ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

Design and Implementation of Enhancements for a Knitting Machine to Enable Hybrid Fabrication

Philipp Kügler

Technical Report – STL-TR-2019-02 – ISSN 2364-7167

stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Philipp Kügler: Design and Implementation of Enhancements for a Knitting Machine to Enable Hybrid Fabrication
Technical report id: STL-TR-2019-02

First published: October 2019
Last revision: September 2019
Internal review: Lora Oehlberg, André Miede

For the most recent version of this report see: https://stl.htwsaar.de/

Title image source: Philipp Kügler

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
https://stl.htwsaar.de

ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Bachelor-Thesis

zur Erlangung des akademischen Grades
Bachelor of Science (B. Sc.)
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Design and Implementation of Enhancements for a Knitting Machine to enable Hybrid Fabrication

vorgelegt von
Philipp Kügler

betreut und begutachtet von
Prof. Dr.-Ing. André Miede
Prof. Dr. Lora Oehlberg

Calgary, 20. September 2019

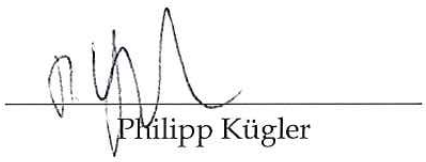
Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit (bei einer Gruppenarbeit: den entsprechend gekennzeichneten Anteil der Arbeit) selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend"zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Calgary, 20. September 2019



Philipp Kügler

Abstract

Physical data representations, or physicalizations, offer alternative ways to explore and understand data. For example, a physicalization can encode data through form, materiality, or texture. During manually crafting a physicalization the maker already discovers the underlying data. People generally author physicalizations through hand fabrication techniques or digital fabrication machines. However, in hybrid fabrication the user performs manual fabrication in collaboration with a digital fabrication machine throughout the whole process of creation, resulting in a meaningful, reflective experience for the user. In this work, I created a hybrid fabrication system for knit data physicalization.

At the beginning I had a look on how fabricating with a knitting machine works. In addition, I explain the All Yarns are Beautiful (AYAB) project, which aims at controlling knitting machines with a computer. I used a paper prototype session to find out what knitting techniques would a potential maker use to create a knit physicalization and to explore the design space of knit physicalization for a simple object like a beer coozy. This helped me to develop concepts of features for a new system, which I called K1M1. K1M1 covers the whole process of turning a data set into a pattern as well as using this pattern to knit a data physicalization. I implemented additional modules for the existing software from the AYAB project. This software is the control element of the hybrid fabrication system. Eventually, I used this hybrid fabrication system to design and create three data physicalization prototypes to demonstrate the functionality of the enhanced software.

The system can assist the user with designing an usable pattern and is able to integrate the user in the creation process. I also propose concepts for software enhancements as future work.

There are two ways to write error-free code;
only the third one works.

— Alan J. Perlis [20]

Acknowledgements

I thank my reviewers, and especially Anne Woelk, for their detailed comments that helped. Moreover, thanks to the participants of my studies, their feedback was crucial for my work. Furthermore, I thank Andre Miede and Lora Oehlberg, my supervisors who supported me and the whole iLab team for giving me a good time there.

Inhaltsverzeichnis

1 Introduction

1.1 Motivation

Since the early days of mankind, people used different techniques to make data accessible and therefore easier to understand. For example, the idea of coordinates was already used by ancient Egyptian surveyors in laying out towns, Greek people who used pebbles and urns as a voting system [16]. These attempts to represent data are called data physicalizations or to be more specific, physical data representations, because of their analogue nature. Many things have changed by now, with the rise of computer systems and the occurrence of digitization, there are new ways to represent data such as digitally plotting charts and diagrams. Using computers and their software applications for the purpose of presenting data in a graphical or pictorial way is called digital data visualization. The visible presentation of data has its benefits in many different areas. Business or research environments are common areas where digital data visualization is frequently used. For companies it is vitally important to understand and predict customer's behavior for running a successful business. Recognizing market trends and coherence enables managers to develop business strategies. It is in the nature of research to collect and analyze large volumes of data. In order get to the core of the data and make it more usable, digital data visualization is indispensable, especially when data is presented to non-researchers. Another field where the continuous collection of data becomes more and more common is the personal environment. Fitness trackers and other wearables, as well as smart phones, are able to gather a variety of data about their owners and their environments. In this context, the visualization of data can be used for self-tracking and self-reflection purposes.

Nevertheless, despite the immense value of digital data visualization, this kind of representing data can reach a limitation which can be overcome by physical data representation [12]. For example, children are not very interested in charts or diagrams, but if they have something they can touch and play with, they will explore it. Moreover, this kind of data representations offers the possibility for blind people to explore the represented data by themselves instead of having to rely on someone describing or explaining it. Furthermore, Sheelagh Carpendale et al.[23] showed that crafting physical data representations by hand boosts data understanding, creates a valuable token and also, depending on the data, fosters self-reflection. Crafting an object by hand is called manual fabrication. In contrast, digital fabrication describes a computer-controlled process that uses digital 3D-designs to manufacture objects. Hybrid fabrication combines both digital and manual fabrication aspects. The problem with hybrid fabrication is having a machine system that interacts with the user, instead of the user just telling the machine what to do or the other way around. On the one hand, if the user just tells the machine what to do at the beginning, without the possibility of making changes during the fabrication, the user will become a passive observer without any deeper relation to the outcome. On the other hand, if the machine just tells the user what to do, there is a risk that people only concentrate on the instructions instead of the data. Additionally, it could be that the instructions are not clear enough, so the user will fail and be frustrated. In order to allow systems to interact with users, it is necessary to improve currently existing systems. In the case at hand that means to augment the knitting machine KH-930E. This knitting machine is located in the FabLab

at the University of Calgary. The goal of this project is to turn this knitting machine into a hybrid fabrication system, which can knit physical data representations and make this process a meaningful experience for the user. This work is an approach to exhaust the full potential and benefits of data physicalization and hybrid fabrication.

There are already a couple of projects aiming at controlling knitting machines using a computer. One that focuses on the Brother KH-9xx range of knitting machines is the All Yarns are Beautiful (AYAB) project. The project team developed a hardware interface that replaces the original built-in computer, in order to connect the machine with an external computer via USB. The knitting machine at the University of Calgary is equipped with such an interface. Furthermore, the AYAB project team wrote an open source software to interact with their hardware. The problem with the current software is, that it does not provide a function to turn a data set into a pattern. The software also offers too little assistance during knitting, but this is necessary for a hybrid fabrication system. However, the AYAB software constitutes the starting point for this work.

1.2 Task and Purpose

The Brother KH-9xx range is popular among hackers. These knitting machines, which have been designed for hobby knitters, are an interesting target for hackers, due to their simple construction and their variety of different functions. The Brother KH-930E at the FabLab is a single-bed knitting machine with a Knit-Carriage and a Lace-Carriage. There is a paper manual called *How to use the Knitting Machine* and the knitting machine can be connected to a computer, due to the AYAB interface. Nevertheless, handling the knitting machine can be challenging, as it can easily lead to frustration, due to its lack of usability. Still, it is a valuable approach to knit physical data representations. In order to make it easier for novice users to get started with data representation,the knitting machine needs to be transformed into a collaborative hybrid fabrication system. The entire system should benefit from both the strengths of the user and the strengths of the machine. The machine should cover monotone and repetitive work, while the creative part is reserved for the user. The user should, also conduct the work that the machine is not able to perform. For instance, decreasing or increasing stitches has to be performed manually, but the machine should keep track when to perform these tasks and communicate this to the user. Additionally, the system needs to provide a possibility to turn data sets into patterns. To create such a hybrid fabrication system, it is necessary to break the work down into more accomplishable tasks.

One of the tasks is to provide the user with a manual on how to handle the different process steps that must be performed manually. With a digital manual it is possible to provide even more detailed information than the printed manual could give. The information is no longer distributed, but concentrated at one point, the software. Furthermore, the printed manual starts to tatter, which cannot happen to the digital one. Another task is to make the machine keep track of the rows, which are already knitted. There was a mechanical row counter with the built-in computer, but by replacing it, the row counter is also gone. One approach to restore it is to add an IR-camera to the system, which can count as soon as the Knit-Carriage changes its direction. The IR-camera can also support solutions with other challenges like telling the user about false settings. It is difficult to forecast the real length and width of the fabrication, due to different settings of tension, different yarn thicknesses and the different shapes a fabrication can have. To knit the right size and dimensions of a data representation, a calculator is needed, which calculates the amount of rows and stitches. Moreover, a pattern maker needs to be implemented, which is able to help transfer the data into a pattern, which can be knitted. This pattern maker

needs to provide the user with the opportunity to mark points in the pattern where manual tasks have to be performed. That way the user gains a better understanding on how the finished knitting may look like.

All these enhancements are important steps to achieve a meaningful experience throughout the fabrication process even for inexperienced users. When a user wants visualize data with knitting just once in a lifetime, it would be disproportionate to first require him master all techniques and settings the system has to offer in order to achieve his goal. If anything, it should be possible to fabricate with just the minimum knowledge about knitting, only by having a vague idea what the end product should look like. To reach this goal, this work focuses on the possibilities of modifying a knitting machine in a way that allows interaction with a human being and data in order to create a valuable output as a physical data representation. This also means, that the software should not be limited to one operating system, instead it should work at any computer which is able to connect with the knitting machine. As the research in this whole area is still on going and this work has a time limitation, the software should be extendable to build in more enhancements or improvements.

1.3 Structure

This work is devided into eight different chapters. Firstly, the key terms and concepts are elaborated in the chapter *Background*. It is subdivided in different sections, each providing a part of the basic knowledge, which is needed to understand this work. The first section is about the knitting machine Brother KH-930, which is used in this project. Then I discussed the AYAB project. The next sections are about the foundations of data representation and the foundations of hybrid fabrication. The closure is a small section about the different evaluation methods I have used.

In the *Requirements Analysis* chapter, I discussed why I used the spiral model in this work. Then I had a look at the current state of the AYAB software, the relationship between the software, the knitting machine and the user were also discussed here. The third section is about the worked out use cases from the user´s point of view. Finally, I explain the paper prototyping study, which I used to identify requirements that are listed in the last section.

A rigorous research of the related work can be found in the chapter *State of the Art*. The gap between the identified requirements from the previous chapter and the presented projects is also addressed here. First of all, I summarize two projects aimed at improving the work with knitting machines. Then I continue discuss hybrid fabrication systems and the lessons I that draw from them. At the end of this chapter, I present tools that can be used to convert data into data representations.

The concept of the functions that I implemented in order to bridge the analyzed gap from the previous chapter, can be found in the *Conception* chapter. In the first two sections I elucidate how I planned to enable the converting of data into a pattern and how this pattern is used to create the hybrid fabrication part of this work. To gain a better understanding how the different parts of the software work together, I created some models, which illustrate this and explained them. GUI drafts of additional windows form the end of this chapter.

The *Implementation* chapter covers how I implemented the developed concepts. I started with describing technologies and frameworks that I used for this work. Furthermore, I put the main focus on idiosyncrasies of Python and PyQt5. The next section is about the system architecture. After that, I discussed the most important functions with samples of the written source code.

In the *Evaluation* chapter, the first section is about how I knitted sample data representations to demonstrate the functionality of the system. After that I discuss technical performances of the system by analyzing the results of various test I ran.

The last content-related chapter is called *Conclusion and Future Work*. The first of the two sections I critically recapitulate this work and its most important aspects. The closing of this chapter is a forecast about the work that can be tackled in the future.

2 Background

In this chapter I explain the basis knowledge that is necessary to fully understand my work. First I discuss relevant knitting techniques and how they are performed by the knitting machine. Then I have a look how the AYAB project enables a user to control a knitting machine with a computer. After these technical fundamentals, I outline the theoretical foundations of data visualization and hybrid fabrication.

2.1 Knitting

During knitting the maker uses needles create loops. These loops are called stitches. Creating the first row of stitches is called cast on. Then the maker pulls new stitches through earlier stitches. Old stitches then drop from the active needle. This creates a line or tube. After the last row the maker has to bind off to stop the knitting from unraveling. The backside of knittings is called purl stitch (see Figure 2.2b). It is also possible to increase or decrease stitches while knitting or to knit with multiple colors. In addition, the maker can also create holes in a knitting, when these holes form a pattern, it is called lace knitting. Knitting can be done by both hand or machine.

2.1.1 Knitting Machine

Abbildung 2.1: The knit carriage from the used knitting machine in the fablab at the UofC

The Brother KH-930 is a computer-controlled knitting machine, coming with a lace and a knit carriage (see Figure 2.1). These machines from the 1980´s were meant for domestic use and small-scale production facilities to produce simple garments. Due to their small size, just about 1 meter length and 30 cm wide, they can be placed on regular tables. The KH-930 uses an on board computer to handle its many functions, but these computers can be replaced in order to connect the knitting machine to a external computer via Universal Serial Bus (USB). This is the reason why these machines are popular among hackers and hobby knitters. The knitting machine at hand has just one needle bed, there are also V-bed knitting machines which have two needle beds that are angled toward each other in an inverted V shape. The V-bed knitting machines can knit both plain (or knit) stitch and purl stitch on the same side, while single-bed machines only can knit plain stitch on the one and purl on the other side.

2.1.2 Knitting Machine Techniques

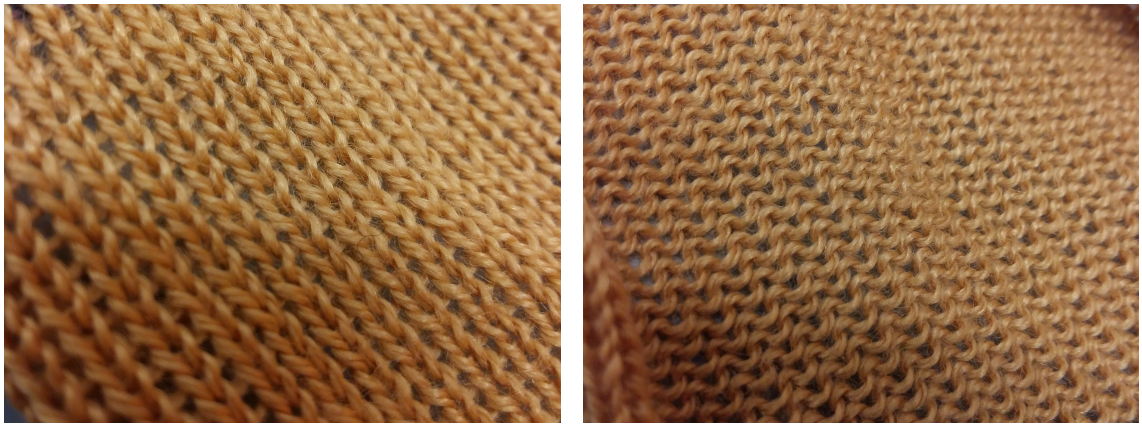
First the user has to activate needles. There are four different positions a needle can have. The position for general knitting purpose is the working position. The more needles are activated, therefore in working position, the wider the knitting will be. Then the user has to arrange the yarn, from the yarn ball over the yarn guide to the yarn feeder of the knit carriage. On the yarn feeder itself, there is a wheel to set the tension of the knitting. If a knitting is knitted with a tension levels from zero to five, its stitches are tight, therefore thinner yarn has to be used, while the tension levels from five to ten are for looser stitches and thicker yarn.

2.1.2.1 Cast On

Knitting always begins with a so-called cast on. It is a set of different technique to knit the first row. Unlike the usual knitting stitches, the stitches in this first row do not depend on earlier stitches. A cast on is a manually process. Some techniques even require hand tools to perform them. After the cast on is done, there is one stitch on every activated needle. To start knitting the user has to move the knit carriage from one side to the other in order to knit one row.

2.1.2.2 Stocking Stitch

When the knit carriage is dragged over an activated needle, it pushes the needle forward, the hook of the needle opens and the needle grabs the yarn, which is provided by the knit carriage´s yarn feeder. Then the knit carriage pulls the needle back again and the yarn in the hook is dragged through the old stitch, adding a new one to the web. This results in alternately knitting a row in plain stitches and knitting a row in purl stitches. This knitting pattern is called stocking stitch. The stocking stitch produces an V on the front side of the knitting (see Figure 2.2a), whereas the back side consists of intertwined loops (see Figure 2.2b). When knitting with the Brother KH-930 the purl side is facing to the user. A drawback of the stocking stitch is that it curls up because of the yarn tension (see Figure 2.3a).



(a) Plain stitch (b) Purl stitch

Abbildung 2.2: Examples of the two stitches

2.1.2.3 Bind Off

After all rows have been knitted, the user has to bind off. Without binding off, the stitches are open loops and will unravel easily. Similar to the cast on, the bind off is a range of techniques that are done manually and also requires hand tools.

2.1.3 Special Techniques

Besides knitting a unicolored rectangle, the knitting machine can also fulfill a range of manual or automatically handled techniques, to give the knitting more variation. I listed the most important techniques in this subsection.

2.1.3.1 Two Color Knitting

The knitting machine is able to knit a given pattern. Older models can use punch cards to input a pattern, the next generation uses on board computers to provide the user with patterns. After enabling the knitting machine to connect with an external computer, it can also use digital two color pictures to input a pattern. The relation from a picture to a pattern is that one pixel from the picture is one stitch in the knitting. For example if a picture is 100 pixels width, the user has to activate 100 needles and every pixel per row is connected to one of the needles. That means, if one pixel is not colored the needle will use the main yarn to fabricate a new stitch. If the pixel is colored, the needle will use the contrast yarn for the new stitch. This results in a knitted picture (see Figure 2.3a). The contrast yarn is also provided by the knit carriage by its second yarn feeder.

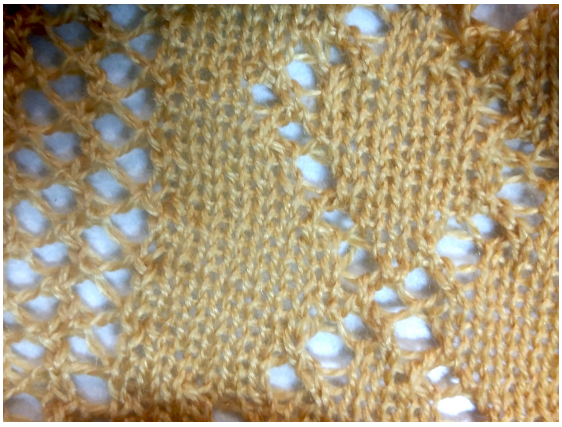
2.1.3.2 Lace Knitting

A pattern is also used for lace knitting, but instead of knitting with a contrast yarn, a hole is produced in the knitting. In order to produce a hole, the stitch from one needle is transferred to another needle next to it. This can be done by hand manipulation as well as by the lace carriage, which is a lot faster. When the lace carriage is used, the knit carriage has to stand on the right side of the knitting and has to be set to plain knitting. After that, the user has to push the lace carriage over the needle bed. When the lace carriage is pushed from left to right, the stitches are transferred to the next needle on the right side. In turn, when the lace carriage is pushed from right to left, the stitches are transferred to the next needle on the left side. The lace carriage has to be used until no more needle is activated, then the knit carriage has to be pushed over the needle bed as many times as the lace carriage has been pushed over. This results in a knitted lace pattern (see Figure 2.3b).

