TU WIEN Informatics

Utilizing Visual Methods to Generate Synthetic Time-Oriented Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Stefan Weiser
Matrikelnummer 0625052

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung
Betreuer/in: Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Silvia Miksch
Mitwirkung: Mag. Alexander Rind

Wien, 16.04.2021

Unterschrift Verfasser/in | Unterschrift Betreuer/in

Technische Universität Wien
Karlsplatz 13 | 1040 Wien | +43-1-58801-0 | www.tuwien.at

TU WIEN Informatics

Utilizing Visual Methods to Generate Synthetic Time-Oriented Data

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Stefan Weiser
Registration Number 0625052

to the Faculty of Informatics
at the TU Wien

Advisor: Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Silvia Miksch
Assistance: Mag. Alexander Rind

Vienna, 16th April, 2021

Signature of Author | Signature of Advisor

Technische Universität Wien
Karlsplatz 13 | 1040 Wien | +43-1-58801-0 | www.tuwien.at

Erklärung zur Verfassung der Arbeit

Stefan Weiser
Adelheid-Popp-Gasse 5/6/19, 1220 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Acknowledgements

I want to thank my supervisor Alexander Rind who supported me during the whole time. He guided me through all necessary steps and gave me helpful feedback during the implementation and the writing part. Additionally he took over the part for the expert inspection, which was essential for the evaluation part.

Furthermore, I would like to thank my colleagues Lukas Mad and Alexander Jakovljevic who participated in the user study. They spent their leisure time for the evaluation of the implemented prototype.

Finally I want to mention my parents who supported me during my whole study, and my wife who always was sympathetic about me.

Abstract

The visualization of time-oriented data is an essential part for analysis, which is solved by software applications in today's digital age. The graphical preparation of time-oriented data is additionally a difficult task, because time is a complex variable. Visualization techniques try to prepare time-oriented data in a graphical way to identify specific patterns and structures. For the development of such techniques sample data is required, which is used for testing and demonstration. Data can be obtained from various sources, in which real data is not always available, e.g. due to legal issues. In such cases synthetic data can be used.

Synthetic data is produced by data generators. There exist several generator, in which most are not specialized in time-oriented data. Even fewer generator are able to visualize the data they generated. In order to close this gap this master thesis presents a software design, which is able to generate time-oriented data in a visual way. Data will no longer be generated first and then visualized, data is generated directly from visualization.

This master thesis describes a software design which is able to generate time-oriented data with visual aspects. An additional aim is to provide a design, which is good enough for both expert and non-expert, so they can work with a corresponding implementation. This design is represented by a prototype, which is also used for the evaluation. An expert and two users, who are not experts on time-oriented data, use the prototype and generate specific data sets.

Kurzfassung

Die Visualisierung zeitorientierter Daten ist ein wichtiger Bestandteil zur Analyse und Aufbereitung, um große Menge von Daten leichter zu erfassen und überblicken zu können. Die grafische Aufbereitung zeitorientierter Daten ist zusätzlich eine schwierige Aufgabe, da die Zeit eine komplexe Variable darstellt. Visualisierungs-Techniken versuchen zeitorientierte Daten graphisch aufzubereiten, um spezifische Muster und Strukturen erkennen zu lassen. Für die Entwicklung solcher Techniken sind Beispiel-Daten notwendig, um neu entwickelte Techniken zu testen, aber auch präsentieren zu können. Hierfür können Daten aus verschiedenen Quellen bezogen werden, wobei nicht für alle Szenarien reale Daten verfügbar sind, z.B. aus rechtlichen Gründen. In solchen Fällen können künstliche Datensätze aushelfen.

Künstliche Daten werden von Daten-Generatoren erzeugt. Es gibt verschiedene solche Generatoren, wobei die meisten nicht auf zeitorientierte Daten spezialisiert sind. Noch weniger Generatoren sind in der Lage die generierten Daten auch zu visualisieren. Um diese Lücke zu schließen wird in dieser Diplomarbeit ein Software-Design präsentiert, welches in der Lage ist, zeitorientierte Daten über visuelle Techniken erzeugen zu lassen. Daten werden nicht mehr zuerst generiert und anschließend visuell präsentiert, die Daten werden direkt aus einer Visualisierung erzeugt.

Diese Diplomarbeit beschreibt ein Software-Design, welches in der Lage ist zeitorientierte Daten über Visualisierungstechniken zu generieren. Ein zusätzliches Ziel ist, dass das Design sowohl für Experten/Expertinnen als auch für unerfahrene Benutzer/innen gut genug ist, sodass diese mit einer entsprechenden Software-Implementierung arbeiten können. Das Design wird durch Implementierung eines Prototypen evaluiert. Ein Experte und weitere zwei Benutzer/-innen, welche keine Experten/Expertinnen für zeitorientierte Daten sind, haben den Prototypen verwendet und spezifische Datensätze erzeugt.

Contents

CHAPTER 1

Introduction

1.1 General Introduction

Time as a physical dimension has been researched for hundreds of years from many different popular and less popular scientists. Albert Einstein (14. March 1879 in Ulm; † 18. April 1955 in Princeton, New Jersey), one of the most famous physicist, was working on time and its properties. He studied this subject in detail and at the end he defined the theory of relativity [12]. This history shows the importance of time and its properties.

Time itself is much more complex than simple numbers. It has many different forms of scale-factors (seconds in minutes, hours a day, etc.) and even such a scale-factor is not regular (the number of days in a month). So time in combination with data (time-oriented data) requires special treatment, because of the factor "time".

The treatment with time-oriented data is not easy and requires software solutions, which are able to analyze and visualize data and the aspect of time in an easy way. For this task software applications and frameworks (e.g., TimeBench [54]) have been developed, which facilitate the work with such data. The implementation of such software solutions dealing time-oriented data is dependent to sample data, which is required for testing and demonstrating the features of the software.

Sample data is essential in many different areas like software engineering, statistics or medicine. It is possible to use real data, which gets collected from records of the real world. But this kind of data could be expensive or cannot be used due to its secrecy. Additionally, it could be possible that specific data constellations are not available, because they do not exist. Theoretical scenarios cannot be analyzed with only real data.

Another way to get sample data is the generation of it. Synthetic data is produced by data generators, which are often used for software development. Such data needs to be prepared in standard formats, so that other applications are able to read and process the generated output. Preparing such data by hand requires heavily much time, time which can be spent on higher valuable tasks. An advanced solution for this problem is a data generator producing sample data, which is able to deal with the aspect of time and to configure the input parameters graphically.

Without such a data generator the development of applications dealing with time-oriented data slows down due to missing data for development and representation of the functionality. It is not possible to test the functionalities of the software in detail. Errors and bugs will not be found and the quality of software solutions is reduced. So it is essential that time-oriented data is provided.

1.2 Background and Motivation

Many data generators for different fields exist and are able to produce sample data. It is possible to receive such data from the internet or by applications. Often the input of the user is done via a simple command shell. But it is also possible to define the input parameters on a graphical user interface. A more advanced possibility is to define the input parameters graphically on a plot.

There already exist a variety of data generators, some of which have already graphical solutions (see chapter 2.5). The user has the possibility to draw shapes and lines which the software interprets as input for the generation process. Afterwards, the user can directly see the generated data visualized on the display. The feature sets of these generators contain graphical input definition but they are not able to deal with time-oriented data and its specific characteristics. Seasonal trends and time loops can not be defined quick and easy.

This master thesis close this gap and provide a software design which uses similar visual feature sets to generate time-oriented data.

1.3 Research Question

This thesis defines a main research question and a sub research question.

1.3.1 Main Research Question

What is an appropriate solution to create synthetic time-oriented data by using visualization techniques for input parametrization?

The answer of this research question will be given after the evaluation of the software design. The software design requires an prototypical implementation, used for the evaluation phase. This prototype will be a data generator with a graphical user interface. The generated data has a direct link to the time, so this prototype will produce time-oriented data. The input parameters will be defined graphically on a plot, i.e. the user is able to define value ranges directly on a graph.

1.3.2 Sub Research Question

What is an appropriate solution to draft a user interface design that experts and non-experts in time-oriented data are able to work, independent to introduction and user manuals?

This question will be answered after the informal user study within the evaluation phase. This user study contains of one expert on time-oriented data and two non-experts. Each person gets a set of predefined tasks and should solve it without a detailed manual or workshop.

1.4 Research Methodology

The research questions are intended to be answered by doing the following tasks:

- Literature research of existing data generation approaches, especially using visualization techniques and producing time-oriented data.
- Selection of valuable methods to evaluate the implemented prototype, so that the predefined research questions can be answered.
- Implementation of the prototype, creating synthetic time-oriented data by using visualization techniques for input parametrization.
- Evaluation of the implemented prototype by using predefined evaluation methods.
- Answering the two research questions based on done evaluation of the prototype.

1.5 Structure of the Thesis

After the introduction this master thesis starts with the state of the art, containing and presenting existing theories, solutions, and designs dealing with visualization of time-oriented data and the generation of synthetic data. Afterwards, the methodology describes the techniques and methods, which are used to research and answer the research questions. The next chapter describes the design of the data generator, which uses visual methods to generate time-oriented data. Then the design evaluation describes the evaluations and its results, which are based on an implemented prototype. At the end of this thesis both research questions are answered in the conclusion. The potential future work of this master thesis is described in the last chapter.

CHAPTER 2

State of the Art

This section deals with previous work in the different fields of visualization of time-oriented data and data generation. The first part describes time as physical size and time-oriented data. After that I will give an introduction of data visualization, especially of time-oriented data. Furthermore, I will describe other existing technical tools and frameworks for the visualization of time-oriented data. Previous data generators will also be mentioned. At the end of this section I will present guidelines and patterns for user interface design.

2.1 Time and Time-Oriented Data

Time is a complex variable and has been investigated for ages. For cultures around the world time influenced their daily life (e.g. rainy or dry season, harvest time, etc.). Time is important in many different fields of science, like philosophy, geology, mathematics or astronomy. The variety of different disciplines, where time is investigated, describes the significance of time and that time has an important status in our daily life.

Isaac Newton (25. December 1642 in Woolsthorpe-by-Colsterworth in Lincolnshire, † 20. March 1726 in Kensington) was an important philosopher and described the time as an absolute size. In his book "The mathematical principles of natural philosophy" [45] he characterized time as follows:

> Absolute, true, and mathematical time, of itself, and from its own nature flows equably without regard to any thing external, and by another name is called duration: relative, apparent, and common time, is sensible and external (whether accurate or unequable) measure of duration by the means of motion, which is commonly used instead of true time; such as an hour, a day, a month, a year. (Quote from [45], Definitions, p. 9)

In contrast to Isaac Newton the physicist Albert Einstein (14. March 1879 in Ulm; † 18. April 1955 in Princeton, New Jersey) described the time as a relative size, depending on each

individual. His publications about the theory of relativity [12] are significant up to the present time. On the base of Isaac Newton and Albert Einstein science continued investigating into the research of time.

2.1.1 Characterization of Time

Time as a complex variable in our lifetime is demonstrated and recorded on many different clocks and calendars. Many different forms of clocks and calendar systems exist. One of the most important calendar systems is the Julian calendar, which has been defined by Gaius Iulius Caesar. It was the defining calendar system in Roman times and is close to our present system. The Gregorian calendar was introduced by Pope Gregory XIII in 1582 and superseded the Julian calendar, which is still up to date in the presence [39]. Several common computer systems are working with the Gregorian calendar (e.g. the class GregorianCalendar[1] in the programming language Java).

The present time calculation around the world is a complex calculation out of years, months, days and further scale-factors. But time is not equal on each place on earth. We introduced time zones to line up our time to day and night. The main time zone is the Greenwich Time in London, the prime meridian. Today we use atomic clocks to define the exact time on earth. We try to define the time in relation from the earth to the sun. Because of the situation that the revolution around the sun and the earth axis is not absolutely regular and has small deviations, corrections to the time are done. So our time adapts to the astronomic conditions of the earth and the sun. The International Earth Rotation and Reference Systems Service (IERS) is able to measure the rotation of the earth and its parameters and arranges time corrections in form of leap seconds.

2.1.2 Time and computer systems

Data and applications often require specific time stamps for operations and saving data. Even the communication between different computer systems requires an exact time measurement. Operating systems, programming languages and applications use different commands to get the current system time. Depending on the technical implementation, they are able to define the time more ore less detailed.

The Unix Time [61] is a time format, which has been defined for the Unix operating system in 1969. It continuously counts the seconds since January 1st, 1970, which is also called the ”Epoch“. The period since this point of time is called the ”POSIX time“ [64]. On a 32-bit system the time is countable till January 19th, 2038. On this date an overflow will occur. So a newer time system is needed for this event. The 64-bit technology would be a solution, but requires a migration of existing 32-bit applications, which are still working with the old Unix Time system.

Microsoft Windows has a different calculation of time to the Unix Time. It calculates the system time as own type, which contains year, month, day of week, day, hour, minute, second

[1] `http://docs.oracle.com/javase/7/docs/api/java/util/GregorianCalendar.html` accessed 2019-11-04

and millisecond. Additionally there exists the ”Windows Time“, which counts the number of milliseconds since the last start of the system. It is a relict for the compatibility to the old 16-bit Windows systems. The ”File Time“ is a 64-bit value. It defines the number of 100-nanosecond intervals since January 1, 1601 (UTC) [43].

Other operating systems, programming languages and applications have their own functions and structure for the definition of the current time. The epoch and range of the different time formats vary and provide different functions and exactness (seconds, milliseconds, nanoseconds). In the following the complexity for calculations of time varies, due to their technical implementation.

2.1.3 The Relation between Time and Data

Industry and science are collecting data for calculations and research. The more data is available the more accurate predictions can be taken. In most cases the time is an essential size. Even the time, when data was measured and collected, is recognized. With the help of such time-oriented data it is possible to note historical records and to analyze these historical data sets. Out of the results of the analysis it is possible to get new knowledge and to make decisions based on the found knowledge.

As presented by Aigner et al. [4] the definition of time in data is different to the physical dimension of time. It is not required that an exact time stamp is saved in combination with a specific data set. It could be enough to save the exact date (year, month, day) without the exact time or only a year or a century. Sometimes it is of interest not to save a point of time, but a period. As it can be seen the context and the model, where time is used, is decisive. So it is necessary that the model is best fitting to the data content and that the time as variable or parameter has a good scaling factor. Time can be categorized into different sections and subsections, which provide different views, scaling-factors and other properties. These categorizations help to distinguish between interesting and not interesting aspects of the time, as it is required for the context and the model.

Figure 2.1 provides a detailed overview of a classification, as mentioned above. Explaining all displayed types of classification would go beyond the scope. The most interesting parts for this work are the factors arrangement and the number of variables.

- Arrangement distinguishes between linear and cyclic time. Linear time is easy and can be visualized with a simple timeline. In contrast a cyclic time defines a reoccurring loop of the time, e.g. a week or a month.
- Different variables can be observed along the time. Their correlation and dependence to each other are of interest.

2.2 Visualization

Visualization is the graphical representation of any kind of data and phenomena. Especially the visualization of data is essential and provides the possibility to get an overview and to identify

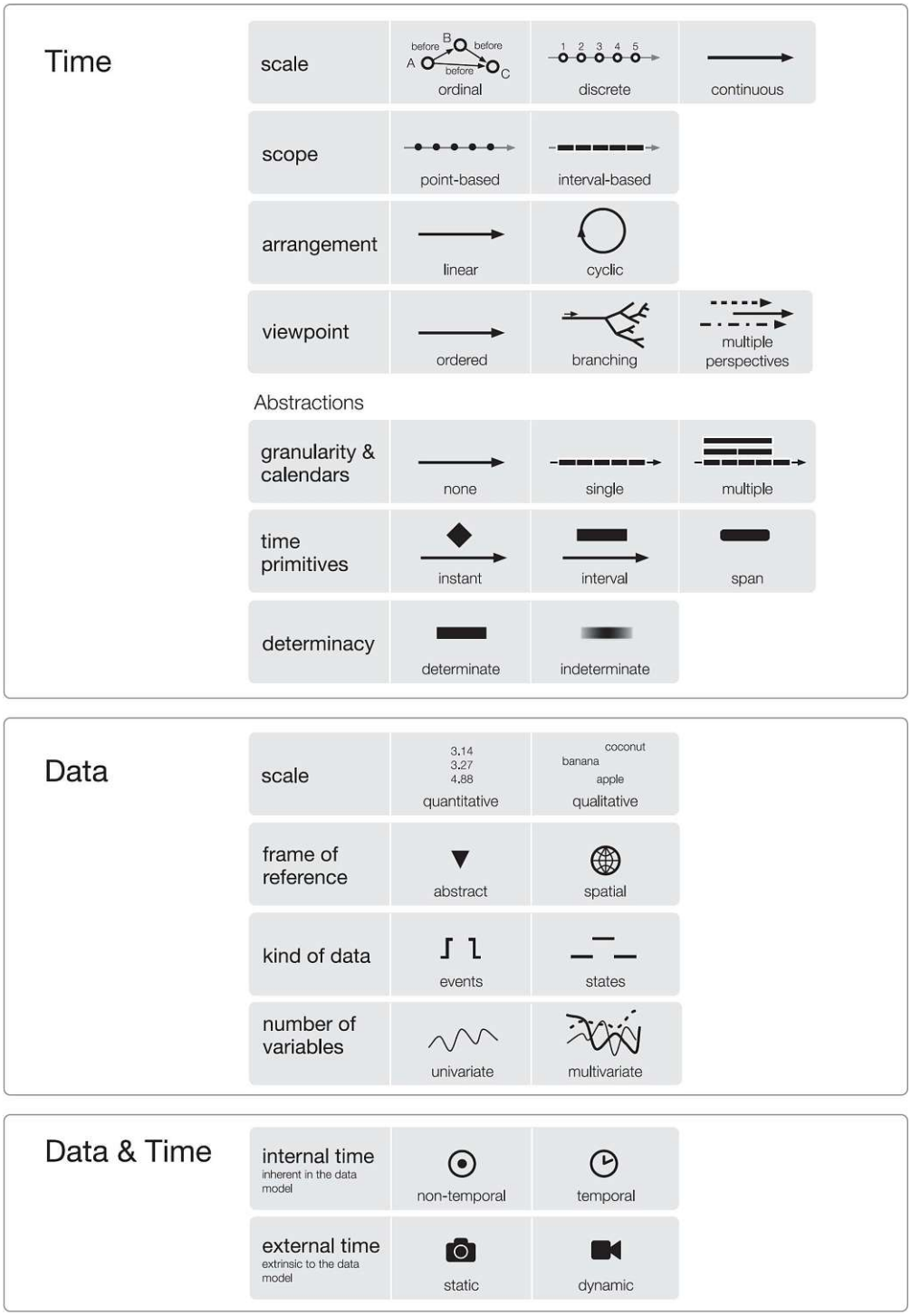

Figure 2.1: Design aspects of time-oriented data [4]

patterns out of huge sets of data. Early forms of visualizations have been done by hand, while nowadays computers and software solutions are responsible for a modern and clear graphical view of a given data set. Depending on the field and the topic different software solutions are used. Card et al. described the field of Information Visualization as follows:

> The use of computer-supported, interactive, visual representations of data to amplify cognition. (Quote from [16], p. 6)

The term ”Computer-Support“ points to the further evolution of information visualization. Computers are situated in the center and the development of faster and better hardware and software also leads to new visualization techniques. While at the beginning of the computer age hardware has been expensive and large, the visualization of information was very expensive. The possibilities of visualization were limited to the technology stack of the time. With the evolution of the hard- and software the opportunities increased and new visualization techniques were introduced.

The topic ”Information visualization“ aroused the interest of many people, so a community of scientists evolved, which specialize in this topic. IEEE InfoVis (IEEE Information Visualization) is an international conference and was initialized in 1999 in San Francisco, California. This conference takes place every year and is the most important meeting for experts on information visualization.

Different visualization techniques are necessary to display data graphically in an appropriate way. Depending on the requirements for the visualization different techniques do exist, which all have advantages and disadvantages, depending on the requirement and the needs. A whole list and description of all existing visualization techniques is impossible and will go beyond this thesis. The book of Robert L. Harris [26] provides an overview of the most important techniques, e.g. Point Plot, Line Plot. Depending on the context further techniques do exist.

Visualization techniques could also be categorized by their dimensions. 2D is a common and simple way for the representation of data in a graphical way. More complex visualization techniques are rendered in 3D. With the additional dimension more data can be displayed on one chart or graph. 4D models are hard to imagine for people but are an adequate instrument to visualize e.g. the time. In the paper of Zollmann et al. [75] a visualization concept is presented, which uses techniques to get an overview of 4D data on different levels of detail. Another example for 4D visualizations are the ”4D Space-Time Techniques“ [67], which also use the time as the fourth dimension to visualize changes on MRI (Magnetic Resonance Imaging) and dynamic SPECT (Single Photon Emission Computed Tomography). The visualization of real-time data is also an area of application, where four dimensions gets used. As example, Bista and Pack [11] describe a 4D visualization system, which takes multiple large scale simulation outputs and loads them into an existing 4D environment.

Many dimensions do not indicate a good visualization technique. Sometimes a simple 2D technique provides enough information and need not to be rendered in a costly 3D technique. Sedlmair et al. [59] provide an empirical study on Scatterplots and Dimension Reduction Techniques. They found out that 3D need not to be a good choice for cluster verification, because a 3D perspective could hide class structures and requires higher interactions costs, but do not add a significant benefit.

2.2.1 Visualization of Time

The visualization of time-oriented data is a subsection of the big topic ”Visualization“. During centuries research and industry have been collecting data, how significant numbers have been changing over time and capital and income has been rising and falling. So the requirement on tools for analysis and visualization increases, especially to depict the aspect of time.

> However, visualizing time-related data is more than ordering them along an axis.
> (Quote from [23], p. 310)

This quote illustrates that the visualization of time is not as easy as it seems initially. Silva and Catarci mentioned that most visualization techniques on time are done through interactive 2D timelines. More advanced approaches are done via 3D techniques. In their paper [23] they give an overview of many different visualization techniques of linear time-oriented data.

The timeline as a visualization technique is the easiest form. The time is displaced on a single axis and continues in one direction. It is commonly used in many different fields e.g., for time schedules or project plans. Despite of its simplicity the timeline is commonly used and also new visualization forms of timelines are designed. Craig and Roa-Seiler [20] present an innovative form of a timeline, where the timeline is vertically oriented (against the classical horizontal form). It is used to analyze human-computer dialogue data. Figure 2.2 shows the simple prototype of this form of a timeline. Another variant of a timeline is the Perspective Wall [41], where the timeline is presented as a 3D wall. The user is able to scroll to a specific sector, receives a detailed view he/she is interested in and also has an overview about the timeline in the past and the future.

Calendars are an old instrument to note events of any type (birthday, meeting, astronomic events, etc.). It is used to note future, past, and reoccurring events, depending on the context. With the use of electronic media visual calendars have been designed. Many different designs for calendar visualizations [8, 21, 42, 47, 69, 71, 74] exist and provide functionalities on event handling, scaling, planning and group synchronization.

2.2.2 Techniques

Time can be visualized in many different ways. Depending on the scaling factor and the context, different visualization techniques provide different advantages and a good balance between overview and detail. Because of the huge amount of different visualization techniques on time-oriented data only a rough overview can be given in the following section. In the book of Aigner et al. [4] many different forms and techniques are listed, which have been invented and designed. Such a list is also accessible in the internet, which is called the ”TimeViz Browser“ [65]. This list is not complete due to the situation that many new visualization techniques have been presented and more will follow. They also provide a categorization for techniques, which is defined as follows:

- **Data**
  - *Frame of Reference* - abstract vs. spatial

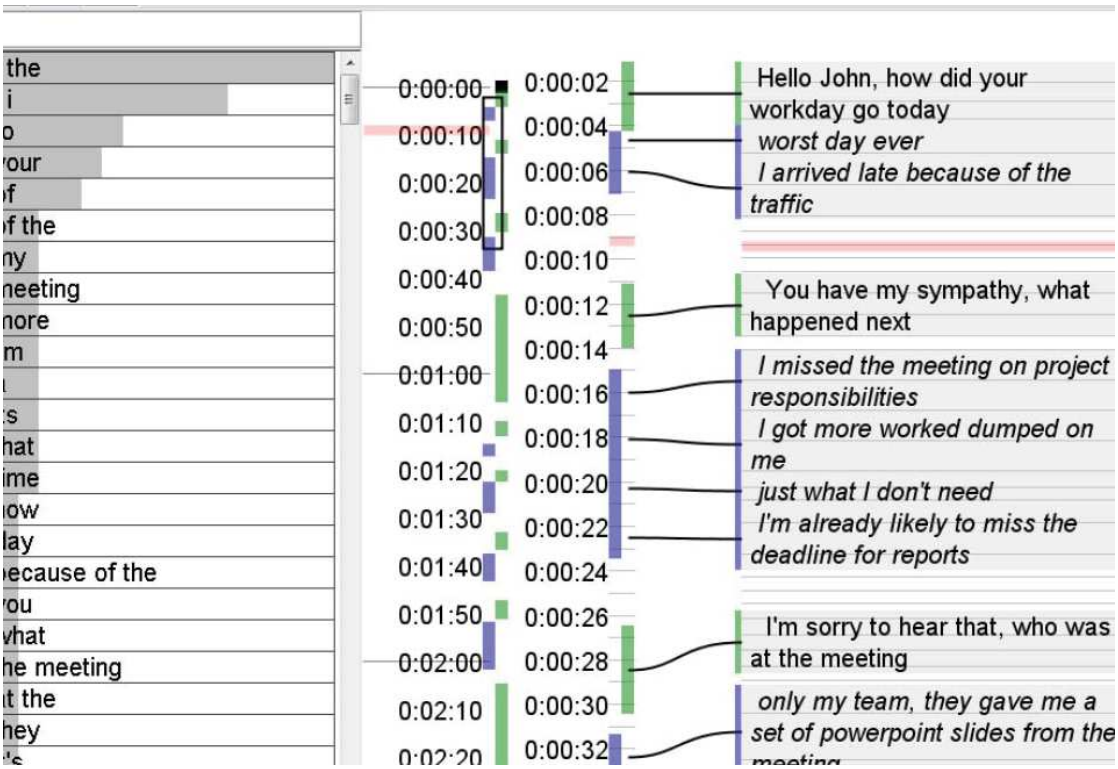

Figure 2.2: The Dialogue Explorer prototype interface with a vertical timeline [20]

  - *Variables* - univariate vs. multivariate

- **Time**
  - *Arrangement* - linear vs. cyclic
  - *Time Primitives* - instant vs interval

- **Visualization**
  - *Mapping* - static vs. dynamic
  - *Dimensionality* - 2D vs. 3D

In the following section the examples of different visualization techniques are all referenced from Aigner et al. [54]:

The *Frame of Reference* defines, if a variable considers a reference to specific area. Spatial oriented visualization techniques mostly use geographic maps, e.g. Icons on Maps, Value Flow Map. Techniques based on abstract data are not using maps, e.g. Line and Cycle Plots.

