show/Hide the grid lines in the chart. This does not hide the tics on the axis.
- LineRankAxis: Set the axis that is used to determine which solution line has which colour from the array of gradient pens. This first line from the top has the first pen in the gradient array.
- MaximumAxesAmount: The maximum amount of axes that can theoretically be created, not considering the length of the solution line.
- ShowGridValue: Show or hide the grid values numbers.
- ValuePowerNotation: The power notation that is to be used when displaying the dot value in the dot data panel. The power notation can be default which only uses an E power notation when the value string becomes too long or it can be an E notation with the amount significant figures decided by the user.
- ValueSignificantFigures: Set the amount of significant figures to use when the ValuePowerNotation is not set to default.
- DataPanelBehavior: Set whether hovering over a solution point or clicking on it will show the dot data panel
- RelativeValueType: Set if the relative values should be treated like a score or just like general relative values
- ValueType: Set whether the input values are relative of absolute

Elemental details window

Finally, the RC was finished and suitable for implementation into Diamond. The location for the RC would become a window that was called the elemental details window. The elemental details window already contained the solution explorer, a solution data panel and a parallel coordinates chart. It was decided to make the RC a panel that could be "hidden" to decrease the impact on the sizes and readability of the other chart in the window due to adding the RC. By clicking "show/hide radarchart" on the right side of the panel a timer would be started that made the chart panel wider of smaller depending on if the panel was already unfolded. This give the illusion of a sliding panel. To fully integrate the RC all data from the solution explorer

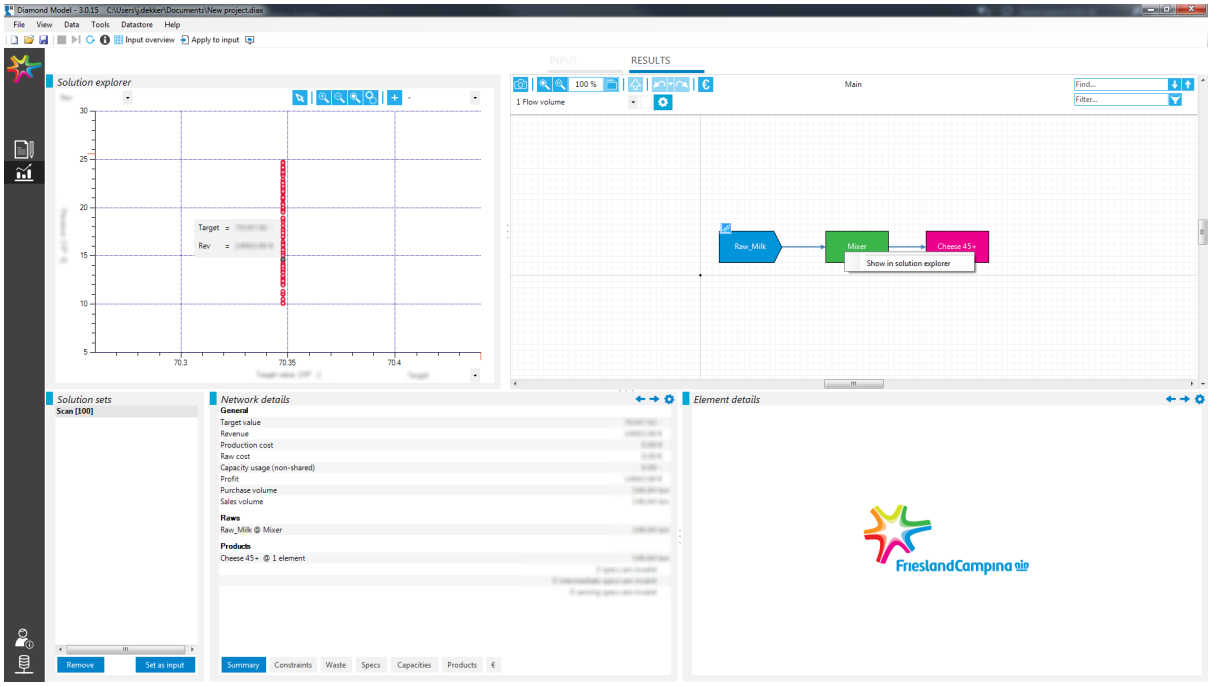
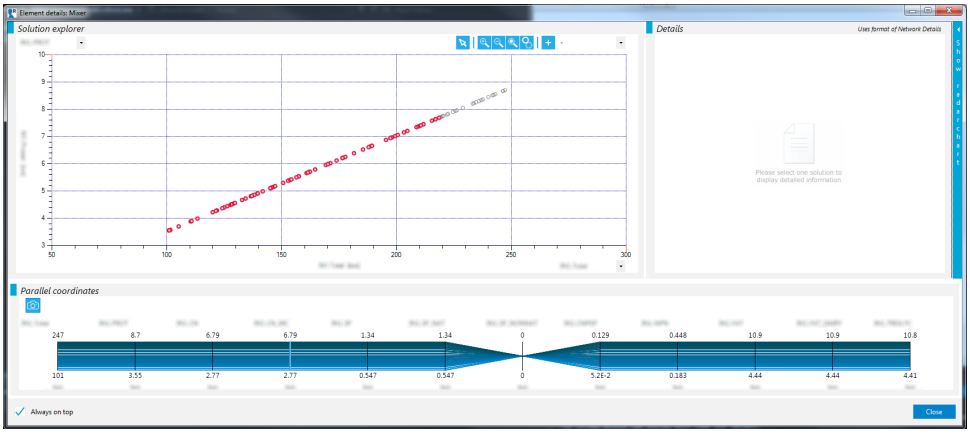