2.1.3.3 Increase/Decrease Stitches

These are also manual tasks. For increasing one stitch, the user simply has to push the needle, next to the already activated needles, to the working position. Increasing more stitches at one time is also possible, but it is more difficult because the user has to fulfill a cast on on the new activated needles. Quite similar to this, decreasing one stitch can be done by transferring the stitch at the edge to another needle next to it. Whereas decreasing more stitches requires fulfilling a bind off for the deactivated needles.

(a) Two color knitting



(b) Lace knitting

Abbildung 2.3: Examples of the two special techniques

2.2 The All Yarns are Beautiful (AYAB) Project

There are already projects with the goal to controlling knitting machines using a computer. One that aims at the Brother KH-9xx range of knitting machines is the AYAB project [17].

2.2.1 Components

The German team build an open source hardware, as well as implemented an open source software. For their hardware module, they used an Arduino microchip and combined it with a custom developed interface. This module replaced the build in control board from the knitting machine. The original control board was responsible for controlling the needles, keeping track of the carriages and counting the knitted rows. The new module now performs these tasks. The board replacement is reversible, that means if the user wants to use the old build in control system, it is possible to do so by switching the boards again. The AYAB module which is used for this work, was build by the Evil Mad Scientists LLC, a small business that designs and produces Do It Yourself (DIY) and open source hardware. They redesigned the AYAB module and sold it in their online shop.

The AYAB team also implemented a firmware, which is written in C++, for the Arduino chip. The hardware interacts through the firmware with the software. The software itself provides a Graphical User Interface (GUI) to interact with the user. Furthermore, the software reproduces functions for editing the pattern, which previously has been supplied by the original control board. The software's user interface is made with Qt or to be more specific, with PyQt5. Qt is an open-source widget toolkit, which can be used to create GUIs as well as non-GUI programs. The GUI is keep as simple as the whole software, it has one main window with a menu bar and buttons on the right side which maintain the most important tasks. The AYAB team designed to guide the user through its functions. This is done by number the steps that have to be performed to start the knitting process. Moreover, these steps are in a vertical order to clarify their sequence. In the biggest widget, the user can see the pattern, which was loaded. During knitting a black bar shows which row is knitted at the moment.

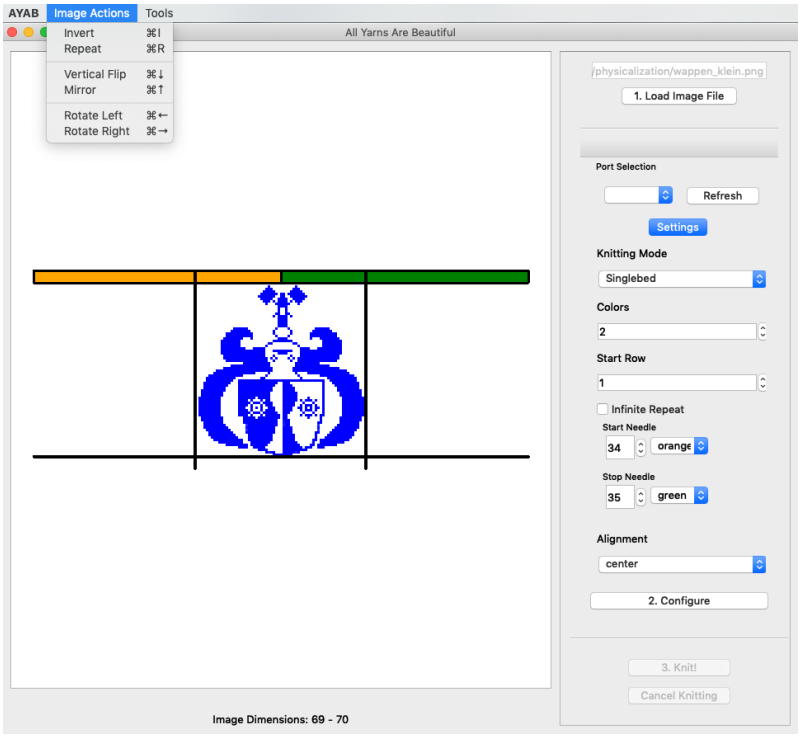


Abbildung 2.4: The GUI of the AYAB Software

2.2.2 Functioning

If the user wants to use special techniques like Two Color Knitting or Lace Knitting, the first step is to load a pattern into the software. A pattern can be a low-resolution picture, which will be loaded from the computer's disk into the software's main window. After the pattern is imported, the user can choose between different options to edit it. The pattern can be inverted, repeated, mirrored or rotated. It is also possible to change with how much needles the user wants to knit, the alignment of the pattern on the needle bed and to set the start row. It can be specified on what kind of knitting machine the user is operating or how many colors the pattern has as well.

After the configurations are set and the start button has been clicked, the software´s Application Programming Interface (API) requests to start a new pattern. Then the software converts the pattern into a bytearray and sends the first line of the pattern to the hardware through the firmware. The control board sets the needles and after the knit carriage is pushed over to the other side of the needle bed, it requests the next line from the software. The software in turn counts the knitted rows, updates the GUI and sends the next line to the hardware. This process will be repeated until the last line is sent. If the user does not manually change the settings of the knitting machine by then, from two color knitting to plain knitting, the last line of the pattern will be knitted repeatedly. However, it is not possible for the user to create a pattern with this software.

2.3 Data Visualization

With the rising amount of data in science, business or even personal environment, it is getting more important to represent it in a way to increase its understanding. A visual representation of one or more data sets is called a data visualization. Data visualization itself has inputs from many different disciplines from computer science and psychology

to graphical design. Aparicio and Costa [1] point out that it is more natural and faster to communicate in images than in written words and numbers. Data visualizations are also understandable despite languages and cultures. Therefore it is important to use it for clearer and more efficient communication.

For the better understanding how a data visualization is designed, Card et al. [4], created a model that shows the different stations from a data set to a data visualization. From *raw data* in its origin form, over *processed data*, where the data is put into tables, into a *visual structure* to the final *visual presentation*. In order to fulfill this transformation, data has to be converted into marks, like points or lines. These marks can have very different appearances, Bertin [2] called these appearances a marks can adapt visual variables. He worked out seven different variable, these are:

position	where the mark is placed in the visualization
size	how big or length the mark is and how often it appears in the visualization
shape	the endless possibilities of which form a mark can have
value	how light or dark the mark is
color	which color it has
orientation	how the mark is aligned in the visualization
texture	the different grains a mark can have

All these can be used to clarify the represented information in an visualization. Besides the almost omnipresent printed or digital data visualizations, there is also the field of data physicalizations.

2.3.1 Data Physicalizations

Jansen et al. [12] defines a data physicalization, or physical data representation, as a physical artifact whose geometry or material properties encode data. This means physicalizations can be very different. For example, on the one hand, there are historical physicalization, like the 5000 year old clay tokens, which represent a jar of oil or a sheep [21]. On the other hand, Kevin Quinn a chief engineer at General Motors, uses Lego bricks to keep track of the progress and problems in the production lines. This is also a form of physical data representation. That shows that physical data representation are offering exciting ways to explore data due to its growing range of different techniques and materials. It can also been seen as a long missing link between art and data visualization [14]. Due to this it is obvious that museums are already start to use data physicalization as a form of infotainment, a medium that informs and entertains at the same time.

Figure 2.7 shows a simple example of a physical data representation, a 3D bar chart built with Lego bricks. The represented data set about alcohol consumption. It is from a workshop called Let's Get Physical [10], a workshop where participants create data physicalizations. Each column represents a different country. The light blue bricks in the front represent the average consumed alcohols of the country in gram. The bars behind showing the ration between the different kinds of beverage that have been consumed. Yellow represents beer, green wine, purple spirits and orange represents a summery of other beverages than these three.

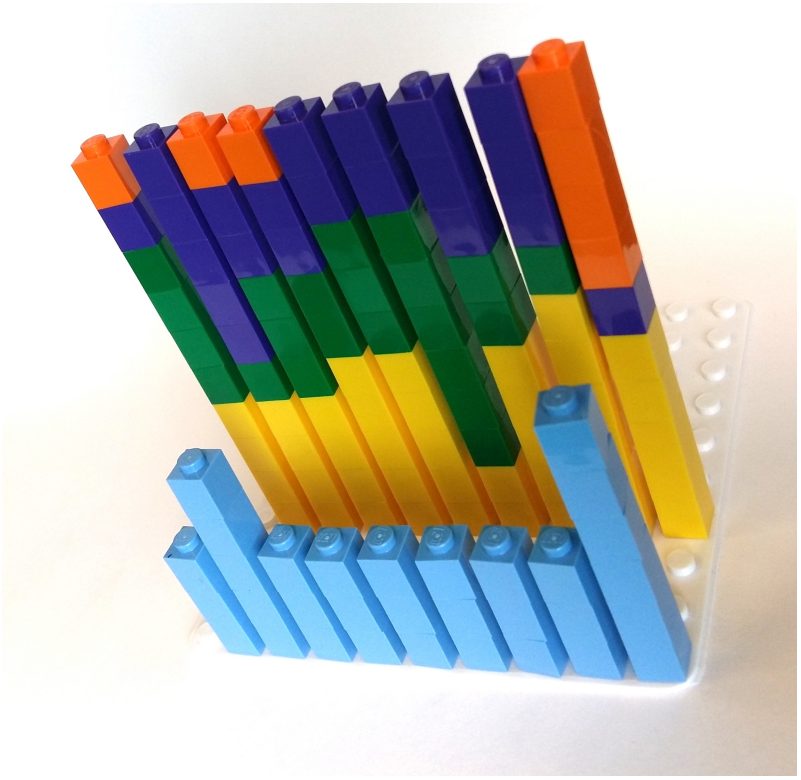


Abbildung 2.5: Example of a simple Physical Data Representation

2.3.2 Benefits of Data Physicalizations

There are a couple of benefits a physicalization has. One is explored by Thudt et al. [23]. It is about a study with nine participants, who gathered data from their everyday life. They subsequently used this data to fabricate physical data representation tokens and used these token for self-reflection. The gathered insights helped the participants to make small self-improvements.

Jansen et al. [12] worked out a whole list of further benefits in their paper Opportunities and Challenges for Data Physicalization, which are as follows:

Leveraging our Perceptual Exploration Skills: This benefit is also subdivided into four groups. The major aspects are that artifacts, which can be explored interactively by touching or walking around, are easier to understand than a displayed graph. Another one is that even the visual perception is the most dominate perception, it is not the only one that can be used to understand the meaning of data.

Making Data Accessible: Especially the non-visually aspect of physical data representation helps to make data perceptible to visually-impaired people. This is a good way to include them into something what is ordinary for most people.

Cognitive Benefits: Like an abacus is helping children to understand basic mathematics methods, other physicalizations have the potential to foster learning as well.

Bringing Data into the Real World: An artifact does not need power to be used and maybe has other properties to put it in an environment where a screen would break. This offers new possibilities to embedding data in our surroundings to foster the correlation between those two.

Engaging People: All the previous benefits come together here and is a major reason to foster research in this field. Due to all this it is possible to communicate data to a wide audience and therefore engaging people to spend more time exploring data, even for more

complex data that otherwise could appear deterrent.

2.3.3 Physical Variables

Similar to the visual variables form the data visualization, data physicalizations also have ways to represent data points. This range is even wider, because more senses can be taken in account and the used materials have properties, which also can support the data encoding. For example, artifacts can feel cold or warm to encode the average temperate of a country.

I worked out some physical variables that the knitting machine at hand is able to perform are:

- Main Color/ Contrast Color (two-color knitting)
- Number of stitches in a row
- Number of rows (in a given color or pattern)
- Shape of the overall piece
- Cable/ Crossed stitches
- Number of rows with different tension

These variables correspond to Bertin's variables. Obviously corresponds color to color and the shape of the overall piece to shape. Moreover, number of stitches and rows correspond to Bertin's size variable and cable/ crossed stitches or number of rows with different tension to texture.

Even if the number of physical variables do not seem to be so many, it is possible to have enough variation within just one variable to represent data on its own.

Furthermore, Gualiumin [9] provides a whole book about different techniques to manipulate stitches by hand. These techniques can also enrich the range of possible physical variables, but due to their partly complex nature and large number, including them into this work would be out of scope.

2.3.4 Workflow for Data Physicalizations

Jansen and Dragicevic [11] developed, based on Card's [4] data to visualization pipeline, an enhanced workflow especially for physical data representations. They described different actions the user has to take in order to converting data into physical attributes.

Raw data can be messy, therefore the first step *Data Transformation* is to convert this raw data in a way that makes it suitable for further processing. *Visual Mapping* is the next step, it means to give the data physicalization its initial form. After this the user can go on with mapping data points to different physical variables of the data representation. This step is called *Presentation Mapping*. The last step the user has to take is *Rendering*, whereby the data representation is converted into an artifact.

They also included the beholder's perception into their workflow and how the beholder processes data representations back into information. The first step that happens is called *Percept Transformation*. The beholder's senses capture the artifact's physical variables. *Integration* means that the beholder creates a mental model of the artifact. At last the mental model is used to extract information from different physical variables represent, which is called *Decoding and Insight Formation*.

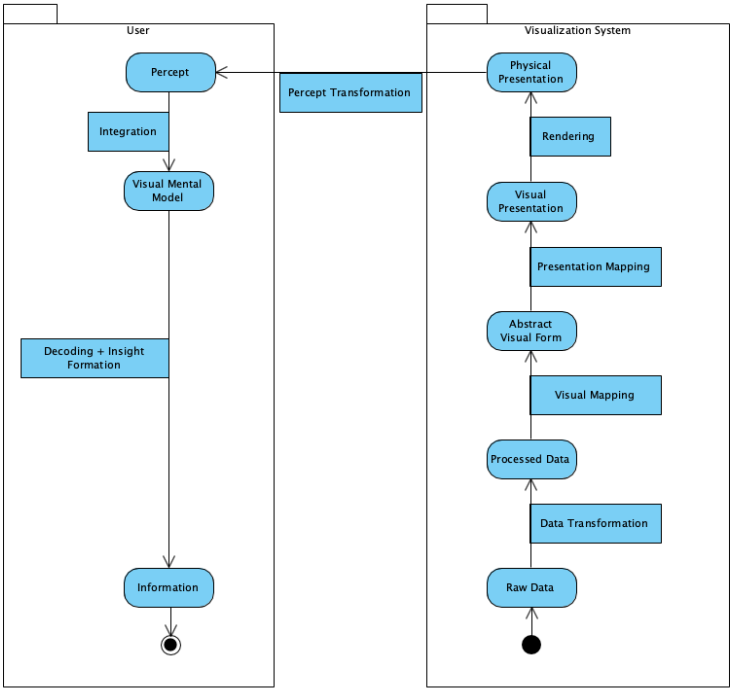


Abbildung 2.6: The InfoVis Workflow based on Jansen and Dragicevic [11]

2.3.5 Knitted Data Projects

To gain a better understanding on how knitted data can look like, I briefly discussed three examples of existing knitted data projects in this subsection.

The currently most popular form of knitted data are temperature blankets. The variation that Liza [6] describes is that the maker knits one row of the blanket everyday for a year. The color of the row depends on the temperature of the region where the maker lives. A table that shows what range of temperature is encoded by which color, helps the maker during the knitting. This usually result is a colorful blanket.

Another one is from an older German women [8], who knitted a scarf using data she captured in real time. The women knitted for one year. She took the train every day two times, whenever the train had a delay less than five minutes, she knitted a gray row. When the train had a delay over 5 and under 30 minutes, she knitted a pink row and for a delay over 30 minutes she used red.

Abbildung 2.7: The so-called Verspaetungsschal by Claudia Weber [3]

Seung Lee [7] also knitted data she collected one year long. Her data set consists of the sleep schedule of her new born baby. With this data she knitted a blanket. Every row of the blanket represents a day and every stitch of a row represents 6 minutes of the day. The

time when her baby was asleep she used a yellow yarn and blue if her baby was awake.

2.4 Authoring Physicalizations

Not only exploring a finished physical data representation supports the understanding of the represented data, but already being involved in the development of the physicalization supports it as well.

2.4.1 Manual Fabrication

Thudt et al.[23] discovered during there studies, that the process of creating already triggers the understanding of the data. Building a data physicalization by hand, or at least partly by hand, forces the user to think more about the data and what it is representing, due to figuring out how to connect the different data variables or how to make the physicalization stable. On one hand, manual fabrication, which means to craft an object by hand, offers a wide range of materials to choose from. Furthermore, it gives a lot of freedom to create and customize artifacts. Due to this, manual fabrication is an important aspect of this work.

One the other hand, crafting can require a lot of experience. A novice user may not know how to handle the specialized tools for the material. Therefore, many machines has been developed to lighten the entrance or to accelerate the crafting.

2.4.2 Digital Fabrication

In contrast to manual fabrication, digital fabrication describes a computer-controlled process to produce an object. A 3D printer is a typical example of a digital fabrication machine. The user designs or downloads a model with a computer and subsequently uses a 3D printer to physically create the model. A 3D printer's approach to create a fabrication is to add layers on each other. Besides this additive technique, there is also a subtractive technique. For example, a laser cutter severs smaller pieces from a panel, so the pieces can be put together to build an artifact. Both techniques are linear processes, which leave little to no room for users to hand manipulate the object during the procedure.

2.4.3 Hybrid Fabrication

However, in hybrid fabrication the user performs manual fabrication in collaboration with a digital fabrication machine throughout the whole process of creation, resulting in a meaningful, reflective experience for the user [5].

Kim et al. [13] discussed that the digital fabrication machines impede the creativity of the user or increases the error rate, because the user can not interfere with unexpected events. In order to tackle these downsides and to change the purpose of the machine from a tool to a co-designer, they created an alternative fabrication pipeline. The current workflow consists of the separation between the user, who designs the model and the machine, which creates the object. Whereas the new workflow also includes the user into the creation process by receiving feedback from the machine as well as taking actions to fulfill manual tasks. Furthermore they identified three aspects to achieve a collaborative hybrid fabrication system:

accessibility through support during the whole process the system enables novice user, as well as experienced user, to work with it

fluidity means to ensure that the user is able to interact with the object during the whole creation

concurrency there are multi-directional actions on the user's and the machine's side, which can happen simultaneously.