The number of displayed *Variables* differ between the techniques. While some techniques only provide the possibility to display one variable (e.g. Point Plot, Line Plot), other ones make

it possible to show multiple variables with several data values at the same time (e.g. Spiral Graph, Spiral Display).

The *Arrangement* distinguishes between linear time lines and a time cycles. Linear time lines represent some kind of time beam with a continuing time line, e.g. Timeline, Gantt Chart. Time Cycles show reoccurring time intervals, e.g. SpiraClock.

Time Primitives are divided into time-points (e.g. Cycle Plot) defining a single point on a time axis and time-intervals (e.g. DateLens).

In a static *Mapping* (e.g. Time Tree) time is mapped to space, where the presentation is not changing over time. In contrast a dynamic mapping maps time to time, where visualizations can change over time (e.g. TimeRider).

The *Dimensionality* defines the amount of dimensions, which are used for visualizations. Examples for two dimensional visualization techniques are Point and Line Plots. Three dimensional visualization techniques are the Worms Plots or the 3D ThemeRiver. In 4D visualization techniques the time is often presented as the fourth dimension and there does not exist a timeline or something else. Such visualizations are working with dynamic mappings, as mentioned before. In the paper of Zollmann et al. [75] an approach of 4D visualization on time-oriented data is presented.

Visualization techniques are also used by analytic software solutions, like Falcon [2]. Falcon uses different visualization techniques to analyze and explore large time-oriented data sets, finding errors and problems in the logs of 3D printers.

As the Line Plot and the Cycle Plot are essential for this thesis I will describe these two visualization techniques in more detail:

2.2.3 Line & Cycle Plot

The line plot [26] is one of the most common forms to visualize time-series data. Line plots are similar to point plots and connect the single points with lines. A line plot is an ideal way for presenting trends. Many different forms of line plots have been created and provide different extensions and subtypes, which provide additional information and advantages for specific tasks.

A subtype of a line plot is the cycle plot [17], which is able to illustrate seasonal trends. Single trends get visualized as small line plots, which are presenting e.g., the values on each Monday. So you can see the value changes and the trend for this single day. Additionally you can see the whole trend e.g., of the month. Figure 2.3 gives an example of a cycle plot, where you can see the trends for each day of the week. You can also identify a weekly pattern with a peak on Wednesday.

The line and cycle plot are essential for this master thesis as they are used for the prototype implementation.

2.2.4 Effects and Dynamic

Visual effects and dynamic provide additional visualization possibilities compared to static techniques.

Simone Kriglstein et al. [37] give an overview of animated visualization techniques of time-oriented data, where the time is illustrated in form of animations. They identified that both, static

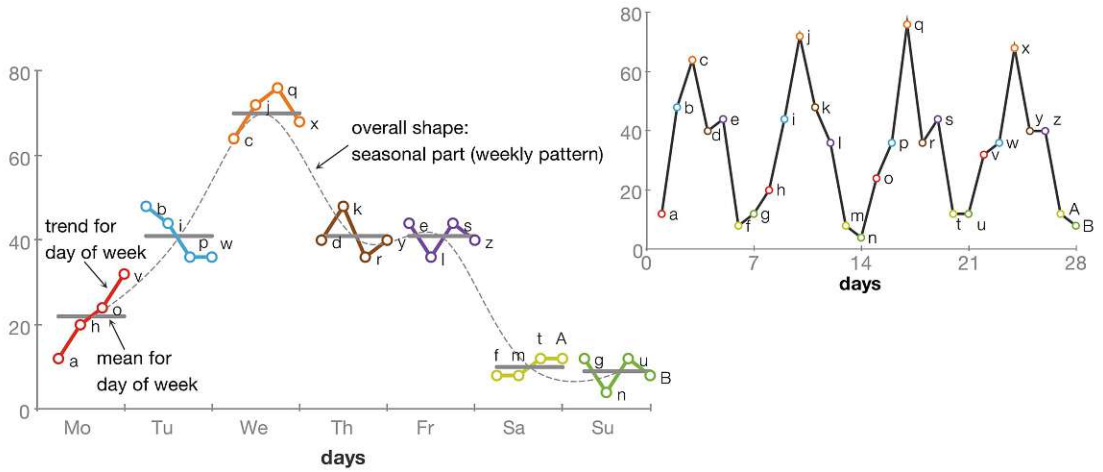

Figure 2.3: Cycle Plot showing a seasonal trend over weekdays, sample illustration taken from Aigner et.al [54]

and animated visualization techniques, have their advantages for different use cases. While animations for small data sets are more powerful, static visualizations techniques have their benefit on big data sets. The possibility to edit the speed of animations and the availability of circular animations are always beneficial.

Sometimes it is of interest to draw user's attention to specific elements of visualization techniques. Such forms of highlighting can be visualized in form of coloring or sizing. Deadeye [36] achieves this by showing each eye the same image, except the element to highlight. The element to highlight is only rendered for one eye . This technique has the advantage that existing images and graphs stay nearly untouched and reduce the probability of misinterpretations.

2.3 Frameworks and Libraries for Visualizing Time-Oriented Data

The combination of visualization techniques and time-oriented data requires software-frameworks, which provide a technical base for efficient implementations. They help developers to focus on the main task. The base for new features begins at the operating system, the developer is implementing on. Furthermore, he/she needs a programming language, a graphical library, a model framework, sample data and other components for the development. To list all essential prerequisites is dependent on the context and the innovation, which get developed.

For a data generator, which generates time-oriented data with visual methods, several prerequisites are essential:

- Model framework to store time-oriented data and calculate with them.
- Graphic library to use visual techniques.

Time-oriented data is usually combined in a tuple, where time is combined with the data of context. Such tuples can be easily displayed in simple tables. Such data can also be stored in relational databases. A uniform data framework for time-oriented data provides TimeBench [54]. This software framework is able to deal with the different properties and characteristics of time (e.g. granularities, instances vs. intervals). It is the only software-framework, which is specialist in modeling time-oriented data in an efficient way and is able to provide a base for a software solution dealing with time-oriented data.

The visualization of data requires more than knowledge about data visualization. Visualization tools and frameworks require advanced programming and graphic programming skills, combined with detailed knowledge in mathematics and statistic. Solving a visualization problem with a technical solution is a task for more than only one expert. Advanced software tools may close the technical gap. Prefuse [27] is a software library written in Java, which provides an open and extendable framework for information visualization. While most software solutions are specific on its context, Prefuse is independent and can be modified and customized. Similar to Prefuse is the InfoVis toolkit of Fekete [22]. These software solutions are available for Java Swing applications and provide multiple visualization solutions.

The JavaScript InfoVis Toolkit by Belmonte [9] is a framework, which is written in JavaScript[2] and available for the web. This solution provides similar usable visualization graphs to the InfoVis toolkit of Fekete [22]. This toolkit can also be integrated into several software solutions of different libraries and programming languages, if a browser widget [3] is available.

Another JavaScript based web library is D3 (Data-Driven Documents) [14]. This library is able to bind different kind of data to a Document Object Model which is then transformed into several forms of textual or graphical representations. The high amount of more than 100 references and projects based on this library illustrates the cardinality and universality of this software product. Especially for this master thesis implementations and examples dealing with time-oriented data are available (e.g. [7], [6], [52]).

Based on D3 a declarative language in form of a visualization grammar was established as Vega [57]. Vega uses JSON[4] to describe graphical visualizations and appearance on a low-level grammar, using simple graphical elements (e.g. axes, legends. etc.). A further stage of expansion represents Vega-Light [56], which uses high-level grammar dealing with interactions. Another framework build on Vega and Vega-Light is Altair [33], a library written on Python[5] for declarative statistical visualizations.

Another graphical approach to visualization with JavaScript provides Protovis [13]. This software has been finalized in June 2011 and is no longer under active development.

In the web you can find several commercial and non-commercial visualization tools as finished applications or web-applications (e.g. Visual.ly [1], Tableau Public [62], Datawrapper [25], Many Eyes [30] (discontinued by IBM 2015)).

[2] Scripting language used in web browsers and applications for interactive interactions
[3] Widget in GUI-libraries to interpret HTML, CSS and JavaScript
[4] JavaScript Object Notation, a commonly used data format for browser-server communications
[5] Programming language, `https://www.python.org/`, Accessed: 2019-09-07

2.4 Generation of Data

Data generators are often required for software tests and demonstrations, if no real data is available due to its secrecy. Sometimes real data does not provide specific data structures, which are of interest. Data generators are used in any fields of software development and are essential for good test coverage. Often such data generators are specified on the used context. E.g. in the article of Renata Georgia Raidou et al. [53] synthetic data sets are used to demonstrate a prototype with a new technique improving the display of parallel coordinate plots.

Data generators exist in several different forms. You can use rich clients, which are directly installed on a personal computer, or generate data from web applications, which are available in the World Wide Web (e.g. generatedata.com [35]). Often the input of the user is done via a simple command shell, like in the programming language R [63]. More comfortable generators provide a user interface, where you can directly specify types and ranges.

2.4.1 Test Data Sets

The generation of synthetic data is often done for testing functionalities and techniques. So test data sets should provide specific data constellations, where patterns and other phenomena in data are identifiable. Bergeron et al. [10] give an overview of test data sets for the evaluation of data visualization techniques. Scientific data can be characterized by their data types (nominal, ordinal, metric). For test data generation three aspects are important which should be recognized for the generation process:

1. *Constraints*: The amount of data values and dimensions of a data set.
2. *Parameters*: The distribution and the correlation between the data values and the dimensions.
3. *Cluster regions*: The position of cluster regions and their properties.

As the generation of reliable test data sets requires statistical properties, like data distribution and correlation, the book ”Einführung in die Stochastik“ [72] provides the statistical base. In this book you can find the most common data distributions (e.g. binominal, Poisson, Gaussian), their statistical properties and formulas.

2.5 Data Generators

There exist three different data generators, which use visual methods for the input definition. The first one is from Albuquerque et al. [5]; a data generator for high-dimensional data sets, which represents the generated data in different graphical plots. The user is able to see the generated output directly on a chart, which is advantageous for identifying a usable data set. Additionally the user can use a painting tool to directly create cluster patterns, as shown in Figure 2.4.

The most modern data generators display the configuration and the outcome of the generation process in one view, i.e. so you can immediately see what you have generated out of your configuration settings. Bremm et al. present such a tool for the generation of multivariate data:

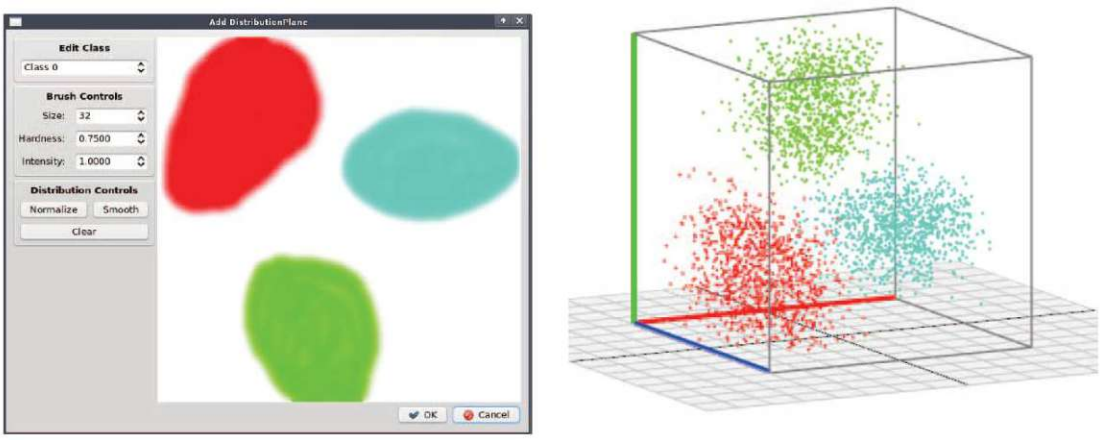

Figure 2.4: Creating a cluster pattern with the painting tool [5]

PCDC [15]. You can see an example of their prototype in Figure 2.5, where the generated output is visualized with a multi line plot. This generator with visual aspects provides a colorful user interface, where each variable is displayed with its own color. The course of each variable is graphically presented and can be edited directly on the plot. Direct editing results in changes of the output. Furthermore, you can define the data distribution individually for each value range. Next to predefined distributions you can also customize user drawn distributions.

Another visual data generator has been designed by Wang et al. SketchPad^{N-D} [73], which is able to generate synthetic data with the use of visual methods. The user defines the input parameters on simple charts and the output gets immediately visualized. Especially the custom drawing features on a drawing board provide direct user interaction without the need of many standard or custom widgets. Sketchpad^{N-D} provides two visualization paradigms:

- Parallel coordinates
- Sketching on Scatterplots

Parallel coordinates are visualized with a multi line bar, similar to PCDC [15]. The user is able to define the number of dimensions, the number of generated samples per cluster, the correlation per cluster, coloring and other input. Additionally the user can directly draw areas where the output should be generated, as shown in Figure 2.6.

SketchPad^{N-D} also supports direct manipulation on scatterplots. Figure 2.7 shows the user interface, which gets used to generate data on a scatterplot. The user initially draws a 2D distribution into a shape and receives a simple scatterplot, which then can be directly edited. It is possible to clean and delete parts of the generated output by rubbing it out, like a rubber on a piece of paper. The produced scatterplot can then be displayed in different views; rotating on the different axis or change the view to a multi line plot.

All three referenced data generator with visual methods represent excellent design and products of software development. They introduce visualization aspects into a tool, which generates data and does not only display datasets.

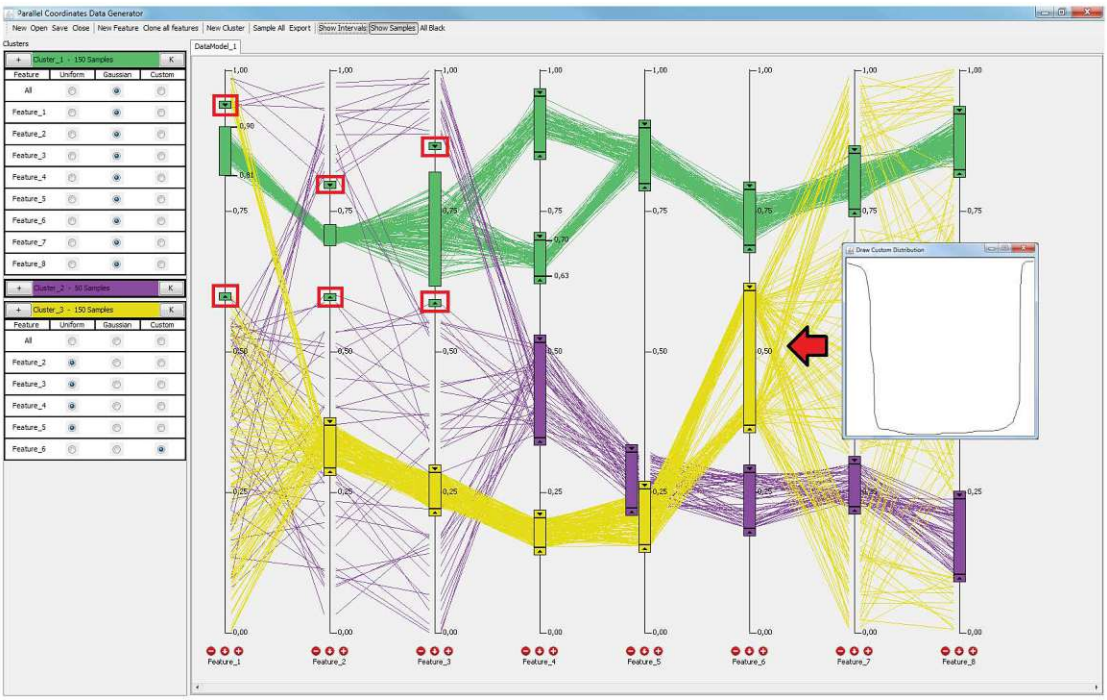

Figure 2.5: Sample view of PCDC with multiple variables [15]

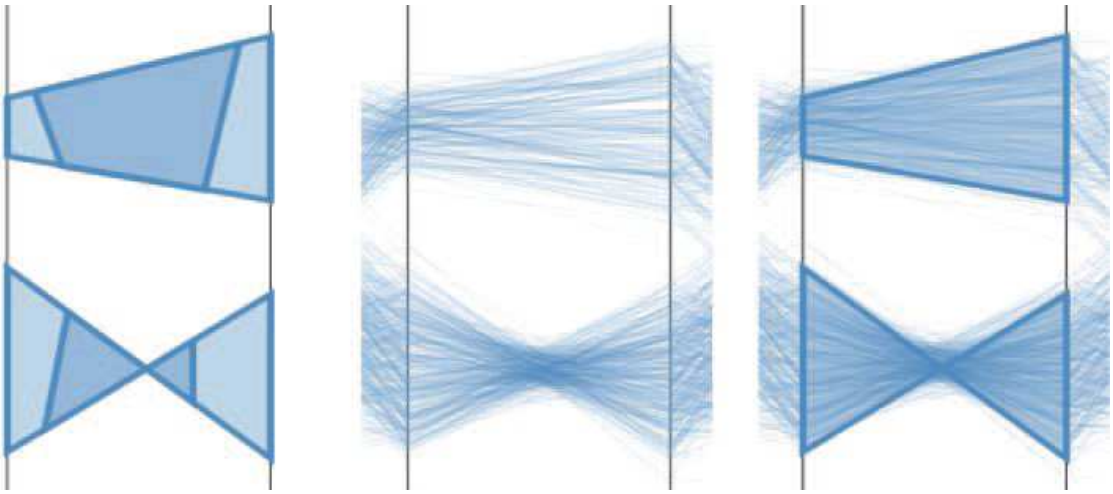

Figure 2.6: Drawing principle of SketchPad^{N-D} [73]

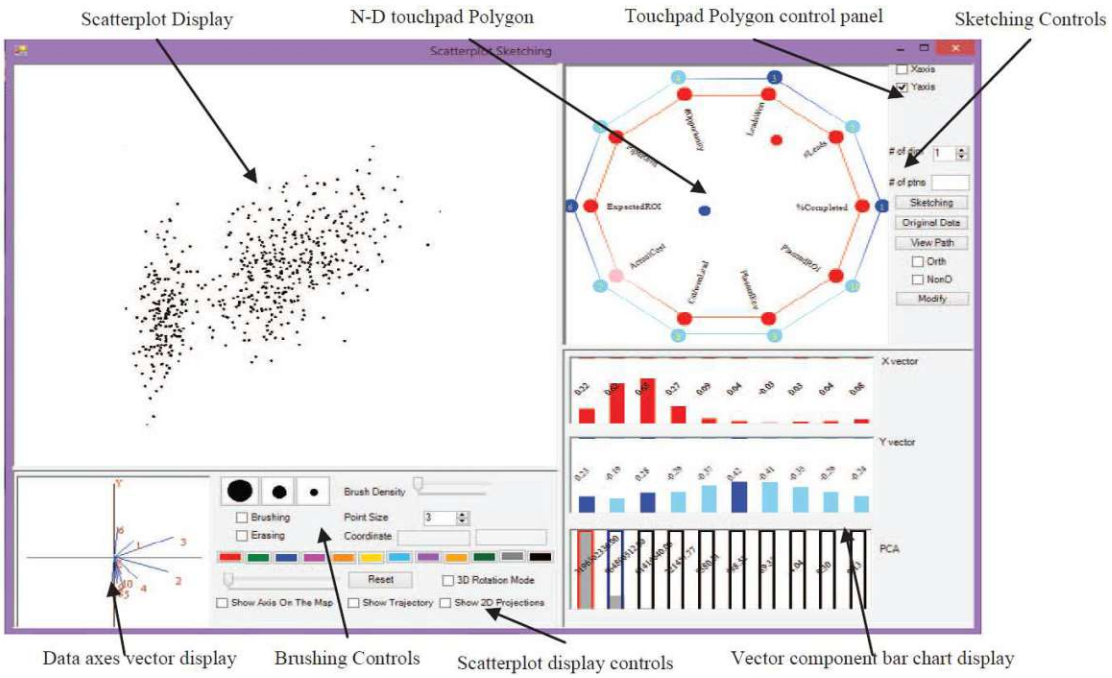

Figure 2.7: Scatterplot sketching interface of SketchPad^{N-D} [73]

2.6 User Interface (UI) Design

For publishing and describing a design study of a software solution, the topic of ”UI Design“ is an essential part. Innovative software solutions, which should solve specific problems or support specific tasks, often provide a Graphical User Interface (GUI), where you are able to interact with keyboard, mouse, a touch screen or combinations of them. The acceptance by the user has high priority; otherwise the software solution does not get used. Even if the GUI has urgent weaknesses the software itself needs not to be useless or a maldevelopment. But it is possible that due to pitfalls in the UI design the success of software is greatly reduced.

2.6.1 Guidelines in Usability

Johnson wrote a book [34], which is a simple guide to understanding UI design rules. It contains and describes several design principles with simple figurative examples, good and bad ones. He also mentions that it is not easy to match all described principles in detail. The designer has to make many tradeoffs, but must find a good balance. Furthermore, he compares design guidelines to laws, as described in the following quote:

> Just as a set of laws is best applied and interpreted by lawyers and judges who are well versed in the laws, a set of user-interface design guidelines is best applied and interpreted by people who understand the basis for the guidelines and have learned from experience in applying them. (Quote from [34], Introduction xii)

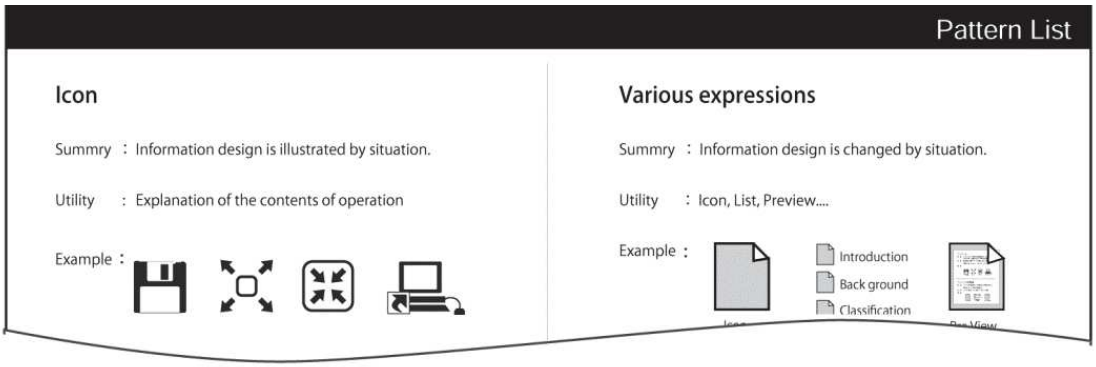

Figure 2.8: Icon as GUI design pattern [28]

As described by Johnson, sometimes it is not possible to find a perfect design, which matches all the design principles he described. To make tradeoffs seem to be unsatisfactory in specific cases. Obeidat and Salim [51] describe UI guidelines with adaptation techniques, which should solve usability problems. Each user is different and interprets the usability of a UI design differently. This difference of interpretation has its origin from the characteristics of each user, e.g. sex, education, profession, skills, culture etc.

Obeidat and Salim [51] solve such design problems with adaptive systems, which adapt the system behavior depending to the user. Therefore, they collected UI guidelines of Nielsen [46] and Shneiderman & Plaisant [60].

2.6.2 GUI Design Patterns

UI Guidelines try to describe Do's and Don'ts in UI Design. More concrete definitions are situated in GUI Design Patterns, which are composed of summary, utility and examples (e.g. Figure 2.8). Hirata and Yamaoka [28] collect 81 patterns and categorize them into seven groups with a cluster analysis. Furthermore, they divide these patterns into four layers. The World Wide Web also provides libraries of UI Design Patterns (e.g. UI-Patterns.com [68], Welie.com [70], Quince [31]).

2.6.3 User-Centered Design (UCD)

User-Centered Design is a design process where the end-user is integrated and can influence the software product before and during the implementation. The user is involved in this process to improve the usability and to achieve a high acceptance on the user side.

Abras et al. [3] summarized history and information and gives a good overview of UCD. The origin of this design process comes from Donald Norman [49]. In his book "The Design of Everyday Things" [48] he describes four basic suggestions of a good design:

- Make it easy to determine what actions are possible at any moment.

- Make things visible, including the conceptual model of the system, the alternative actions, and the results of actions.
- Make it easy to evaluate the current state of the system.
- Follow natural mappings between intentions and the required actions; between actions and the resulting effect; and between the information that is visible and the interpretation of the system state. (p. 188) [48]

These suggestions are as important as they have been in 1988, where the user is the midpoint of the software design. The user should be able to learn and use such a design with a minimum of effort.

To involve the user into the design is not an easy job and requires high financial and human resources, as described by Abras et al. [3]. So it is not possible that each software solution gets developed with the pattern of UCD.

The user is not the only one, who is touched with a software solution. People, who are working or communicating with the user, are also involved. Tasks and workflows are dependent to the used software. As you can see involving all affected parties is a difficult task and requires high effort in time, financial resources and organization. It is necessary to strike a balance between UCD and profitability.

2.6.4 User Input via Sketching

Many user interfaces only deal with simple user input by keyboard and mouse, e.g. typing characters or clicking or dragging buttons and items. A less common user input technique is free-hand drawing of figures which are interpreted by the software. AxiSketcher [38] is a prototype which is able to define axes on scatterplots by free-hand drawn lines and curves. Bahador et al. [55] studied 12 different graphical input encodings where users are forced to interact directly on graphical elements instead of manipulating graphs via sliders. Potential future user input interactions may be found in the virtual reality where users are able to interact in a 3D area. In [19] a first formal user study was done where graphs and networks are visualized on virtual reality platforms. There user where not able to directly interact with the 3D models but started to gesticulate on the 3D visualizations.

