Verisium Drone

- PinDown ML-Debug Visualisation Tool

LUND UNIVERSITY
Campus Helsingborg

LTH School of Engineering at Campus Helsingborg
Department of Computer Science

Bachelor thesis:
Pontus Henry André Persson

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden

Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2023

Abstract

In collaboration with Cadence this thesis work has created a web based interface which aims to visualise large sets of data in a more human readable way. Cadence is a company specialising in electronic system design and is offering other companies solutions for silicon design creation, simulation, implementation and signoff of analog and digital circuits and a lot more. This interface will also be replacing an old 3D based interface called CityScapes. The goal of this thesis was to create a new interface for the debugging tool PinDown with the help of user tests and established design principles. An interface for PinDown would function as a visual medium for the large amounts of data PinDown generates after a test-suite.

The thesis consisted of three main phases: planning and information gathering, prototypes and user testing and lastly completion of the final report. The prototype development went through three iterations, lo-fi, mid-fi and hi-fi. During the lo-fi iteration three different designs were explored and in later iterations only the treemap design was kept and continued to be worked on.

The resulting interface became an interactive treemap model with clickable elements, where each element in the treemap represents either a folder or file in a larger file system. When the user left clicks on a folder element it brings them into that folder meaning that only one layer of the file system is visible at one time. The user can also sort and filter out folders and files based on the data they have provided in the interface.

Keywords: User Experience, user-friendly, web application, data plotting, visualisation, prototype fidelity

Sammanfattning

I samarbete med Cadence har detta examensarbete skapat ett webbaserat gränssnitt vilket kommer att visualisera stora mängder data på ett mer läsbart sätt för människor. Cadence är ett företag med inriktning på elektronisk system design och erbjuder andra företag lösningar för silikon design skapande, simulation, implementation och signoff för analoga och digitala kretsar och mycket mer. Detta gränssnittet kommer också att ersätta ett äldre 3D baserat gränssnitt kallat CityScapes. Målet av den här rapporten var att skapa ett nytt gränssnitt till debugging verktyget PinDown med hjälp av användartester och etablerade designprinciper. Ett gränssnitt för PinDown kommer att funka som ett visuellt medium för dem stora mängder av data PinDown genererar efter en test-svit.

Examensarbete bestod av tre huvudfaser: planering och studier, prototyper och användartester och till sist färdigställning av slutrapporten. Prototype utvecklingen gick igenom tre iterationer, lo-fi, mid-fi och hi-fi. Under lo-fi iterationen utforskades tre olika designer och i senare iterationer så valdes en av dessa till fortsatt utveckling.

Det resulterande gränssnittet blev en interaktiv treemap modell med klickbara element, där varje element i treemapen representerade antingen en fil eller mapp i ett större filsystem. När en användare vänster klickar på ett mapp element tar det dem till en ny vy av innehållet av den mappen. Det medföljer därför att användaren kan bara se en nivå av filsystemet. Användaren kan även sortera och filtrera mappar och filer baserat på deras inmatning i gränssnittet.

Nyckelord: Användarupplevelse, användarvänlig, webb application, visualition, prototype fidelity

Foreword

I would like to show my gratitude towards Cadence for giving me this opportunity and giving me an adequate work life experience. It has been really fun and challenging and I hope this can be used in the future, either as an inspiration or framework. I would also like to thank my mentor Kirsten Rassmus-Gröhn who has guided me through this endeavour, she has been really helpful especially during the writing of this report.

List of contents

Chapter 1
Introduction

1.1 Background

This thesis is done in collaboration with Cadence Design Systems, Inc. Requested by one of Cadence's regional offices located in Lund. This office was previously known as Verifyter and was acquired by Cadence a few years ago. Cadence is a company specialising in electronic system design and is offering other companies solutions for silicon design creation, simulation, implementation and signoff of analog and digital circuits and a lot more [5].

The project that has been asked to do is help Cadence visualise complex data from one of their software programs. The software in question is called PinDown and is used for predicting and locating bugs in their ASIC programming workflow. Currently PinDown outputs a file with column names and thousands of rows with data. This is not practical for human readability and is therefore needed to be transposed into a visual medium with easier access to the information being produced.

Cadence already has a solution for this called CityScapes that shows all the data from PinDown in a 3D environment. But Cadence isn’t satisfied with this solution and wants an implementation of the same system but as a 2D projection instead. The goal of this thesis is therefore to find the best 2D projection of CityScapes that provides a satisfying result for Cadence and its customers. The new software should have a more understandable UX (User Experience) than its predecessor CityScapes and still provide the same amount of information.