Not only these aspects have to be considered by people who built a hybrid fabrication system. There is also another challenge for building such a system, namely to decide what task is worth to do by hand and what task has the machine to fulfill. This has to be analyzed for each system anew.

2.5 Summary

In this chapter I explained the most important knitting techniques, the components of the AYAB project and how they work. In addition, I worked out why this approach of creating data representations is valuable and how it can be achieved. In the next chapter I will have a more detailed look at the AYAB software. Then I will analyze what modifications are missing to enable the design and creation of a physical data representation.

3 Requirements Analysis

In this chapter I explain the agile method I used to analyze and also implement this work. Then I have a more detailed look at the current state of the AYAB software. I also discuss two paper prototype sessions that were held and requirements that I identified through that process.

3.1 Project Context

Dr. Lora Oehlberg, a professor from the University of Calgary and part of the iLab, initiated the work at hand. It was research based and mainly experimental. With its unique interaction of the used components, the actual output was hard to predict. Therefore a close cooperation with the initiator of the project was necessary to face unexpected challenges and to develop new ideas to realize the desired goal. Constant feedback and meetings were important for this work, to include the initiator of the project into the creation process. This way it was ensured that my work maintained on the right track. Due to these circumstances, I chose to use an agile procedure model for the analysis, as well as for the whole work process, namely the spiral model. During the analysis I went through three iterations, during each iteration a prototype was created in order to analyze and work out improvements for the next prototype. This also means that the analysis is overlapping with the conception and the implementation at some points. Still only the worked out findings and requirements are presented in this chapter.

The first prototypes were created with paper prototyping. Paper prototyping is a common method in Human Computer Interaction (HCI) to design and analyze drafts of a software. I tested the designed software with potential users, in order to collect valuable feedback. This helped me to come up quickly with new ideas for better software designs.

In the next section, I discussed how a knitted data physicalization is made currently. Then a brainstorming session was held, it led to ideas of different use cases. In order to work out requirements of these use cases, a first paper prototype session toke place, which was focused on turning data into a physicalization. The results led to the first prototypes for the user interface that covers the data converting process. I held a second meeting with group interviews to test and improve these prototypes. Eventually, the gathered feedback was used to create the first improvements in the existing software from the AYAB project. In this chapter the different findings of the individual iterations are summarized.

3.2 Current State

After I had a look at the current possibilities to create a knitted data physicalization, I analyzed the current state of the AYAB software and its function within the whole system.

3.2.1 Knitting Data

Currently, there is no system that fully supports a user who wants to create a knitted physical data representation. One way to fulfill that task is to knit by hand. In this case the user has to be experienced in hand knitting. The first step is to design the knitting. This

can be done in different ways, for example, by sketching or just in the head of the user. In every case, the data points have to get mapped to physical properties of the knitting. After the design is done, the user can start knitting. During this process the user has to keep track of the amount of rows and stitches, which can be difficult and therefore lead to mistakes. Another method to knit a data physicalization is to capture the data in real time. To do so, the person how knits uses different knitting techniques to capture certain events during a particular time period. For example, the knitter could watch an ice hockey game, knits one row for every minute of the game and after every goal there is a change of color.

Another way to knit a physical data representation is to use a knitting machine, which is also the way that is fostered by my work. For my work I used the knitting machine KH-930. Like knitting by hand, there is no software for this knitting machine that covers the design of a pattern based on data and supports knitting this pattern. Even though this knitting machine can be controlled by computer software, there is no software that is designed for these tasks. The current AYAB software can only knit low-resolution pictures. This means, the user has to create a picture of the pattern with an external tool and transfer it into the software. At this point, the external tool does not know about the restrictions the knitting machine has, which can lead to an unusable picture. After a usable picture has been transferred into the software, the pattern can be knitted. To perform this task, a novice user must read the manual or can be taught by a more experienced user on how to use the knitting machine. The machine itself as well as the software does not support the user with the different manual tasks.

3.2.2 System Architecture

To gain a better understanding of the existing software, the architecture of the AYAB software is illustrated in figure 3.1.

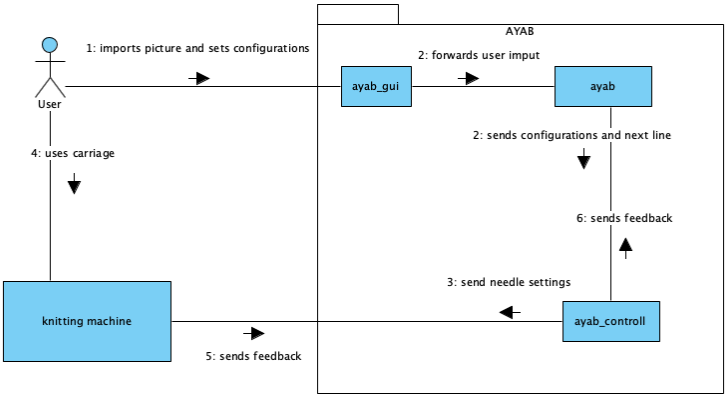


Abbildung 3.1: Overview of the current System

The user can import a picture from the disk to the AYAB software or chose one of the software's sample patterns in its resource directory. Within the AYAB software, the ayab class receives the picture and enables the user to set configurations or options to edit the pattern, such as inverting or mirroring. After everything is set and the user wants to start with the knitting process, the ayab class sends the configurations as well as the first line of the converted picture to the ayab_controller class. The ayab_controller class converts the received line into instructions for the knitting machine on how to set the needles and sends these instructions to the knitting machine.

Now the user moves the carriage to knit one row of the knitting. The knitting machine recognizes the movement and sends feedback about the new carriage position to the

ayab_controller class. The ayab_controller class converts this feedback and forwards it to the ayab class, which uses the feedback to keep track of the knitted rows and updates the information displayed to the user.

This also shows what kind of components is needed for the system to work. The system consists of a computer or laptop, which is able to run the AYAB software. This computer has to be connected to a knitting machine. The AYAB software supports all knitting machines models from the Brother KH-9XX series. Finally, a user is needed to first import a picture and set the configurations and then operate the knitting machine.

3.3 Use Cases

A first brainstorming session with the initiator of the project, resulted in a variety of ideas on how to turn the knitting machine into a hybrid fabrication system, which can be used to knit physical data visualizations. These worked out ideas were documented and subdivided by their commonalities, resulting in the identification of the following use cases (see Figure 3.2).

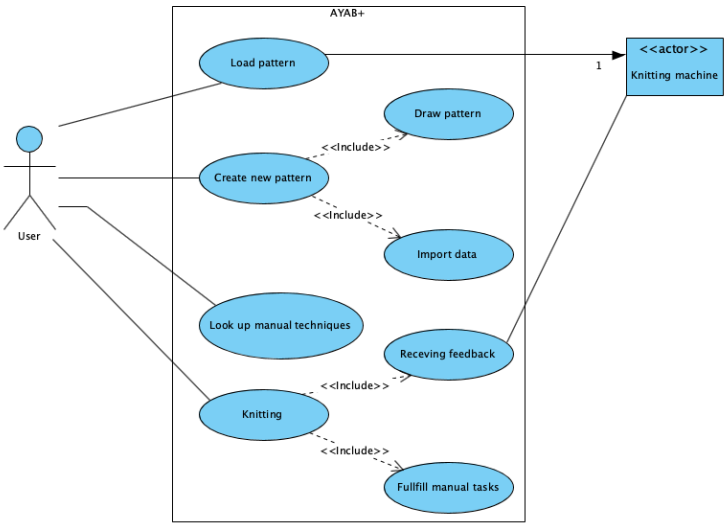


Abbildung 3.2: Use cases

When the user wants to create a new pattern, there are two options. On the one hand, the user has a data set and wants to import this set into the software in order to design a pattern for a physical data visualization. On the other hand, it is possible to create a pattern from scratch, just by drawing it. This could be used to capture data in real time during an event.

Knitting a data physicalization requires not only to work with the software, but also to do tasks manually. Furthermore, one task can have various techniques with different outcomes. In order to look up these techniques, which have to be performed, without using any additional resources, it is necessary to provide a section where these tasks are explained. This way the user can easily reread it at anytime.

Loading a pattern into the machine is necessary to start knitting. Then this pattern gets send to the knitting machine. This is performed by sending the settings of the needles for every row, from the software to the knitting machine.

The last use case is knitting. This also includes receiving feedback from the knitting machine as soon as the user has to fulfill a task manually, which also can mean to switch

settings of the knitting machine. Therefore, the system should provide supporting tools for this use case. Fulfilling a task is included into the use case as well.

3.4 Paper Prototyping

After I worked out the use cases, I used them to plan my further procedure. I identified one use case as the main task and held a paper prototype session to gain more insights of this task.

3.4.1 Physicalizations

The Create New Pattern use case is the major task because of the different characteristics a pattern has. These characteristics form the basis for the actions the knitting machine and the user have to take, in order to transform the pattern into knitting. On this occasion, I held a paper prototype session, in which five attendees (one female and four males) took part. The participants got a data set from a workshop, which is about making data physicalizations [10]. The data set is about the alcohol consume in different countries. I asked them to build a paper beer coozy, which represented the data or a part of the data in any way. The small group session lasted approximately one hour and most attendees came up with two or three prototypes with a wide range of shapes and appearance (see Figure 3.3). Especially the different approaches were interesting, because it helped clarifying which functions are needed in order to encode the data set into a data physicalization.

Abbildung 3.3: The different beer coozy prototypes

3.4.2 Findings

During the paper prototype session, I made some observations. The attendees used the data set during the whole process of the creation, in order to match the data with the physicalization. That leads to the conclusion, that the chosen data set has to be displayed during the process of the pattern creation in the software as well. Another observation was that especially at the beginning the attendees were not sure about the different techniques and what possibilities they open up to encode the data. Providing a reference guide during the creation process could solve this.

After that session, I analysed the prototypes itself. I identified some recurring ideas of data representations, but also some prototypes boasted unique characteristics. I focused

on the ones that had similarities, because could I use these prototypes to create a paper prototype interface, in which the user can perform the process of data transformation

Most attendees used some kind of bars for their representation, some used different colors to create a bar, and others used the shape of the coozy itself to create bars. Moreover, the colors matched the colors of the flag from the country which data was represented in their prototype. This indicated that the two color knitting as well as increasing and decreasing techniques have the potential to be very important. Therefore, I chose these techniques to be supported in the design process, as well as in the creation process.

While the two color knitting function was already well developed and covered by the existing software, there was no possibility that the machine provided feedback to the user telling him to decrease or increase stitches. Furthermore, the knitting machine did not differentiate between a pattern with laces or a pattern with two color knitting. At the moment it depended on which carriage was used and if there was a contrast yarn in the yarn feeder or not. As a result, I decided that it had to be possible to assign more than just one type of information to a pixel in a pattern. That way the software can give feedback about the tasks a user has to fulfill on the current row.

3.4.3 Interface

I created a draft of a user interface and showed it to the same attendees as in the session from the first paper prototyping session. I asked to use it like it was a real software application. Then I asked them to give me feedback based on their thoughts and experience with this interface prototype.

3.4.4 Findings

The attendees came up with a wide range of improvements. One of the ideas to improve the user interface was to add a undo function. If a user has made a mistake during the design process, it should be possible to undo this mistake. Therefore a history panel that displays the last actions is helpful to maintain the overview about the actions performed and the ability to return to a specific state.

The user often did not want to set just one single mark, instead the user wanted to fill a whole line or even section with the same marks at once. To enable this the pattern maker tool should provide a function, which enables the user to do so.

Different views, such as preview, zooming in or zooming out, could help to maintain the user's overview of the pattern. Therefore these functions could also be built in.

The attendees also suggested some smaller improvements. One of them was that the most important functions should be able to get activate by a certain key combination. These short cuts are a common and convenient way to activate a function in most other picture editors.

Another one was to equip the buttons with icons in order to raise the usability. That way it is possible to recognise the function of the buttons more easily.

3.5 Found Requirements

I grouped the worked out requirements for this work according to their importance like it is used in the MoSCow analysis.

3.5.1 Must Haves

It must be possible to import a Comma-separated values (CSV) file into the software and read out the data in the file. Furthermore the software has to give feedback if the file cannot be converted and what have to be changed in order to make the conversion possible. There must be a possibility to create a pattern for a physical data visualization and also to save the created pattern on disk. In addition, while using the software for creating a pattern, the stitches and rows of the pattern have to be adjustable by the user. The look up for regular used manual techniques has to be accessible by any time. The user needs to know what dimensions the knitting is going to have before he starts knitting, especially if the relation between length and width is an important part of the data visualization. Therefore, the software must have a calculator for this task. This means that the user has to give input about the tension and the thickness of the yarn and indicate the number of needles he wants to use. When the user wants to do a knitting with an uneven edge, the software should give feedback when the user has to decrease or increase stitches.

3.5.2 Should Haves

The pattern maker tool should provide functions that are most common in other picture edit tools, such as drawing a line, changing views and undo the last actions. The imported data should be displayed during the pattern creation process, if the user wants to.

3.5.3 Could Haves

The software could keep track of the tension from the incoming yarn, because it is a common source of error, for example the yarn get tangled up or can put itself into knots at the yarn guide. When the tension raises it is often a signal, that something does not work smoothly at the yarn guide and the software could warn the user about this situation. In order to achieve to that, there could be sensors attached to the knitting machine, which can measure the tension. The software could have access to an Infrared (IR) camera in order to compare the actual settings of the knitting machine with the settings that are needed to perform a certain task. Furthermore, it also could have access to a projector. This projector could support the user during the knitting process, for example, by highlighting what needles have to be activated.

3.5.4 Won't Haves

It is not possible to knit with more than two colors in one row, due to the restrictions the knitting machine has as a single bed knitting machine. The same reason makes it impossible to change between purl and stocking stitch on one side of the knitting, without labor-intensive hand manipulation techniques. This is why there will be no supporting tools in the software for these techniques.

3.5.5 Non-functional Requirements

It is prescribed to use the knitting machine Brother KH-930 is, therefore the software has to be interoperable with this knitting machine.

One important aspect of this project is to lower the threshold for knitting physical data visualizations, therefore a good usability is essential. The user has to be able to operate the software after a short introduction.

Usually there are more functions and requirements identified than can be implemented during the time frame of a bachelor thesis. In order to implement these functions as well,

even after this work is concluded, the software should be extensible. This is especially necessary, if there will be following projects that will use this work as a starting point.

Furthermore, the developed application should not be restricted to one operation system instead the software should run on every common operation system.

To keep the modules manageable, especially for future work, the source code of the new modules should be small and therefore consists fewer than 300 lines of code.

Another requirement to keep methods manageable is to give methods names, which are explaining its task. Furthermore, more complex methods should have comments to explain its task more detailed.

The software does not need to display complex graphical representations. In addition, it is not necessary to perform complicated calculations, so the software should use little Central Processing Unit (CPU) and working memory allocation.

3.6 Summary

After I discussed the method I used, I had a look at the current state of the AYAB software. Together with the paper prototype sessions I was able to identify the main requirements of the work. The main goal is to cover the pattern design process this includes to convert data points into physical variables, in order to create a pattern which can be used get knit by a knitting machine. Furthermore, the machine has to give feedback about the current state of the knitting process and which task has to be performed by the user.

Due to the experimental nature of this work, it is not possible to identify all requirements during the analysis. This requires to test and to reconsider the functionality of the system constantly. Still the determined requirements will generate a solid base for this work.

After this analysis I will have a look at other projects, which have some similarities to my work. Furthermore, I will explain on the basis of these projects what gap I bridge with my work.

4 State of the Art

In this chapter I discuss other projects that overlap my work and what I could learn from them. First I focus on other projects that are aiming at knitting or knitting machines. Then I have a look at other hybrid fabrication systems. Finally I discuss applications that transform data into data visualizations.

4.1 Knitting Projects

Using soft materials to create fabrications is an interesting research field in Human Computer Interaction (HCI), because fabrications have the ability to come in many different variations and they can still change their from after the production. This holds great potential to create dynamic data physicalizations. This is the reason why there are various projects about knitting and the use of knitting machines. In this section, I discussed two projects that offers ideas, which can be used for my work.

4.1.1 Automatic Machine Knitting of 3D Meshes

One of these projects involves a computer-controlled knitting machine as well. Narayanan et al. [18] implemented a software that is able to turn a 3D model into executable instructions for knitting machines, in order to create knitted 3D meshes.

4.1.1.1 Functionality

At first, the user loads a 3D model as input into the software. This input has to be a 3D triangle mesh. The mesh also needs to have a time function, so the software does know where to start and where to end. The next step is called remeshing. It means that the software generates a directed graph to fit the row-column structure of a knitted good. The software has to consider many constraints, such as being helix-free, in order to generate a fully functional graph. After the generation of the graph, the software uses it to create different knitting operations. The operations are represented by different marks. This step is called tracing the knit graph. The software's last step is scheduling, hereby turns the software these knitting operations into instructions for the knitting machine. These instructions can be used by the knitting machine to finally knit the mesh.

4.1.1.2 Limitations

This transformation from a model into a knitted good offers an interesting base for my work. The software uses a given pattern to generate knitting operations in form of marks. Then it creates instructions for the knitting machine based on these operations. In my work, I also applied this idea to generate marks, which were used to create instructions. Instead of creating instructions for the knitting machine, I created instructions to let the user know what kind of task has to be performed at a certain point. Another similarity is that, in both projects, a computer is used to track progress and calculate the dimensions of the outcome. Furthermore, Narayanan's work provides groundwork for 3D knitting. This can be used to create data representations as 3D models, but due to the restrictions of

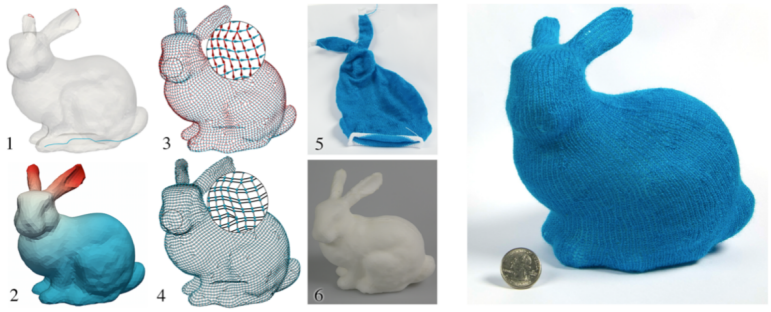


Abbildung 4.1: The different steps 3D Meshes [18]. 1) The input model. 2) The defined time function. 3) & 4) The generated graph. 5) The outcome. 6) A foam model of the input model.

the knitting machine in my work, it is not possible to adapt these ideas easily. Therefore, this is out of scope. Moreover, their project does not have options to design a model out of data or involves the user into the creation process.