Free-hand drawings, gesticulations and other forms of input extend the well known input possibilities and can also be used for data generation.

2.7 Evaluation

Evaluation techniques are an essential part of this master thesis. The research question asks for a data generator on time-oriented data, which should be intuitive for experts and non-experts in time-oriented data. Evaluating the prototype and its usability is done in form of user studies. There exist different techniques, how a user study can be established. Sunny Consolvo et al. [18] describes the following techniques:

2.7.1 User Study Techniques

2.7.1.1 Contextual Field Research

This technique gathers qualitative data. The participants are observed during their common activities. There does not exist a lab-environment. The research is done in the normal user's environment. This has the advantage that the participants of the user study are not influenced by an unfamiliar environment. Additionally they are not acting with samples; they are doing their real activities.

2.7.1.2 Intensive Interviewing

Doing intensive interviews on the people, who participate in the user study, is a very efficient technique for a user study. The participants answer on specific questions and out of the given answers a set of qualitative data should be emerged. This technique has the advantage that it need not to be done in the user's environment and no specific technical equipment is necessary (expect a camera or a sound recorder). Furthermore, the interviews could be individual for each participant. On the other side doing interviews is a time consuming task and requires much time by the participants and by the interviewer. Additionally it is not easy to stay neutral, because the participants should not be influenced in any case.

2.7.1.3 Usability Testing

Usability tests are a common technique for a user study and provide empirical data very easily. This technique has the advantage of low costs of time and effort and provides much data in a very short time. A usability test can be done in a lab-environment, where the users can be observed easily. Disadvantageous of this technique is the artificial examples, which may lead the participants to alter the normal behavior. Furthermore, good usability tests are no guarantee that a design or a prototype will be successful in practice.

2.7.1.4 Lag Sequential Analysis

The Lag Sequential Analysis is a technique, where the user tests are done during their normal activities. In contrast to the Contextual Field Research they are working with the software and hardware, which should be tested. This kind of user study provides very good quantitative data about the usage and the effectiveness. The disadvantage of this technique is the very expensive methodology. The environment has to be prepared for the user study and cannot be done in a lab-environment. And the evaluation of the gathered data requires much time. As there are no predefined tasks or questions, the result of data is a surprise and the interpretation may be difficult.

2.7.1.5 Expert Inspection

Reviews by domain experts provide qualitative information about the usability. Tory, M. and Moller, T. [66] identified that reviews by expert are a good possibility to review early prototypes. Experts can identify critical problems very early. This has the advantage that major problems

can be identified and fixed with little effort and very fast. But it is also mentioned that expert reviews should not be used exclusively. Expert reviews can only complement user studies but not replace them.

2.7.1.6 Usage Scenarios

Another evaluation method is defined by usage scenarios. This technique is based on the researchers themselves. They describe "how they used a new visualization approach to solve/improve upon a certain problem without a (strong) involvement of domain experts". (Quote from "A Systematic Review on the Practice of Evaluating Visualization", Evaluation Scenarios, p. 2822) [32].

2.7.2 Validation on a nested model by Tamara Munzner

Tamara Munzner [44] provides a specific nested model for visualization design and validation. Her model is divided into four layers, as presented in Figure 2.9. The upper levels delivers the input for the lower levels. When mistakes in upper levels occur they also affect lower levels.

- The top layer focuses the domain itself. It is necessary that the target audience is able to understand design and planning, e.g. the common vocabulary of a domain must be used, as expected by the audience.
- The second layer covers the data and operation abstraction design. The layer should produce descriptions of predefined operations and data types. This also covers the transformation of raw data into predefined data types and forms.
- The third layer concentrates on encoding and interaction designs. The interaction between the design and the user is focused. The design is not usable if the end users are not able to interact with it or do not understand its behavior.
- The fourth and last layer concentrates on technical problems, like efficiency of algorithms or memory consumption. Algorithms may be implemented in a bad way, so at the end the design does not work as expected because of technical issues. This layer is not of major interest for this master thesis. By the way efficiency and technical problems should always be avoided (even in the protoype) but are not the major criteria for this thesis.

2.8 Reflection

The current state of the art provides a huge list of available visualization techniques. Aigner et al. [4] provide a good overview by summarizing and categorizing them. Especially, the line and cycle plot are of interest for this master thesis, used within the implemented prototype. Different software libraries dealing and handling time-oriented data are available. TimeBench [54] is the chosen one, used for modeling time-oriented data in the prototype. A small set of visual data generators are available but no one focuses on handling time-oriented data in a professional

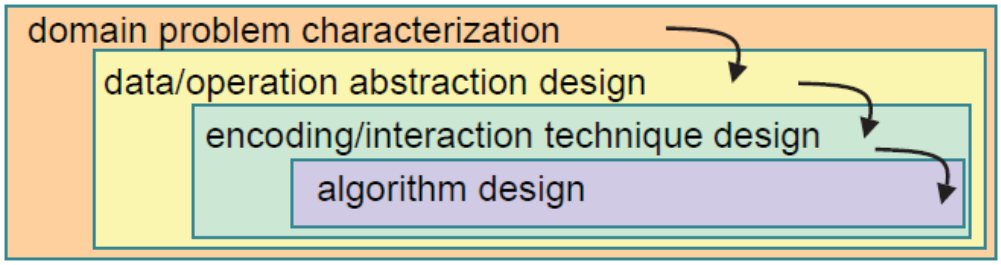

Figure 2.9: Model of visualization creation with four layers [44]

manner. Therefore, the prototype should close this gap. Evaluation techniques are used to answer the defined research questions and should help to analyze the implementation results for this master thesis. Intensive interviewing, expert inspection and usage scenarios are in focus of the evaluation phase.

The next chapter describes the methodology; the implementation of a prototype and it's evaluation.

CHAPTER 3

Methodology

The methodology of this master thesis splits up into 5 parts: The first part consists of a requirement analysis, a description of requirements and expectations of a software design. The second part consists of the software design itself, generation visual time-oriented data by using visual methods. The third part is the implementation of the software design, producing an executable prototype. The fourth part deals with the evaluation of the implemented prototype. The evaluation is composed of usage scenarios, expert inspections, and usability tests. In the last part the results of the evaluation are elaborated. The evaluation and its results are presented in Chapter 5.

3.1 Requirement Analysis

A new software design is initialized by a requirement analysis. The analysis describe the necessity of a software and functional and non functional requirements. Commercial requirement analysis expect a detailed specification book, describing all relevant detail of design and implementation. This master thesis concentrates on a new design approach. Therefore, the requirement analysis concentrates on an innovative software design approach, less on technical details and subtleties.

3.2 Software Design

Generating time-oriented data by using visual methods requires a well structured and predefined software design. Therefore, it is necessary to predefine a generic design of a data generator consisting of several technical and functional artifacts. Technical decisions are essential for the setup and build of the prototype but only have low importance for the evaluation phase. Functional decisions are more interesting as they define the used visualization methods, the generation process, parameter definitions, etc. At the end of the initial phase the software design is ready to be implemented in form of a prototype.

3.3 Implementation of a Prototype

To receive an answer to the research questions a prototype was implemented, based on the software design of the initial phase of this master thesis. This prototype demonstrates an innovative design, where visual techniques, the time and the generation of synthetic data are combined into one software solution. The user input for the generation process is done by directly editing visualization methods, in case of this master thesis line and cycle plots. Line and cycle plots are common visualization techniques for time-oriented data and the easiest ones for user input. Therefore, these techniques should show the feasibility of the design. The generation of synthetic data is established by generating CSV[1] files, which contains the generated data.

3.4 Evaluation

Within this master thesis a set of different evaluation techniques is used, which are applied on the implemented prototype.

Intensive interviewing will be part of the user study on the implemented prototype. The people should give feedback to the design and should show strengths and weaknesses after testing and interacting with the implemented prototype. The participants may provide additional or new design aspects, which have not been recognized during the design time. Such an input by participants provides additional possibilities for improvement and redesign.

Usability tests are required and part of the user study. In [29] Holzinger divides usability enineering methods into inspection and test methods. Inspection methods are used during the design and implementation phase of a software. Test methods are done by the end users. The evaluation of this master thesis uses “Aloud Thinking”, where the participants are enforced to think loudly while using the software, and “Questionaires”, interviewing the participants afterwards. Predefined tasks have to be solved by each participant. A larger set of ten or more participants exceeds the operating expenses for this master thesis. So only a small set of three people will participate, one expert on time-oriented data and two non-experts with advanced IT knowledge.

As this master thesis is supervised by Alexander Rind, he is the expert for reviewing the implemented prototype and will participate in the expert inspection.

Usage scenarios are another part of the evaluation phase. They are a possibility to get evaluation results in a quick manner. The researcher is represented by the author itself.

3.4.1 User Study of the Prototype

The user study of the implemented prototype consists of a usability test and interviews. For the usability test the participants have to solve artificial tasks. Speed and correctness of the tasks are not the core targets of this evaluation, because these aims may produce stress, which may influence their mind. The participants are encouraged to “Think loudly”. It is of interest to identify their thoughts and ideas, which occur during the user study. Their statements are recorded for the evaluation and provide additional information for the interviews, which are done

[1] File format for saving comma-separated values as plain text.

after the user study. Interviews are the second part of the user study and should give information, which parts of the prototypical design are intuitive and which parts require redesign or does not provide good usability. After all they have the possibility to mention their ideas; what is missing and which parts they would have designed in another way.

The participants consist of one expert in time-oriented data and two people with advanced skills in the IT. The expert additionally gives an expert review, which should point out the major problems for the domain of time-oriented data. Usability tests of users without technical skills are not done, as it is expected that they will have individual or technical problems and do not show real usability problems. They would represent outliers. The handling with outliers is difficult, but single user problems should not be reflected as actual usability problems, as described by Følstad et al. [24].

3.4.1.1 Tasks for Usability Test

CSV files are expected as output during the usability tests, which contain a set of time-oriented data, generated with the implemented prototype. These files represent the results produced by using the prototype. Out of these files it is possible to evaluate, if the participants were able to produce specific constellations of sample data. The predefined tasks describe which properties the output should have. It is expected that the exported CSV files are similar to real world scenarios, as artificial data should not be distinguishable to real world data.

The usability tests should cover all important designs and functionalities, which are provided by the prototype. So the tasks also contain specific hints and subtasks, so that specific features get used in specific scenarios. But these hints and subtasks must not be too detailed to avoid easy cooking recipes. If the tasks are too detailed, the participants need not to think and experiment. An adequate balance between specification and latitude is required to receive useful results.

Task 1: Simple dataset for one year. "Generate a weekly dataset for 2013. You want to provide sample dataset for the level of bookings for a place on an Austrian lake. In the summer months the peak times have 50,000 tourists. The level of booking is rising in spring and falling in autumn. In winter the level is lowest with 500 tourists, except in December with 5,000."

This task contains a simple example for a dataset over the whole year. The difficulties for the users lie in setting the weekly interval and defining the time and variable scale. Five markers for the definition of the dataset would be the ideal definition. The participants have to define the time scale, the value scale and the value interval.

Task 2: Simple dataset for one year with drawing feature "Repeat the first task and use the drawing feature to define the markers. "

This task should hint the user to the drawing functionality, where the markers can be defined by drawing a line of the average trend. The user should be able to compare two different ways of using different features for the same task. It is expected that using the drawing functionality is much faster and more intuitive than defining the markers manually. The quality of the result is expected to be equal to the first task.

Task 3: Seasonal trend on days of week "Generate a daily dataset for 2013. You again generate datasets for the same place in Austria. Now it is of interest to have a seasonal trend on the days of week. From Tuesday to Thursday is the lowest number of tourists. On Monday and Friday the amount of tourists is 10 % higher than from Tuesday to Thursday, because many people enjoy an extended weekend. On Saturday and on Sunday the amount of tourists is 30 % higher than during the week. In the summer months the peak times have 50,000 tourists. In the winter months the level is lowest with 500 tourists, except in December with 5.000."

This task requires the usage of the cycle plot, where the participants are able to define seasonal trends. Because of the situation that the users already should have used the drawing feature, it is expected that they use it for this task again. The difficulty for the user is situated in the combination of the seasonal and the general trend.

Task 4: Level of booking in combination with the temperature "Repeat the second task. Additionally introduce the general trend of the temperature as a second variable.".

This task introduces the feature of multiple variables. Now the participants have to handle more variables, which should illustrate the handling with multiple variables. As the configuration possibilities are active for the currently selected variable, the user has to handle the switch between different variables. As multiple variables make the scenario more complex, it is expected that the participants require more time to solve this task.

3.4.1.2 Interviews

The second part of the user study consists of interviews of the participants, after they did the usability tests. The interviews are individual to the people, but contain similar questions, which should be answered from each participant.

The first question asks for the difference between the first and the second task. It is of interest, which technique is more efficient or more intuitive. Which technique do they prefer? It is expected that the user prefer the direct drawing feature, because it should require less interactions of the users for the same task. Each interviewed person should also give information about the usability: which problems occurred and what can be improved.

The second question asks for the representation of the cycle plot. As the cycle plot is a specific visualization technique for time-oriented data, it is expected that the experts on time-oriented data do not have problems in understanding this feature. In contrast to the group of experts, I expect that non-experts will need time to understand the cycle plot and the combination with the normal plot. The experience with the cycle plot is of high interest, as this form of visualization has never been used for the generation of data. The interesting sub question asks if this feature provides essential benefits or if it is more confusing.

The third question asks for the handling with multiple variables. As each additional variable makes a task more complex, the user should give information about his/her experience. This question asks mainly for problems when handling with more than one variable. It is expected that the users will have different understandings, how the handling of multiple features should work.

The last question asks for the general experience with the prototype. It is expected that nearly all participants will have ideas for additional required features. Much more interesting

are general problems, which occur during the usability tests. The users should give hints, which aspects of the design are good and intuitive, but furthermore, which parts of the design require changes. The input of these questions is material for future work.

3.4.1.3 Usage Scenarios

After all the designer and author describe specific usage scenarios, how and where such a software design can be used. These scenarios should point out the potential benefits of the design and how it improves the development of visualization techniques for time-oriented data. Such scenarios should define the advantageous in combination of using conventional sources of sample data. Such sources are real world scenarios, synthetic data from other generators or manually produced examples.

3.5 Summary

The core topics of this master thesis consists of an initial software design where end users are able to generate synthetic time-oriented data by using visual methods and the evaluation phase, evaluating the design based on an implemented prototype. After the evaluation phase the predefined research questions are answered based on the final results.

The next chapter describes the requirement analysis and how the analysis is coupled to the predefined research questions.

CHAPTER 4

Requirement Analysis for visual data generation

Software design traditionally begins with a requirement analysis and specification. [40] mentions that the initial definition of system requirements is an essential part, most time the hardest part of the whole project. A requirement analysis describes functional and non functional requirements, e.g.

- Functionalities
- Participating domains
- Infrastructure
- Techniques and system environments
- Scalability
- Fail-safety
- Performance

This listed items represent only a small subset what requirement analysis and especially a specification book must contain. On the initial analysis of requirements several stackholders of management, technicians and end users participate on this process of definition. Therefore, the definition of a specification book often requires a long time and often consumes more time than expected. The invested time on a detailed specification book should focus to a common understanding what a software should do and how it should work.

This master thesis describes an innovative software design approach on a data generator using visual methods for the generation process. The following stackholders are involved within the software design, it's evaluation and its documentation within this thesis:

- Supervisors: Alexander Rind and Silvia Miksch
- Developer and designer of the software approach, author of this master thesis: Stefan Weiser
- Participants of the user study: Lukas Mad and Alexander Jakovljevic

As this master thesis do not focus on an industrial finished software solution this requirement analysis is much shorter and more informal. Financial and time oriented budget are not of interest as no sponsors or customers do exist. The main focus on this software approach is to answer the predefined research questions. The following sections describe the established requirements and expectations to the software design.

4.1 Generation of time-oriented data

The supervisor Alexander Rind mentions that no adequate approaches for generation of time-oriented data exist on the market. Many different data generators are available for many different types of data and combinations of it. Therefore, an approach is required which provides the possibility to generator data focusing on the specific characteristics of time. The necessity of such a data generator is to test and implement new visualization techniques dealing with time-oriented data. Adequate and meaningful tests and showcases require larger sets of data and even specific characteristics and data constellations.

It is expected that a design and its prototypical implementation is able to fill this gap.

4.2 Visual approach for parameter definitions

Instead of simply plotting data in form of a text based format or visualizing it on some kind of diagram it is expected to use a visual approach for parameter definitions. Alexander Rind is asking if direct interaction on visualization techniques is more intuitive than using simple controls for parameter definitions. Drawing and wiping techniques conquered the world of mobile devices (smartphones, tables, etc.) and can also be used for the data generation process.

It is expected to draw the trend of time-oriented data directly on diagrams. Then, the generator interprets and generates data in any kind of table-oriented form and also visualizes the results directly on the diagrams.

4.3 Usage of TimeBench library

TimeBench [54] is an open source project dealing with algorithms on time-oriented data and its visualization. This project was initialized by the supervisors and could directly be used for an initial prototypical implementation. It provides the necessary data structures and visual approaches but is not mandatory for the software design itself.

This project is available on GitHub[1].

[1] `https://github.com/ieg-vienna/TimeBench` accessed 2019-11-02

4.4 Usability

The supervisor Alexander Rind is interested into the question if the visual approach for parameter definitions is intuitive. Therefore, the usability of the software design should be tested and evaluated. It is expected that a visual approach is more user-friendly than traditional designs of data generators, using classic controls for input definitions. The second research question should be answered after the evaluation, based on user study of the implemented prototype.

4.5 Summary

Out of the requirement analysis the predefined research questions have been established. As this thesis is focusing on research, identifying and evaluating a new software approach, the list of functional and non functional requirements is not very long. Even when non functional requirements are not established explicitly it is expected that the software design and the implemented prototype do not show urgent issues on stability, scalability and performance. Otherwise the evaluation phase (especially the user study) may be negatively affected because of technical issues.

The next topic describes the software design in detail.

CHAPTER 5

Design of a Data Generator using Visual Methods

This design combines three elements, which are used. The generation of synthetic data represents the first and most essential part, as it delivers the output. The second part defines that the output contains of time-oriented data. The third element of this design deals with visual methods. The design is presented in form of an implemented prototype, as it illustrates design decisions more clearly.

The prototype is developed in Java[1] and uses Swing[2] for the implementation of the user interface. This decision is based on the technical implementation of TimeBench [54], which is also implemented in Java and uses Swing components. As TimeBench gets used for the prototype, Java and Swing provides a good technical base for an ideal compatibility. TimeBench delivers the visual methods for the presentation of time-oriented data. All visual output of the prototype is based on technical implementations of TimeBench.

As this design also deals with good usability, it was planned to provide a design with a minimum amount of interactions by the user. Too many steps for configuration and other stuff should be avoided. The user should be able to achieve fast results with an interesting and useful data set.

5.1 Basic Structure

The basic structure of the design contains of a menu, a toolbar, configuration parts and the visualization part. The most important part is the visualization part, where charts and the graphical output are represented. The other elements, which are not that important (but necessary), should require as little space as possible to provide more space for the visualization. Figure 5.1 shows the partition of the different elements and how they are arranged.

[1] `http://www.java.com` accessed 2019-11-04
[2] `http://www.java-tutorial.org/swing.html` accessed 2019-11-04

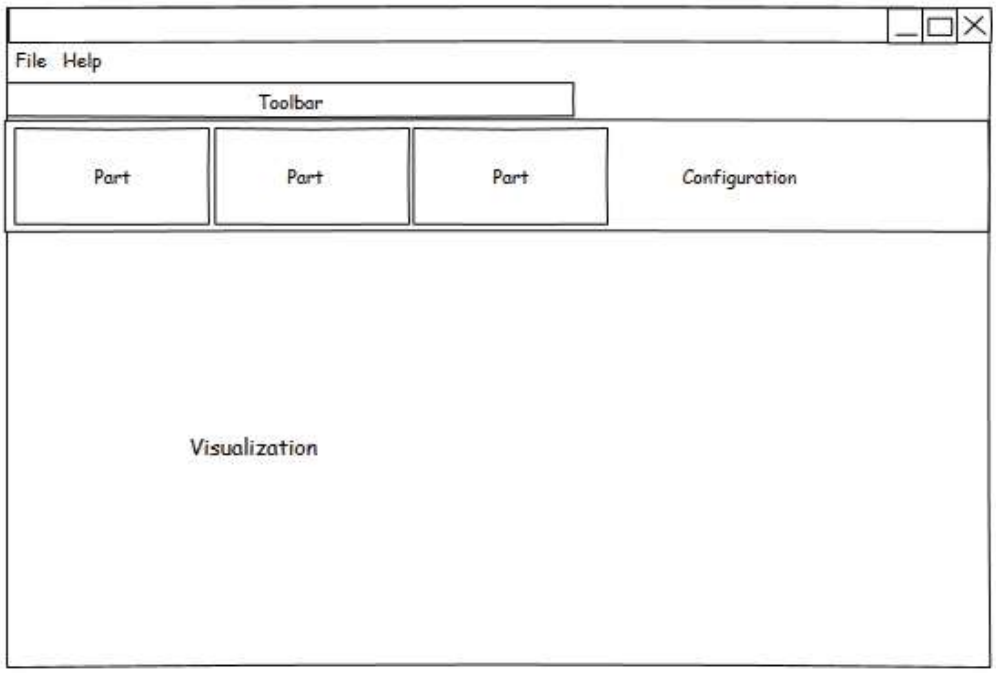

Figure 5.1: Basic Structure: on top a menu and toolbar, above the configuration parts followed by the visualization

As many possibilities for the configuration will be necessary and helpful, the configuration parts may become too many. A system of fading in and out is required, so that different configuration parts can be made visible and invisible. This provides the advantage that the user can reduce its display to the necessary elements and fade out these one, which are not required in the situation. This feature provides a possibility of user specific configuration, as described by Obeidat and Sali [51]. Other software solutions provide a similar system, e.g. Eclipse[3]. This platform divides its user interface into views and perspectives, which are individualized to specific tasks. Furthermore, the user herself or himself is able to configure the organization of views on his/her own, which is saved when closing and reopening the application.

In the prototype the visibility of the configuration parts is configurable in the menu or the toolbar of the application. For each part a toggle button and menu is added, which controls one specific part. If the number of parts does become too many in the future, menu and toolbar buttons become impractical and should be replaced by a more dynamic system (e.g., search function for fading in). The prototype provides a small set of configuration parts:

- The *time scale* defines the horizontal area as timeline, which is visible on the plot. Zooming in and out takes places in the specified time scale. If values occur outside of this area, these values do not disappear, as they are simply not in the visible area.

[3] `http://www.eclipse.org` accessed 2019-11-04

- The *value scale* defines the vertical area and defines an axis for decimal values. The functionality is equal to them of the time scale.
- The *variables and their properties* can be configured in a single part. Each variable has a unique name and color. Variables can be added, removed and set to visible or invisible. The current shown variable is the active one. The drawing functionality and creating value ranges refers always to the active variable.
- The *distribution* is definable in one part. The prototype provides the uniform and the standard normal distribution, which are used for the generation of random values.
- The part of the *value properties* defines the distance between the values. The prototype provides intervals for hours, days, weeks and years.
- The cycle plot configuration provides the possibility to switch between the view of a simple line plot and a cycle plot. The cycle interval defines the period of a cycle. The prototype provides the following intervals: weekdays, weeks of the month and quarters of a year.
- The *value sector configuration* provides the possibility to configure value sectors. When selecting a value sector the current upper and lower limit are displayed and can be edited. This provides the possibility to define value ranges in detail. When using the drawing functionality the number of created value sectors can be defined. For each drawn line the specified amount of sectors is created.

5.2 Interaction with Visual Techniques

This design is focused on the usage of visual techniques for producing synthetic data. This design uses single line plots and cycle plots for defining data sets and representing the output on the same chart. The value ranges must be definable by the user, so that the desired output can be produced. On the other side the output itself should be presented in form of a simple line or cycle plot. As it can be seen there must be a separation between in- and output elements.

5.3 Value Ranges

Value ranges define the area where the values of sample data are defined. The user is able to define as much value ranges has need. But it is not necessary to define a range for each value point. This would not provide any benefit, as the user cannot define ranges for 200 values or more. The design requires at least two value ranges, one for the start and one for the end. The points between this value ranges are calculated by interpolation, as it can be seen in Figure 5.2. The trend of the values runs against all defined value ranges, till the last one is reached.

A value range can be moved and re-sized by Drag and Drop[4]. The value sector configuration provides an additional possibility to re-size a selected sector, when exact upper and lower limits are required.

[4]Method to move elements on a user interface with a mouse

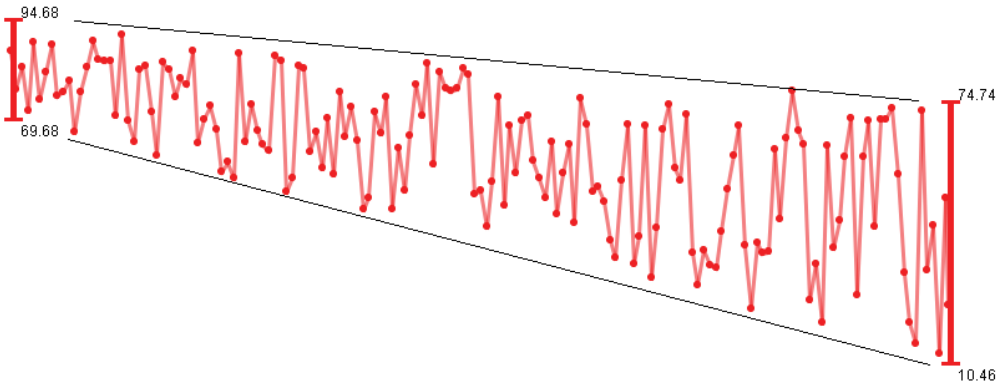

Figure 5.2: Value Interpolation

5.3.1 Drawing of Value Ranges

When a bigger set of value ranges is required for a more complex scenario, defining them with Drag and Drop is awkward and requires too much time. Such an example can be a sine wave, which could not be configured with only two value ranges. In such a case the value ranges can be drawn on the line plot.

Drawing a set of value ranges provides the possibility to create an unlimited set of value ranges with two mouse clicks. The drawing functionality can be enabled by a button in the tool or menu bar. Then the user has the possibility to draw a line on the plot, like in a graphical program (e.g. Microsoft Paint). The drawn line defines the trend. Figure 5.3 illustrates this process. The first step shows the beginning of drawing a specific trend. By continuing drawing the trend gets defined where the values will be created. The second step shows the finished drawn line, how the trend should look like. The third and last step shows the automatic created value ranges, which occur immediately after the finishing drawing. The position and order of the value ranges represent the trend. Afterwards they can be edited manually, if the user wants to alter them.