1.1.1 CityScapes

CityScapes is a previous iteration of Verisium Drone created in a 3D environment. It displays the amount of bugs in each file as a coloured block, ranging from green to red. The blocks are placed on top of layers of grey planes meant to visualise the folders in the file system. Navigation through the world is done by flying around the camera with the arrow keys or the WASD keys on the keyboard. The problem with this solution is the UX, it's neither easy to use nor intuitive. To find a file a user searches for it in a search bar, which then lights up the block with an overhead light. The user then has to navigate the camera using the keyboard and mouse to the block which can take a while depending on the distance the user has to fly. Also if the user just wants a quick overview of what they are looking at they have to hover all of the blocks to get information about what files or folders the user is looking at. The goal of this thesis is therefore to reimagine this interface into a better UX.

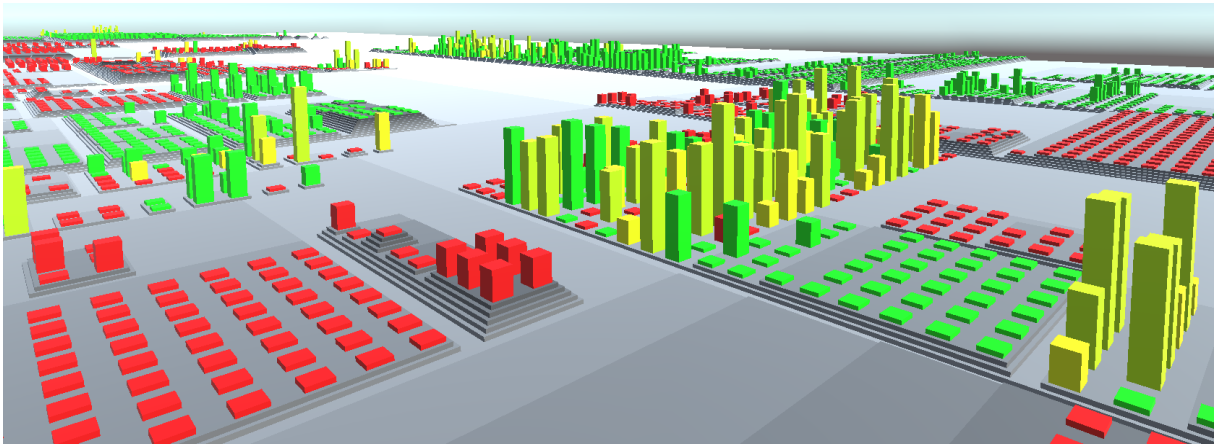

Figure 1.1.1 An image of the Cityscapes program

1.1.2 Previous work

As seen in CityScapes the interface creates a model representing some sort of filesystem. In the case of this thesis the filesystem is derived from files that have been uploaded to a version control system. When a user uploads files to a version control system they are called a commit and are usually accompanied by a message called commit message. CityScapes looks at all the commits to the version control system and recreates a rough copy of the project files based on all the files that have been uploaded or changed.

Visualising a filesystem has been done many times before and it is therefore interesting to look at some already existing solutions for this thesis. WinDirStat [4] is a well known example, it is used to visualise a computer's file system to get an overview of the space allocation on an harddrive. To get a quick overview WinDirStat has implemented a treemap model as well as a list view of the files in the system. This will be used as a reference going further into this thesis.

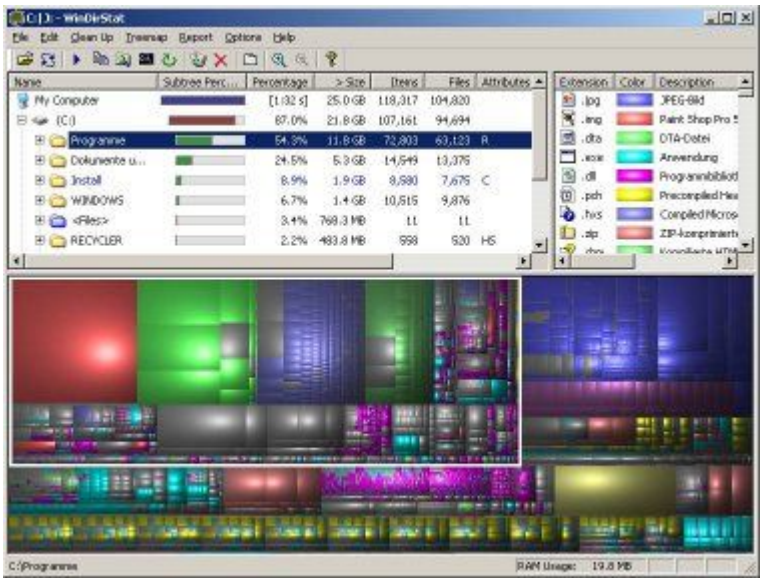

Figure 1.1.2 WinDirStat file overview

1.2 Purpose

The purpose of this project is to develop a 2D interface that summarises all the bug information PinDown outputs after a test suite. This data will be put into a data structure that has been chosen after research of what data model fits best for this kind of data visualisation. The expectation is that this will be a better UX than the current 3D chart program CityScapes.