4.1.2 Knitting Visualizer

Yang [25] is connecting the design of a knitting pattern with writing code in Javascript. With her work she wants to show the relationship between these two techniques.

4.1.2.1 Functionality

The interface of this (see Figure 4.5) prototype shows three fields, the first one displays the Javascript code. To create this code, the user can either write it or design a pattern and generate the code from the pattern. The one in the middle shows the knitting pattern. It is structured in a grid, which represents the stitches and rows of the pattern. Each cell is one stitch and can have a mark, which stands for a specific knitting technique. The last field shows a rough preview how the knitting good would actually look after it is knitted.

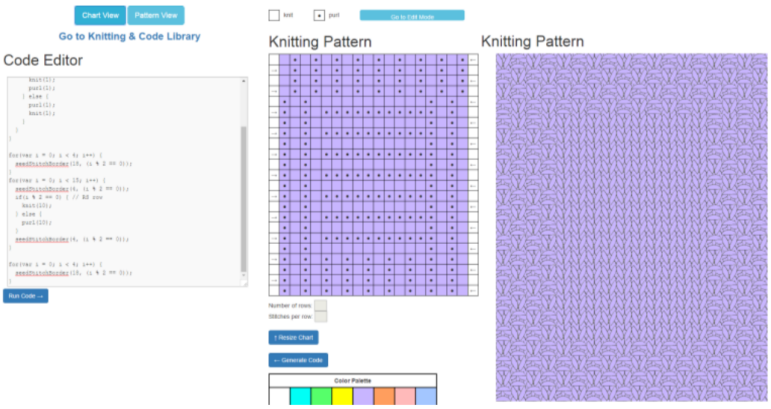


Abbildung 4.2: The software interface of Knitting Visualizer [25].

4.1.2.2 Limitations

The Knitting Visualizer covers the creation of the pattern design, how it is also utilized in my work. The use of a grid and marks to represent the pattern is used by both Yang's and

me. One difference is, that Yang uses Javascript as a base to generate the pattern. While in my work the base is the data, which the user wants to create a data representation out of it. Furthermore, the created pattern is intended for hand knitting instead of knitting the pattern with a knitting machine.

4.2 Hybrid Fabrication Systems

To be involved during the creating process of a physical data representation has its benefits, but how can a system achieve such interaction between user and machine? This research question is tackled by different approaches. The most successful will be discussed in this section.

4.2.1 Being the Machine

Devendorf and Ryokai [5] built a hybrid fabrication system whereby the user acts like a manual 3D printer. Their goals is to rethink who should be in control during a making process, as well as open the field of usable materials and encourage the user to trade precision with unexpected forms.

4.2.1.1 Functionality

The system consists of a laser pointer that shows the user where to put the material. The laser is attached to two servomotors in order to move it. Furthermore, there is also a computer that runs the associated software and a wireless key fob, which allows the user to go back and forth with the instructions.

At first the user has to load a digital model of the desired outcome into the system. Then the user has to choose a material, which the outcome should exist of. The material can be chosen complete freely, for example, one user built her model with Magnolia leaves or another used pancakes. The digital model gets converted into a so-called G-Code file. These files usually are getting used from 3D printers. At this point it is also possible to set the building parameters to customize the model to the dimensions the material has. After the file is imported into the system, the current layer and the whole model will be displayed on the system's screen. The next step is to set the laser in the right position and then start with the first instruction. The laser starts to show where the user has to put the material. After the user hits the next button on the key fob, the laser moves to the next point. Then the user puts down the material and pushes next again and so on. When the laser goes of, it means the current layer is completed. This procedure goes on layer by layer until the whole model is completed.

4.2.1.2 Limitations

The approach of the Being the Machine system is to combine the strength from both, the computer and the user. By using the computer to calculate the size of the outcome, the calculation is more accurate. Changes in dimensions can also be recalculated faster than the user could do it. Additionally, the computer is able to display a visualization of the model and keeps track of the current state of the outcome. This way the user can concentrate on other things, which the computer, in turn, is not capable to handle on its own. Handling unexpected events is one of the things a human can do better. Another one is to manipulate the wide range of different materials, which can be used to build the outcome. Except from different materials, my work also includes these aspects. The system can calculate, visualize and keeps track of the work process as well. Unlike Being

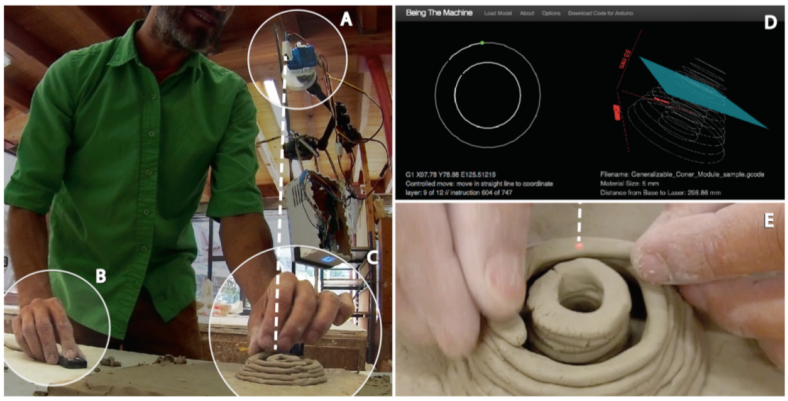


Abbildung 4.3: The Being the Machine system [5]. A) The laser pointer. B) The key fob C) & E) The outcome. D) The software interface.

the Machine my work is limited to one material, therefore the system is able to give a better support to the user during the creation process. Moreover, the knitting machine performs repetitive tasks, which otherwise would be hard to perform by novice user. And last, there is no possibility to create a own model within the Being the Machine system, therefore it is more complicated to create a data physicalization with it.

4.2.2 Robotic Modeling Assistant (RoMA)

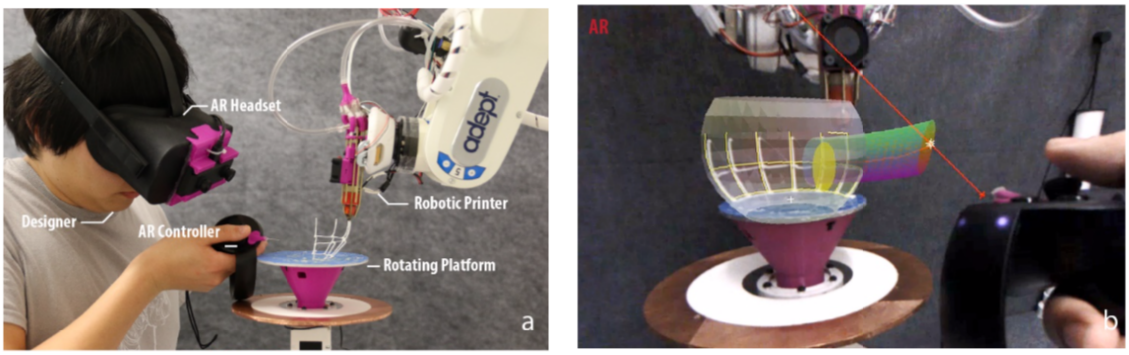
One hybrid fabrication system, that allows to design a model and to create this designed model, were developed by Peng et al. [19]. The RoMA enables the user to create a digital model by using an Augmented Reality (AR) headset and controller, while a robotic printer creates the model simultaneously. Furthermore, this system is able to work on an already existing physicalization.

4.2.2.1 Functionality

At first the user has to put on the AR headset and controller. Then the marking menu will appear in which it is possible to choose between tools like revolve, extrude, loft and sweep. After selecting one, the first plane can be drawn on the platform between the user and the robotic arm. During the whole time, the system displays all the drawn planes with the help of the AR headset. As soon as the first planes are finished with designing, the user can validate this by pressing the confirm button on the controller. Then the robotic arm starts to print the part of the design, which is next to it on the platform. The user can still go on with designing the model. To add something to a part that is currently printed, the user can stop the robotic arm and modify this part.

4.2.2.2 Limitations

Like my work, the RoMA is specialized to one material and covers the whole process from designing to creating the actually outcome. Moreover, both systems provide the possibility to make changes through the creation process itself. On the other hand, the user's only manual interaction is cutting strands off. Due to this separation into design and fabrication process, it is not a pure hybrid fabrication. Instead Peng et al. [19] rather uses the term interaction fabrication. AR technology could be used on the proposed system, like for supporting manual tasks by guiding the user with visual advises, but this would mean a considerable additional effort. An effort that would not have much



(a) The overview of the RoMA system

(b) The user's view

Abbildung 4.4: The RoMA system [19]

more of an advantage than simply showing advises on the computer screen. Therefore AR technology was not used in my project. Another Limitations is that there is no possibility to create a data-driven outcome.

4.3 Tools for Data Conversion

In this section one tool for fabricating physical data representations and two for creating digital data visualizations will be discussed.

4.3.1 MakerVis

There are many tools, which are covering the whole process from raw data to a digital data visualization. These tools also provide a wide range of different ways to visualize the data. However, this is not yet the case for creating physical data representations, Jansen et al. [12] already pointed out, that there have to be more tools to cover bride this gap. To ensure that a data physicalization can take full advantage of the different materials it can have, there must be more tools, which are able to process these materials. One of these tools is (the proof of concept prototype) MakerVis, which functionality is discussed more detailed by Swaminathan et al. [22].

4.3.1.1 Functionality

In order to design a data representation with MakerVis the user has to fulfill six different steps. As already discussed the data representation process starts with the raw data that is transformed into processed data, by transferring the data into a table. Then the processed data can be imported into the MakerVis software. Hereby is the CSV format as input supported. Next the user can choose how the data should be represented. There is a selection of different visualizations, like a bar or a line chart. This gives the initial form of the representation and leads to the next step, mapping data to visual variables. The user sees the different data dimensions and can decide how to arrange them.

After that, the software shows a 3D preview of the physicalization and the individual parts the physicalization consists of. This steps also help to gain an understanding how big it will be. This step also results into the abstract visual form of the data physicalization. Step number four is setting its geometry. The changes will also update the preview, this way the user is able keep track. In Jansen and Dragicevic's [11] InfoVis pipeline, this step is called presentation mapping and results in the visual presentation of the data.

Now the next step is choosing which fabrication machine will be used for the creation process. The user is also able to set the default material, as well as the maximum size of the single pieces. Eventual, the user can download the design as a file, in order to create the data presentation. An instruction on how to build and assemble the pieces can also be downloaded.

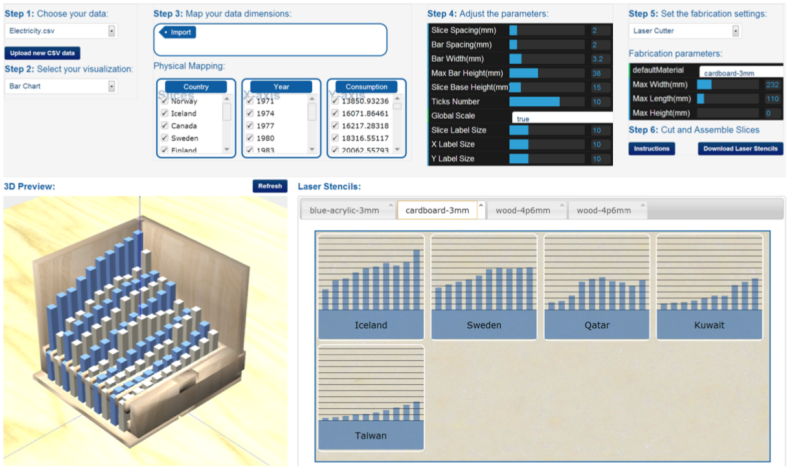


Abbildung 4.5: The software interface of MakerVIs [22]

4.3.1.2 Limitations

The MakerVis tool covers the whole process of designing a data presentation. The outcome is a file that can be used for various fabrication machines. My work covers the process of designing a data presentation from processed data as well, but its outcome only aims at one specific fabrication machine, the knitting machine Brother KH-930. With this restriction it is possible to create a system that not only covers the design, but also the fabrication of the data representation. Furthermore, MakerVis supports rigid materials that can be cut with a laser cutter, processed by CNC milling or material used by a 3D printer. My work aims at different materials, namely wool or yarn, which are soft and supple. Therefore, my work is also meant to enrich the possible materials a physical data representations can have.

4.3.2 DataInk

DataInk offers an artistic way to represent data digitally, with a focus on creative and customized visualizations. It was created by Xia et al. [24]. With this application the user encodes data points into glyphs. Since this application was developed for tablets, it is controlled by pen and touch input.

4.3.2.1 Functionality

The GUI has one big panel and two smaller panels on the left side. While the big panel serves as a canvas, the smaller ones are the glyph panel and the layout panel (see Figure 4.6). First the user draws different shapes, which the glyph can have. Then the user chooses one of these drawings and binds data points to its dimensions. The application automatically creates an randomly amount of glyphs on its canvas window. To modify a glyph the user has to tap on it, then a visual-data palette appears as a ring around the glyph. This way it is possible for the user to map the glyphs dimensions to certain data

points. The changes apply to every glyph that is mapped to the same data point. To maintain the user's overview, the glyph panel can be used to check the dimensions that are mapped to certain data points. After the mapping, the glyphs can be organized in groups based on common data points. Then these groups can be formed and arranged so that their similarities can emerge even more.

Abbildung 4.6: The DataInk GUI [24]

4.3.2.2 Limitations

DataInk's simple GUI and direct manipulation of the glyph creates an easy entry for novice user. These are characteristics that I also adapt for my software. Furthermore, the idea of creating glyphs that can be modified to map data points is a valuable approach, which can be adapt by the user while creating a knitting patter. Moreover, DataInk creates unique and artistic data representations. Due to that these representations are more memorable. This is also strength of my work. However, the output of DataInk is not suitable to be knitted afterwards.

4.3.3 Excel

Probably one of the most common used tool for data visualization is the one which is build into excel.

4.3.3.1 Functionality

Excel itself is able to open CSV files or data can be written into it directly. After gathering or importing the data, the user has to mark the cells with the data points. Then the user clicks on the button of the diagram, which the user wants to use to represent the data. The program creates a first draft for the data visualization automatically. Now the user can customize the visualization by clicking on the diagram. The options for customizing open and are displayed in a submenu. Changes can be made, such as inscribing labels, rearranging data points or changing colors of the different sections. All changes in the data will lead to changes in the diagram instantly.

4.3.3.2 Limitations

Excel covers the whole creation process for digital data visualizations from raw data to the final visualization. Besides the step from raw data to processed data, this is similar to the work at hand. Even though, it is possible to load raw data into the system and edit the

data, it is not intended to do so. Excel only creates data visualizations that can be presented on a display, but does not support visualizations that can be rendered into physical data representations. Furthermore, the user can only choose from prescribed diagrams, this makes the creation process faster and more convenient, but also restricts the user to this prescribed diagrams. In contrast, in my work the user can create the representation without these restrictions, but also has to consider some knit specific limitations.

Summary

I explained how I can include ideas from the presented projects to my own work and also what pieces are missing to fulfill my work out requirements.

In the next chapter I will explain concepts that i designed to bridge these gaps.

5 Designing K1M1

In order to meet requirements from the analysis chapter, I conceptualized modifications. I discuss in different sections how these modifications convert data-sets into pattern and how they enable hybrid fabrication. Moreover, I present models to illustrate different aspects of my software. At the end I show and explain GUI drafts of the software.

5.1 Converting Data

In the second paper prototyping session, I asked a group of people to use the developed GUI draft as it would be an already existing software, to evaluate the usability of the draft. The gathered feedback showed that the basic concept works, the participants had only ideas of minor changes, like adding icons to the button description or having short cuts for some of the main functions. Together with this feedback and ideas to improve the usability I developed this concept.

5.1.1 Functioning

In order to access the new function, which lets the user create a pattern, adding a new menu item to the main menu of the program called New Pattern would be a simple solution. Through clicking on this menu item, the window of the pattern maker will be displayed. When the user wants to create a data physicalization it is necessary to import the data into the system. Therefore, there can be a button, which allows the user to import a CSV file. If the file contains raw data, this data can be rearranged and processed in a particular window. Already processed data can also be imported and would make the process faster, therefore this way would be recommended. According to Jansen and Dragicevic's [11] workflow for physical data representations, the next step is to give the physicalization its initial form. To fulfill this task, the user can be provided with an empty table in the main window. The cells of the table are square-shaped to recreate pixels in a picture. Furthermore, each cell represents one stitch of the physicalization. These cells can be filled with marks and these marks, in turn, represent the different techniques the knitting machine can fulfill. There need to be a button for each kind of mark, the different marks are as followed:

O represents a hole, which is used for lace knitting

X decreasing a stitch, which also means that if there is an X in the row beneath, but not anymore in the next row, there will be an increasing of a stitch

Color a cell can also be filled with a color to show that a contrast yarn has to be used for this stitch. On the other hand, a white cell means using the main yarn and knit stitch

The button for the color mark will open another window with different colors to choose from. Another button is the undo button, to enable the user to cancel the last action. The undo action is commonly linked with a well-known short cut, namely Ctrl+Z, so this is a

good addition here, too. The way this undo function works is that the software saves the current content of the entered cells in an additional variable. The entered cells itself are saved in an additional variable as well. After calling the undo function, it fills the saved content into the last entered cells. However, this works best if only one cell is entered. When the users hold the left mouse button to enter more cells at a time, the cells are not filled at the same time, but one after the other. Therefore, it is only possible to undo the contents of the cells that were last filled.

The calculator button will open another window to calculate the length and width of the fabrication. The user can specify how many rows, stitches and what tension the fabrication should have, then the software calculates the length and width. When the user is satisfied with the settings, the rows and stitches can be transferred into the main window and will be applied to the rows and columns of the table. If the new number of cells in length and width exceeds the ability of the table area to display all cells simultaneously, it is not possible to zoom in or out due to the restriction of the table. A cell in the table must be large enough to display at least one letter. This means, the user must use the sliders at the bottom and right of the panel to change the display of the cells. After the user used the marks to create physical variables and mapped the data points to these variables, the pattern is finished and can be transferred to the main window of the program.

To do so, the pattern table has to be converted into a picture, that the main software is able to handle. Therefore, a new picture will be created with the length and wide of the pattern. Each cell has to get checked what mark it contains. If it does not contain a mark the allotted pixel in the picture will be drawn white. The same works for the X mark, but if a cell is colored, the allotted pixel in the picture will be drawn in the same color like the cell. Furthermore, an O mark leads to a black pixel. In this way, an image is created that can be further processed.

Sometimes the user is not finished with the creation of the pattern, but wants to work on with the current draft at another time. The solution is to save the current draft on the disk, therefore a button will be provided, which is able to perform this task.