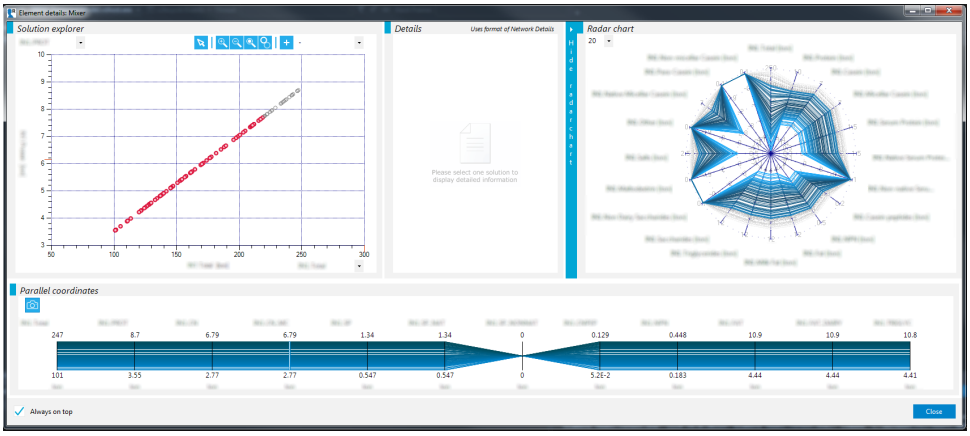
Figure 4.13: Diamond start screen. The elemental details window can be found by right-clicking a flowchart element.

had to be converted to be set for the RC. Furthermore, the solutions that are selected in the solution explorer will also appear selected in the RC and parallel coordinates chart and vice versa. In addition to that the excluded lines in the solution explorer will also become excluded in de RC. Once the implementation was finished the last days of the assignment were also reach and only a few minor visual adjustments were made in the end. The final result can be seen in figure 4.14.



(a) The radar chart is hidden.

(b) The radar chart is open and dots have been selected in the solution explorer.



(c) The selected dots have been excluded in the solution explorer and are also excluded in the radar chart.

Figure 4.14: Elemental details window.