5.4 Variable Definition

For more complex sets of time-oriented data, more than one variable can be handled. The variables can be handled in the variable part, where all settings for variables can be done and are displayed. A variable has to be unique, which is achieved with a unique name, defined by the user. The name of the variable is also essential for the output, when the data set is exported to a CSV-file. For differentiation of the variables on the plot each variable also requires a unique color. A unique color is required to clearly distinguish between variables on the plot.

Variables can be added and removed. At least one variable is required, so it is not possible to remove the last existing variable. To handle multiple variables on one plot, it must be clear, which one is currently active and can be edited on the plot. Adding new value ranges or using the drawing functionality can only be done on the currently active variable. Other variables

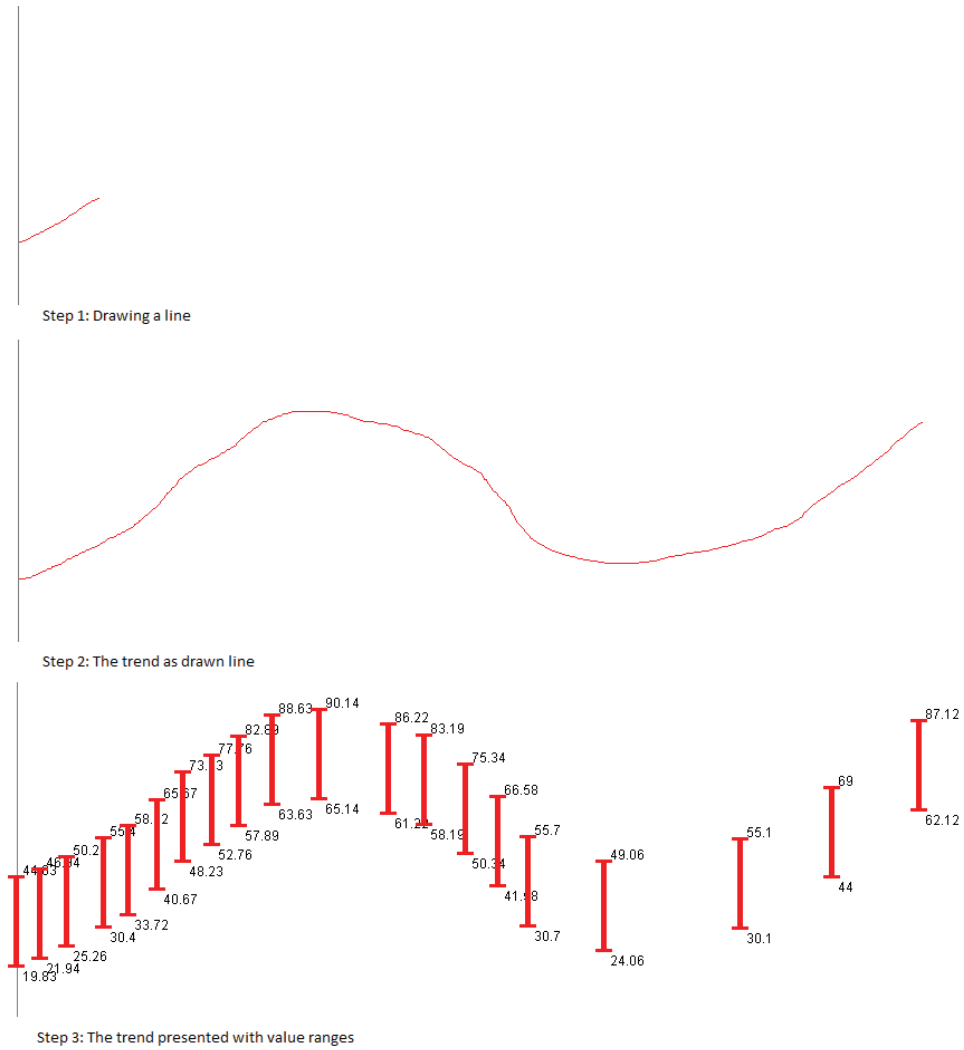

Figure 5.3: Drawing of Value Ranges

are only editable, if the are the current active one. This is a quite simple strategy, but has to be recognized when working with this software design. Switching between all variables is quite easy and fast, but it is recommended working on one variable and nearly finish the configuration, before continuing with another one.

A huge set of value ranges from multiple variables can lead to an overloaded plot with too many value ranges and too many colors. Figure 5.4 illustrates the problem with four different variables. For each variable all value ranges can be faded out. This provides the possibility to do fine adjustments on one specific variable, without getting disturbed by an overloaded plot. The prototype provides the possibility to fade out all value ranges of all variables, which can be useful to visualize only the generated values without the associated value ranges.

There may be some situations, where variables with greatly different scale factors exist, e.g. the first variable is situated between -10 and +10, the second variable has values between -1000 and +1000. As variables should be generated on one and the same plot, the visualization together becomes a bit tricky. The normal representation is not sufficient, independent to the configuration of the scaling factor of the plot. One variable is always out of sight. One possibility is that the user has to make sure that the variables are representable on one plot, as solved in the prototype. Another possibility is to provide a technical solution. The statistical normalization is a solution to transform values of any scale factor to a common scale, so that all variables can be visualized together.

The variable definition provides many possibilities for more detailed variable configuration, e.g. show and hide the output, the legend of the values or the average trend in form of a line. The prototype provides a small set of configuration possibilities, but illustrates the main concept of handling multiple variables.

5.4.1 Linear time line

Generating values on a linear time line is the basic concept of this software design. The linear time line of this design has no starting point or end point. It only shows a small section of the whole time. As the time is an unlimited variable, the line plot must be able to scale in any direction of the time, forward and backward.

The same behavior has the axis of the values. The line plot is able to scale in both directions without any restrictions.

5.4.2 Time cycle

As specific units of the time are repeating itself (e.g. a day, a week, a whole year) events and trends occur in a similar form each cycle again. Such seasonal events have a regular form, which is repeating again and again. Additionally to the seasonal trend there also exists a general trend, which shows changes over several cycles.

The visualization of such time cycles requires a specific technique for visualization. As the visualization should also be used for the generation process of time-oriented data, the cycle plot represents an easy technique, which is a specific form of a normal line plot. Depending to the unit of time the cycle has a given amount of time points, e.g. the months of a year or the days of a week.

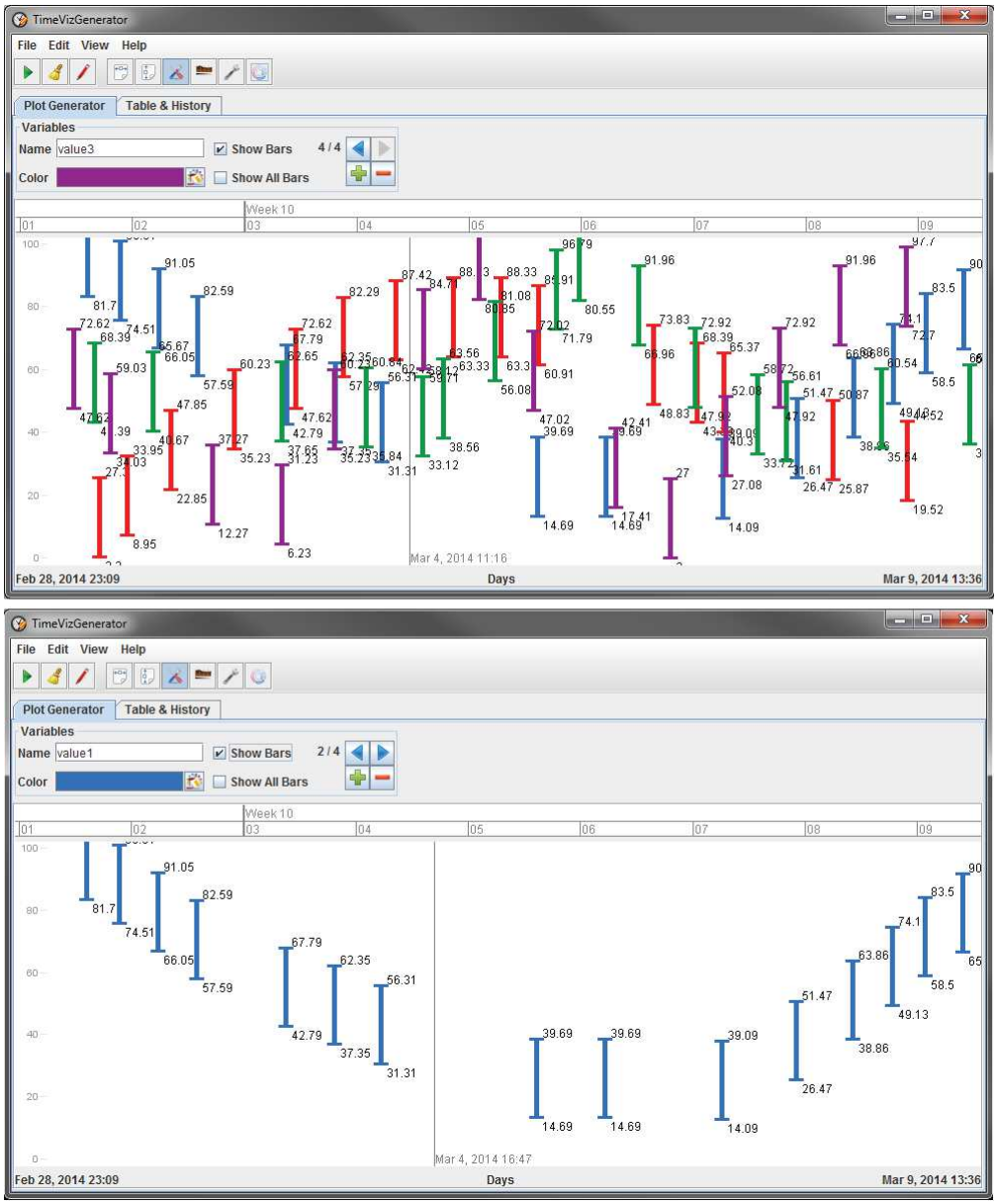

Figure 5.4: Handling with Multiple Variables

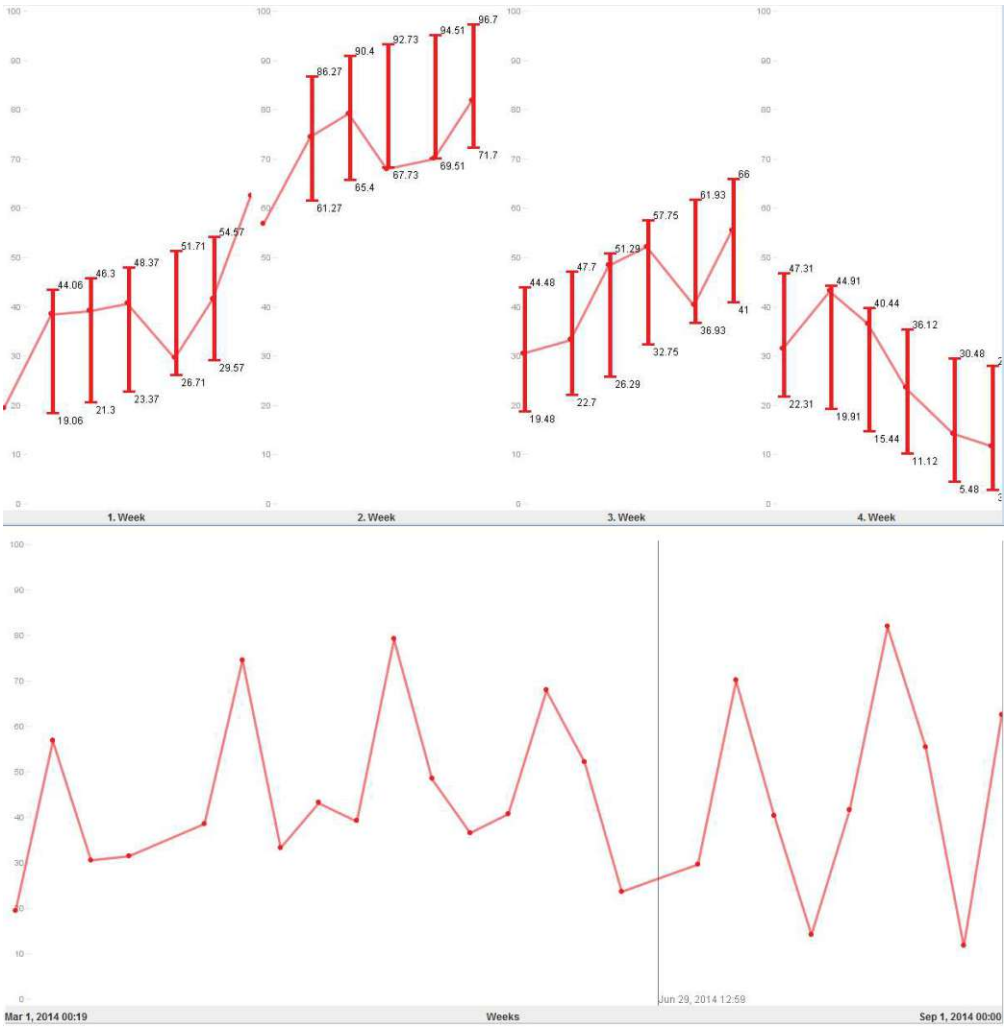

Figure 5.5: The same dataset as cycle plot (top) and line plot (bottom)

For the software design of the data generator the cycle plot provides similar functionality to the simple line plot. In contrast the cycle plot has predefined time points for the value ranges, because for each time cycle one value range is required for each unit. An example could be a seasonal trend over the weeks for a whole year, where the amount of Mondays of the years defines the amount of value ranges. As existing in the simple line plot it is also possible to use the drawing functionality to define the trend of one unit.

An interesting feature is the switch between the cycle plot and the simple line plot. When generating data for a seasonal trend, the results are also visualized on the line plot. Figure 5.5 shows the same dataset, on top in form of a cycle plot on the weeks of a month, on the bottom as line plot. As it can be seen the different techniques of visualization produce different pictures and are focusing on different aspects of the data.

5.5 Generation process

The generation of time-oriented data requires a defined set of value ranges. Without the definition of any value ranges, the definition is unclear as the value ranges define time and possible values.

For the simple line plot the first and the last value range define the starting point and the end point of time for the generation of a data set. So the simplest set consists of at least two value ranges. This rule is applied to all variables of a data set, so each variable requires at least two value ranges.

When using a cycle plot the starting and the end point of time are defined by the defined time scale. For each interval all existing time points between the starting and the end point represent a value range, even if they are not predefined. A simple example represents a cycle interval on weekdays. If we have a starting time of 2014-01-01 and an end time of 2014-02-28, we have to identify all Mondays, which occur in this period. These are the 6^{th}, the 13^{th}, the 20^{th} and the 27^{th} of January and the 3^{rd}, the 10^{th}, the 17^{th} and the 24^{th} of February. This is also done for all other days of the week. If the user does not define a value range of point of time, the vertical defined scale gets used as value range. So the cycle plot needs no predefined value ranges, as they are computed from the time scale and (if not defined by the user) from the value scale.

Value ranges define the range of possible values for a specific point of time. It contains a lower limit with the minimum value and an upper limit with the maximum value. The generator uses a random number generator to create a random value between the defined limits. Furthermore, it is also possible to define a distribution. The distribution provides the possibility to modify the probability of values within the value range. When using a uniform distribution, the probability of all possible values is the same. When using the normal distribution e.g., within the values 0 and 100, the probability of 50 is much higher than 10 or 90. So the generated values are dependent from the defined value range and the defined distribution.

As it is not comfortable to define each time point manually, values should be generated in a defined interval. The amount of generated values in a given time range can be defined by specifying the interval of generated values, e.g. in the time range of one week with a value interval of an hour we produce 168 values, one value for each hour of the week. The prototype provides the possibility to generate values every hour, day, week or year. There are many more possibilities to define intervals. The implemented prototype only deals with a regular interval, but it may be possible that also an irregular interval is of interest.

The generation itself should be very fast. Depending to the amount of values, which have to be calculated, the process could be represented in form of a process bar. A process bar gives direct feedback of the generation and the required time. Furthermore, the generation should not block or freeze the whole program, when calculating a large set of values. Values should be generated and added to the output continuously, without hindering the user.

5.6 Representation of generated data

After the generation process the generated data set is visualized directly on the plot, where it has been defined by the user. So the plot displays the input (value ranges) and the output at the same

time. It may be of interest to visualize only the output, so the value ranges can be hidden.

The raw data are also represented on a simple table and can be exported as CSV file. Depending to the use case different export formats are possible, as the generated data set should be used for any visualization techniques.

5.7 Summary

The generator design describes the basic structure and functionalities how users will be able to generate time-oriented data with visual methods. Configuration parts are used to set paramteres like scale factor, time lines usw. The visualization parts represents the most interesting part and presents data graphically and also provides the possibility to edit plots with input for the generation process.

The evaluation of the design requires an executable prototype which will be implemented in form of a standalone Java program. The prototype is ready for usage and is used in the evaluation phase, described in the next chapter.

CHAPTER 6

Design Evaluation

The evaluation of the software design consists of three different parts: The first part describe usage scenarios, for which the software design could be a useful and efficient solution. The second part contains an expert inspection, where an expert of time-oriented data gives detailed feedback to design details. The last part is the user study, where two participants solve predefined tasks. These people are no experts in time-oriented data, but have adequate knowledge in information technology and software development.

6.1 Usage Scenarios

The generation of time-oriented synthetic data provides many potential usage scenarios, where sample data of a generator can be used. Especially when developing new visualization methods and techniques, there is often the problem that no useful data sets are available for presentation and demonstration. Privacy issues are a common problem. The interesting data sets do exist, but they are not allowed to be used for research. And the second problem is that useful data sets may not exist, because specific scenarios or data constellations never happened or are unknown. All these cases are a potential field for using a data generator.

For the user study there are four prepared tasks, which should be solved by the participants. These tasks represent potential situations, where the generation of time-oriented data is useful. All tasks are fictional situations for a hotel next to a lake somewhere in Austria. In the winter we have an average temperature of approximately 0°C, while in the summer months 30°C or more are possible. Austrian lakes are popular holiday locations in the summer, because of the warm temperature and the possibility to swim. In the winter months lakes are not a frequented holiday destination. These rough framework conditions are given for the participants, as the results should meet realistic data sets. We are interested in data sets, which represent the booking situation and the amount of hotel guests over the year. In the summer months is high season, while in the winter they can only count few guests. Only in December the level of booking is a bit higher because of Christmas and New Year.

A description of usage scenarios for evaluation is available in chapter 2.7.1.6.

6.1.1 Preparation of the Scale Factor

For preparing the generation of the data sets the scale factor has to be configured. At peak times the level of booking counts 50,000 guests. So we have to define the visible range of values between 0 and 60,000. It is advisable to define the visible range larger than the upper and lower limits, as the survey is getting better and the whole data set can be visualized on the screen. For defining the range of the values the configuration part "Value Scale" should be used. The plot provides feedback in form of the vertical labels. There it can be seen that the upper value is set to 60,000.

When the vertical value range is defined, the horizontal scale has to be configured. The task asks for a data set of the year 2013. So the start time is set to the January 1, 2013 and the end time to December 31, 2013. These initial steps define the visible area of the plot. This part of configuration can be done anytime during work, as data does not get lost. Changing the scale only results in a change of the visible area.

The defined scale factors can be seen on the plot by looking at the labeling. The vertical value axis provides a legend on the left side of the plot. The horizontal time axis shows the current visible start and end point bottom to the plot, as it can be seen on Figure 6.1. The time axis does not show the predefined end point. The plot can be scrolled by using the mouse wheel to enlarge or reduce the zoom factor. When scrolling to the minimum zoom factor the predefined start and end date can be seen in the status line of the time axis.

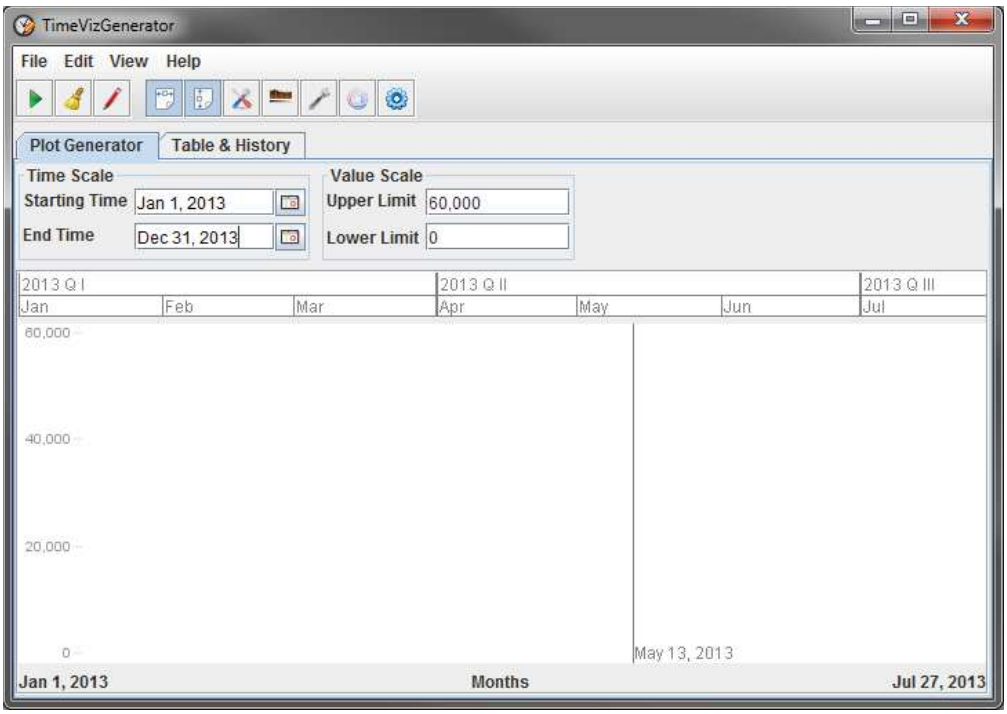

Figure 6.1: Prototype showing the line plot with defined scale factors

The configuration of the scale factor requires four input values, two for each axis. The scale

factor does not influence the output of the generated data set directly, but the user gets influenced in the definition of the general conditions for the generation, as he/she can only see the defined scale factor. Values outside of the displayed view are calculated, but are not perceived by the user. So the perspective should be chosen, so that the whole area of interest is visible.

6.1.2 Drawing the Expected Trend

The drawing functionality provides a quick and easy way to draw the desired trend. We expect that the curve starts at approximately 500 guests in January and continues rising. In the summer months an amount of 50,000 tourists is reached. The high season may start at June and ends in August. So the curve rises to 50,000 in June and stays on this level till August. Then the level of booking drops till October to 500 people again and stays on this level. In the middle of December the level of booking rises again to 5,000 and stays till the end of the year, because of Christmas and New Year.

When drawing the desired trend on the plot, the drawn line is displayed on the plot. Around this line value ranges are created. These value ranges indicate the area of potential values, which are generated randomly. The amount of such value ranges can be configured in the configuration part "Value Sector Configuration". There exists a spinner, where the amount of generated ranges can be defined, when using the drawing functionality. For the given example approximately ten ranges could be useful. It is also possible to use more, depending how detailed the curve of values should be. Drawing the trend has not to be done with one single line. It is possible to interrupt the line and continue at the same or at another position again.

After finishing drawing the result looks like in Figure 6.2. The curve is inaccurate and does not fit exactly to the value as required. Fine tuning is done in the next step.

6.1.3 Fine Tuning of the Value Ranges

The defined value ranges are all equal. The lower limits of some ranges have a negative value, what does not make sense, as the values represent the amount of customers in a hotel. The minimum value must not be lower than 0.

For fine tuning of the value ranges it is possible to resize the ranges with the mouse by drag and drop. Using the mouse for defining the lower and upper limit provides an easy way to achieve the desired output. For more accurate results the configuration part "Value Sector Configuration" provides the possibility to manually define the limits with native text fields. The text fields show the limits of the current selected value range. By altering the text fields the limits of the value ranges get changed, which is directly visualized on the plot.

It is not required to use all generated value ranges, created by drawing. They can be removed by selecting and remove them with the context menu, opened with the right mouse. In the following Figures 6.3 and 6.4 eight value ranges are used for generation. At the end it has to be defined, which interval should exist between the values. This can be done by using the configuration part "Value Properties". There exists a combo box, which offers some possibilities for an interval. The interval of an hour is not an adequate choice, as we would receive too many values, which would overload the line plot. The interval of a day is shown in Figure 6.3, compared to an interval of a week in Figure 6.4. Using an interval of a day also produces many

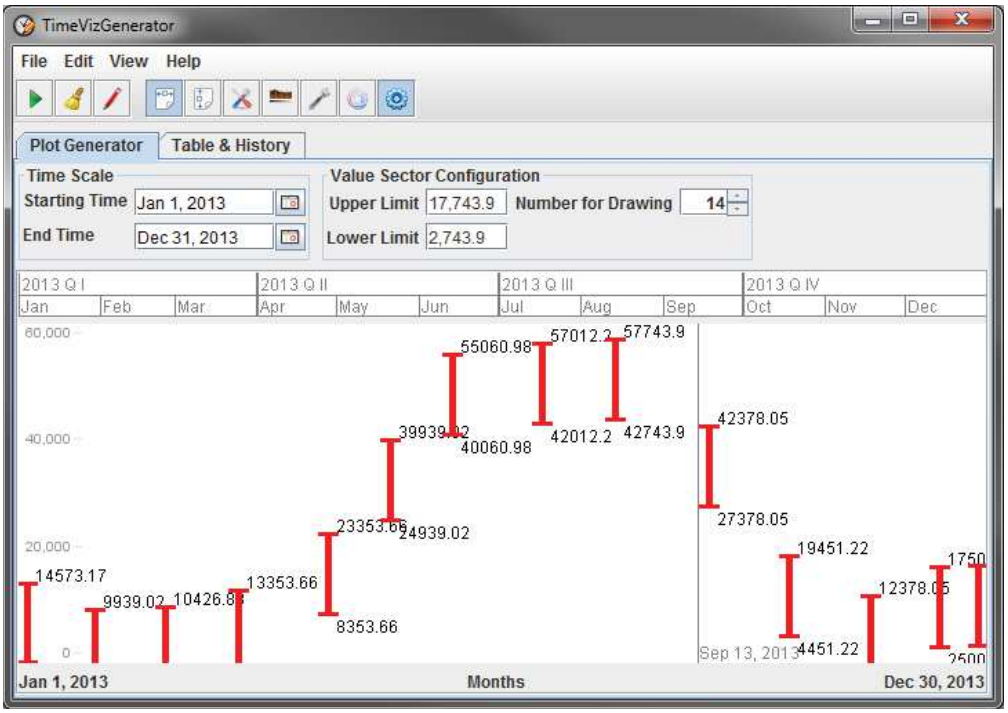

Figure 6.2: Prototype showing the generated value ranges after drawing the trend

values. A better choice is the usage of a week as an interval, as the visualization is much more clearly. The interval of a year makes no sense, as the generated values are only placed within one year.