5.1.2 Restrictions

Due to the restrictions the knitting machine has, there are also various restrictions, which have to be take in account during the pattern design process. One restriction is that an O and an X cannot be in the same row, because, for one thing, the user has to move the lace carriage over the selected needles to produce a row with lace holes. Then again, for two color knitting the knit carriage has to be used, so it is not possible to knit one row with lace holes and different colors. Knitting more than two colors in one row is also not possible, that is the reason why there can be more than two colors in one row of the pattern. It is only possible to decrease or increase a stitch at the edges of the fabrication, therefore can an X only stands at the edge of the pattern or next to another X mark. In addition, an increase or decrease can only be performed on sides on which the knit carriage stands. As the pattern is knitted from bottom to top, it is necessary to calculate where the knit carriage will stand at which row. At the start, the knit carriage always stands on the right side. That means, for an even total of rows, the amount of X marks on the left side of the pattern can only change in even rows and the amount of X marks on the right side can only change in uneven rows. In turn, for an uneven total of rows, the amount of X marks on the left side of the pattern can only change in uneven rows and the amount of X marks on the right side can only change in even rows. This restriction was not be implemented, because I would have to lock even or uneven rows for one side and copying the X marks from the row below. This would result in hardly manageable `if` statements.

O marks represent stitches, which are transferred to the needle next to them. Therefore

it is not possible to have two or more O marks next to each other. This would create dropped stitches. Furthermore, there must be a free row after a row with one or more O marks, due to the mechanism of lace knitting. If there were two consecutive lines with O marks, the following two rows would have to be empty lines. In addition, non of these O marks could be in the same column. To simplify this restriction, it is only allowed to set an O mark in a uneven row if the amount of rows is uneven and to set an O mark in a even row if the amount of rows is even. This way it is ensured that the user only have to push the lace carriage one time to the left and one time to the right, before using the knit carriage again. In general it is very difficult to use lace knitting and two color knitting in one knitting and is therefore not recommended. Due to these restrictions there will be no rotate button, otherwise it might be not be possible to observe all the restrictions after the pattern is rotated around 90 degrees. Moreover, the user is only able to fill in any mark into a cell by using the provided radio buttons. This way it is ensured that the user can only create a pattern with marks, which the system can handle.

The knitting machine has a needle bed with 200 needles on it, because of this the maximum cells a pattern can be width is 200 as well. The same applies for the length. The initial AYAB software restricts the length on 200 pixels, because it is possible to rotate the pattern in the main window. This function, as well as inverting, repeating and mirroring, will be disabled, because it would be mess up with the array that contains the marks and manages the notifications for the user. The feedback from the third analyze iteration took into consider, it will be possible to select multiple cells and set one mark in all selected cells at once as long as the restrictions are not violated.

5.2 Hybrid Fabrication

In order to build a hybrid fabrication system, it can be challenging to decide which task the user should perform and which should be fulfilled by the machine. In the case at hand, it is not that complicated, due to the limitations of the tasks the knitting machine can perform. In this section, I explained the different tasks and how the system can help the user to perform these tasks.

5.2.1 Fundamentals

The initial software from the AYAB team is able turn different colors into instructions for the knitting machine, but it only gives minimal advises to the user. There is a small notification at the beginning of the knitting process about the settings of the carriage and the amount of already knitted rows, as well as the total amount of rows are displayed. That means if the user wants to increase, decrease or switch between two color knitting and lace knitting, it is indispensable for the user to exactly remember at which row these tasks has to be performed. Consequentially, a physicalizations with uneven edges and a mix of two color and lace knitting techniques requires the user to be highly concentrated on tasks which could fulfill the machine more precisely.

To support the user with a higher amount and more detailed advises the system has to know at which point what kind of task is required. Due to the different shapes and forms a knitted data physicalization can have, the pattern that will be used to knit needs additional information, so the system is able to give the right feedback at the right time. The way the software works is, it takes the given picture of the pattern turns it into a bytearray and send it to the knitting machine. It is not possible to put additional information on the picture itself beside different colors. Different colors for different tasks, like decrease, would work for single bed knitting machines and could help to maintain a good overview

for the user, but would also cause problems if the software would been used for a knitting machine that is able to knit more than two colors. Therefore the marks of the pattern form the pattern design process, will be saved separated in a two-dimensional array. For every row in the picture, there is a row in the array, which includes all marks form that pattern row. When the machine knits one row, a row counter will be increased by one. This row counter can be used to find the right row from the array and this row can be checked for marks. If the inspected row contains one or more marks this indicates that a task has to be performed by the user, so the system has to give feedback. This can happen through a new window that appears on the screen with a notification of the task or tasks the user has to perform. From this notification window the user can switch to the help section if needed (see Figure 5.1). Not every time the user needs to read the complete instruction, for example if the user have to increase stitches for the third or fourth time, the user likely does not need to read how to increase needles again. Therefore the whole help section will not pop up every time the system gives feedback.

Abbildung 5.1: Concept of the Notification Window

5.2.2 Increase and Decrease

Decrease and increase are common techniques to hand manipulate a knitted fabrication. These techniques also give a feeling about the differences of the data points, if the shape of the knitting is, for example, a bar chart.

To generate the notification for decreasing stitches, it is important to know on which side the task has to be fulfilled. Furthermore, the amount of stitches, which have to be decreased is necessary, therefore it is not enough to simply counts the X marks in one row. For each row there has to be a comparison between the current amount of the X marks on the right side and the amount of X marks from the row before. The same applies also for the left side. If the amount of one or both sides in the current row is higher as the amount from before, then the notification has to tell the user how much stitches have to be decreased and also on which side. On the other hand, if the amount of X marks is lower, than the user has to increase one or more stitches.

One more thing has to be considered, the purl side of the knitting is facing to the user. This means, the sides of the knitting is flipped vertical, so the right side of the pattern, is the left side on the needle bed. This is important for the notification's text and must be included into it.

5.2.3 Cast On and Bind Off

The other two major tasks, which have to be performed by hand, are to cast on and bind off. Both tasks are easy to integrate into the existing software, because they are independent from the pattern and are always at the same step in the process.

At the beginning the user always have to cast on. Therefore there will be a notification window at every start. This function will be inserted into the start knitting function from the initial AYAB software.

The same applies for the bind off. When the row counter and the total amount of rows are equal, the user has knitted the last row of the pattern, the notification window will tell the user to bind off.

5.2.4 Two Color and Lace Knitting

One challenge is the starting point of the initial AYAB software. At the moment the software only starts to interact with the knitting machine, if the knit carriage is set on two color knitting or if the lace carriage is in use. Therefore it would be one solution to include the instructions for these settings into cast on notification. If these settings are not in action, the system does not count the knitted rows and only performs plain stocking stitches.

5.2.4.1 Camera

To fulfill the requirement of lowering the entrance for novice users, one concept was to include a camera into the system. This camera should be oriented in a way that enables it to recognize the current settings. Then this information could be used to compare it to the setting the knitting machine must have to perform a certain technique. First I tried to archive this by connecting a kinect camera to system. Therefore, I implemented a module to access the kinect, which was connected to the laptop through USB. One main challenge was that kinect cameras are Microsoft products. As a result I was not able to have access to it with my Apple laptop. I was able to use the laptop's camera, but not the kinect. Furthermore, a kinect camera is designed for gesture recognizing and motion capture. Thus, I decide to use an IR camera for this task. By attaching IR reflective tape on the knitting machine, it is possible for the IR camera to recognize current settings of the knitting machine.

Since the implementation of the kinect module was already time-consuming and this enhancement did not have a high priority, I decided to move on to other tasks. Therefore is this a task for future work.

5.2.5 Help Section

Another solution to support the user is to integrate a help section that includes a collection of instructions of the manual tasks. More experienced users will already know how to perform most of the manual tasks, but novice user do not, therefore it is still a valuable enhancement. Furthermore, there are various techniques for most tasks and the help section will provide a selection of them. The different techniques have different outcomes, this increases the possible looks the knitting can adopt, so even more experienced users can have a benefit from the help section. At the beginning, the help section provides just some of the most common tasks and techniques, but if this work will be continued, the help section can be easily extended.

In order to enable access to the help section at every time while the user is working with the system, the help section will not only be accessible through the notification window, but also through the main menu bar at the top.

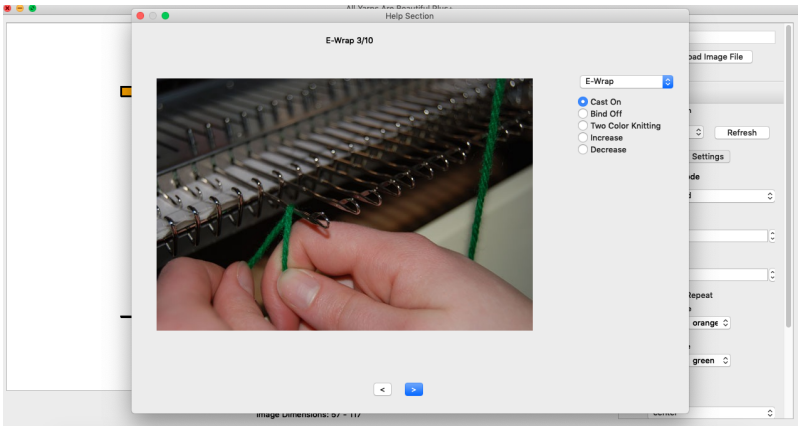


Abbildung 5.2: Concept of the Help Section

5.3 Models

To gain a better understanding of the interactions between the classes, I created a class diagram and discussed it in this section. In addition, I created two more diagrams, a sequence diagram and an activity diagram.

5.3.1 Static Model

The creation process involves various classes. One class is the CSV Converter class. It can be used to import a CSV file into the pattern creator class. The class can read a CSV file, as well as write given information into a CSV file. The pattern creator class itself creates a new pattern. To do so, the user will have to set the amount of pixels for the width, as well as for the length. Furthermore, with this class the user can insert different marks into the pattern. These marks have different meanings, such as decrease stitches, knitting a lace hole or knitting with the contrast yarn. The Calculator class can calculate the size of the knitting and transfer the amount of rows and stitches to the Pattern Creator class. The created pattern is also getting send to the Pattern Painter class, which converts the pattern into a low-resolution picture. This picture is sent to the origin AYAB software in order to get separated into its different lines and forwarded to the AYAB interface's firmware. The overview of the different classes and their relationships can be seen in figure 5.3

5.3.2 Dynamic Model

The two dynamic models I created are a sequence diagram and an activity diagram. While the sequence diagram covers the whole process from importing data to knitting, the activity diagram focuses on the process of creating a pattern. To simplify the sequence diagram (see Figure 5.4) the Calculator class were seen as part of the Pattern Creator class.

At the beginning, the user imports the data, which should be represented in a physicalization into the CSV Converter. The CSV Converter then converts the data in a way it can be read by the Pattern Creator, which creates a simple pattern by itself using some of the given data. Generating such a draft is a common procedure by other data visualization tools as well. Now it is possible to edit the pattern if wanted. During the editing the pattern will be displayed in real time to show all the changes. Afterwards, the user can choose to save the pattern or import the created pattern into the AYAB software. To import the pattern, it will first be converted into a low-resolution picture by the Pattern Painter class. Then the AYAB software sends the lines of the pattern to the AYAB firmware,

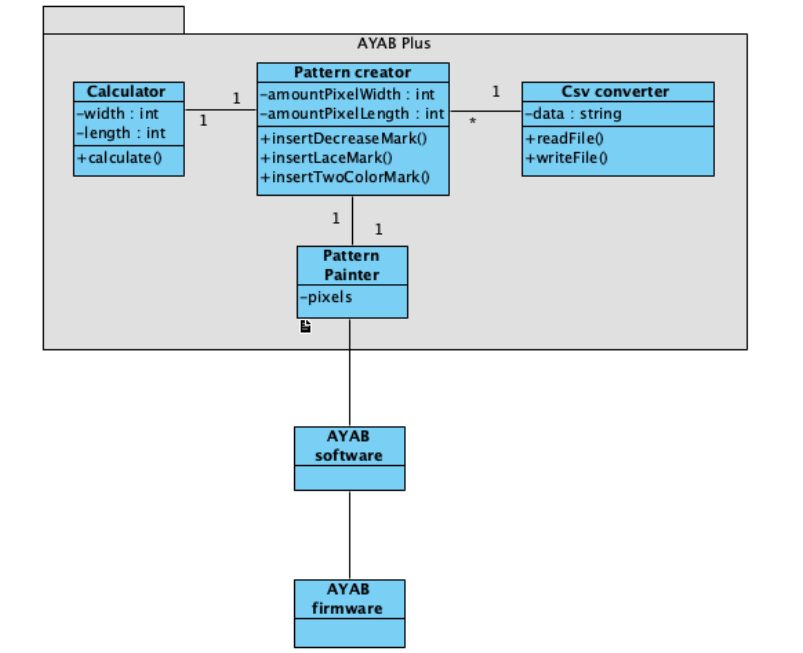


Abbildung 5.3: Class diagram

when the firmware requests the next line. The AYAB software will also provide feedback about the different tasks the user has to fulfill, like increasing stitches or which carriage to use.

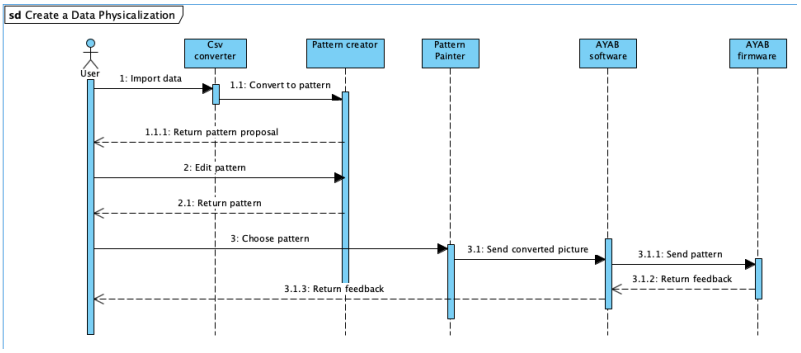


Abbildung 5.4: Process of knitting a pattern

Converting a data set into a physicalization only makes sense, if the beholder of the physicalization is able to recognize the represented data. This is why the user of the knitting machine needs a wide range of modifications to choose from to create a fabrication. Since the pattern is the origin for the knitting, it is worth to take a more detailed look at the creation process (see Figure 5.5).

When the user decides to create a new pattern there are two options, importing a data set or design a pattern without data. Importing a data set provides the user with a draft pattern Otherwise the software generates a blank pattern. In both cases the user is able to set marks into the pattern as long as the user wants to. After that the size of the real knitting gets calculated, based on the created pattern. If the size is not satisfying, the user can go back and adapt the pattern until the knitting will have the desired size. At the end of the process, the user can choose to save his pattern in order to knit it later or import the pattern into the AYAB software to knit it right away.

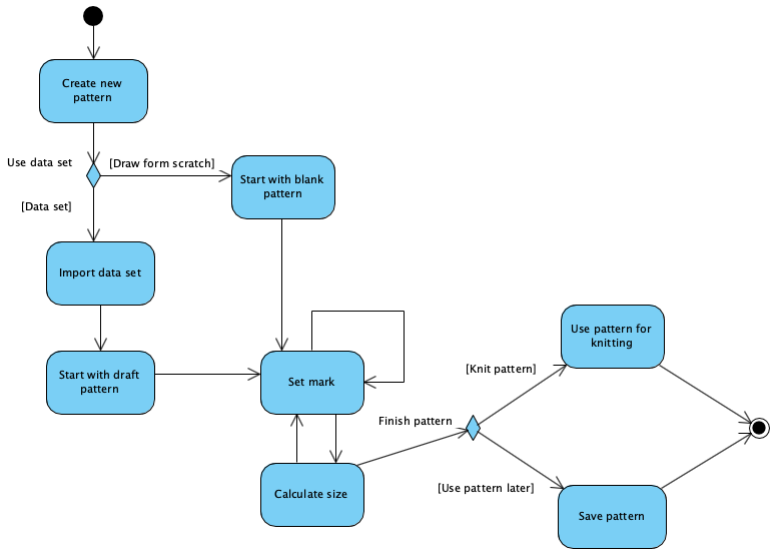


Abbildung 5.5: Process of creating a pattern

5.4 GUI Draft

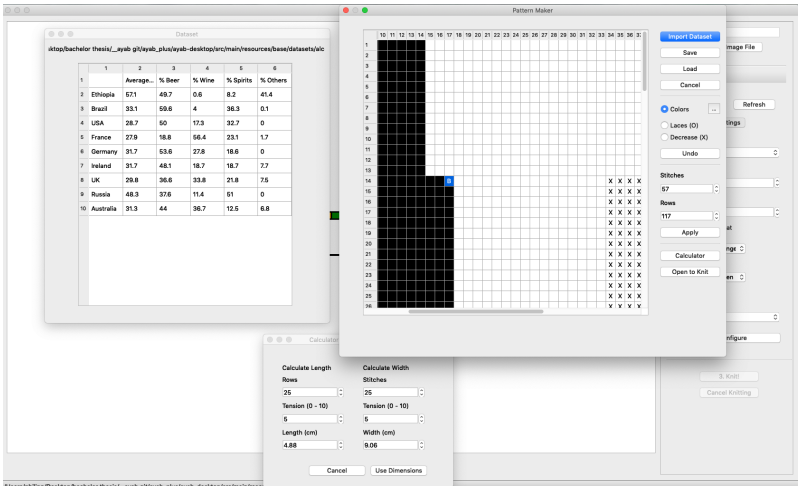


Abbildung 5.6: Concept of the Pattern Maker

Based on this concepts, I devised a draft of a GUI for the pattern creation process (see Figure 5.6). It includes a window with the imported data set, another window with the pattern, as well as a window with the calculator. The pattern itself comprises of a grid, which subdivides the pattern in its cells. Beside the pattern, there are buttons with different functionalities, which are divided in four groups. In the first group the button on the top is for importing the data set. The second one for saving the pattern on the disk. The third one for loading a previously saved pattern and finally a fourth button for canceling the process and returning back to the main window. Then next group of buttons consists of five buttons, their task is to set the different marks into the cells. The first is for coloring a cell, the button right of it opens a further window where the user can choose a color. Beneath there is the button to set a hole for a lace pattern. The mark for creating a hole is an O. The next button in this group is the button for decreasing, which is represented in the pattern by an X shaped mark. The last button can be used to undo the last action the

user has taken. In the next group there are input fields to change the amount of cells in length or width. The only restriction is that the width cannot have more cells than needles on the needle bed, which are 200 for the knitting machine KH-930. After the Apply button is clicked, the properties of the pattern will change to the set amounts. The first button of the last group opens the calculator window. The calculator window itself consists of six input fields in which the user can fill in the rows, stitches, tension, length or width. If one of the fields changes, the others will be new calculated. The calculated amount of rows and stitches can be transferred into the pattern creator. The last button is used to import the created pattern into the main window, so it can get knitted right away.