Chapter 5

Recommendations

The following chapter contains a few recommendations concerning the code that has been worked with.

During adjustments to the solution explorer control the biggest problem was finding the right location of code segments. The reason for this is due to the size of the class. The solution explorer control is one of the oldest items in MrReves and because of that has grown to an + 5000 lines size before realizing it. It is recommended to try to find the time or a person to at least split the class two files, for example a class that manages the empty chart parts (like axis, grid, tics etc.) and a control for the rest. Also, a public SetExcludedIndices would be useful for the solution explorer since changing the excluded lines in the RC in the solution explorer does not synchronise it with the solution explorer.

The RC does contain the most essential options to be practical, useful and visually appealing. Further additions to make it more advanced could be the possibility to display upper and lower boundaries like those used for the solution explorer in the chart and to auto exclude lines outside the boundaries. Furthermore, the display of data values in the data panels for the solution dot and the grid line values could be improved to be formatted better and handle significant figures in a better way. In addition, the ability for each axis update its grid maximum and minimum value could also be useful when excluded solutions are hidden (permanently). Finally, a method for the relative score values type conversion could be implemented.

Chapter 6

Conclusions

During this internship new programming skills like C# and object oriented programming were learned and a finished ‘product’ was delivered. The adjustments to the solution explorer were received with positive feedback and are already being used in practice. The created radar chart does fit nicely in the solution explorer panel and should be a useful addition to the already available tools. The results in the elemental details window can be seen again in figures 6.1 and 6.2.

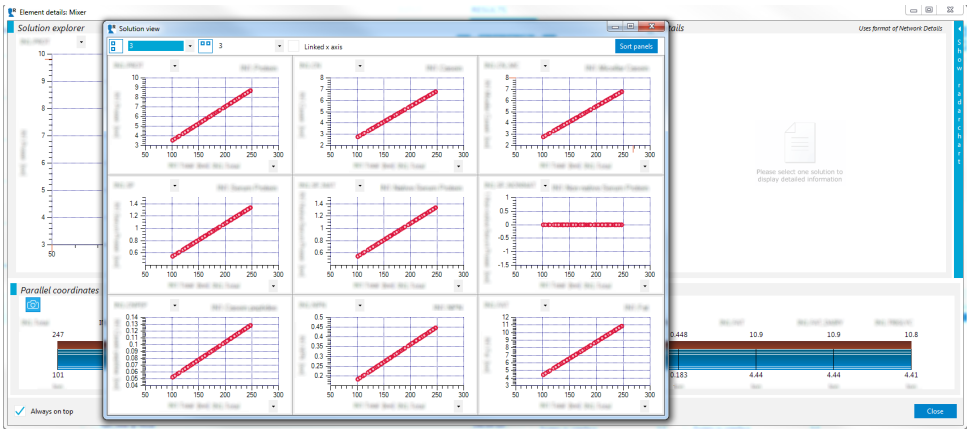


Figure 6.1: Solution explorers in the elemental details window.