If we want to have a more detailed view, we zoom to the desired section. Zooming can be done by using the mouse wheel. In Figure 6.5 we have a more detailed view of the same data set with a value interval of a day. When the amount of values gets too high, zooming to a specific section provides more clearly view.

6.1.4 Export of the Generated Data Set

Each generated data set gets stored, independent how often and how many data sets have been created. The tab “Table & History” has a list of all data sets. Each data set is represented by the time, when it was created. On the right side, there is an overview of the generated values in form of a table. Exporting a data set can be done by clicking on the button “Export CSV”.

6.1.5 Multiple Variables

When using multiple variables for one data set there could be the situation of different scale factors. Different scale factors may be a problem for the visualization as it is not possible to find a common view for all variables. This problem occurs in the fourth task, which the participants should solve. The fourth task is based on the first and second task and adds the temperature as

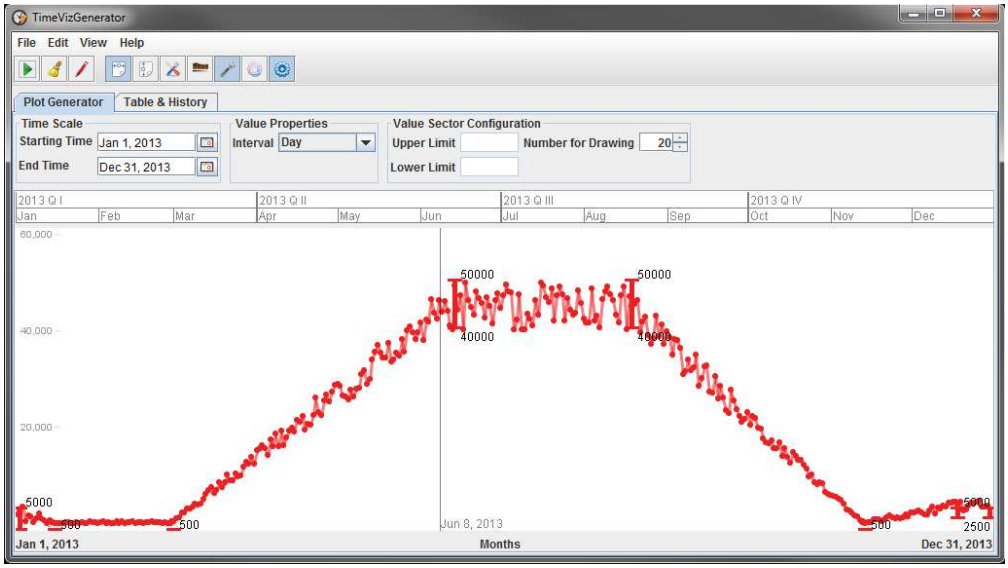

Figure 6.3: Generated values with an interval of a day

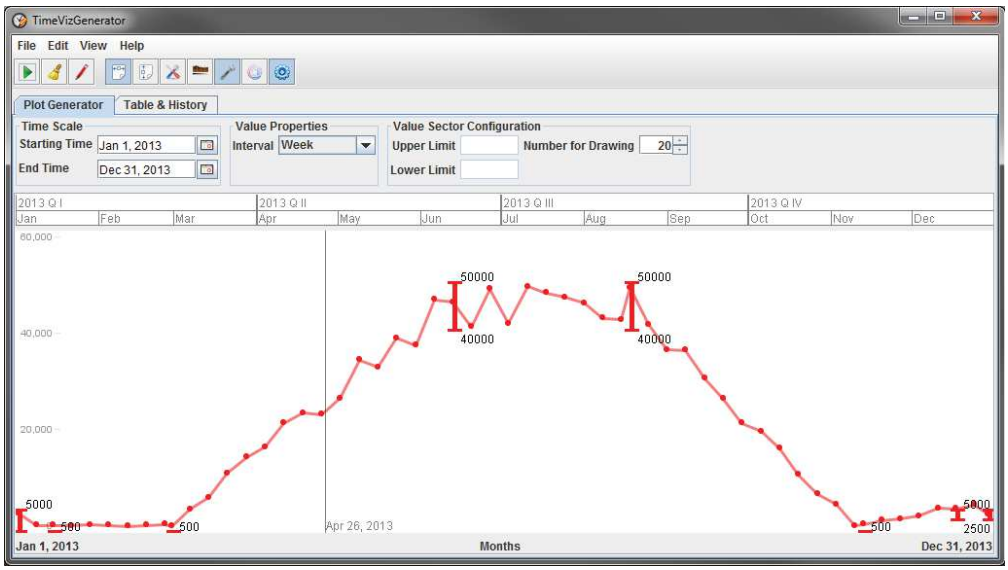

Figure 6.4: Generated values with an interval of a week

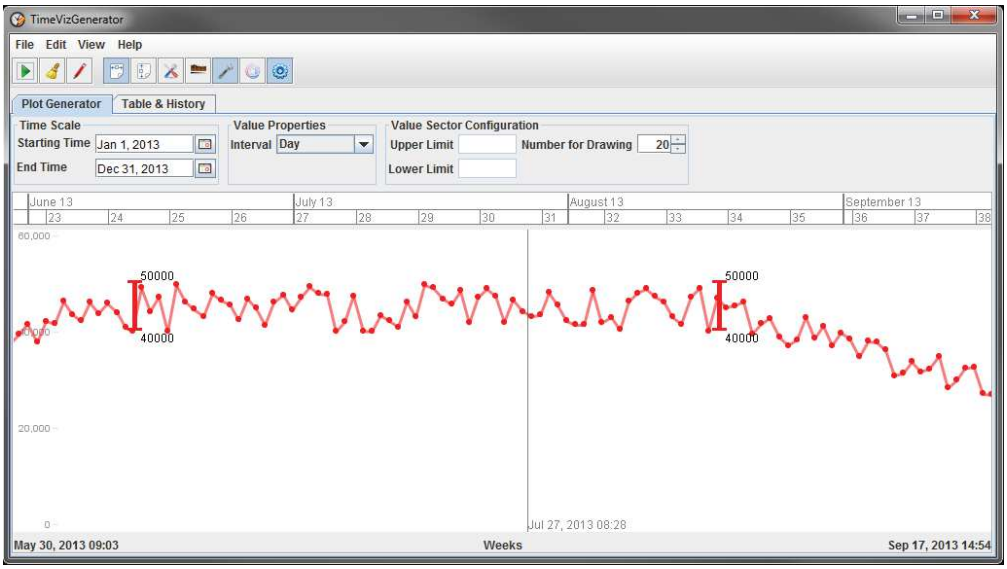

Figure 6.5: Detailed view by using the zoom

a second variable. The level of booking has a upper limit of 50,000 while the temperature in Austria normally does not exceed 40 degree. Furthermore, the temperature could fall below 0 degree, which is not possible for the level of booking. The level of booking could not be lower than 0.

The current prototype does not support a common scaling for different variables, so the user is responsible to find a solution. A possible solution is the normalization of the values. All values of both variables are between zero and one. The trend stays the same. To receive the real values at the end the user has to convert the values with a meaningful factor. This solution gets used for solving the fourth task.

The horizontal scale is the same as in the first and second task, so it starts on January 1, 2013 and ends on December 31, 2013. The vertical scale has a lower limit of zero and an upper limit of one as we use a normalized scale factor.

After defining the scale factors the trends are drawn. The first variable defines the booking level and has the same trend as in the first and second task. Because of the situation that we only have normalized values we have to estimate the values, but the trend stays the same. Then the second variable is added. The temperature has its peak times from June to August and the lowest values in January and February.

After defining all value sectors the generator displays the plot similar to Figure 6.6. This figure shows the mix of both variables. The booking level gets represented in red, while the temperature is represented in blue. After generation of the data set both variables are visualized on the plot, including the value sectors which belong to the variables, as presented in Figure 6.7. Figure 6.7 shows the problem when using multiple variables. The value sectors disturb the overview of the generated values, so it is recommended to hide all value sectors, as shown in Figure 6.8. Now we have an overview of both trends, the booking level and the temperature.

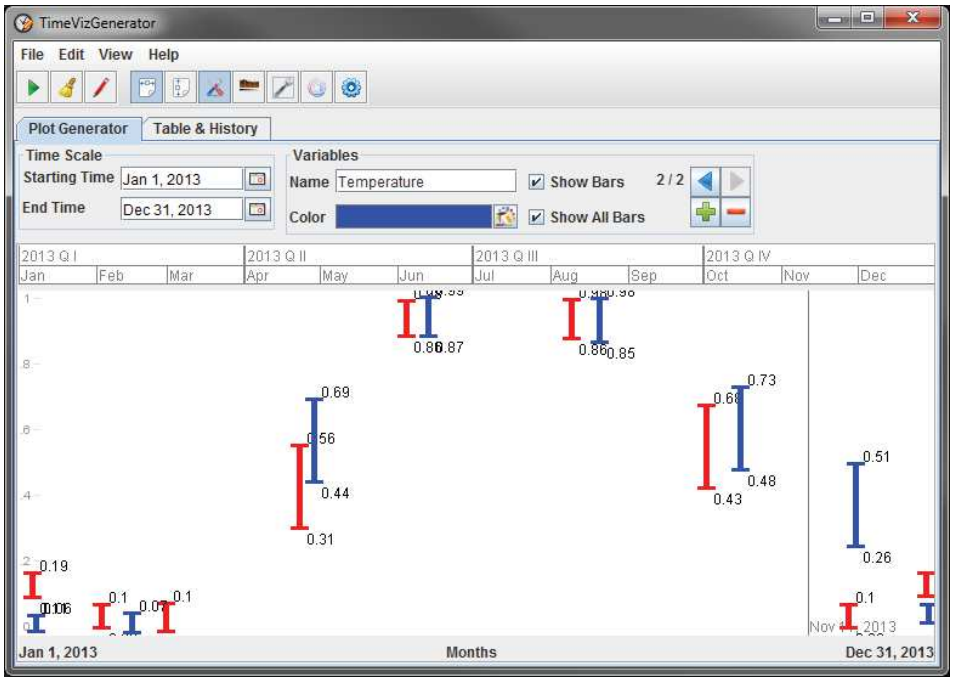

Figure 6.6: Defined value sectors with the booking level in red and the temperature in blue

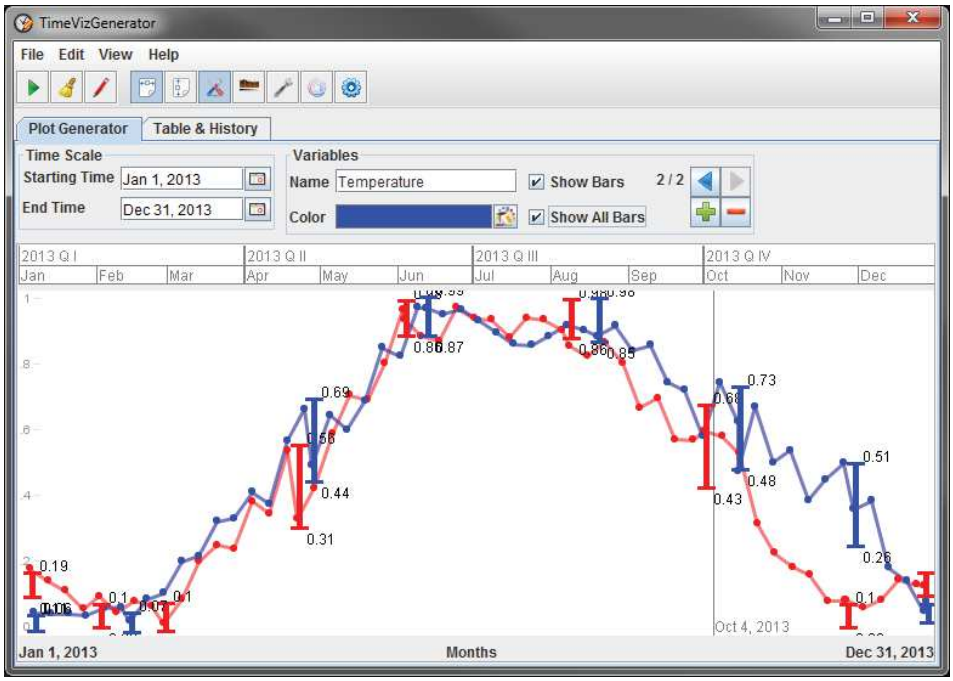

Figure 6.7: Generated values and the value sectors on one picture, which flood the line plot with colors

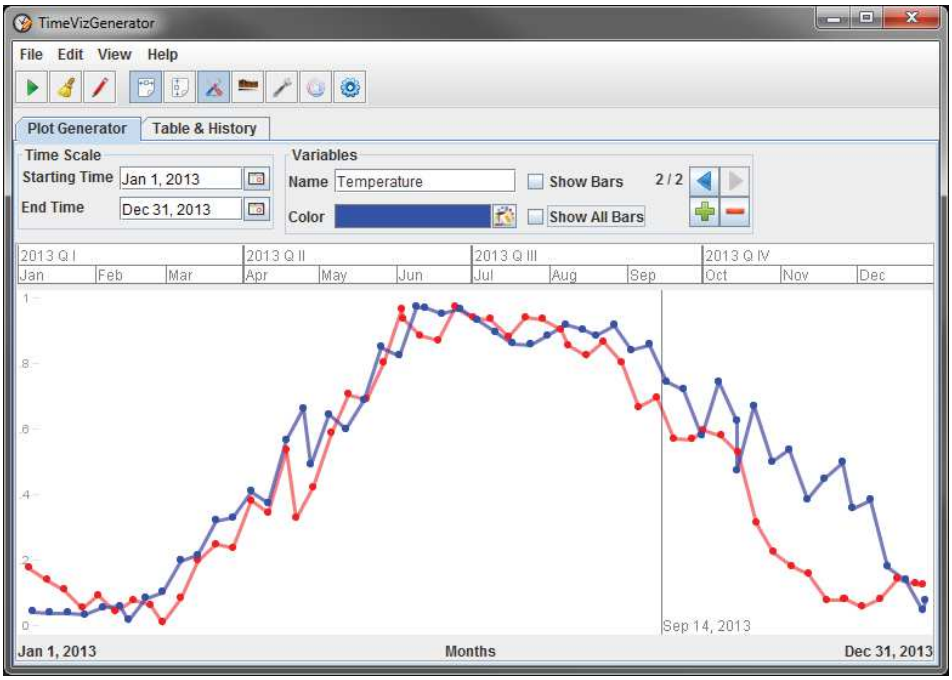

Figure 6.8: Generated values showing their trends, combined on one line plot

6.1.6 Usage of the Cycle Plot

Using the cycle plot for generation of time-oriented data provide the additional possibility to generate a cyclic set of data. The prototype provides three different forms of cycles:

- A week with all seven days of the week.
- The weeks of a month. The implementation of the prototype assumes four weeks in a month for an easier calculation.
- The four quarters of a year.

The precondition of using the cycle plot feature is an adequate time scale. It makes to sense to use a cycle interval of a week, if the time scale is only set to a week or less. The prototype does not check the meaningfulness for the cycle plot in combination with the time scale. The user is responsible to use the features in a meaningful way.

The third task of the user evaluation is similar to the first and second task, but asks to use the cycle plot for the generation of the data. We have a seasonal trend on the days of week. First it is required to enable the cycle plot view, which can be done by enabling the checkbox in the cycle plot configuration. The cycle interval is set to the days of the week. Then the software shows line plots for each day of the week, which should represent the view of a cycle plot, as shown in Figure 6.9. Bottom to each plot the responsible day of the week is labeled. On top the time scale is set from the beginning of January to the end of December 2013.

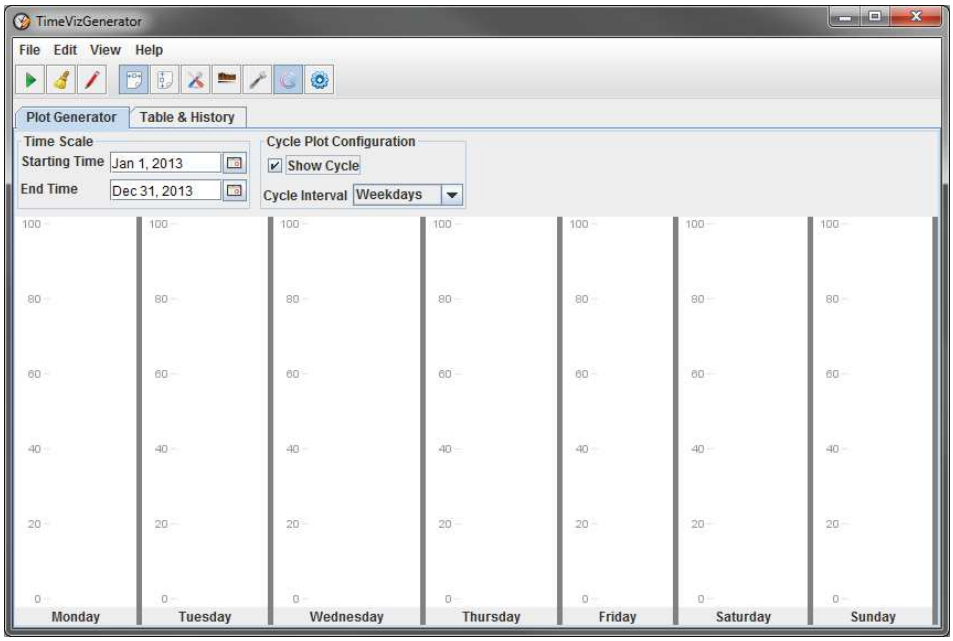

Figure 6.9: Enabled cycle plot view with a line plot for each day of the week

We have to notice that the plot for each day of the week represents the whole year 2013. So it is necessary to draw the general trend seven times. For Monday and Friday the general trend should be 10 %, on Saturday and Sunday 30 % higher. So it is required to consider the different values depended to the days of the week.

The generated amount of values is calculated out of the defined settings of time scale and the cycle interval. In case of a seasonal trend on the days of a week for each day a value gets generated. In case of the whole year 2013 the result consists of 365 values, one value for each day.

The usage of the cycle plot for data generation requires much space, as it is necessary to display seven line plots next to each other. The more space is available on the screen, the easier the value ranges can be defined. In case of the third task two screens are advantageous, as each line plot then receives more than 500 pixels when using two screens with 1980 pixels each. Figure 6.10 illustrates the necessity of more screens for this scenario.

Figure 6.10 shows the result of the generated data set. Each day of the week shows similar output of the generated values. Saturday and Sunday contain higher values compared to the rest of the week. Now it is of interest to switch to the simple line plot, which combines all values on one plot. Figure 6.11 shows a trend with much more irregularities compared to Figure 6.4. The higher values on the weekend cause this result.

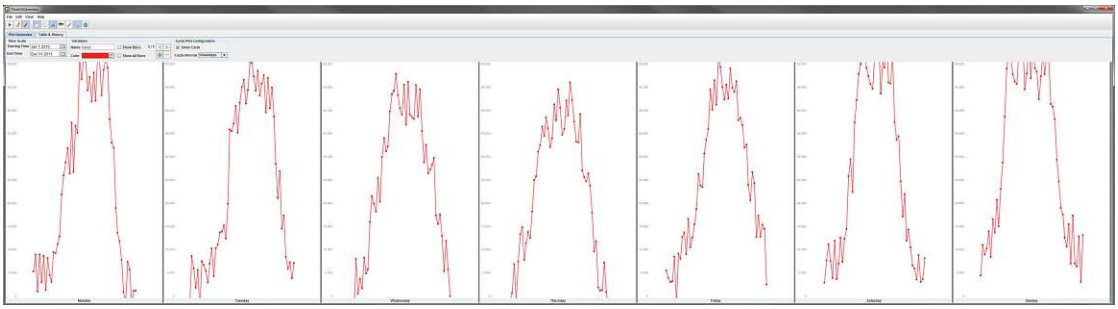

Figure 6.10: Cycle plot view using 3960 pixel on 2 screens

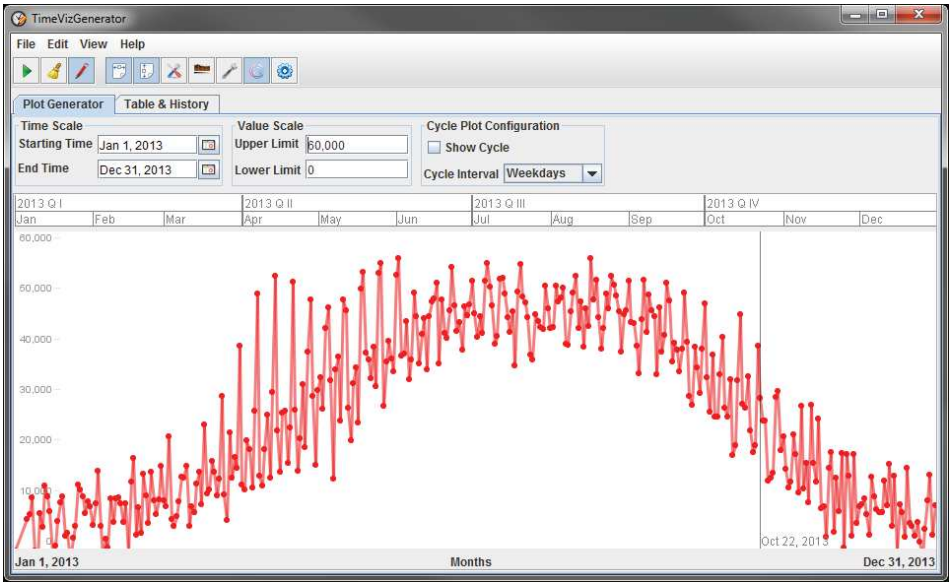

Figure 6.11: Line plot combining the values from the generation of the cycle plot

6.2 Expert Inspection

The expert inspection of the prototype is done by Alexander Rind, supervisor of this master thesis. He evaluates the prototype and its design against guidelines and usability and looks for problems, which are mainly identified by experts in time-oriented data. It is expected that he finds design issues, especially from the expert view, and points to other or better solution, how specific design problems could be solved. Technical details, problems and improvements are not in focus, as the prototype should only illustrate a design.

The following enumeration contains questions concerning the usability and potential problems, evaluated in the prototype. The expert is not asked to answer these questions in detail. They should support the expert during the inspection, so that several essential aspects are recognized.

A description of expert inspections for evaluation is available in chapter 2.7.1.5.

6.2.1 Questions, Usability Issues and Problems

6.2.1.1 Clearness and the Goal

An important part of the design is situated in the clearness. It is asked that the user can identify what the software is used for. The user has to identify the goal and how the achieve a meaningful result.

- Is the basic structure and the layout understandable, so that the user can identify all essential parts and functionalities?
- Are there any labels and descriptions missing or unclear for any parts or widgets?
- Can the basic structure and functionalities by identified very fast, so that the user knows what to do?

6.2.1.2 Intuitiveness & Recognition

The user experience provides the possibility to identify repetitive patterns, which are equal or similar in many different software solutions and designs. Because of previous experience with other software solutions the user should be able to identify the structure and many patterns very early, without the need of any documentation.

- Does the user identify well-known patterns of software solutions?
- Does the prototype contain individual solutions, which are not common on the software design? If so, are these individual solutions justified?
- Are there any unusual features of the prototype, which should be solved in another way?

6.2.1.3 Simplicity

Simple software designs do not require much documentation, as the user cannot do it wrong. Low complexity avoids many mistakes and error, which may occur during run-time. Software should provide automatic solutions for specific problems, as simple as possible?

- Does the prototype contain any features, which are more complicated than necessary?
- How good is solved the reduction of complexity to generate time-oriented data?
- Which aspects of the prototype can be simplified?
- How much documentation and experiments are necessary to get into the software and its design?
- How much redundancy of features does exist in the application and how disturbing or helpful is it?

6.2.1.4 Mouse Interactions

The mouse as a simple tool for doing inputs solves many problems. It simulates the movements of the hand directly on the screen and the user can touch elements (as long as there is no touch screen available).

- How handy are mouse interactions on the plots?
- Are there any essential mouse interactions missing or disturbing?
- Does the direct feedback of the application provide enough information for the user to identify, what is happening at the moment?
- Are there any features solved with the mouse, which are faster or more comfortable with the keyboard?

6.2.1.5 Quick Result & Little Effort

The user should be able to achieve quick results with a minimum of effort. If using the software requires too much time, it may be easier and even faster to define sample data by hand.

- How fast it is possible to achieve meaningful results?
- How much effort is necessary to define and generate a specific set of time-oriented data?

6.2.1.6 Drawing

As the prototype uses a drawing feature to define the trend of a set of time-oriented data, it is of interest that the user is able to use this feature quick and easy. This software design is no graphic program, but uses graphical aspects to define and to visualize data sets.

- Is it easy to identify the drawing feature very quickly?
- Does drawing a trend produce a meaningful result?
- Does drawing a trend accelerate the definition of a data set, so that the output can be achieved faster?
- How handy is the drawing feature on the simple line plot?
- How handy is the drawing feature on the cycle plot?

6.2.1.7 Error Tolerance

Errors and mistakes can occur quite often. Sometimes the user does not use the application, as designed and errors may occur. Exactness is required by the application, but the user as human is not infallible. The application should help the user to use and interact in a correctly way. Incorrect input should be avoided.

- How good does the prototype prevent the user from incorrect input?
- Are there too strong restrictions, so that the user is restricted in his/her possibilities?
- Does the feedback by the prototype provide enough information for the user to identify incorrect usage?

6.2.1.8 Usefulness & Efficiency

The most important question asks for the usefulness and the efficiency of the software. If the software is not useful for the desired purpose, it is the death of any software solution. Software should help and accelerate the user to solve specific tasks.