Some buttons provide functions, which are illustrated as well-known icons. These icons are displayed on the corresponding buttons in the draft, but yet these icons are not displayed when the software is running.

5.5 Summary

I showed that the user will be provided with a system that helps to cover the process of converting raw data to a physical data representation. The system will consist of a tool for pattern designing and tool for fabrication. The pattern design tool enables the user to import data and uses a table to create the pattern. The pattern itself will be converted to a picture that is readable by the initial AYAB software. With the support of the additional information from the pattern's marks, it is possible for the system to give feedback to the user about tasks, which have to be performed manually. Finally, the help section provides guidance if needed.

Even more functions for pattern design could be provided and more manual task could be supported, but the ones that are supported are the most common, important and basic ones.

My next step was to implement these worked out concepts into a functional software. Therefore, I will explain the main functions of my software in the next chapter.

6 Implementation

At the beginning of this chapter I present the technologies and frameworks, which I used for my work. In addition, I explain the software's architecture. Then I discuss functionalities and idiosyncrasies from the source code in more detail.

6.1 Technologies and Frameworks

First I explain Python and its aspects that are important to understand my source code properly. Then I discuss the framework PyQt5 and what I used from it to implemten K1M1.

6.1.1 Python

The existing AYAB software is written in Python and therefor I also used Python to ensure compatibility between the existing software and the enhancement. Furthermore, Python helps to achieve the non-functional requirement of being independent from a specific operation system, due to its platform independence.

Guido van Rossum developed Python as an easy to write and easy to learn programming language in 1991. It supports various programming paradigms like object-oriented, aspect-oriented or functional programming. I used object-oriented paradigm in my work, Python supports the common functionalities such as class attributes, inheritance or creating instances of classes. However, compered to other commonly use programming languages, Python has some idiosyncrasies. In this subsection I briefly discussed the ones that occur in my work's source code.

One of these idiosyncrasies is, for non static methods the first parameter of the method has to be `self`. The `self` parameter is a reference to the own instance. This is necessary to have access to the instance data. Methods can be recognized by the keyword `def`. Programmers who use older programming languages to write source code, have to specify the type of variable they want to create. In contrast, phyton uses a approach called intuitive interpretation. This means the type of the variable is unclear until the first time a value is assigned to the variable. After this the type of the variable cannot be changed anymore. Another idiosyncrasy is the structure. Python does not use curly brackets at the beginning and end a to define blocks. Instead whitespace indentation is mandatory. In addition, Python uses different terms for some data types. A collection of elements is a `list`, instead of an array as in other programming languages. While most arrays only can go from the first to the last element, a list can also be accessed from behind by using a negative integer in square brackets behind the name of the list. Example: `sample_list[-1]` will return the last element of the list `sample_list`. Dictionaries are associative arrays, means a key is assigned to the elements in a dictionary. To look up a specific element in the dictionary the programmer must use the key instead of an integer.

One library that I used in my work is the CSV library. This library provides interfaces to read and write CSV files. The `csv.reader` function returns on object, which can be used to iterate over a CSV file. The first parameter is a string, which represents the path to a CSV file on the disk. One optional parameter is the delimiter, this delimiter can be

any symbol that is used to separate the different data points in the file. Therefore, this parameter enables to iterate over the different data points within a line of the CSV file. The `csv.writer` function is quite similar. It returns a object that writes a CSV file. Its first parameter is a string with a path where the file has to be saved. As well as the reader, the writer has also a optional delimiter parameter, which is used to separate the different data points in the file.

6.1.2 PyQt5 Framework

The AYAB project uses PyQt5 to create their GUI. To ensure compatibility between the existing software and the enhancement, I created the GUI in my work with PyQt5 as well. In this subsection I introduce the most important features of PyQt5 that I used in my work.

Every GUI of an application has a main window, in PyQt5 this main window inherits from the QMainWindow class. The QMainWindow class itself, inherits from QWidget class, which is also the base for all user interface objects. The QDialog class is for more transient windows and will show up in the center of its parent window. It can be set as modal, which means the dialog window has to be closed to use the parent window again. By default, the dialog window can be closed with the Escape key. There is also a specific class if the user wants to open a file from the disk or to save it to the disk. This class is called QFileDialog. It receives a path of a directory as a parameter and displays the content of this directory. Then the user can choose a file or a different directory to open or save a file. After this have been done the window closes.

Various widgets can be placed on a window, the most common ones are buttons and labels. PyQt5 also provides a range of these widgets through classes like QPushButton, QRadioButton, QLabel, QSpinBox and even QTable.

QPushButton provides a command button, which can trigger a function by clicking on it.

QRadioButton provides a radio button combined with a text label. Usually the are some radio buttons on a window, but the user can active only one button at a time.

QLabels displays text or also images in a label.

QSpinBox provides a input field, which also has two arrows, one up and one down. The user can input numbers in it directly by entering from a keyboard or by clicking on the arrows to increase or decrease the number in the field.

QTable provides a table, which columns and rows can be adjusted in total with the `setRowCount` or `setColumnCount` methods as well as adding single rows or columns to the table.

QComboBox displays its items in a drop down menu, then the user can choose one of the items by clicking on it.

Developer use the Qt Creator application to design QWidgets to use them creating GUIs. Qt Creator offers various templates and configurations, which support the developer during the design. One way to use the designed widget is to save it as a `.ui` file, then the developer converts this file to a Python file with the UI code generator from PyQt5. The Python file includes a class with two methods, which are building and setting the GUI like designed in Qt Creator, the class can then be used normally.

To display an image in the GUI with QtPy5, the image has to be in a resource file. This resource file is a XML file with the suffix `.qrc`. It also can be converted into a Python file using a the Resource Compiler from PyQt5. Only after the file has been converted it is possible to use the images in it.

QtPy5 also provides a class to draw, this class is called QPainter, but it needs a instance from the QImage class to draw on. With the QPainter it is possible to draw circles, squares or lines. In addition, individual dots can be drawn on the image by transmitting the pixels to the function as a parameter. Furthermore, these functions can be connected to so-called paint events, such as clicking with the mouse. This way a user could draw freely on the image.

6.2 System Architecture

The software architecture has three tiers. In most software architectures these three tiers are presentation tier, logic tier and data tier. The K1M1 does not have a data repository and therefore there is no data tier, instead it controls the knitting machine. Thus the third tier is the knitting machine controller tier. There are some benefits to this architecture in my work. The agile procedure model leads to many changes throughout the whole development process. Due to the different tiers it is possible to work with one tier while leaving the other tiers unaffected. Moreover, the knitting machine control tier is comprehensive and already implemented by the AYAB team. Due to the tier architecture it does not have to be modified, which saves a lot of time.

The overview (see Figure 6.1) illustrates the workflow of the architecture. The user interacts with the GUI of the software, which is the presentation tier. Interactions with the GUI triggers functions from the logic tier. This, again, has an influence on the third tier, which controls the knitting machine. The knitting machine is also controlled directly by the user.

Furthermore, I used a second architecture model, namely the plugin architecture. My work provides the ground work for a broader project, which will continue even when my work ends. Therefore, it is necessary to build the software in a way that makes it easy for future developers to fit in more enhancements. One possible enhancement is to monitor the knitting machine to recognize its settings. This can be used to compared it to the settings, which the knitting machine should have to perform the upcoming task. To enable this plugin to interact with the existing software, there have to be a interface for this plugin.

6.3 Source Code

In this section I discuss selected details from the source code to explain some difficulties of my work.

6.3.1 Locking Cells

The main part of the pattern maker is its table. To stop the user from filling in marks, which the software is not able to process later, the cells have to be locked for normal input. The QTableWidget class does not have such a function, but a QTableWidget instance consists of QTableWidgetItem instances, which are represented as a cell. These items can be set so it is possible to modify them or not. A item has certain flags, which are managing the modification setting. These flags are bits, so to change the flags, the first step is to query the current flags. Then I used a xor bit manipulation with the current flags and the flag that

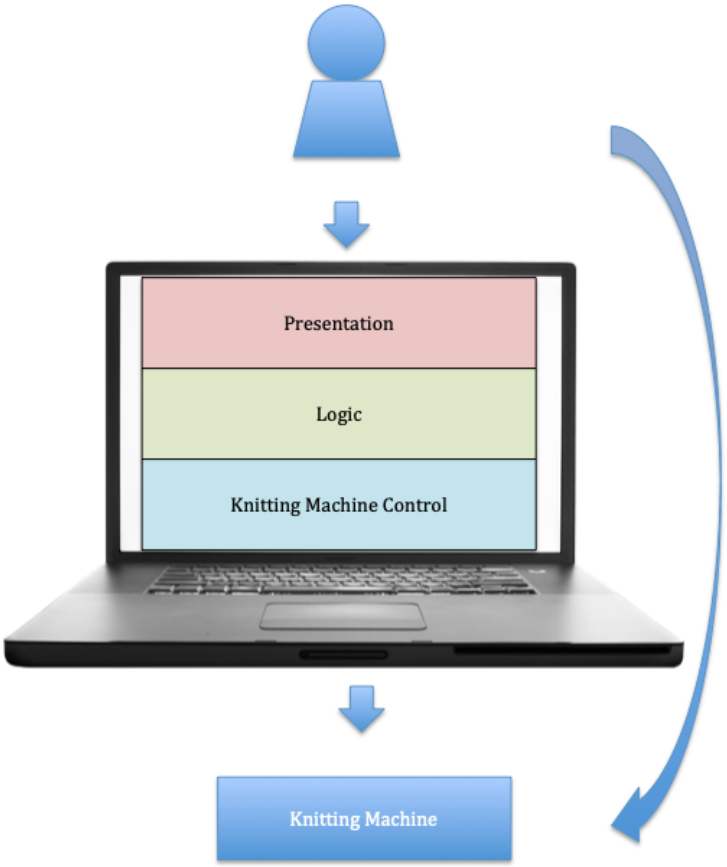


Abbildung 6.1: Overview of the Architecture

should change. Qt provides flags for this occasion, such as the `Qt.ItemIsEditable`, which I used in this method (see listing 6.1). In the method two while loops are run through to get every cell of the table and set their flags. The method is called during the construction of the pattern maker object to define and set all items. This will affect all items, which are existing at that time. When the amount of rows or columns get increased, the new items also have to be set as locked, therefore the method gets called again. To maintain the current data of the items, only the new items have to get set. Moreover, the setting of an item would be reversed if the flags would be set again the same way, therefore the already set flags have to be skipped

Listing 6.1: The lock_cells Method

```
    def lock_cells(self):
        row = 0
        while row < self.pattern_maker_ui.table_pattern.rowCount():
            column = 0
            while column < self.pattern_maker_ui.table_pattern.
                columnCount():
                if self.pattern_maker_ui.table_pattern.item(row, column)
                    is None:
                    self.pattern_maker_ui.table_pattern.setItem(row,
                        column, QTableWidgetItem(""))
                    current_flag = self.pattern_maker_ui.table_pattern.
                        item(row, column).flags()
```

```
                    self.pattern_maker_ui.table_pattern.item(row, column)
                        .setFlags(current_flag ^ Qt.ItemIsEditable)
                column += 1
            row += 1
```

6.3.2 Calculator

To know how big a knitting will be, is important for the user.I implemented a calculator for this purpose. After the calculator window opens, the user is able to change the amount of rows and stitches in the spin boxes on the GUI, as well as the setting of the tension. When the user changes one of there numbers, a event is triggered that calls either the method `calculate_length`, `calculate_width` or `tension_changed`. For the length the tension is multiplied by 0.015 and added by 0.12. The result of this gets multiplied with the numbers of rows. The same procedure also applies to the width, but here other figures are used. 0.0225 is multiplied, 0.25 is added and the result is multiplied by the numbers of stitches. The results are displayed immediately in the spin boxes of the length and width.

To calculate the used figures, I made various sample knittings with different tensions. Then I measured these knittings. I used the measurements to calculate the average increase of length and width, and utilized the result to implement the calculator. Due to the flexibility of a knitting the real size of the outcome is still hard to predict. Furthermore, other factors affect the size, such as the thickness of the yarn or the material the yarn consists of. Therefore, a large number of rows or stitches can cause deviations easily. However, the calculator still helps to gain a good impression on how tension, the amount of rows and the amount of stitches affects the size.

6.3.3 Filling in Marks

Additional to the already discussed marks in the concept chapter, there are also marks for the different colors, this makes the functions easier to implement. The colors that can be used are black and white, white also can be used to delete other marks. Furthermore, I also implemented red, blue and yellow, the three primary colors. By adding marks for the different colors, it is also possible to give feedback to the user during the fabrication process in which row the user has to change the contrast yarn. Furthermore, the created pattern can be saved as a CSV file. This works in such a way that the file is structured in rows and columns as well. The different marks are saved as letters on the same position in the file as in the table. This way the Csv Converter class can be used to save and load the patterns. The Csv Converter is also used to load a data set into the window that displays the data set in a table.

I implemented three radio buttons on the pattern maker window provided. These buttons allow the user to fill in only certain marks in the cells of the table. Each button for one kind of task. When the user clicks on a cell in the pattern maker window, an event is triggered. This event is connected to the `fill_cells` method (see Figure A.1), which checks what radio button is selected. Then the mark is inserted according to this selection. If the color radio button is selected, the background of the cell will also be changed according to the chosen color. By just clicking on one cell, only this cell is filled with a mark. In order to fill more cells at a time, the user can hold the left button on the mouse and select various cells at once. After clicking or selecting cells, the cell filling method calls another method to check whether any restrictions were violated. In that case, a notification window pops up to inform the user about the violated restriction.

6.3.4 Restrictions for Marks

To implement the identified restrictions that are described in detail in the concept chapter, I created a new class called mark checker. This class has one method, which is meant to communicate with the pattern maker class. This method, called `check_marks` receives three parameters. The first is a string that represents a mark, which needs to be checked to see if it can be inserted into a cell. The other two parameters are the number of the row, as well as the number of the column of the clicked cell. Then the mark gets checked to call the right method.

When the mark is an O, the method `check_lace` is called. This method is the biggest and most complicated one, because lace knitting has many restrictions. First this method checks if there are color marks in the row. To do so, it goes through a while loop to address every cell in the given row. To facilitate extending the range of colors to choose from in the pattern maker, the mark is checked if it is not any other mark than a color mark, instead of the other way around. If the content of a cell is a color mark, the loop is interrupted and false is returned. The text of the notification window is also edited, so it informs the user about the violated restriction. After that the table's amount of rows is taken modulo two, in order to check if the amount of rows is even or uneven. If the amount of rows is even, but the row of the mark is uneven, false gets returned and the text of the notification window is edited. The same applies if the the amount of rows is uneven and the row itself is even. At last, the cells next to the selected cell are checked if they contain an O mark. In this case the notification's is modified and false gets returned.

When the mark is an X, the `check_decrease` method is called (see listing 6.2). This method checks if the X mark is at the edges of the pattern, by comparing the received number of the column to zero and to the total amount of columns. Only if none of these conditions apply, the cells next to the selected cell are checked if they contain an X mark. This prevents the software to try to access an element, which is out of bound. In case all these conditions do not apply, the text of the notification window gets modified and false is returned.

Listing 6.2: The check_decrease Method

```
    def check_decrease(self, row, column):
        if column == 0:
            return True
        elif self.pattern_maker.get_table().columnCount()-1 == column:
            return True
        else:
            if self.pattern_maker.get_table().item(row, column - 1).text
                () == "X":
                return True
            elif self.pattern_maker.get_table().item(row, column + 1).
                text() == "X":
                return True
        self.pattern_maker.note_text = "X marks can only be set next to
            edges or\n other X marks"
        return False
```

When the mark is neither an O or a X mark, it has to be a color mark. Therefore, the method `check_color` is called. First, the color white, which is just a empty string, is sorted out. I implemented it this way, because this mark is also used to delete other marks. For this reason this mark can be placed in every cell. Like in the `check_lace` method a while loop is undergone to check if there is already an O mark or another color mark in this row. This is done by checking if every other cell is empty, an X mark or the received mark.

If this condition does not apply to one cell, the loop is interrupted, the notification text is modified and false is returned. The source code of this methods can be found in the appendix (see Figure A.2).

6.3.5 Pattern to List

To provide the ayab class from the initial AYAB software with additional information about the manual tasks, the created pattern has to be saved in a way that it is represented with rows and columns. Like saving the pattern as a CSV file, the pattern is also transferred into a list with rows, columns and the content from the cells. To do so, a method was implemented with two local list variables (see listing 6.3). One gets returned and the other is just temporary used to create the first one. Two nested while loops are passed through, the temporary list is created blank in the outer loop and receives content from the cells at the inner loop. At the end of the outer loop, the temporary list is appended to the list that is returned. This way a two dimensional list is created, which can be used to generate feedback for the user during the knitting process.

Listing 6.3: The pattern_to_list Method

```
    def pattern_to_list(self):
        pattern_as_list = []
        row = 0
        while row < self.pattern_maker_ui.table_pattern.rowCount():
            column = 0
            templist = []
            while column < self.pattern_maker_ui.table_pattern.
                columnCount():
                templist.append(self.pattern_maker_ui.table_pattern.item(
                    row, column).text())
                column += 1
            pattern_as_list.append(templist)
            row += 1
        return pattern_as_list
```

6.3.6 Pattern to Image

The initial software needs a low-resolution image in order to generate instructions for the knitting machine. Therefore, the created pattern has to be converted into an image. To do so, I implemented the pattern painter class, which receives the amount of rows and columns from the pattern and creates a QImage object with the same dimensions the pattern has. The class has a method called `draw_points`, this method uses a QPainter object to draw on the image by coloring pixels, which represent cells with coloring marks. Depending on the mark, the color with which the pixel is drawn changes. For the comparison between the mark and the color a dictionary called `marks_dict` is used, which stores the marks as keys and the colors as values. After the image is drawn, it is saved temporary in the pattern directory of the AYAB software.

Listing 6.4: The draw_points Method

```
    def draw_points(self):
        for row in range(self.image_height):
            for column in range(self.image_width):
                mark = self.pattern_maker.get_table().item(row, column).
                    text()
```

```
                if mark == "B" or mark == "R" or mark == "U" or mark == "
                    Y":
                    self.set_brushcolor(self.marks_dict[mark])
                    self.painter.setPen(self.brushColor)
                    self.painter.drawPoint(column, row)
        self.update()
        file = self.app_context.get_resource("patterns")
        file += "/temp.png"
        self.image.save(file)
```

Then the ayab class method `load_image_from_string` is called. The path of the temporary image file is given as the parameter for this method, which transfers the image to the main window, enables the configurations and allows the user to start knitting. Then the temporary image is removed from disk, using the `remove` method from the os library.