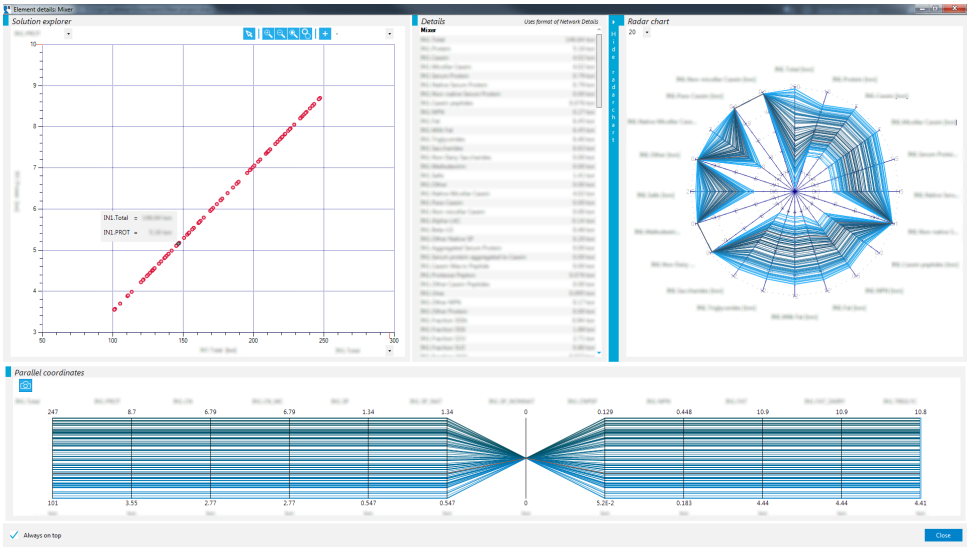


Figure 6.2: Elemental details window.

References

[1] Wikipedia. (2017) New product developement. [Online]. Available: en.wikipedia.org/wiki/New_product_development

[2] Reden. (2017) Mrreves knowledge builder. [Online]. Available: http://www.mrreves.nl/

[3] ——. (2017) Mrreves business solutions. [Online]. Available: http://www.mrreves.nl/businesssolutions

[4] Wikipedia. (2017) Object oriented programming. [Online]. Available: https://en.wikipedia.org/wiki/Object-oriented_programming

[5] ——. (2017) Form (programming). [Online]. Available: https://en.wikipedia.org/wiki/Form_(programming)

[6] ——. (2017) Radarchart. [Online]. Available: https://en.wikipedia.org/wiki/Radarchart

Appendices

Appendix A

Reden

Reden is a small company that is part of the TechnoDome Group (‘mother company’). Reden specialized in virtual product testing by creating models of products and systems. The models can then be used to optimize the systems or products without the need to test it outside of the virtual environment.

All people within Reden are engineers and the ambiance is one of an engineering company. Since most work can be done behind the computer the social level in the company can be described as social in an engineers kind of way. To have some social interactions during working it better to be able to understand (and speak) Dutch since all workers are Dutch and casual talks in English can be difficult when you are not used to it. It is possible within Reden to have some flexible working hours. Not every workday needs to be 8 hours and not every week needs to be exactly 40 hours. It is possible to shift hours as long as this is discussed with the supervisor. Reden is also not a company that requires their workers to work from 8 to 5 starting and ending times can be flexible however usually most people arrive between 8:30-9:00 and have a 1 hour break during lunch time.

Appendix B

Reflective essay

During my internship at Reden I confirmed my feelings that I do not want to go into research or stay at the University longer than needed. I have enjoyed the work environment in a small company and would probably prefer that too in the future. This internship has also given me a better view on what kind of work I would like to do in the future. I've always had an interest for programming however never really considered it as an inclusion in my future jobs until recently. Programming a user product was very interesting and kept me concentrated for relatively long times, which is not that common for me. I enjoyed the mental work of placing yourself in the consumer/user and considering what they want or would need, while keeping in mind that it should be logical and useful. Of course, there is room for improvement in that part since it was my first real time doing it.

From what I have gathered from my ex-colleagues was that they did enjoy me as a colleague at work. I did not receive negative comments on my style of working or socializing with everybody however I did get some useful feedback about letting myself be 'known'. Since most of the times I'm a quiet person with new people or people that I'm not close with I tend to blend into the background. Though in a company it is important to be noticed a show your presence and let them know you are doing a good job. That is why it has been advised to me to give more progress updates/feedback even when things are going great.

Before finding an internship I used to be afraid that it would give the same demotivating experience as my bachelor assignment. However the main problem with the bachelor assignment was the feeling of needing to repair unfinished work. To keep motivated at work I need to have the feeling that I can give my own style to it and create something which works and can and will be used. This is something I'll keep in mind for the further when accepting an assignment or job.