- How is the overall impression of the software design?
- Is the principle of the design understandable?
- Does the software design accelerate or simplify the generation of time-oriented data?
- Are there any parts of the design, which fail and are not usable in their current state? If so, are these problems solvable so that they become useful?
- Which parts of the design are most helpful and satisfy their purpose best?

The expert and supervisor Alexander Rind tested and evaluated the prototype on 25.02.2015, without the presence of the author. At first he started to experiment with the user interface. After getting involved with the prototype he did the predefined tasks of Chapter 3.2, which are also solved by non-experts in the user evaluation. The work with the prototype was recorded with a screen recorder. Next to the work on the tasks he also provided videos with bugs, errors and other critical problems he identified during the evaluation.

6.2.2 Session state

The expert described that the prototype does not persist the state of the user session and its settings. Every time when the prototype is restarted, all controls and settings are reset to the default settings. It is asked to save and reload the state, so it is possible to continue with the software at the same state, as it has been closed last time.

Many applications provide the possibility to keep the settings of the user interface even after closing and restarting. It is much easier for the user to reenter work, if after restart the application looks like as it has never been closed.

6.2.3 Quick Result

The property of creating a quick result is not clearly fulfilled, as it is not possible to create any output after starting the prototype. The user has to predefine at least two value sectors, before it is possible to produce some output. The expert did not know how to interpret it, because he did not get any response by the software. The scenario is not a bug or other kind of an error. The user should receive some kind of response; otherwise he stays unclear about the state of the program. An error message is better than no message.

6.2.4 Undo & Redo

Applications often provide the functionality to undo or redo the last operation. Often the user executes something he did not want to and reverts the last operation. Sometimes he changes his/her mind and executed the operation again; he redid the operation.

The prototype does not provide such functionalities, as more effort would have been required for implementation. Nevertheless "undo" and "redo" would be practical functionalities, as cleaning the whole plot may occur very fast. The expert recommended to use reconfirmation dialogs for critical operations (e.g. cleaning the plot), if no "undo" and "redo" is available.

6.2.5 Text Fields for Upper and Lower Limit

The horizontal and vertical scale of the plots is defined by text fields, where it is possible to define the upper and the lower limit. The input of the text fields only makes sense, if the lower limit is lower than the upper limit. To guarantee a valid input, the last edited text field changes its background color to red to indicate that the input is not valid and has to be changed. Additionally it is not possible to leave the text field, as the program does not allow setting the focus anywhere else, if the input is not valid.

The expert claimed about freezing the focus to the text field, if the input is not valid. When he wanted to change the general limits down or up he has to make sure to edit the correct text field, dependent to the direction of change. When he wanted to change the general limit up, the upper limit should be changed first. Otherwise it is possible that the lower limit exceeds the upper limit, which is not allowed. He started with the wrong text field and was not able to define the limit he wants to. So he had to define a valid value, then change the limit of the other text field and finally define the value, he wanted to define first.

Freezing the focus to a given control, if it is not valid, is a way to guarantee the consistency of the input fields. This works fine as long as the input field is not dependent to any other inputs fields. So this behavior is not advantageous for the text fields defining the upper and lower limit. If any input fields are dependent to each other, all of them should be editable. Freezing the focus is not necessary. It would be enough, if it is not possible to generate data. This would have the same effect and does not restrict the user in his/her possibilities.

6.2.6 Handling of Value Sectors in the Line Plot

The handling of the value sectors has some weaknesses. Generating a sector manually is done with the right mouse button. The sector always appears vertically centered on the plot, indepen-

dent to the position of the mouse. It only notifies the horizontal position of the mouse. It would be more intuitive if the mouse position is also essential for the vertical position.

The width of the value sector is implemented in a static way and has always the same width. Guessing the correct with is not possible. Instead a user friendly behavior could be reached by saving the width of the last created value sector and then create the next one with the same width.

It is possible to remove all value ranges with only one click, but the plot stays as long as a new generation process is started. The expert asked to remove the plot but keep the value sectors. This feature is not necessary, as it does not prevent the user from operation. On the other side the user wants to keep the plot clean and is interested in removing all unnecessary visual details, e.g. the last created plot.

Drawing the value sector is an easy way to produce a quick result. The prototype generates value sectors based on a user drawn line, but may produce clusters of value sectors. Based on the mouse speed the distance between the value sectors varies. This problem is illustrated in Figure 6.12. The expert started at March 2^{nd} to draw. At March 3^{rd} he continued drawing very slowly, at March 4^{th} he accelerates the mouse speed. At the end he received a clustered output of value sectors on March 4^{th}, as shown in Figure 6.12. The value sectors should be created in a regular distance, which is expected by the user.

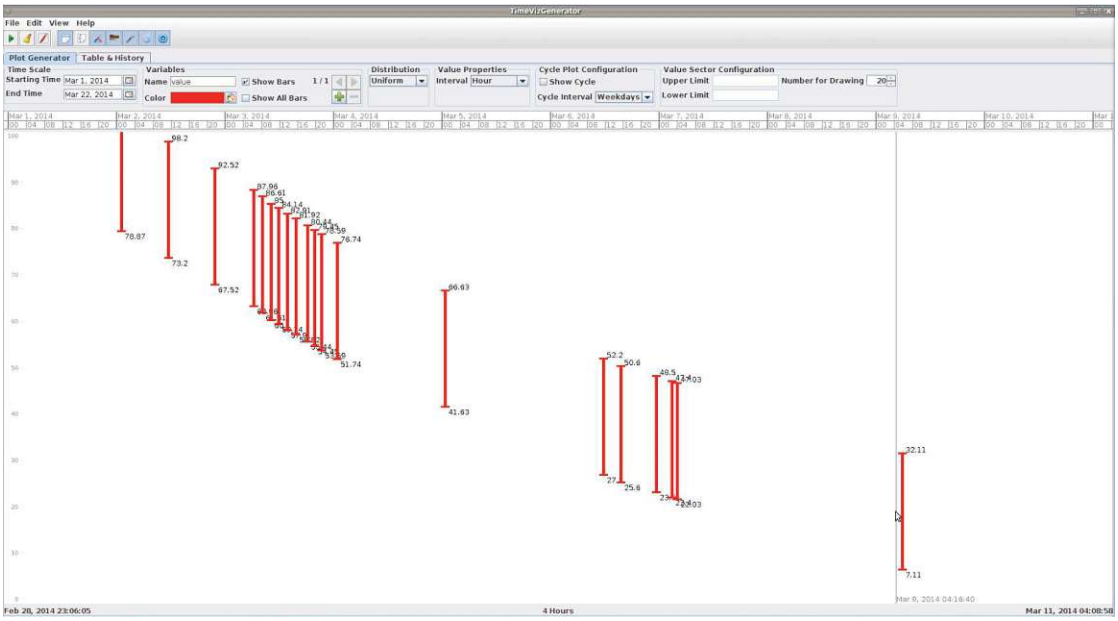

Figure 6.12: Clustered Value sectors after drawing during the expert inspection

6.2.7 Handling of Value Sectors in the Cycle Plot

The usage of value sectors in the cycle plot is different to the line plot, as for each values a value sector is reserved. So creating value sectors as in the line plot does not work identically. The expert claimed about the different behavior. He cannot create line plots manually and has to

use the drawing feature. Furthermore, when drawing the trend on the cycle plot, it is possible to miss sectors on the left and on the right side. The user does not see the missed sectors and so receives an unexpected result after generation. This problem is illustrated in Figure 6.13. The expert initially starts a generation of values to identify the horizontal position of the values. Then he tries to generate value sectors manually, but fails, because this is not possible in the prototype. Instead he draws the trend to receive them. Furthermore, drawing on the cycle plot view removes all existing value sectors, existing before. This may unexpectedly remove the user's previous work.

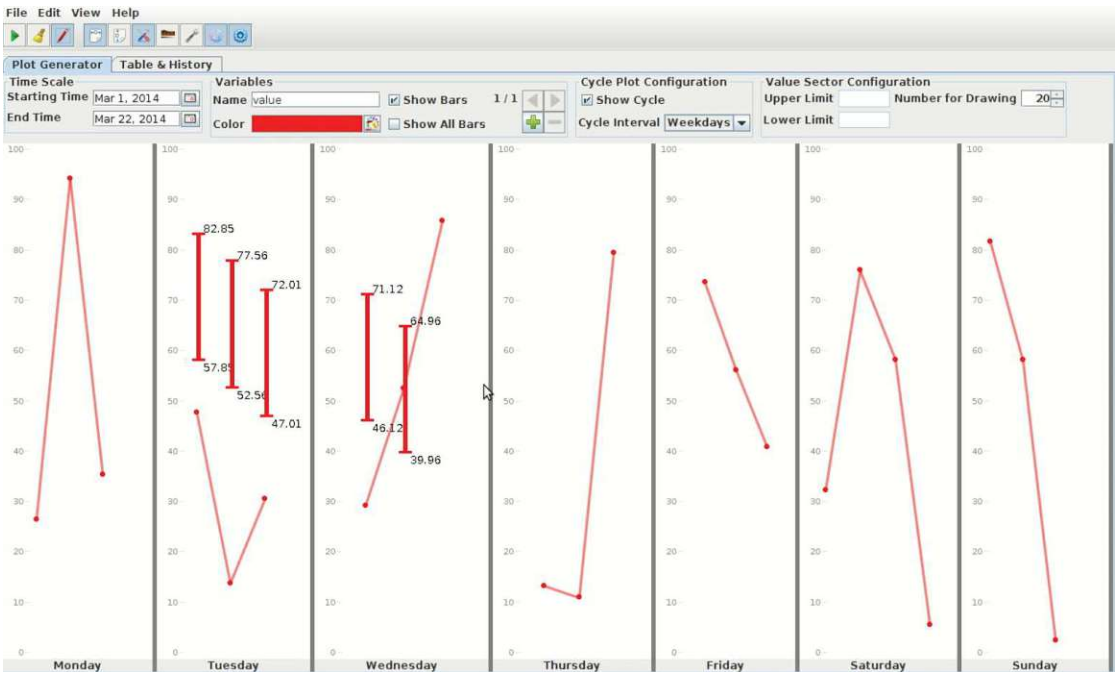

Figure 6.13: Value sector problem in the cycle plot view during the expert inspection

The value sectors in the cycle plot are fixed and cannot be repositioned horizontally. So it would be advantageous if their position would always be visible. The value sectors should also be producible without the drawing feature. These problems may be solved by always displaying value sectors, which are not removable. The user has still the possibility to make them invisible, if he does not want to see them.

The implementation of the prototype expects that the user defines the trend for each part of the cycle, e.g. the trend for each day of the week. The expert recommended that it should be possible to take over the settings from one part to another, or even define the trend for all parts. Then it is much easier to adapt the trend for each part of the cycle plot. Also the output is more regular and the user has not to take care of creating similar trends for each part.

The handling of value sectors in the cycle plot has some weaknesses and is not as good as in the line plot. Even the difference to the behavior in the line plot is the biggest point of criticism, because the expert expected the usability to be nearly equal. Such differences should be avoided,

as it improves the recognition of functionalities.

6.2.8 Labeling and Formatting

Choosing meaningful names and labels are very helpful for the users, when working with new software. The best case is reached, if the wording in the prototype is good enough so that no additional help or documentation is required. This is not always possible, as complex or extraordinary functionalities cannot be explained in a view words. On the other side the complexity could be the problem and should be reduced by redesigning the functionality.

The expert identified a few labeling problems in the prototype. The labels "Value Properties" and "Interval" are confusing and do not describe their functionality detailed enough. The expert related the term "value" to the variables, the term "interval" to some interval data. "Value Properties" contain configuration possibilities for the generated output, in the case of the prototype the interval between the single generated values (e.g. hour, day, week, etc.). He would define the functionality with "Temporal Granularity". All in all this case of labeling problem requires a better description.

The spinner control "Number of Drawing" is unclear. This spinner is responsible to define the number of created value sectors, when drawing the trend. For each drawn line the specified amount of value sectors is created. The expert did not identify this functionality. If the expert was not able to interpret its meaning, the chosen term was totally incomprehensible. A meaningful link to the value sectors is missing, so the user is not able to understand its meaning. On the other side it is questionable, if this feature is useful or required anyway, as the expert was not looking for this feature.

The used formats in the history are not a good choice, because the chosen formats are too complex. The overview is suffering, which can be improved by using simple formats. A more advanced solution could be the possibility to provide several formats and let the user chose his/her favorite, as offered by Microsoft Excel.[1] This would also help if the user has individual preferences, e.g. the expert required seconds in the data list.

The prototype uses different time and date formats in different areas, which may confuse the user. The expert advises to use a general easy format and not to mix up different formats.

6.2.9 Value Precision

The time point of the generated values is dependent to the first point, which gets generated. The following values have an exact regular distance to each other (e.g. a day, an hour, etc.). The first value is a random hit and has no dependence to the granularity. When generating values for each day, the value for a day should be at the beginning, so at 00:00. This has the advantage of reduced noise in the data, because all units smaller than the granularity are not of interest and should not be displayed. The time formats stay simple and are easy to read.

[1] Commercial spreadsheet application developed by Microsoft

6.2.10 Inconsistency

Inconsistency is a disagreeable behavior, which confuses the user and makes an application unnecessarily complex. The prototype contains a few inconsistencies:

The behavior of the button for activating the drawing feature is different between the cycle and the line plot. On the line plot the button unselects after drawing, while in the cycle plot the button stays selected. The expert did not expect a different behavior. He executed drawing actions on the cycle plot he did not want to, because the buttons stayed selected, in contrast to the line plot.

Resetting the plot works differently in the cycle plot and the line plot. While in the line plot the generated values and the value sectors are removed, in the cycle plot the value sectors remain.

In the line plot the interval combo box defines the distances between the values. In the cycle plot this combo box has no effect, as the distance is defined by the cycle interval. This is an unnecessary introduction of additional controls. Both combo boxes should be merged together, because they are responsible for the same value (defining the distance between the values).

6.2.11 Conclusion of the Expert

The summarizing feedback of the expert (Mag. Alexander Rind) describes the design and the implemented prototype as good functional and intuitive. He experimented with the prototype on his own and also tried to solve the predefined task, which are used for the user study. The tasks contain some challenging details, which show the limits of the prototype and its design. An essential problem is situated in the interaction with multiple variables with different scale factors. The user is responsible for matching scale factors, which are visualizable.

The expert also identified some realistic improvement opportunities, from which the design and the prototype would benefit. One critical point of interest is the consistency within the application. Inconsistencies provide different behavior and representation, which can be unified.

6.3 User Study

For this user study only two people participate. Both have an adequate knowledge of information technology and work in the area of software development. They represent two non experts, as they have no experience with time-oriented data.

Descriptions of user studies and intensive interviewing for evaluation are available in chapter 2.7.1.6 and 2.7.1.2.

6.3.1 1. Participant: Student of Business Informatics

The first participant Lukas Mad is studying business informatics at the business university of Vienna. He has finished his bachelor study and continues with the master study. He worked as a trainee for an insurance company in the department for software development for eight months and has good knowledge in the development of Java applications. As he never worked with

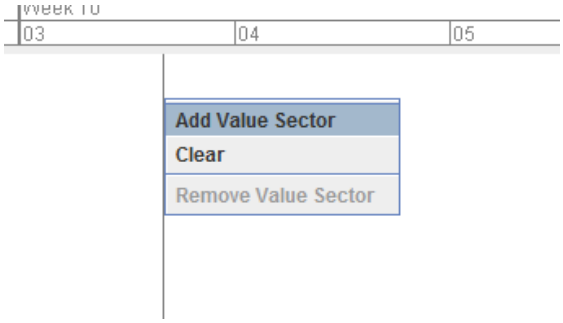

Figure 6.14: Context menu of the plot, which is required for adding new value ranges

time-oriented data at the university or professionally, he has no experience with time-oriented data and its specific characteristics.

Before the user study started, he received basic information about the topic of time-oriented data. He got the information that the prototype is a data generator and should produce sets of time-oriented data. He took this time to read and understand the tasks he should solve with the prototype and was asked to think loudly, which should give more information about his mind and intention during the work. It was planned that he do not receive too much information about the design, the prototype and other details of the product itself, as this could influence the first touch with the software design.

1. Task: A Simple Dataset

The first task was to define a simple dataset for one year, as described in chapter 3.2. This first task should give the participant the possibility to become familiar with the prototype and its design. I expected that the participant requires the most time for the first task, as he had to identify the functionalities and the behavior of the prototype. "Learning by doing" is a common way to become familiar with new software. As people in the information technology often try to learn using software by simply using it, I expected that the participant would identify the basic functionalities within ten minutes.

Producing initial Output After the first approaches I identified that the participant had problems producing any output. As the prototype requires at least two value ranges, which define the start and the end of the data generation, he was surprised that he was not able to produce any output quickly. The design is based on the definition of value ranges. As long as he did not identify, how to define them, he was not able to produce any output. He required the hint that he had to use the mouse wheel and the right mouse button on the line plot.

As the user did not identify the possibility to add a value range, he was not able to produce any output. Furthermore, the information that at least two value ranges are required was missing. In this case additional hints by the software would be helpful. The requirement of the value ranges and the minimum amount for an output have to be communicated to the user anyway. Additionally, value ranges should also be addable without the right mouse button. The

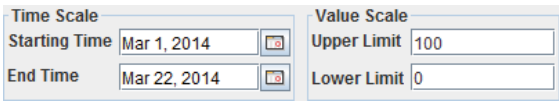

Figure 6.15: Configuration parts for defining the horizontal time scale and the vertical value scale

participant gave me the advice that the right mouse button is not the first action, when he gets familiar with any new software. Menu or toolbar buttons could be helpful. Another possibility to help the user is the usage of tooltips or other small popups, which advices the user how to use the software. Figure 6.14 shows the popup menu of the prototype, when using the right mouse button clicked on the plot. The participant did not expect that the right mouse button has such a high priority.

Scaling The plot provides the possibility to define the scale factor for the horizontal time axis and the vertical value axis. When dealing with different scaling factors it is necessary to define the visible area. This can be done by using the configuration parts for the time scale and the value scale. I expected the participant to define the visible area. The participant identified the configuration parts, but he did not exactly know what they define or how to use them. Figure 6.15 shows these parts of the prototype, where the horizontal time scale and the vertical value scale can be defined. In his first intention he mixed the definition of the visible area with the definition of value sectors. I had to give him the advice that these two configuration parts are only responsible for the visible area. This problem illustrated that the user is not able to combine these two configuration parts with the panel itself. On the one side more applicable naming and labeling is required, on the other side the definition of the visible area should also be possible in a different way.

The participant expected to use the mouse for defining the scaling factor of the visible area. He identified the possibility to scroll the horizontal time line by pushing and holding the left mouse button. But he also expected this possibility for the vertical axis. Also the mouse wheel for zooming the visible area was intuitive. Consequently he tried to use the keyboard in combination with the mouse to identify additional scaling functionalities. A common combination is CTRL and the mouse wheel, which he tried to use, but the prototype did not provide such functionality for this combination of keyboard and mouse. Many familiar applications use this combination for scaling and zooming, e.g. in Microsoft Word this combination alters the zoom factor, in Mozilla Firefox the font size changes. The participant expected this behavior of the prototype, as the mouse is used for scrolling on the plot.

The problems of the participant illustrates that the common concept for using scaling and scrolling should be used, as users expect a similar behavior to common applications, e.g. Microsoft Word or Mozilla Firefox. In such a case the configuration parts would not be as important as in the prototype, but provide a possibility to define a more detailed configuration of the scaling.

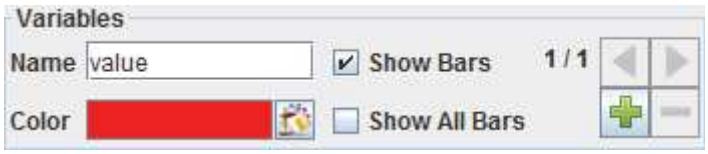

Figure 6.16: Configuration part for defining variables

Variables vs. Value Ranges This software design provides the possibilities to define multiple variables, which can be displayed on one single plot. The definition of the variables can be done through a configuration part, as shown in Figure 6.16, where new variables can be added and their properties can be defined. For each variable multiple value ranges can be defined, where the generated values should be positioned.

The participant had a problem to distinguish between these two basic elements of the software design. In his first intention he created for each value range a new variable, as he mixed up these two concepts. I had to explain the difference; otherwise he would have run into a wrong direction and would have never received any output. As each variable requires two value ranges to produce some output, he would have never received any feedback, because he would have created multiple variables with one single value range.

In this case a differentiation is required between the definition of the variables and the value ranges. Similar to the problem of getting an initial output, the user requires feedback that each variable requires at least two value ranges. There are many possibilities to solve this problem. One potential solution could be a warning, when the user tries to start a generation, where a variable has not enough value ranges.

The Value Interval The value intervals define the interval of the generated value points. Depending to that interval a different amount of values is generated, e.g. when defining an interval of one hour, each hour a new value is generated. This is independent to the defined value ranges. Only the first and the last value range define where the generation starts and where it ends.

The participant was looking for such functionality, as he achieved his first useful result. He had the problem to understand the labeling. This problem illustrates that the wording of the labels is not good enough. Furthermore, the user should receive hints and explanations for specific terms, if they are not clear. This can be implemented by using tooltips.

Summary of the 1. Task The participant required 21 minutes to solve the first task. As this should also have been an introduction into the design and the prototype, the main concepts and principles of this software design have been identified by the user. He was surprised to work only on a single plot. He expected the generation process starting from a table to a graphical output, but identified the other way around. He additionally mentioned that this is not a negative aspect, as working direct on the plot accelerates the pre-generation process. On the other side he expects from a software product that it provides a short introduction of its usage.

Some critical problems were identified. Producing a quick initial output was not that easy. A user wants to receive a fast result, when experimenting with new software. The fast result was

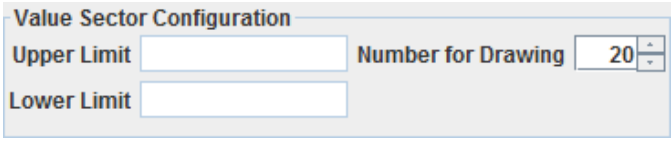

Figure 6.17: Defining the number of generated value ranges as unused input field in the prototype

prevented from at least two required value ranges. Another critical problem is the handling of scrolling and scaling, which does not work equal to other familiar software solutions. As it was not the aim to break a proven concept, these functionalities should be adapted to the common functionality.

Another critical problem is situated in the naming and labeling of some features. The user did not identify the controls and configuration possibilities with their intended functionality. As the user is no expert of time-oriented data, he is also not familiar with specific terms and expressions. Nevertheless this problem could be solved by providing tooltips and other helpful hints, which are provided by the software itself. Then it should be much easier for a new user to get familiar with the design.

2. Task: A Simple Dataset With Drawing Feature

The second task was to repeat the first by using the drawing feature, provided by the software. The drawing feature provides the possibility to draw a trend directly on the plot, like a pencil on a peace of paper. As the drawing feature is the most interesting part of this software design, the expectations on the usability and the feedback by the user are very high.

Identifying the Drawing Feature The participant did not identify the drawing feature by solving the first task. He received the hint through the second task. His first action after start was looking for a button in the menu or toolbar. There he was able to identify this feature very fast, as it is represented by a simple pencil. The pencil as a symbol is many common programs (e.g. Microsoft Word, Microsoft Paint, GIMP) and is always used to draw something freehand. So the recognition of the symbol is given.

He understood the drawing feature very fast, as he was able to draw a line from left to right. He started to experiment and identified that drawing the line was only possible from left to right. When trying to draw a line the other way round no line is visualized. A time axis is commonly ongoing from left to the right. He commented this restriction as helpful as it prevents the user from creating something useless. The output is always useful.

Configuration of the Generated Value Sectors The prototype provides the possibility to define the amount of value sectors, which gets generated when drawing a line with the drawing feature. The default amount is set to 20, which should provide an adequate amount for the most use cases, as shown in Figure 6.17 from the prototype.

The participant did not identify how to regulate the amount of created value sectors, but he even did not ask for such functionality. He did not modify much after drawing and was happy with the result. After finishing the second task I gave the advice that it is possible to define the amount of generated value sectors. He answered that there was no need to alter the amount, because the default configuration was enough for the use cases. He also mentioned that it is much more comfortable to draw a view lines and then delete some value ranges again if they are not required.

This case illustrates that the possibility to configure the amount of generated value sectors is not required by the user. This shows that there is also the situation of having unnecessary features which are not used by the user. He identified another way which is more comfortable than the predefined feature.

Problem of the Value Interval On the other side the value interval is also essential when using the drawing feature. He had problems to identify the configuration part again, where it is possible to adjust the interval between the values. As the user had the same problem again one task later, this seems to lead to the conclusion that defining the value interval has to be solved someway else. The recognition of the symbol in the toolbar and the configuration part itself is not sufficient.

Summary of the 2. Task As the participant required six minutes to solve the second task; this illustrates the good recognition for identifying the tool and its functionalities. He did not have any problems to use the drawing functionality and understood the principle and its details very fast. He also started to experiment and did not require any advice or other hints. The task was solved very quickly.

He gave the information that the drawing functionality is a quite fast and comfortable feature. As it is a bit uncommon to draw a simple line on a line plot and receiving an output immediately, the understanding is very good and intuitive. Furthermore, it provides the possibility to draw the basic trend and fine tune the generated value ranges afterwards.

3. Task: Seasonal Trend by Using the Cycle Plot

This task included the usage of a cycle plot, as the generation of a seasonal trend was required. The user was asked to generate a data set, where the values differ between the days of the week.

Problems Understanding the Cycle Plot The biggest problem was situated in understanding the behavior and functionality of the cycle plot. As the cycle plot is a specific visualization technique for time-oriented data, it is hard to understand without introduction. As the participant did not have any experience with time-oriented data, he also did not know the cycle plot and its specific properties. As I expected the participant had problems to understand and interpret the visualization. Furthermore, he was asked to generate data by using the cycle plot visualization, which provides more complexity than understanding a simple cycle plot without interactions. He was confused and did not know how to use it, even if he knows the principles of the software design and its characteristics.

I had to explain the concept of the cycle plot in detail, that he was able to use the cycle plot functionality of the prototype. This demonstrated that a cycle plot is a more complex visualization technique and requires previous knowledge of this technique. It would have been advantageous if the user would have received an introduction to the cycle plot. People without knowledge about the cycle plot are not able to use the technique in combination with the design, as they cannot identify the structure and functionality.