6.3.7 Feedback Algorithms

The first algorithm to give the user feedback during the knitting process, is the simplest one. To implement it, the `start_knitting_process` method from the ayab class is extended. The method then generates a new notification window and modifies the text of the label from the window. This way the user receives the first feedback right at the start, which tells the user to do a cast on. The notification window also has a button to open the help section. Moreover, the notification window is used every time the user receives feedback.

For the second algorithm, I implemented a new method called `knit_with_enhanced_pattern`. It is responsible for giveing feedback if the user has to change the contrast yarn, use the lace carriage and if it is necessary to increase or decrease stitches. Furthermore, the user is informed where to increase or decrease the stitches. After transferring the image into the initial AYAB software, the list with the marks is allocated to the `pattern_as_list` class attribute from the ayab class object. Every time a new row is knitted, the method receives the current amount of knitted rows, as well as the total amount of rows of the pattern. The first step is to go through the current row in the `pattern_as_list` attribute and count all X marks on the left side. Then all X marks on the right side of the pattern are counted. The results are compared to the results from the last row. Differences between the results mean that the user has to increase or decrease stitches, therefore the text of the notification window is modified. To find out if the user has also to change the contrast yarn, the current row is checked on coloring marks as well. When there is a coloring mark in the row, it is compared to the last founded color mark. A different mark means that a message is generate about what color the new contrast yarn has to be. Then this message is added to the label text of the notification window, which is then displayed on the screen. In the same loop that checks the current row for color marks, also checks if there are marks for lace knitting. When there is a mark for lace knitting in the current row, but not in the previous row, a message is added to use the lace carriage for the next row. While there are additional O marks each row, no new message will be added. As soon as there is no O mark in the current row anymore, the notification window is displayed with the information to stop using the lace carriage. This is achieved by comparing two boolean variables with each other. One represents the status of the current row, while the other represents the findings from the previous row. A important issue that I had to consider was that during the knitting progress the knitting machine knits from bottom to top. That means, while the transferred number of the current row counts from zero to the total amount of rows, the real number of the current row counts from the total amount of rows to zero. Therefore, the real number of the current row has to be calculated at the beginning of the method.

The last feedback algorithm is implemented in the `knit_with_enhanced_pattern` method as well. To inform the user to perform a bind off on the right time, the total amount of the rows is compared to the current row. When the total amount and the current amount of rows are the same, the notification window appears with a modified text, which tells the user to bind off.

6.3.8 Help Section

The help section can be entered through the main menu bar or through the notification window. The GUI has various radio buttons on the left side, the functionality of each button and the other elements are defined in the GUI's controller class. These radio buttons are representing different manual tasks and by clicking on them a guideline on how to perform such a task is displayed in a label. To technically display an image in label, the path of the image and its name has to be given to a QPixmap object. This object can then be used to display the image in the label, but only if the image is listed in the resource file. A manual task can be performed with different techniques, therefore the chosen task is saved in a variable, which helps to display the right techniques for the manual task in a combo box. Depending on the task, the items of the combo box are customized. To do so, first all items of the combo box are deleted, then new items of the technique are created and filled into the combo box. The name of the image, which is required to display it, consists of the name of a technique, as well as a number from one to ten. This number indicates the position of the image within the order of the image gallery and is allocated to a variable called `image_number`. When the button for the next picture is pressed the image number is increased by one and a new pixmap object is created, which replaces the previous image in the label. The same applies for the button that allows the user to go back in the image gallery, but the image number is decreased by one instead of increased. By using modulo, the image gallery starts from the beginning if the image number is more than ten or less than one. In order to maintain the user's overlook, the current technique and image number, as well as the total amount of images is displayed in another label above the label with the image and is updated along with the image. The entire help section is designed so that it can be easily expanded to display additional tasks and techniques.

6.3.9 Plugin Camera Interface

To realize the plugin architecture, I implemented an interface for future work. This interface helps to integrate a future function that allows to check the knitting machine settings with the help of an IR camera. Python does not have a formal interface contract, therefore a class `AbstractCameraClass` is implemented with the method `calculate_current_setting`. This method is called in the same method, which checks the pattern list for marks. When a mark appears that indicates a task, which requires a change of the settings, the current settings are queried and compared with the settings that are needed to perform the task.

6.4 Summary

I showed that I implemented the worked out concepts in the program language Python. I considered the architecture guidelines. Therefore I implemented interfaces and expendable function for future work, as well as adapt the three tiers, namely GUI, logic and knitting machine control. I did not change anything in the knitting machine control tier, but enriched the GUI and logic tiers with more classes. An overview of all classes can be found in the appendix (see list A.4). I used the PyQt5 framework to create and convert GUI

classes. The non-functional requirements of expandability is given through architecture and through the allover structure of the created functions.

In the next chapter I will discuss the evaluation of the implemented functions. I have a look if they fulfill the analyzed requirements and functionalities.

7 Evaluation

To evaluate a build system can be challenging, there are various ways to assess the value of a new system. In order to be able to choose the right evaluation technique, it is important to have a look at the aspect to be examined and what the goal of the system is. Ledo at al. [15] identified four different evaluation techniques, which are commonly used in HCI community. In this chapter, I discuss how I used two of these techniques and why they are appropriate to evaluate this work. Furthermore, I discuss the findings as well.

7.1 Evaluation by Demonstration

The goal of this work is to build a system that can be used to create physical data representations. Therefore a method that aims at the question what can be build with the system, is a suitable approach to evaluate the value of the system. Evaluation by demonstration is such a method. By creating novel examples of knitted data physicalizations, it is possible to demonstrate the ability a potential user has to have and what can be achieved by using the system.

I used the paper prototype beer coozies from the analysis as a template for new created physicalizations. I chosen two prototypes, which are representing different approaches to encode the data. In this way, I was able to evaluated the variation of the system. The first prototype combines two approaches to represent data points. On one hand, the overall shape is a bar chart that represents data. The width of each bar is the same, but the lengths are different. On the other hand, it creates a second bar chart by using a second color to represent different bars. For this colored bar chart, the length of the shortest bar from the shaped bar chart, is the absolute height. Furthermore, there are letters on each bar for the name of the country. I omitted the letters on the data physicalization, because of the knitting's small size (see Figure 7.1a). The second prototype uses different colors that are representing different data points, to write the name of countries. The maker divided the country names in three groups and placed shifted all over the beer coozy. This means there are more than two colors in some rows, no matter in which direction the coozy is knitted. Therefore the system is not able to knit this prototype, but I adopted the basic idea of colored country names to knit a second data physicalization (see Figure 7.1b). At the end I made a third data physicalization. For this one I used lace knitting to evaluate this technique as well. The physicalization has lace holes in it, which I used to put through a string in a different color as the main color. This way the yarn created a bar chart (see Figure 7.1c). Moreover, I designed it based on what I learned from the design challenges of the past approaches.

However, one challenge of this evaluation technique is that the creation is not performed by a potential user. Therefore it is hard to proof if the system really lowers the entrance. Moreover, the technique shows what is possible, but does not say much about how well it's usable and usability is one of the non-functional requirements, inter alia, to lower the entrance for novice user.

The evaluation process is divided into two parts, Converting data and hybrid fabrication. This is because the pattern design part is mostly independent, while the knitting process relates to the initial AYAB software.

7.1.1 Converting Data

In this subsection, I discuss how I created the patterns of the prototypes. Then I talk about the findings during that process.

7.1.1.1 Practices

(a) The Prototype with the Bar Charts

(b) The Prototype with the colored Country Name

(c) The Prototype with the Lace Knitting

Abbildung 7.1: The three created Beer Coozies

At the start of the creating of the first beer coozy, the interface is still separate from the knitting machine. To facilitate the design process. In the complete hardware setup, the laptop is placed behind the knitting machine in order to have a good view of the laptop's screen (see Figure 7.2). This way it is easy to recognize the feedback notifications during the knitting process, but this placement of the laptop is obstructive during the pattern design process.

After the start of the software, the New Pattern option in the menu bar is used to open the pattern creator window. A click on the Import Dataset button opens another window, which displays the files and directories on the disk. It only allows to select CSV files. When a CSV file is chosen, the window with the files closes and another window pops up with the file's content displayed in a table. The content was arranged like saved in the file. In the next step I measured the girth of the can and opened the calculator window. Then I increased the amount of rows until the right length was displayed in the length spin box. I transferred the amount of rows and stitches to the pattern creator window and started to use X marks to design the overall shape of the knitting. After that, I used the yellow color marks to design the colored bar chart. I had to use a additional calculator to calculate the right amount of stitches that have to be knit in the contrast yarn. Finally, I saved the pattern on disk, in case something goes wrong during the knitting process and transferred the pattern to the main window of the initial AYAB software.

The design process for the second knitting went similar. The steps of the importation of the CSV file were the same. I did not had to measure the can again, I could use the amount of rows and stitches from the last design process. The main difference was that I did not use the X marks, but instead I needed to use all of the available color marks to design the patter. Moreover, I had to used a external calculator again and also saved this pattern on disk as well. This time I did not start the knitting process right away, instead I closed the software and restarted it for knitting later that day. To do so, I opened the pattern maker

window and used the load function the import the pattern into the pattern maker and then transferred it to the main window of the initial AYAB software.

By the time I designed the pattern of the third physical data representation, I was already faster than the first two times, because I could build on my already acquired experiences. The only difficulty was to obey the restrictions lace knitting has. Still I was done with the designing process quickly, saved the pattern on the disk and transferred the pattern to the main window.

7.1.1.2 Findings

During the design processes I recognized some drawbacks of the system and also advantages, both were discussed in this subsection. Using a external calculator was one of the drawbacks, which the system has. This did not come up during the analysis, but has to be a part of the future work, as it is obstructive for the user's workflow. After loading a saved the pattern into the pattern maker, the colors are not displayed, only the marks of the different colors. This can irritate a user and should be fixed. It turned out that the calculated length of the knitting does not fit the real size of the knitting. The basis how the length was calculated worked but the numbers by which it is multiplied and added may have been incorrectly measured. Due to the soft and malleable texture of a knitting it is hard to calculate its real size. Furthermore, neither the texture of the yarn, nor the effects of two colored or lace knitting has not been included in the calculation. Therefore the calculator must be revised again. Despite its errors, the calculator could be used to calculate the correct size for the second and third beer coozy, which shows that it is still a valuable part of the system.

However, I was able to import a CSV file into the pattern maker, which helped to match the data points to the physical variables. This matches one of the must have requirements for the data converting process. Moreover, it is possible to save the created pattern as a CSV file on the disk. Just as important as saving the file, is to load it again into the pattern maker. This also works, but with the already discussed drawback of losing the colors. The amount of rows and stitches are adjustable on two ways, on one hand it can be changed on the pattern maker window itself or through the use of the calculator.

On the whole it is possible to create patterns and offers a wide range of possibilities to design these patterns.

7.1.2 Hybrid Fabricating

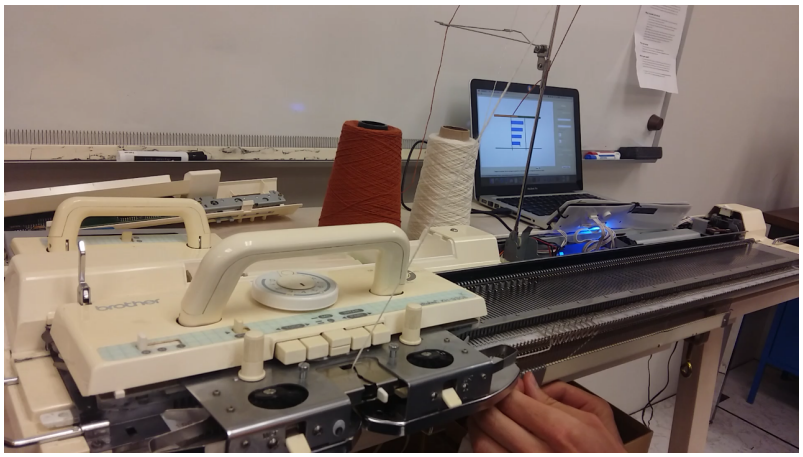


Abbildung 7.2: The Setup of the K1M1

7.1.2.1 Practices

After I transferred the pattern design for the first data physicalization into the initial AYAB software, I started knitting. The first feedback appeared right after I pressed the start button. The notification window was displayed with a message to cast on. I also had the option to open the help section window, which I did. I chose the cast on section and fulfilled this manual task while being guided through the technique by a series of pictures. Then I used the knit carriage to knit the rows until I received some more feedback about increasing stitches or later decreasing stitches. I noticed that sometimes a few needles were not in the right position for two color knitting, so I corrected them manually. At the last row the notification window showed a message to bind off, but before binding off I attached the stitches from the first row to the activated needles and knitted one additional row. This way I gave the knitting the form of a loop. Then I used the help section again to perform the bind off task.

The main differences in the process of creating the second prototype, was instead of increasing or decreasing stitches, I had to change the color of the contrast yarn. The beginning and the end was still the same, therefor I did need to use the help section again for these tasks.

For the third prototype I first had to cast on and then place the knit carriage on the right side of the needle bed. I put its settings on plain knitting and attached the lace carriage on the left side of the needle bed. I first used the lace carriage to transfer the first stitches and then the knit carriage to knit rows. I decided to keep this workflow, even when there was no O mark in the pattern. This was more convenient for me, because I did not had to switch the settings every couple of rows. I still received feedback to use or stop using the lace carriage. At the last row I created a loop and did a bind off again. Finally, I put the yarn through the holes.

7.1.3 Findings

This subsection is also separated in drawbacks and advantages that the system has to offer. Some of this disadvantages are not based on the part of the software I created in this work, but I still discussed them here for the sake of completeness.

One of these drawbacks from the initial AYAB Software is that the contrast between white and light blue is not high enough to be detected by the system. In this case, the system does not set the needles right. Even light blue is not part of the color palette from the pattern creator, it is still displayed in the main window because of the following drawback. The initial AYAB software also has trouble to display the right color in its main window. For example blue turns into red and red into blue, while yellow turns into light blue. On the other hand, the software still gives the right feedback. The message in the notification tells the user to change to the contrast yarn to the color from the initial pattern. It also turned out that because of the soft material, the shape of a bar chart was not very successful. The longest bars are not stable and therefore roll over. Furthermore, the edges of the knitting are curling, which makes it hard to recognize height differences of the bars. This drawback is more a pattern design issue and therefore a user decision, which can only be addressed by the software, but not solved. In addition, I implemented the feedback for lace knitting to use it when there are mixed techniques in one knitting. It informs the user when to start and stop using the lace carriage. Due to this the notification window is redundant if a user only wants to perform lace knitting. Furthermore, the user has to stop knitting and close the notification window every time to see the pattern in the main window again.

One requirement was that the help section can be opened at any time. This requirement

Tabelle 7.1: The average Duration Time of the different Methods in Seconds

Methodname	Average Duration Time
Fill Cells	0.00018
Choose Color	0.00902
Open to Knit	0.50620
Check for Feedback	0.00545
Import	4.15858
Open Calculator	0.01397
Apply Clicked	0.11851
Save Clicked	2.19636
Load Clicked	1.68798

has been met. Moreover, the software suggests to open it at the right time, namely when the user has to perform a manual task. The software also gave feedback about what kind of task has to be performed. For example, when I had to increase stitches, the software opened the notification window with a message where I had to increase stitches and how many. By the arrangement of the hardware the instructions were good to see. All in all, it is the software fulfills the functional requirements, which I identified as must haves, as well as could have requirements.

To evaluate the quality of the software, the next section discusses the non-functional requirements.

7.2 Evaluation by Technical Performance

To test how well the software works, benchmarking against thresholds is a common evaluation method [15]. It is used to quantify technical performances, as well as the quality of the software and its components. The technical performance test was subdivided into various characteristics, which were as followed.

Reaction Time how long does it take for the software to react to user input?

Usability is the software able to be used by novice user?

Robustness how fault tolerance is the software?

Manageable how easy is it to extend the software by other programmers for future work?

Lightweight how much CPU does the software use and how much working memory allocation?

7.2.1 Technical Findings

7.2.1.1 Reaction Time

Benchmark: The software should be react under 0.5 seconds.

I used the `time` method from the time library, to measure the time that is passed between the start of a method until its end. For methods that access the disk, I measured the

duration between start and appearing of the file selection window. The measured time was printed on the terminal. Opening the calculator, choosing a color, filling cells or adjusting the amount of rows and columns took around or less than 0.1 seconds in average. These are also the most used functions and therefore the user expects a fast reaction time. Furthermore, the method for giving feedback during the knitting process, took also less than 0.1 seconds. The method of transferring a pattern to the main window takes longer or shorter depending on the size of the pattern. Nevertheless, in average it takes a little more time than 0.5 seconds (see table 7.1). When the file selection window is opened, it will take longer than 0.5 seconds. Especially, when the disk is accessed for the first time, the function took up to five seconds, after that it took about one to two seconds. However, this is due to the hardware of the laptop, the software itself does not have much influence on it. Other computers with more advanced hardware could fulfill these functions faster.

7.2.1.2 Usability

Benchmark: The knitting machine should be usable without its printed guide line.

Through the help section it is not essential to use the knitting machine's printed guide line to fulfill the manual tasks. Due to the structure of the initial AYAB software, it can be difficult for the user to recognize all functions directly. Therefore, a short introduction is still needed. This can be overcome by a new structure of the overall software.

7.2.1.3 Robustness

Benchmark: The software should not crash during operation.

During all three processes for creating prototypes, all functions and possible user input has been tested. The software only crashed one time, when the USB cable was plug out during the knitting process. Otherwise the software runs stable. The input fields only allow input that the software can process. This applies to the table, the file selection window, as well as to the spin boxes.

7.2.1.4 Manageable

Benchmark: The software's source code should be separated in different modules and tiers. A module should not have more than 300 lines of code.

While the initial AYAB software has two main modules with more than 600 lines of code, the biggest module in my work has only 269 lines of code and consists of two classes. This module provides the pattern maker function and also controls its GUI window. It has access to several smaller modules, such as pattern painter or csv transformer. This way it is also possible to replace this module without losing these other functions. The smaller modules have about 100 lines of code.

7.2.1.5 Lightweight

Benchmark: he software should not use more than 10% CPU and less than 100 Megabyte (MB) memory allocation.

During the process of designing and creating a physicalization, the laptop's activity indicator was used to observe the statics. The used laptop has 4 Gigabyte (GB) working memory and a 2.5 Gigahertz (GHz) CPU. The values of CPU use fluctuated between 1.0 % when the there were no interaction, to 7.4 % during function calls. Furthermore, the working memory allocation fluctuated between 43 MB to 82.5 MB.

7.3 Summary

I showed that my software and the system that includes it, are meeting most of my worked out requirements. I used two common evaluation techniques to demonstrate this.

The next section will be a summary of the whole work and will also be a look out to future work.