Missing Legend of the Time Axis Additionally the user claimed about the missing legend of the time axis. The time axis was faded out to receive more space for the visualization. Furthermore, the day of week already represented a legend for each part of the cycle plot. For better understanding the time axis should not get faded out.

Not enough Space for Interaction Multiple plots leads to another problem. As there exist seven days of the week we need seven small line plots, positioned next to each other to represent a cycle plot. The small plots for each day of the week are very small. Although the user was working with a wide-screen monitor, there was not enough space for comfortable interactions. The usage of the drawing functionality was very inaccurate and did not produce a good result for the third task.

For using the cycle plot, more than one monitor is required for comfortable interactions. The program gets expanded to two or more screens and each plot receives enough space.

Switching between Cycle and Line Plot The design provides the possibility to switch the view between the cycle plot and the line plot very quickly. The participant had no problems to do so and it was clear that both techniques of visualization should represent the same set of data.

The synchronization between the techniques was a bit confusing for the participant. He generated data with the cycle plot and identified the generated data set in the simple line plot again. But when generating data with the simple line plot, he did not see anything on the cycle plot. This was identified as a missing part of the design, as there is no logical problem to represent the data from the line plot also on the cycle plot.

Only the synchronization of the value ranges could not be mapped 1:1. When generation data on the line plot with any amount of value ranges, they could not mapped exactly on the cycle plot. The cycle plot only allows value ranges for specific time points, e.g. when using a cycle plot on the days of a week, the part for the Mondays only allows value ranges for a Monday. Consequently value ranges from a simple line plot need to be only on Mondays. This synchronization problem could be solved by interpolation. The software has to calculate the value range for each Monday a represent it on the cycle plot. Then the synchronization of both visualization techniques would be more clearly and the user does not get confused about any missing representation.

Summary of the 3. Task The user required more than ten minutes to solve the task in an appropriate way. The cycle plot, its characteristics and functionalities were difficult to understand, as he had no experience with it. Another problem was situated in the incomplete synchronization

between the cycle plot and the line plot. Switching from the simple line plot to the cycle plot did not visualize the data, when generated data on the line plot.

As it can be seen the topic of the cycle plot is much more difficult. The user requires basic knowledge about the visualization technique, as he is not able to deal with the technique without any experience.

4. Task: Interaction with Multiple Variables

For the last task the user had to use multiple variables on the single line plot. I expected that switching between the different variables would not be very easy for the user. But the participant did not have any problems. He was able to create an additional variable for the temperature and identified how to manage both on one single line plot. The concept of the current active variable seems to be intuitive, as he switched between the variables without any hints or help.

Different Value Ranges of the Variables As the software design does not provide any functionality to normalize the values of different variables; the user is responsible to deal with it. The participant understood the problem and mentioned that the normalization of the values would be a nice feature for future work. As the prototype did not recognize it, the participant used the configuration possibilities to configure each variable and its trend. Because of the drawing feature, this problem stays unproblematic for him.

Feedback and Impressions by the Participant

The participant was very impressed about the drawing functionality of the software design. It is currently an uncommon way to define a trend, but is much more intuitive than working with bars, widgets or other common stuff. When working with such a data generator for time-oriented data, he prefers to use the drawing functionality in each situation, as it makes it possible to create much data in a very short time.

The cycle plot is a specific visualization technique for time-oriented data. As this is a context specific part of the software design, only people involved into the context are able to use that feature. The participant was not an expert on time-oriented data and had problems to identify the concept and its functionalities, as he did not know the cycle plot before.

When getting touched with any new software, the participant expects tooltips, hints for other form of help, which leads the user to the correct usage. The prototype provided tooltips, but they were sometimes not sufficient. Especially each configuration part should have quick information, what can be configured and what impacts each widget has to the plot. Short labels are often not enough so that a user does not exactly know how to use that part of the software.

At the end the participant is convinced about the software design. He thinks that the generation of synthetic time-oriented data becomes much easier with such a tool. The weaknesses and problems, which have been identified, are solvable and most of them are technical issues. The concept of the design shows the potential and the possibilities, when spending more time and effort into the conception and technical implementation. He did not identify conceptional pitfalls and classifies the problems as mistakes during the implementation or simply missing functionality.

6.3.2 2. Participant: Professional Software Developer for an Insurance Company

The second participant Alexander Jakovljevic has an education for software development and works as software developer for an insurance company in Austria. He has good knowledge about Java applications and has much experience with the development of rich client applications, especially in Java and SWT. He has no experience in time-oriented data. As he never studied on a university or similar form of education, he has no academic graduation. So he will focus on software usability, as it is expected and developed in the private sector.

Before starting with the user study he asked for basic information about the prototype, the context of the topic and the aim of this study. He got the information that the software, he should test, is a prototype for the visual generation of time-oriented data. Furthermore, I explained the term what time-oriented data is and that this prototype should be able to produce synthetic data. The use case for such a software design is situated in the problem that real data is not available due to privacy issues or such data is just not available. To receive adequate samples for visualization techniques, the generation of sample data is of interest.

1. Task: A Simple Dataset

During the first task the participant tried to get familiar with the context and the prototype. He started to experiment with the software and tried to understand the different functionalities, the prototype provides. While getting familiar with time-oriented data, he wanted to understand more about time-oriented data and how this topic is connected to this software design.

While testing several features of the prototype, he identified configuration possibilities, he did not understand immediately. For example he experimented with the switch between simple line plot and the cycle plot. I did not want to tell him too many details before solving the task, so I only gave him the information that he will identify most functionality when working on the later tasks. As the user should identify most features and functionalities on his own, he only received basic information but no specific details to the prototype itself.

Drawing Functionality as the Point of Start He identified the drawing functionality very early, because of the familiar symbol in the task bar: a pencil. The pencil is a well-known symbol for drawing features in many common software products, e.g. Microsoft Paint[2], GIMP[3] as shown in Figure 6.18. These products are common image processing tools and are used for private and for commercial tasks. He did not need any help to identify the functionality of this feature and was able to create value ranges within a few minutes. I expected that the user would identify the drawing functionality in the second task, but he started to work with it, while he did not notice the possibility to add and configure value ranges manually. I did not interrupt him, as it is an interesting situation for this user study.

[2] https://support.microsoft.com/de-at/help/4027344/windows-10-get-microsoft-paint accessed 2019-11-04

[3] http://www.gimp.org/ accessed 2019-11-04

Figure 6.18: Pencil in the toolbar of GIMP

This effect illustrates that a familiar symbol with an expected behavior of the software is essential. The user is familiar with a specific set of symbols and can map its meaning to a correct utilization. He did not need any hints or other forms of help.

Meaning of Value Ranges He continued experimenting but could not match the meaning of the value ranges, which were created by the drawing functionality. He identified to move and re-size them, but was not sure of their functionality and their meaning. In this case I had to explain him what these value ranges mean and how he can use them to generate synthetic data. I noticed that it was not clear for him that the value ranges mark the upper and the lower limit for the value generation.

Their importance for the prototype and the software design became clearer, when he started to generate some data the first time. He saw that the line of generated random values is positioned inside the value ranges, so he recognized their relevance for the design.

The Value Interval For the initial generated value set he defined an interval between the values of one hour over one year. As an interval of one hour is the default value, he did not know that he has to define the interval for this task. This resulted to a set of more than 8,000 values. He

was looking for the possibility to configure the interval between the values, but could not find the configuration part for it. So I had to give him the advice to the correct configuration part.

This situation illustrates that the labeling and the used icon were not enough to detect this feature by the user without any help. The participant mentioned that he was looking for exactly that functionality but he was not able to identify it based on the label or icon. In this case a better labeling and choice of icon is required to improve that situation.

Scrolling on the Line Plot While solving the first task he identified the different mouse features on the line plot. He was able to scroll and zoom on the plot with the mouse. He gave a good rating for zooming with the mouse wheel and the possibility to use drag and drop for scrolling. The first impression was very good, as he had no problems to interact with the plot. Afterwards he mentioned a few weaknesses:

When zooming with the mouse wheel he identified that the plot expanded only to the right, not on both sides. This was unusual for him, as he expect that both sides are expanding from the position of the mouse cursor. This is a specific implementation detail and a mistake in the implemented prototype but very important information for the handling of plot.

Value Points and Value Ranges While solving the first task he mentioned that he would expect the possibility to define an exact value, not only a range. I gave him the advice that this is possible by re-sizing the value range to one single point. This would solve the problem for the given situation but is not comfortable enough for that use case. Using a single value point could be a common use case in many situations and should be considered.

Summary of the 1. Task At the end the user required 18 minutes to solve the first task. As expected the first task was also used for getting familiar with the software design and the prototype. Initially he started to experiment with different functionalities and tried to identify a huge set of features within a short time. Surprisingly he identified the drawing functionality very soon and started to use it. The possibility of manual defining value ranges was not as essential at the beginning. He identified that possibility afterward.

He described the first encounter with the prototype very good and was impressed about the possibilities. Especially the drawing functionality was a very interesting feature and seduced him to experiment and play with it. He mentioned that drawing a trend is much easier than trying to configure it with widgets and other controls. Drawing should be the main feature, while the manual configuration should be used for fine tuning.

2. Task: A Simple Dataset With Drawing Feature (already solved)

Because of the situation that the participant identified the drawing functionality in the first task, there was nothing to do for the second task. As mentioned before this is the best case, as the user identified the feature without any advice.

The user identified the restriction that it is not possible to draw a line from right to left, only from left to right. He did not expect this behavior but mentioned that this restriction is useful in that case. The time is visualized horizontally. It is common that a time line starts on the left

side and continues to the right side. So there is no necessity to draw a line the other way round. The participant prefers the restriction only drawing from the left to the right because the line continues only into the logical correct direction. As the user has no possibility to create any strange input, the drawn line stays always clear.

3. Task: Seasonal Trend by Using the Cycle Plot

When starting with the third task the user was looking for the cycle functionality and identified the required configuration part, which is necessary for this task. Switching between the simple line plot and the cycle plot was intuitive for the user and it was clear for him that both visualizations should display the same content in a different way.

Understanding the Cycle Plot The main problem using the cycle plot is situated in the missing know-how of the participant. As he has never worked with time-oriented data and did not know how a cycle plot works, he had problems to use it. He simply saw seven small plots, one for each day of the week but he did not know how data gets visualized on these plots. The correlation between all plots and the simple line plot was not clear and made it impossible for him to use it correctly. I gave him more detailed information about the cycle plot and its properties, how a cycle plot is working and what it visualizes.

After giving him some contextual information about the cycle plot I additionally had to explain how to use the cycle plot on the prototype. He expected to define the general trend on the simple line plot and then adapt the trend in the cycle plot. I gave him the information that he only has to work on the cycle plot view; the simple line plot only visualizes the overall output of the generation process. After a longer introduction he was able to create a seasonal trend with the drawing feature.

He gave me the information that using the drawing feature for the cycle plot is not as comfortable as in the line plot. The available space for each part of the cycle plot is not enough. This leads to the problem that drawing an exact line is not very easy and the error rate and inaccuracy is rising. Furthermore, the trends for each day of the week are not as similar as desired. In the following the overall output is not as good as expected and the user has to fine tune on the cycle plot very often.

The user expected to use the line plot for defining the general trend and switch to the cycle plot view. Then it would be much easier to configure the seasonal deviation, as the user has only to adapt some parts. Furthermore, the connection between the line plot and the cycle plot would be more clearly, as a cycle plot is produced out of a line plot. He does not have to think so much, because the gets presented a possible solution without any hints. Adapting is much easier than recreating everything.

Display Another problem is the generated amount of value ranges. For the third task the user had to generate data for a whole year, with a cycle plot for the days of the week. At the end each part of the cycle plot displayed 52 value ranges. These are too many and leads to the problem that the user has no overview. He only sees a cluster of value ranges, as shown in Figure 6.19. This illustrates those specific constellations result to a non optimal solution. If the amount of

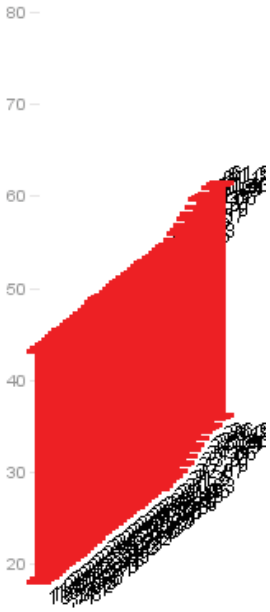

Figure 6.19: Cycle plot view with too many value ranges

value ranges is too high, the graphical visualization is not useful any more. The more interesting part is the trend and the generated values. When showing too many value ranges, the user cannot identify the value any more.

"Undo" and "Redo" Sometimes the user cleaned the whole plot, but did not want to do that. Then he had the problem that he removed the entire configuration on the plot and has to restart again. He asked for a "Undo" button, which undo's his last step. In that case I identified that the concept of "Undo" and "Redo" should be available. Most common software solutions provide such functionality, as users often do things, which they did not want to do. If the user does not have such a possibility, he has to undo it by hand or repeat the whole task again.

Summary of the 3. Task The participant required 15 minutes to solve the third task. Because of problems in understanding the context and functionality of the cycle plot, he had problems to produce the result, as requested in the description. He misinterpreted the functionality of the cycle plot and showed up that in specific situations the usage of value ranges does not lead to an acceptable result.

Because of the problems with the cycle plot, I identified the necessity of a redesign for the handling of the cycle plot. The user is not able to understand the functionality out of the work and there may be situations where good usability is available.

4. Task: Interaction with Multiple Variables

The last task provided the challenge to use two variables on one plot. Only using more variables is not the difficult part. The tricky part is situated in the scale of both variables. The scaling factor is very different between both variables. The first variable requires a maximum value of 50,000 while the second variable has an upper limit of approximately 40. As the prototype does not provide any functionality or other support to handle this problem, the user is responsible for it.

Different Scale Factors The participant had problems to find a solution for this problem, as he saw that the prototype did not provide any functionality as solution. As the participant has never visited a university or other higher education, he had no idea how he could solve this task. I gave him the advice that the problem could be solved in two different ways. The first possibility is to define a common scale for all variables and then scale up with a factor for each variable. The other possibility is to use the scaling possibility of the prototype to receive an adequate view for each variable. In this case the user has to work iteratively for each variable. Mixing up and working on both variables at the same time would be impractical.

The user claimed about the missing possibility to handle the scale problem for multiple variables. He is of the opinion that the design should be intelligent enough to support the user for this interaction, as it is an essential core feature. When working with multiple variables, the user also requires the possibility to handle the problem of a common scale factor. Otherwise the usage of multiple variables is not very attractive.

Another disadvantage of the missing tooling support is the view itself. Because of the very different scale factors only one variable is visualized in an acceptable manner. Either the one or the other variable is not visible and so the visualization suffers because of the missing tooling support. This gets visualized in Figure 6.20 and Figure 6.21. Both figures visualize the same data set, which would have been a possible solution for the fourth task. In Figure 6.20 we can see all values, but the blue variable representing the temperature is too small. We can only see a big line at the bottom, but no details. On the other side Figure 6.21 presents the worst case. In this scene we can only see the blue variable, but not the red one, which should represent the amount of tourists.

The only solution in that case is to define a common scale factor for both variables, where the values can only be situated between zero and one. The user has to scale up the generated values with a factor by hand for the given variable. This cannot be visualized by the prototype at this time. For the future the design should provide a feature, which is able to deal with different scale factors. Furthermore, the line plot should also display the values for each point, so it is possible to identify the exact value.

Summary of the 4. Task The participant interpreted the last feature of multiple variables as unfinished, because the tooling support is not enough for the requirement of generating values of multiple variables with multiple scaling factors. This would be a common scenario and should be solved by the software itself, so the user receives fast feedback and a usable result. He is of the opinion that it is not reasonable that the user has to calculate manually the exact values, as

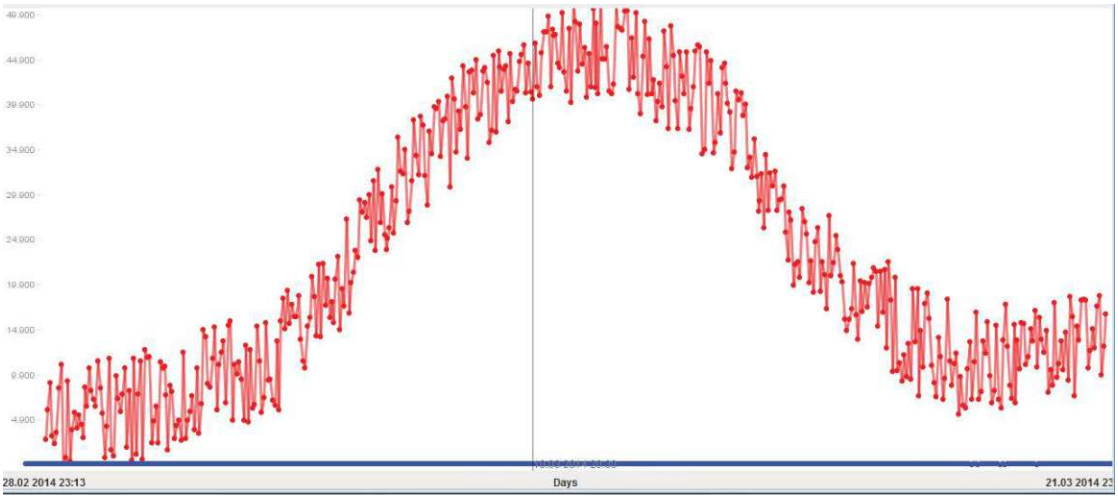

Figure 6.20: Scaling problem, where the values with an upper limit of 45 are not visible

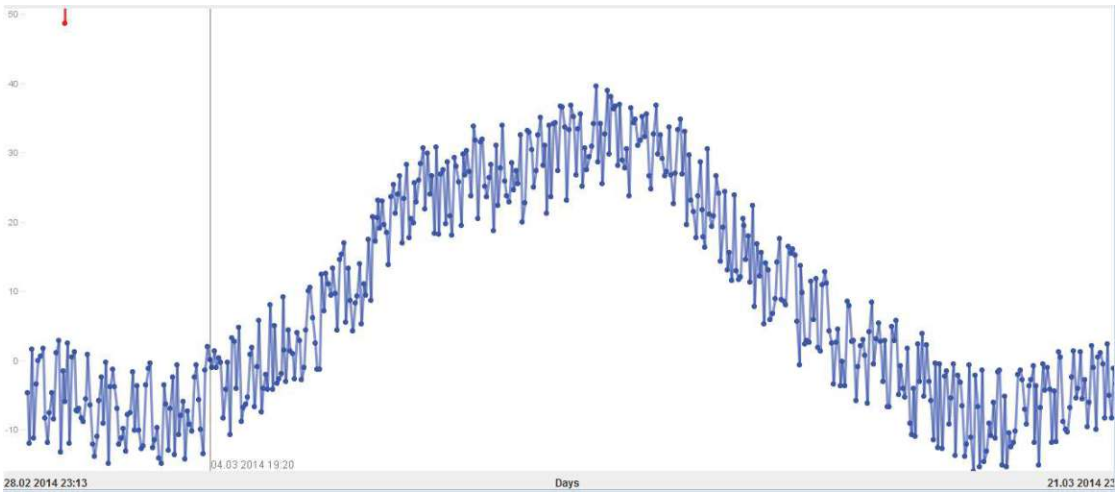

Figure 6.21: Scaling problem, where the very high values vare not visible

this is an automatic task which could be solved by the software. Humans tend to make mistakes, which could be avoided by an automatic solution.

Feedback and Impressions by the Participant

The participant gave all in all a positive feedback after the user study. He identified several interesting and useful features, which are not a common practice for generating synthetic data. The fact that he was not able to identify and understand all feature after a short time of introduction was caused by his missing know-how of time-oriented data. Especially the cycle plot is a tricky visual technique, which is unknown for non experts. Most people, who are not dealing with time-oriented data, are more familiar with the simple line plot. The simple line plot can be understood by a wide range of people. The cycle plot is a special technique and requires specific know-how.

The most impressive feature of the prototype was the drawing functionality, which provides the possibility to simply draw a trend. It is a very simple and fast way to achieve an interesting result and makes it possible to configure complex trends. The conventional way of defining the trend with simple widgets and input values can be very exhausting. When using an implementation of this software design it is preferred to use the pencil for drawing.

The possibility of including more than one variable is very important, but requires additional tooling. The solution of handling variables with different scale factors must not lie in the responsibility of the user, as humans tend to make mistakes. As calculating the scale factor for each variable, there can happen several mistakes. These mistakes could be avoided by the software itself. Furthermore, the visualization suffers due to mistakes by the user. Because of the situation that the visual output need not to provide a good overview with multiple variables, the importance of better tooling becomes more illustrated.

The situation of many configuration possibilities was solved by showing and hiding the different parts above the plot. Many other problems solved this problem by using dialogs. Microsoft Word uses menu and toolbar items, which opens a simple dialog. When using dialogs there is no problem of too little space, as it is the situation in the prototype. As the configuration should be applied initially to the plot of this design, a dialog must not be modal. [4] Then it is possible to configure and work on the plot simultaneously, so that it is not required to close the dialog.

For a better usability the software should provide more tool tips. There is often the situation that the user requires additional information for a field, text box or a whole panel. Short labels often do not provide enough information to understand its functionality. Furthermore, each configuration part could provide a button for getting help or additional information. It is not required to provide a global documentation of the whole software (as documentation is not read very often by many users). Small hints and help texts would help the user to get familiar with most functionality.

All in all the participant liked the prototype and was convinced that the design has much potential. Most of the identified problems can be solved by adding additional features and new

[4] A modal window is positioned to the front. The main window cannot be used, as long as the modal window is open.

functionality. Spending more time into the implementation and essential requirements could result into a valuable product.

6.3.3 Findings from the User Study

Concluding out of the user evaluation both user mentioned positive and negative aspects, which were identified during the evaluation.

6.3.3.1 Paint Functionality

The most impressive feature of this software design is the drawing functionality, which provides the possibility to simply draw a trend on a line plot and then receive sample data. Both participants had no problems to understand the meaning and the possibilities, which are provided by this feature. The second user identified this feature very early and started to use it before it was required. The reason was situated in the chosen icon. A pencil is a common icon in many other software solutions and all these solutions use this icon to indicate something to draw, depending to the task. Especially graphic programs provide a pencil to draw something freehand, with the same usage like in the prototype. So the recognition of the used image was sufficient and did not require further description.

Because of the situation that drawing a line is only possible from left to right, both users were surprised about this limitation. Often users are annoyed about restrictions, but not in this case. Drawing a line on a time line is normally done from left to right, as in the prototype. The other way round is illogical and not intuitive. How should a line be interpreted if it does not continue to the right? So both participants mentioned that this limitation does not hinder the usability. Instead it leads the user to the correct usage. So this situation illustrates that this limitation is useful and perhaps necessary.

After drawing a line on the line plot value ranges are created. Value ranges are the base for the generation process, as they define lower and upper limits for possible values. The users did not expect them after drawing the trend. On the other side they had no information what else they would have expected. This illustrated the situation when using new or unknown user interface functionality. They were not sure about the behavior and mentioned that they would need further experience in creating sample data. So it stays unclear if this behavior is a positive or negative aspect. In that case a larger set of participants would have been needed. Long-time evaluations would also bring more information into this situation, because then the users would have experience and may identify benefits or weaknesses.

6.3.3.2 Value Ranges

The usage of value ranges are the core concept for defining the trend for the generation of time-oriented data. They can be added manually by menu items or automatically by drawing the trend. The forms of the value ranges are similar to box plot.[5] A box plot divides its data into its quartiles, as illustrated in Figure 6.22. It shows the highest and the lowest value with a short horizontal line, equal to the value ranges of this design. But the recognition was not given, as

[5] Diagram for grouping numerical data into quartiles

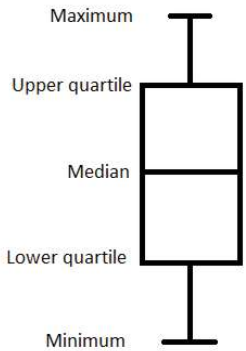

Figure 6.22: Box plot

Figure 6.23: Global value range as alternative for single value ranges

both participants did not know what they illustrate. Even after the first generation of data they were not sure how to interpret them. This problem could lie in the situation that both do not have much experience in statistics. As box plots are often used in statistics, statisticians and people working with statistic would have better recognized the value ranges in their meaning. Furthermore, these people would expect real box plots.

Because of the bad recognition of the meaning, there is the question, if another presentation for the upper and the lower limit would have been more useful. Another possibility would have been displaying the limits over the whole plot, as illustrated in Figure 6.23. Instead of several single value ranges, a global value range is displayed. This global range has the advantage that the borders are always visible and the space between single value ranges does not stay undefined on the plot. On the other side additional content is placed on the plot and may overload the line plot much faster.

The prototype requires at least two defined value ranges for generation, because it requires

a start and an end point of time. Both participants understood this behavior and interpreted it as logical. On the other side the first participant had problems to generate anything in the first task. He was surprised that nothing happens with only one single value range. When he identified the situation, he did not claim about the requirement but about a missing hint. In this situation it would have been better if the user gets informed that two value ranges are required. As both users said that the requirements of two value ranges are understandable, this need not to be changed and only requires some hints. This would not occur with a global value range, as illustrated in Figure 6.23. There we always have a start and an end point and the user would not stumble over this problem.

Another problem was illustrated in Figure 6.19. There we can see that to many value ranges overload the plot. The user has to hide all value ranges to see the generated output. In this situation also the global value range would be a possible solution.

Summarizing the problems of the value ranges it seems that another form of visualizing upper and lower limits for the generation would have been better. The global value range could solve the problems, which occurred with the single value ranges. This has to be evaluated in future work. There would be the question, if the single value ranges or a global value range is preferred for the configuration of upper and lower limits on a line plot.

6.3.3.3 Cycle Plot

The cycle plot is a special technique to visualize time-oriented data. This form of technique has the advantage that a trend for a given interval can be visualized. By example a trend for a specific day of the week is visualized in detail and provides the possibility to identify patterns and other interesting details, which cannot be identified with a line plot.

Experts of time-oriented data are familiar with this visualization technique. But people without knowledge about this technique have problems to get familiar with. This was shown by the user study, as both participants had problems to identify the functionality and the meaning of the visualization. Both users mentioned that they would need more information about this visualization technique. Even samples would have been nice but they would have required much time, till they understand this technique. People, who are not involve into the context of time-oriented data and the cycle plot, are not able to interact with this technique. They require detail hints, examples and documentation, which explains all details.