8 Conclusion and Future Work

Finally, this last chapter once again summarizes the entire work and examine its strengths and weaknesses. In addition, it provides an overview of possible tasks for future work.

8.1 Conclusion

With the software I implemented, it is possible to take a data set and design a pattern, which properties are mapped to data points from the data set. Then this pattern gets used by a knitting machine to fabricate this physical data representation. This happens in a way that provides a meaningful experience by combing the strengths of human and machine. To be more specific, the user contributes to the fabrication process by fulfilling tasks, which are helping to discover the data, while the machine performs tasks that would distract from discovering the data. To fulfill this task, I divided it into two parts. The first part is to convert data into a pattern. The second part supports the user during the fabrication process.

While these tasks seem like an easily manageable project, they turned out to be a collections of even more subtasks, which each could consume months of work to be satisfyingly done. Due to the experimental nature of the whole work, it was already a complex challenge to work out requirements. While the transformation from a data set to a visualization for a screen is a known field, the knitting of data is an unconventional way to fabricate a data physicalization. Moreover, hybrid fabrication, a mixture of manual and digital fabrication, is also a relatively new research field in the HCI community.

Therefore, my first step was to find out how people would encode data into a knitted data physicalization. This knowledge was the basis to build a pattern maker function and subsequently, implement functions that support tasks for fabricating these patterns. The centerpiece of the pattern maker is its table, which represents a pattern. Using a table has benefits and drawbacks. One drawback is that it only imitates a paint program, but cannot provide all functions of it. For example, filling in various marks at one time leaves the last marked cells unfilled. The undo functionality of these fillings only works on the last filled line of cells and not on all recently filled cells, zooming in and out is not possible as well. Still it is the best approach, because of its benefits. These benefits are that the table helps to display the single pixels more precisely and makes it possible to insert additional information into the pattern in the first place.

The functionality for giving feedback to the user is based on the list, which provides different marks from a created pattern. Nevertheless, the mapping of tasks to defined points on the pattern makes it unfeasible to use pattern modifying functions from the initial AYAB software. Furthermore, the hand manipulating techniques, which are supported through feedback and included in the help section, are limited to the most common, important and basic ones. This does not inhibit a user from performing other manual tasks, but in this case the user receives no support from the software. The major benefit of the list is that it is used to give detailed feedback. For example, how many stitches have to be increase or decrease and on which side, as well as which color the contrast yarn have or when to use the lace carriage.

This shows that the software is already able to provide a range of variations to work

with. However, this work should be seen as a basis for future work, to provide a benefit for every possible user.

8.2 Future Work

The calculator is a good example for unexpected difficulties. It demands much more work than just adding a constant for every row and to get the right size of a knitting as a result. A lot of different factors have to be taken into account, such as the used tension, what kind of yarn is used, amount of rows, amount of stitches, what kind of stitch and if lace knitting or two color kitting is used. To cover all these factors in one calculation was out of scope for this work, but can be a project in the future. Besides a special calculator for the size of the knitting, a calculator for basic mathematics has proved helpful. During the pattern design process, the user often needs a to calculate the right amount of stitches or rows. Especially for percentage based data points or data sets with big figures. Then the user has to calculate the ratio between the data points and the stitches or rows. This means the current calculator should be expand to perform these mathematical operations as well.

The techniques that I included in the help section are illustrated partly by photos from websites or from the printed guide line. To raise the understanding and usability, these quickly taken photos could be replaced by better illustrated photos with an useful description. Another option would be to film the task and include it as a Graphics Interchange Format (GIF) or video.

Furthermore, to exploit the full potential the whole system can accomplish, more techniques to create physical variables should be supported. For example, the tension can be changed during the knitting process. That means it is possible to knit rows with different space between them. On the other hand, this complicates the calculation of the knitting's size. Another example is using different stitches. By dropping a stitch on purpose and repairing it, a user is able to create a purl stitch. This offers new possibilities to create different stitches, like the garter stitch.

A more complex approach to enhance the system is to involve a IR camera and a projector. The IR camera can be used to detect the current settings of the knitting machine. These settings could be compared to the settings that are necessary to perform a certain task. Then the project could highlight what have to change. Moreover, the projector could be used to identify errors in the activated needles. By highlighting the needles that should be activated, the user can compare if these needles really are activated. This procedure can support the error-prone knitting machine. This is a valuable approach to strengthen the feedback functionality. Therefore, I already implemented an interface for this enhancement.

Another source of frequent errors is the yarn guide. On one hand, when one of the yarns have too much tension. It could snap and the knitting process cannot be continued. On the other hand, when one of the yarns does not have enough tension, it can be tangled up in a carriage. The same applies when the knitting itself does not have enough weight. It can slip off the needles and also be tangled up in one of the carriages. In order to prevent this, the knitting machine's claw weights, as well as the yarn guide can be equipped with sensors. These sensors could monitor the tension and send a warning if the tension is to high or to low.

Finally, a couple of smaller changes can help to increase the usability. The initial AYAB software's GUI is designed to guide the user through its functions. This is done by number the steps that have to be performed to start the knitting process. In addition, these steps are in a vertical order to clarify their sequence. The extended functions from my

software can be called through the main menu bar and is therefore not included into the initial software's process sequence. This makes it unclear for novice user where to start. Therefore it would be a sensible task to design the whole GUI in such a way that the new functions are perceived as part of the software's sequence. I could not implement the functions properly, so the icons on the buttons are displayed while the software is running. Solving this problem would also improve the understanding of the software. In the calculator window, the tension input can be changed from a spin box to a vertical slider. This gives the user a better overview and understanding of whether the selected tension is high or low. In the choose color window, more colors can be included to choose from. This also means to extend the range of marks for the feedback. Moreover, instead of choosing from a prescribed set of colors, the user could define its own colors. In order to archive this, there have to be a generator for random marks that transfers their meaning to the other classes. Another option to handle color selection is that the user can only choose from colors, which are available in the fablab. This could also mean, that the main color has to change. At the moment the main color in the pattern is white by default. To provide the user with a better preview what the finished knitting might look like and how the real colors look in combination, changing the main yarn in the pattern maker is a important component.

Due to the agile procedure model, even more ideas can occur in the future while working on with this project. This shows the dimensions of the whole project and that this work was only the beginning.

Literatur

[1] Manuela Aparicio und Carlos J. Costa. „Data visualization“. In: *Communication Design Quarterly Review* 3.1 (2014), S. 7–11. DOI: 10.1145/2721882.2721883.

[2] Jacques Bertin. *Semiology of graphics: diagrams, network, maps*. University of Wisconsin Press, 1967/83.

[3] Kathrin Braun. *'„Verspätungsschal'“ im Internet versteigert - mit dieser Summe hatte wohl keiner gerechnet*. [Online; accessed 2019-08-27]. 2019. URL: https://www.tz.de/muenchen/stadt/mvv-muenchen-org81486/verspaetungsschal-im-internet-versteigert-mit-dieser-summe-hatte-wohl-keiner-gerechnet-11057429.html.

[4] Stuart Card, Jock Mackinlay und Ben Shneiderman. *Readings in Information Visualization: Using Vision To Think*. Jan. 1999. ISBN: 978-1-55860-533-6.

[5] Laura Devendorf und Kimiko Ryokai. „Being the Machine: Reconfiguring Agency and Control in Hybrid Fabrication“. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2015, S. 2477–2486. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702547.

[6] Liza Eckert. *WHAT IS A TEMPERATURE BLANKET?* [Online; accessed 2019-09-11]. 2016. URL: http://www.lionbrand.com/blog/what-is-a-temperature-blanket/.

[7] Flowingdata. *Blanket pattern visualizes baby's sleep data*. [Online; accessed 2019-08-05]. 2019. URL: https://flowingdata.com/2019/07/25/blanket-pattern-visualizes-babys-sleep-data/.

[8] CNN Gianluca Mezzofiore. *Woman records train delays by knitting them on a scarf*. [Online; accessed 2019-08-05]. 2019. URL: https://www.cnn.com/2019/01/07/europe/woman-knitting-train-delay-scarf-intl-scli/index.html.

[9] Susan Guagliumi. *Hand- Manipulated Stitches for Machine Knitters*. Create Space Independent Publishing Platform, 1990.

[10] Trevor Hogan, Samuel Huron, Pauline Gourlet, Uta Hinrichs und Yvonne Jansen. „Let's Get Physical: Promoting Data Physicalization in Workshop Formats“. In: Juni 2017. DOI: 10.1145/3064663.3064798.

[11] Yvonne Jansen und Pierre Dragicevic. „An Interaction Model for Visualizations Beyond The Desktop“. In: *IEEE transactions on visualization and computer graphics* 19 (Dez. 2013), S. 2396–405. DOI: 10.1109/TVCG.2013.134.

[12] Yvonne Jansen, Pierre Dragicevic, Petra Isenberg, Jason Alexander, Abhijit Karnik, Johan Kildal, Sriram Subramanian und Kasper Hornbæk. „Opportunities and Challenges for Data Physicalization“. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, S. 3227–3236. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702180. URL: http://doi.acm.org/10.1145/2702123.2702180.

[13] Jeeeun Kim, Haruki Takahashi, Homei Miyashita, Michelle Annett und Tom Yeh. „Machines As Co-Designers: A Fiction on the Future of Human-Fabrication Machine Interaction“. In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '17. Denver, Colorado, USA: ACM, 2017, S. 790–805. ISBN: 978-1-4503-4656-6. DOI: 10.1145/3027063.3052763. URL: http://doi.acm.org/10.1145/3027063.3052763.

[14] Robert Kosara. „Visualization Criticism - The Missing Link Between Information Visualization and Art“. In: Aug. 2007, S. 631–636. ISBN: 0-7695-2900-3. DOI: 10.1109/IV.2007.130.

[15] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg und Saul Greenberg. „Evaluation Strategies for HCI Toolkit Research“. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: ACM, 2018, 36:1–36:17. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173610. URL: http://doi.acm.org/10.1145/3173574.3173610.

[16] M.Friendly. „A Brief History of Data Visualization“. In: *Handbook of Computational Statistics: Data Visualization*. Hrsg. von C. Chen, W. Härdle und A Unwin. Bd. III. (In press). Heidelberg: Springer-Verlag, 2006, ???–???

[17] Andreas Müller. *AYAB - all yarns are beautiful*. [Online; accessed 2019-08-06]. 2015. URL: https://ayab-knitting.com.

[18] Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros und James Mccann. „Automatic Machine Knitting of 3D Meshes“. In: *ACM Trans. Graph.* 37.3 (Aug. 2018), 35:1–35:15. ISSN: 0730-0301. DOI: 10.1145/3186265. URL: http://doi.acm.org/10.1145/3186265.

[19] Huaishuu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch und François Guimbretière. „RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer“. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2018. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3174153.

[20] Alan J. Perlis. „Special Feature: Epigrams on Programming“. In: *SIGPLAN Not.* 17.9 (Sep. 1982), S. 7–13. ISSN: 0362-1340. DOI: 10.1145/947955.1083808. URL: http://doi.acm.org/10.1145/947955.1083808.

[21] Denise Schmandt-Berrerat. „Tokens and Writing: The Cognitive Development“. In: *General Studies in Writing* (2015), S. 97–116.

[22] Saiganesh Swaminathan1, Conglei Shi, Yvonne Jansen, Pierre Dragicevic, Lora Oehlberg und Jean-Daniel Fekete. „Supporting the Design and Fabrication of Physical Visualizations“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2014, S. 3845–3854. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557310.

[23] Alice Thudt, Uta Hinrichs, Samuel Huron und Sheelagh Carpendale. „Self-Reflection and Personal Physicalization Construction“. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2018. ISBN: 978-1-4503-5620-6. DOI: 10.1145/3173574.3173728.

[24] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo und Daniel Wigdor. „DataInk: Direct and Creative Data-Oriented Drawing“. In: Apr. 2018, S. 1–1. DOI: 10.1145/3170427.3186471.

[25] Stephanie Yang. „Knitting Visualizer: Connecting Craft and Code“. In: *Proceedings of the 2017 Conference on Interaction Design and Children*. IDC '17. Stanford, California, USA: ACM, 2017, S. 705–708. ISBN: 978-1-4503-4921-5. DOI: 10.1145/3078072.3091985. URL: http://doi.acm.org/10.1145/3078072.3091985.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

List of Abbreviations

AYAB All Yarns are Beautiful

CSV Comma-separated values

GUI Graphical User Interface

IR Infrared

DIY Do It Yourself

API Application Programming Interface

RoMA Robotic Modeling Assistant

AR Augmented Reality

HCI Human Computer Interaction

CPU Central Processing Unit

MB Megabyte

GB Gigabyte

GHz Gigahertz

USB Universal Serial Bus

GIF Graphics Interchange Format

Anhang

A Erster Abschnitt des Anhangs

Listing A.1: The fill cells Function

```
'''Fills the chosen cells with marks'''
    def fill_cells(self):
        time_start = time.time()
        self.save_for_undo(self.pattern_maker_ui.table_pattern.
            selectedItems())
        mark = self.__get_mark()
        if self.mark_checker.check_marks(mark, self.pattern_maker_ui.
            table_pattern.currentRow(),
                                         self.pattern_maker_ui.
                                             table_pattern.currentColumn
                                             ()):
            cells = self.pattern_maker_ui.table_pattern.selectedItems()
            self.__fill_in(cells, mark)
            if self.pattern_maker_ui.button_color.isChecked():
                for cell in cells:
                    cell.setBackground(self.__brush)
        else:
            self.note_window.set_label(self.note_text)
            self.note_window.show()
        duration = time.time() - time_start
        print("Duration fill cells: ", duration)
    ''' Checks with Radio Button is selected and returns the fitting mark
        '''
    def __get_mark(self):
        if self.pattern_maker_ui.button_decrease.isChecked():
            return "X"
        elif self.pattern_maker_ui.button_laces.isChecked():
                return "O"
        elif self.pattern_maker_ui.button_color.isChecked():
            if self.__brush.color() == Qt.black:
                return "B"
            elif self.__brush.color() == Qt.white:
                return ""
            elif self.__brush.color() == Qt.red:
                return "R"
            elif self.__brush.color() == Qt.blue:
                return "U"
            elif self.__brush.color() == Qt.yellow:
                return "Y"
    def __fill_in(self, cells, letter):
        tempcolor = self.__brush.color()
        self.set_brush_color(Qt.white)
        for cell in cells:
            cell.setBackground(self.__brush)
            cell.setText(letter)
        self.set_brush_color(tempcolor)}
```

Listing A.2: The check marks Function

```
    def __init__(self, pattern_maker):
        super(MarkChecker, self).__init__()
        self.pattern_maker = pattern_maker
    def check_marks(self, mark, row, column):
        if mark == "O":
            if self.check_lace(row, column):
                return True
        elif mark == "X":
            if self.check_decrease(row, column):
                return True
        else:
            if self.check_color(mark, row):
                return True
        return False
    def check_lace(self, row, column):
        column_index = 0
        while column_index < self.pattern_maker.get_table().columnCount()
            :
            temp_mark = self.pattern_maker.get_table().item(row,
                column_index).text()
            if temp_mark != "" and temp_mark != "X" and temp_mark != "O":
                self.pattern_maker.note_text = "It is not possible to set
                    an O mark in a row with a Color mark"
                return False
            column_index += 1
        if (self.pattern_maker.get_table().rowCount() % 2) == 0:
            if (row % 2) == 1:
                self.pattern_maker.note_text = "It is not possible to set
                    an O mark in a \neven row " \
                            "if the amount of rows is even"
                return False
        if (self.pattern_maker.get_table().rowCount() % 2) == 1:
            if (row % 2) == 0:
                self.pattern_maker.note_text = "It is not possible to set
                    an O mark in a \nuneven row if " \
                            "the amount of rows is uneven"
                return False
        if self.pattern_maker.get_table().item(row, column - 1).text() ==
             "O" or \
            self.pattern_maker.get_table().item(row, column + 1).text()
                == "O":
            self.pattern_maker.note_text = "It is not possible to set an
                O mark next to \nanother O mark"
            return False
        return True
    def check_color(self, mark, row):
        if mark == "":
            return True
        column = 0
        while column < self.pattern_maker.get_table().columnCount():
            temp_mark = self.pattern_maker.get_table().item(row, column).
                text()
            if temp_mark != "" and temp_mark != "X" and temp_mark != mark
                :
                self.pattern_maker.note_text = "It is not possible to set
                    an color mark\n in a row with a O " \
                            "mark or other colors marks"
                return False
            column += 1
        return True
```

Tabelle A.1: The Duration Time of the different Methods in Seconds

Iteration	Fill Cells	Choose Color	Open To Knit
1	0.00029	0.00945	1.09116
2	0.00014	0.00857	0.36612
3	0.00015	0.00966	2.31230
4	0.00013	0.00918	0.00949
5	0.00016	0.00913	0.20185
6	0.00018	0.00881	0.03971
7	0.00015	0.00882	0.25175
8	0.00020	0.00885	0.03600
9	0.00021	0.00885	0.07065
10	0.00019	0.00896	0.04718

Tabelle A.2: The Duration Time of the different Methods in Seconds

Iteration	Check for Feedback	Import	Open Calculator
1	0.00100	5.91795	0.01319
2	0.01078	1.58499	0.01506
3	0.00099	2.13002	0.01287
4	0.01061	1.53944	0.01346
5	0.00108	16.8064	0.01471
6	0.01021	7.46871	0.01374
7	0.00401	1.63418	0.01472
8	0.00213	1.15560	0.01374
9	0.01066	1.41437	0.01378
10	0.00308	1.93320	0.01444

Tabelle A.3: The Duration Time of the different Methods in Seconds

Iteration	Apply Clicked	Save Clicked	Load Clicked
1	0.00585	8.18318	3.31181
2	0.00515	1.31591	1.61022
3	0.00112	1.21407	1.25578
4	0.00853	1.45407	1.52982
5	0.47698	1.32010	1.45179
6	0.03914	1.81495	1.80738
7	0.50475	1.12831	1.12491
8	0.05969	1.95048	1.22670
9	0.07685	2.34466	1.69327
10	0.00705	1.23787	1.86819

Tabelle A.4: Implemented logic Classes

Name	Lines of Code	Desciption
$\text{mark}_{c}hecker$	69	checks if mark can be set into a cell
$\text{pattern}_{m}aker$	248	controls the pattern maker window
$\text{pattern}_{p}ainter$	42	turns a pattern into a picture
$\text{calculator}_{c}ontroller$	80	calculates the size of the outcome
$\text{csv}_{t}ransformer$	48	reads and writes CSV files
$\text{datasetview}_{c}ontroller$	19	shows the content of a CSV file
$\text{help}_{s}ection_{c}ontroller$	100	controls the help section window
$\text{note}_{c}ontroller$	25	controls the notification window

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.1). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10.95 – (B)426.79135pt – (H)688.5567pt