As expected before both participants were not able to use the cycle plot to generate data by using the prototype. They mentioned that they had not enough information about the context and how this technique works, what is presented and how to interpret the visualization. This leads to the conclusion that the cycle plot is not usable to non-experts of time-oriented data. The interaction for the data generation is another complex step, too much for laymans in this context.

Another weakness for the cycle plot was the missing connection to the simple line plot, when changing from line plot view to cycle plot view. Both participants expected that they would see something, the generated data displayed on the simple line plot and translated to the cycle plot. If this link would have been existing, the functionality of the cycle plot would haven been a bit more transparent. Nevertheless the cycle plot must always display the same data as presented in the simple line plot. There must not be any difference. It should not matter if you change

the view from line plot to cycle plot or the other way round. Both ways must work and has to visualize always the same data in a different form and technique.

The weakness of synchronization is also a problem when introducing additional visualization techniques in the future. All visualization techniques have to show the same data. Chancing to different visualizations is the interesting part of this design and provides many possibilities.

6.3.3.4 Show and Hide Configuration Parts

The prototype provided the possibility to show and hide configuration parts, which are required for the configuration of the generation process. These configuration parts should not require much space, as the plots and the visualization techniques should get as much space as possible all the time. So I tried to keep all configuration parts very small.

Reducing the required space of all parts to a minimum leaded to the problem that there was not much space to arrange controls and information in many different ways. Because of this problem the labels were not sufficient to explain the meaning of all controls. The naming was too short or too inaccurate, so the users had problems to understand how to use the configuration parts and what they really do.

Other software solutions solved this problem by using dialog windows, where the user is able to configure all details. Using dialog windows has the advantage that you need not to keep it as small as possible. There is the possibility to provide extended labeling and more detailed naming, so the user can identify the meaning and the functionality more easily. Additionally the user is able to move the window where he/she wants to. The position is not limited as when using a fixed position somewhere on the main screen.

The usage of dialog windows seem to be a better solution for this software design, because the visualization receives much more space and it is not necessary to keep the configuration as small as possible. The only necessity is to fade in the dialog windows. This can be done by using menu or toolbar items. Most software applications provide a menu and a toolbar, where each item triggers an action or opens a dialog.

6.3.3.5 Missing Help and Tool Tips

The users claimed of missing help and tool tips. They often had the problem that they did not know the exact behavior and functionality of some features. Some tool tips were available but often not sufficient. In this case they only had the possibility to try out the features. Users always try out unknown features, but with additional documentation and help content the functionality becomes much more clearly and does not require much time to experiment.

The prototype provided tool tips for many components, but most time they did not provide enough information. Such tool tips must provide enough information for the user. In the case of the prototype the tool tips are very short, apparently too short. The tool tips should provide more information. Longer phrases or even whole sentences are possible and should be considered, before the tool tips do not provide enough help.

All configuration parts contain a specific set of common functionalities and summarize them with a short term. The name of some parts was not concise enough, so the user did not know the meaning and the functionalities provided. The user mentioned that they would expect a kind

of help button, which provides additional help and information. Furthermore, each part should also have a short description. This description should be placed on top of each part and should contain an explanation and its contained features.

6.3.3.6 Handling of Multiple Variables

The prototype provide the possibility to handle multiple variables. There is always one selected variable, which can be edited. When editing other variables, they have to be selected. The current selected variable is always displayed by the variable configuration part.

Both participants had no problems to handle multiple variables. The system of one selected variable seems to be intuitive, as they had no problems to switch between the variables. They mentioned that displaying the currently selected variable is important. The prototype provided enough feedback to the user.

The handling of variables with different scale factors is not supported by the prototype. The second participant claimed about the missing feature. He said that the handling with multiple features is only useful, if the software is able to manage different scale factors for each variable. The user should not have to make manual calculations. Humans are making mistakes and these mistakes can be avoided by the software. Manual calculations require much time, which can be reduced by tooling.

Another problem of the different scale factors is situated in the visualization. The common visualization of variables with different scale factors makes no scene, if there is always a variable which is not visible. The correlation between variables can only be visualized, if the variables have a common upper and lower limit on the display.

Concluding the handling of a common scale factor is required and should be provided in future work. In this case each variable requires an upper and a lower limit. The prototype has one upper and lower limit for all variables. So when adding a common scale factor to this software design, the value limits must be defined for each variable. The common scale factor has to be calculated by the software. Then the visualization should be able to provide a good overview of all variables, even if there are extremely different value limits.

6.4 Discussion

The evaluation of the implemented prototype contains of three parts: an expert inspection, a user evaluation of two non-experts and usage scenarios. For all parts of the evaluation a set of predefined tasks was available, which should be solved by the users. While the usage scenario only describes the way of usage for the given tasks, the user study and the expert inspection provide further details on usability and intuitiveness. Answers on the main and the sub research questions are expected.

CHAPTER 7

Conclusion

7.1 Main Research Question

What is an appropriate solution to create synthetic time-oriented data by using visualization techniques for input parametrization?

The main research asks for a software design, combining visual techniques, the aspect of time and the generation of synthetic data within one solution. The appropriate solution is represented by the implemented prototype, representing a possible approach and software implementation. It is implemented in Java, uses the visualization framework TimeBench [54] to visualize data and uses the Java Random Engine to generate data, in this case time-oriented data. The design is not dependent to a specific library, programming language or system. The prototype represents a functional implementation, based in the design.

The simplest form to generate time-oriented data with visual techniques is to define the trend manually on a plot. The prototype provides value ranges to define the borders for possible values. The user positions them directly on the plot, defines the minimum and the maximum value and then generates some random values, which are in the range. A range also contains a distribution, which define the probability for the values within the range. The expert and the non-experts required time to identify the exact behavior of that feature in the prototype, but had no problems after some time.

Another possibility to define value ranges was provided by mouse interactions. The prototype offers a drawing functionality to draw the trend in form of a line. The exact generated values are all positioned around to the drawn line. The prototype interprets the drawn line in form of generated value ranges, which can then be edited manually. The expert and the non-experts appreciated that possibility, as it provides the possibility to create fast results.

The most common plot for time-oriented data is a line plot, which is easy to understand even for non-experts of time-oriented data. When using a different visualization technique, the non-experts had problems to understand the visualization technique, as they did not know the meaning. The prototype provides the possibility to generate time-oriented data on a cycle

plot. The non-experts were not able to interpret this technique correctly and required additional information and explanations. The expert already knew this technique.

It is conceivable that further visualization techniques for time-oriented data can be used for the generation of time-oriented data. Depending to the properties and complexity of each technique non-experts would have problems to understand the technique and even work with it. Experts would probably not have this problem, if they already know a technique.

As it can be seen two major components define, how a software solution for visual data generator on time-oriented data can be designed:

- The definition of the input, how borders and limitations are defined.
- The used techniques to visualize data graphically.

Each combination of input definition and visualization technique are conceivable for a software solution.

7.2 Sub Research Question

What is an appropriate solution to draft a user interface design that experts and non-experts in time-oriented data are able to work, independent to introduction and user manuals?

The sub research question deals with the usability of the software design. The usability is an indicator for the quality of a software solution and was measured against the implemented prototype in form of a user study and an expert inspection. For non-experts the user study is of interest, while the expert inspection provides answers on the expert's view.

During the expert inspection the supervisor Alexander Rind experimented with the implemented prototype and solved the predefined tasks, which were defined for the inspection and the user study. He identified weaknesses in the technical implementation, where exceptions occurred during run-time and inconsistencies could be identified. These problems were not of major interest, as they are based on technical problems and implementation faults.

The non-experts did not claim about major technical problems or inconsistencies. They interpreted the implementation of the prototype as useful and did not have problems to understand the principle of the design.

The concept of using value sectors for the manual definition of a trend is an adequate solution for simple tasks. Experts and non-experts understood its usage after some time. Non-experts had initial difficulties, as they did not identify the principle of a box plot. For inexperienced users in time-oriented data and visualization techniques some better labeling or a short description would have been required for more clarity in the prototype.

Experts and non-experts identified in the drawing concept a good solution for achieving quick results. It is easy to use and intuitive, even for non-experts. For the further processing of the drawing functionality there exist different possibilities and each user may have another preference. For simplicity the prototype only converts the drawn line into a set of value sectors.

The simple line plot is easy to understand for all parties and does not require further detailed explanations. In contrast the cycle plot is hard to understand for non-experts, as they do not

know this specific visualization technique for time-oriented data. Detailed labeling and hints are required so that non-experts are able to get involved into the matter of the cycle plot.

All in all, an implementation of a software design is dependent to a good naming and labeling, so that users are able to understand functionalities in the correct way. In some cases labels and names are not sufficient for the user. Then additional material in form of quick tips or examples (e.g. tutorials) would be helpful. A the end, the expert and the non-experts identified weaknesses and not blocking issues within the design approach. So, the appropriate solution is represented by the implemented prototype, showing strengths and weaknesses in desired design decisions.

7.3 Reflection

After answering the main and sub research question there still exist open issues and tasks which would be interesting to be solved and done. They can be done as part of future work and will be described shortly in the next chapter.

CHAPTER 8

Future Work

The implemented prototype of this master thesis provides a small subset of possibilities, how time-oriented data sets can be generated. It only uses two different visualization techniques and is limited to basic configuration possibilities.

8.1 Additional Visualization Techniques

Such a generator may provide additional visualization techniques, which can be used for the generation of synthetic data. There exist a mass of different techniques for time-oriented data, which may be used for an implementation of such a data generator. Chapter 2.2 of this master thesis provides an overview of different visualization techniques. A generator should be able to switch between each available technique and should visualize the same data set. Therewith not only the generation of synthetic data would be of interest, also different techniques are of interest to visualize and identify patterns in the data set. Relationships and structures can be found and the data set can also be edited to experiment on its properties.

8.2 Generative Data Models

Instead of inventing time-oriented data by using drag-and-drop and drawing features it would be interesting to use generative data models, as described in [58] and [50]. The definition of models is sometimes easier and less time-consuming than manual definitions of generation settings, especially when data models are already defined and well-known. Data sets representing the same generative data model can provide data scaling in any dimensions and can be used to benchmark the scalability of software dealing with time-oriented data. Christoph Schulz et al. say:

> Abstract and formal descriptions are more compact, easier to transfer, and thus leverage availability for experiments, while usually also being more tangible and thus beneficial for analysis. (Quote from [58], p. 1)

The data generator can interpret given generative data models in form of equations. Fixed parameters or variables can be defined and the generator calculates endless number of data sets matching the given generative data model.

8.3 Persist Data Configuration Settings

The prototype of this master thesis does not provide the possibility to persist configuration settings for later purpose. The user has to repeat all adjustments in the software and loses much time. Storing user settings is a pleasant feature for users and provides the possibility of individualization.

Next to the user configuration settings the settings of the input parameters for the data generation could also be stored. The prototype of this master thesis is not able to persist the generation settings, which may be very interesting. The repetitive generation of similar data sets could be supported, as the user is able to close the application, start it again and continue his work without difference. Such exportable input definitions could also be interesting for sharing the settings instead of the generated data sets.

8.4 Further Inspection of Drawing Feature

The drawing feature received positive feedback from the user study and the expert inspection. The participants used the mouse for drawing the trend and were able to receive fast results. Manual adjustments were used for fine-tuning the input parameters.

The prototype of the master thesis provided only one way of using a drawing board for the input definition (drawing a trend in form of a line). It is also conceivable that other input parameters could be defined by using the mouse. For example the trend could be split into a general and the periodic trend, where the user has to draw two different lines. The distribution could be drawn in form of a probability density function, as it could be done in PCDC [15]. Or the correlation between two variables could be drawn on a chart.

The usability and effectiveness of such drawing features should be evaluated in more detail. The evaluation of this master thesis shows that a small set of users prefer the possibility to draw the input, instead of defining it manually. A larger set of participants is required to receive a more meaningful result in form of statistical evaluations. Further investigations in any form of direct drawing functionalities are of interest and could provide new innovative software designs.

8.5 Value Range Definition

The prototype used value range definitions to define the upper and lower limits for generating values. Such a value range looks similar to a box plot without the upper and lower quartile. From the usability point of view the usage of box plot similar range definitions are not very clear for non-experts. Other forms of value ranges may improve the situation and could be evaluated in future work.

One possibility is the usage of real box plots. Box plots contain basic information of the value distribution. When using real box plots, including the median, upper and lower quartile, the user has a visual feedback of the distribution directly on the plot.

Another way to solve the value range definition is the usage of a universal value range, as demonstrated in Figure 6.23. The borders of the range are visible for the whole area and the user is able to alter the borders, e.g. by dragging the border with the mouse to another position.

A design of such a generator may offer more than only one way of value range definition. The user can choose between different techniques. More techniques offer more possibilities to all users and the starting problems for the non-experts could be reduced.

The advantages and disadvantages of different value range definitions could also be inspected by another work. Such a work lists all known value range techniques, describes its benefits and weaknesses and evaluates, which technique is preferred for which tasks and usage scenarios.

8.6 Plug-in Mechanism

A software design for a data generator on time-oriented data has a great variety of possible features. Different visualization techniques, import and export of data or drawing the trend for the generation contains a huge set of possibilities. Such a data generator could be extendable by adding new features in form of plug-ins. Such plug-ins could introduce new visualization techniques, analytical algorithms or other useful additions. Such a generator requires an extendable design, which could be solved in future work.

A possible solution could be the usage of OSGi,[1] a technology to modularize Java applications. The Java development platform Eclipse[2] uses OSGi to support the development of OSGi based applications. An extendable data generator on time-oriented data could be based on the Eclipse platform and could then be the base for new visualization techniques and other technical innovations. The visualization framework TimeBench [54] uses Swing as GUI technology, while Eclipse is based on the Standard Widget Toolkit (SWT). A combination of Eclipse and TimeBench is nevertheless possible, as SWT is able to use Swing components.

8.7 User Study with Larger Set of Experts and Non-Experts

This master thesis evaluated the implemented prototype with only one expert and two non-experts. An evaluation with a larger set would have required higher effort, which would have exceed the scope of this thesis. A large inspection on an advanced prototype should be done.

A larger inspection involving a higher set of participants (i.e. ten or more participants) containing experts in time-oriented data and non-experts can be done. The larger amount of participants provides the possibility to do a statistical evaluation, e.g. accuracy and speed of solving predefined tasks. Different usage and handling of the prototype between experts and

[1] `http://www.osgi.org/Main/HomePage` accessed 2019-11-04
[2] `http://www.eclipse.org` accessed 2019-11-04

non-experts could be evaluated and identified. Evaluations on specific design details (e.g. feedback of drawing feature, usage of value sector, etc.) could be done and summarized by future work.

Bibliography

[1] Visual.ly. http://visual.ly/. Accessed: 2019-11-04.

[2] Chad A. Steed, William Halsey, Ryan Dehoff, Sean L. Yoder, Vicent Paquit, and Sarah Powers. Falcon: Visual analysis of large, irregularly sampled, and multivariate time series data in additive manufacturing. *Computers & Graphics*, 63:50–64, 02 2017.

[3] Chadia Abras, Diane Maloney-Krichmar, and Jenny Preece. User-centered design. *William Sims Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications*, 37(4):445–56, 2004.

[4] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of Time-Oriented Data*. Springer, 2011.

[5] G. Albuquerque, T. Lowe, and M. Magnor. Synthetic generation of high-dimensional datasets. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2317–2324, 2011.

[6] Animated Data, Brighton, UK. F1 timeline. https://charts.animateddata.co.uk/f1/. Accessed: 2019-11-04.

[7] Machine Learning Visualization Lab at Decisive Analytics Corporation. D3 timeseries with brush. http://mlvl.github.io/timeseries/. Accessed: 2019-11-04.

[8] David Beard, Murugappan Palaniappan, Alan Humm, David Banks, Anil Nair, and Yen-Ping Shan. A visual calendar for scheduling group meetings. In *Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, CSCW '90, pages 279–290, New York, NY, USA, 1990. ACM.

[9] Nicolas Garcia Belmonte. Javascript infovis toolkit. http://philogb.github.io/jit/. Accessed: 2019-11-04.

[10] R. Daniel Bergeron, Daniel A. Keim, and Ronald M. Pickett. Test data sets for evaluating data visualization techniques. In Georges G. Grinstein and Haim Levkowitz, editors, *Perceptual Issues in Visualization*, IFIP Series on Computer Graphics, pages 9–21. Springer, 1995.

[11] S. Bista and M.L. Pack. Real-time massive data simulation visualization. In *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 543–548, 2008.

[12] Max Born. *Die Relativitätstheorie Einsteins*. Springer, 7. edition, 2003.

[13] M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1121–1128, 2009.

[14] Mike Bostock. Data-driven documents. `http://d3js.org/`. Accessed: 2019-11-04.

[15] Sebastian Bremm, Tatiana von Landesberger, Thomas Heß, and Dieter W. Fellner. PCDC - on the highway to data - a tool for the fast generation of large synthetic data sets. In *EuroVA International Workshop on Visual Analytics*, 2012.

[16] Stuart K. Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[17] William S. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[18] Sunny Consolvo, Larry Arnstein, and B. Robert Franza. User study techniques in the design and evaluation of a ubicomp environment. In *Proceedings of the 4th International Conference on Ubiquitous Computing*, UbiComp '02, pages 73–90, London, UK, 2002. Springer-Verlag.

[19] M. Cordeil, T. Dwyer, K. Klein, B. Laha, K. Marriott, and B. H. Thomas. Immersive collaborative analysis of network connectivity: Cave-style or head-mounted display? *IEEE Transactions on Visualization and Computer Graphics*, 23(1):441–450, Jan 2017.

[20] P. Craig and N. Roa-Seiler. A vertical timeline visualization for the exploratory analysis of dialogue data. In *Information Visualisation (IV), 2012 16th International Conference on*, pages 68–73, 2012.

[21] B. Engin, M. Cetinkaya, E. Ayiter, M. Germen, and S. Balcisoy. Maestro: Design challenges for a group calendar. In *Information Visualisation, 2008. IV '08. 12th International Conference*, pages 491–496, 2008.

[22] Jean-Daniel Fekete. The infovis toolkit. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pages 167–174, Washington, DC, USA, 2004. IEEE Computer Society.

[23] S. Fernandes Silva and T. Catarci. Visualization of linear time-oriented data: a survey. In *Web Information Systems Engineering, 2000. Proceedings of the First International Conference on*, volume 1, pages 310–319, 2000.

[24] Asbjørn Følstad, Effie Lai-Chong Law, and Kasper Hornbæk. Outliers in usability testing: How to treat usability problems found for only one test participant? In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pages 257–260, New York, NY, USA, 2012. ACM.

[25] Datawrapper GmbH. Datawrapper. https://www.datawrapper.de/. Accessed: 2019-11-04.

[26] Robert L. Harris. *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, Inc., New York, NY, USA, 1999.

[27] Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 421–430, New York, NY, USA, 2005. ACM.

[28] Ichiro Hirata and Toshiki Yamaoka. A logical design method for user interface using gui design patterns. In Masaaki Kurosu, editor, *Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments*, volume 8004 of *Lecture Notes in Computer Science*, pages 361–370. Springer Berlin Heidelberg, 2013.

[29] Andreas Holzinger. Usability engineering methods for software developers. *Commun. ACM*, 48:71–74, 01 2005.

[30] IBM. Many eyes. https://www.boostlabs.com/ibms-many-eyes-online-data-visualization-tool/. Accessed: 2019-11-04.

[31] Inc. Infragistics. Quince. http://quince.infragistics.com/. Accessed: 2013-11-10.

[32] T. Isenberg, P. Isenberg, Jian Chen, M. Sedlmair, and T. Möller. A systematic review on the practice of evaluating visualization. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2818–2827, Dec 2013.

[33] Scott Sievert Jake Vanderplas. Altair. https://altair-viz.github.io/. Accessed: 2019-11-04.

[34] Jeff Johnson. *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2010.

[35] Benjamin Keen. generatedata.com. http://www.generatedata.com/. Accessed: 2019-11-04.

[36] Andrey Krekhov and Jens C. Kruger. Deadeye: A novel preattentive visualization technique based on dichoptic presentation. *IEEE Transactions on Visualization and Computer Graphics*, 25:936–945, 2018.

[37] S. Kriglstein, M. Pohl, and C. Stachl. Animation for time-oriented data: An overview of empirical research. In *2012 16th International Conference on Information Visualisation*, pages 30–35, July 2012.

[38] B. C. Kwon, H. Kim, E. Wall, J. Choo, H. Park, and A. Endert. AxiSketcher: Interactive nonlinear axis mapping of visualizations through user drawings. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):221–230, Jan 2017.

[39] H. Lenz. *Universalgeschichte der Zeit*. Marixverlag, 2013.

[40] G. A. Lewis, E. Morris, P. Place, S. Simanta, and D. B. Smith. Requirements engineering for systems of systems. In *2009 3rd Annual IEEE Systems Conference*, pages 247–252, March 2009.

[41] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, pages 173–176, New York, NY, USA, 1991. ACM.

[42] Jock D. Mackinlay, George G. Robertson, and Robert DeLine. Developing calendar visualizers for the information visualizer. In *Proceedings of the 7th annual ACM symposium on User interface software and technology*, UIST '94, pages 109–118, New York, NY, USA, 1994. ACM.

[43] Microsoft. About time. `http://msdn.microsoft.com/en-us/library/windows/desktop/ms724186/`. Accessed: 2019-11-04.

[44] T. Munzner. A nested model for visualization design and validation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):921–928, Nov 2009.

[45] Sir Isaac Newton. *The mathematical principles of natural philosophy*. Knight & Compton, Middle Street, Cloth Fair, 1. edition, 1803.

[46] Jakob Nielsen. Usability inspection methods. In *Conference Companion on Human Factors in Computing Systems*, CHI '94, pages 413–414, New York, NY, USA, 1994. ACM.

[47] Y. Nomura, S. Mihara, and H. Taniguchi. Implementation of a practical calendaring system conforming with ambiguous pattern of recurring tasks. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, pages 1011–1016, 2012.

[48] D.A. Norman. *The Psychology of Everyday Things*. Basic Books, 1988.

[49] Donald A. Norman and Stephen W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986.

[50] Ronald J. Nowling and Jay Vyas. A domain-driven, generative data model for big pet store. *2014 IEEE Fourth International Conference on Big Data and Cloud Computing*, pages 49–55, 2014.

[51] M.Z.A. Obeidat and S.S. Salim. Integrating user interface design guidelines with adaptation techniques to solve usability problems. In *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, volume 1, pages V1–280–V1–284, 2010.

[52] Maruf Rahman. Time series chart(irregular interval) with shared x-axis and zoom. `http://bl.ocks.org/marufbd/7191340`. Accessed: 2019-11-04.

[53] R. G. Raidou, M. Eisemann, M. Breeuwer, E. Eisemann, and A. Vilanova. Orientation-enhanced parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):589–598, Jan 2016.

[54] Alexander Rind, Tim Lammarsch, Wolfgang Aigner, Bilal Alsallakh, and Silvia Miksch. TimeBench: A data model and software library for visual analytics of time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2247–2256, 2013.

[55] B. Saket, A. Srinivasan, E. D. Ragan, and A. Endert. Evaluating interactive graphical encodings for data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1316–1330, March 2018.

[56] A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, Jan 2017.

[57] A. Satyanarayan, R. Russell, J. Hoffswell, and J. Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, Jan 2016.

[58] Christoph Schulz, Arlind Nocaj, Mennatallah El-Assady, Steffen Frey, Marcel Hlawatsch, Michael Hund, Grzegorz Karch, Rudolf Netzel, Christin Schätzle, Miriam Butt, Daniel A. Keim, Thomas Ertl, Ulrik Brandes, and Daniel Weiskopf. Generative data models for validation and evaluation of visualization techniques. In *Proceedings of the Sixth Workshop on Beyond Time and Errors on Novel Evaluation Methods for Visualization*, BELIV '16, pages 112–124, New York, NY, USA, 2016. ACM.

[59] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2634–2643, 2013.

[60] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[61] Benjamin Sintay. Unix time stamp. `http://www.unixtimestamp.com/`. Accessed: 2019-11-04.

[62] Tableau Software. Tableau public. `http://www.tableausoftware.com/public/community`. Accessed: 2019-11-04.

[63] R Project Contribution Team. The R project. http://www.r-project.org/. Accessed: 2019-11-04.

[64] The IEEE and The Open Group. Portable operating system interface. *IEEE Std 1003.1*, 2001.

[65] Christian Tominski and Wolfgang Aigner. The timeviz browser. http://survey.timeviz.net/. Accessed: 2019-11-04.

[66] M. Tory and T. Möller. Evaluating visualizations: do expert reviews work? *Computer Graphics and Applications, IEEE*, 25(5):8–11, Sept 2005.

[67] M. Tory, N. Rober, T. Möller, A. Celler, and M.S. Atkins. 4d space-time techniques: a medical imaging case study. In *Visualization, 2001. VIS '01. Proceedings*, pages 473–592, 2001.

[68] Anders Toxboe. UI patterns. http://ui-patterns.com/. Accessed: 2019-11-04.

[69] I. Tsuda, K. Uchino, and I. Matsui. WorkWare: WWW-based chronological document organizer. In *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, pages 380–385, 1998.

[70] Martijn van Welie. Welie.com. http://welie.com/. Accessed: 2019-11-04.

[71] J.J. van Wijk and E.R. Van Selow. Cluster and calendar based visualization of time series data. In *Information Visualization, 1999. (Info Vis '99) Proceedings. 1999 IEEE Symposium on*, pages 4–9, 140, 1999.

[72] Reinhard Viertl. *Einführung in die Stochastik*. Springer, 3rd edition, 2003.

[73] Bing Wang, Puripant Ruchikachorn, and Klaus Mueller. Sketchpadn-d: Wydiwyg sculpting and editing in high-dimensional space. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2060–2069, 2013.

[74] S. Yuan, A. Tabard, and W. Mackay. Streamliner: A general-purpose interactive course-visualization tool. In *Knowledge Acquisition and Modeling Workshop, 2008. KAM Workshop 2008. IEEE International Symposium on*, pages 915–919, 2008.

[75] S. Zollmann, D. Kalkofen, C. Hoppe, S. Kluckner, H. Bischof, and G. Reitmayr. Interactive 4d overview and detail visualization in augmented reality. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 167–176, 2012.