1.3 Project goal

The goal of this project is to find the best suited data model for visualising the data from PinDown and then implementing that model. The result will be a static website so later development can focus on integrating that website into the Cadence toolbase. If it is successful Cadence will have a new prototype user interface for its debugging tool PinDown. The goal of the project would then be to create a new interface that Cadence is satisfied with since they already have a lot of ideas and requirements of how the interface should look and function.

1.4 Problems

The project poses a lot of research questions about UX and technical aspects of the program. For example, what types of data structures are best used for visualising this type of data in the most pedagogic way possible? The biggest concern regarding this is the visualisation of the commits itself. Since the commits have revisions grouping them together and also information about all the revised files we need to figure out what data structure fits best to visualise this relation. Aside from that we also have information about what type of bugs and where the bugs have occurred that also needs to be visualised. Then regarding the technical aspects we need to firstly figure out what type of programming language and frameworks are going to be used. Secondly we need to figure out how we read the data input from the files into our program and where this data comes from.

The following questions will be answered:
1. What type of data structure will be used for visualisation?
2. How do we display the bugs that have occurred in the commits?
3. What type of programming language and framework will suit us best?
4. How do we read data from files and use it in the interface?

1.5 Motivation

This project is a great way for me to show my competence in programming and learning new things quickly in the industry, since at the time of writing I don’t have a lot of experience working for companies in my field.

As for the company, working with schools is a way to look for new competence and also to save time and resources since they don't have to hire someone to make prototypes and can instead rely on students to explore new possibilities for them whilst the student gets knowledge and work experience.

This thesis can also be used to help others figure out what type of data structures fits their project, or just help them get inspiration for any other data structure related problem. Since We are going to explore different data structures and their pros and cons, it can be used as a guide for what type of data structure to choose.

1.6 Boundaries

The program that Cadence has requested should work for all platforms since it's a web application but they only want it to show all the bugs. There is more information that can be viewed but this is not something this project will deal with unless there’s time for it.

Chapter 2
Technical Background and Theory

2.1 Technologies and programs

These programs and technologies have been used during the development process of the Verisium Drone website. They were all required to make the thesis and development work and had no alternatives.

2.1.1 HTML Canvas

The canvas tag in HTML is used to render either a two dimensional image or a three dimensional world depending on what context you choose. It has no function without JavaScript and is therefore needed for basic function. The JavaScript functions for drawing are made up of shape functions and some text functions. There is no per pixel control out of the box but JavaScript has functions for almost all thinkable scenarios. [12]

2.1.5 CSV

Comma-separated values is a file format where each line of text is a data record. Each record can contain one or more fields which then can be described with a header on the first line of the file. This header is mostly only useful for human readability or cross product interactions. To differentiate fields a separation character is used between each value. The format isn't fully standardised as sometimes the separation character can be any character but most of the time it is either a tab, comma or semicolon. [11]

2.1.6 PinDown

PinDown is made for the use of ASIC verification, i.e., regression testing of hardware description languages. ASIC verification requires a large amount of resources and uses large test farms to run the regression testing in parallel. Because of the long test executions, continuous integration is not used which means that there's usually over a hundred commits before every test suite. When a test suite has failed PinDown ranks all the code contributions with a risk factor, low to high, depending on if the code might contain the fault or not. This saves time and resources since instead of manually debugging over hundreds of commits we can instead review 4 high risk commits.

2.1.7 TigerVNC

TigerVNC is a platform-neutral implementation of VNC (Virtual Network Computing) used to connect to a remote machine and use its resources. It is used in this project to connect to Cadence virtual machines since there's confidential data in those file systems that can't leave the Cadence network.

2.2 Packets and Technologies

These were the packets and technologies used during the development process of the Verisium Drone website.

2.1.1 Chart JS
Chart JS is a JavaScript library used to create charts in the browser. It comes with a lot of features such as scale stacking, which means you can stack several charts on top of eachother. Advanced animations letting you add animations to any of the elements in the chart you're making. Mixed chart types making it possible to mix different types of charts with each other in the same canvas. It is also possible to make elements clickable which makes the element call on a function when the user clicks inside the bounding box of an element in the chart. Chart JS uses the canvas element in HTMLto draw its charts and has a lot of community made extensions allowing many different types of data models. It is open source and made under the MIT licence making it available to everyone. [14] There exists more alternatives to plotting data models in web applications but Chart JS is the most used library and therefore has the most documentation and extensions which is why it was chosen for this thesis.

2.1.2 Treemaps JS
Treemap JS is an extension of Chart JS, it adds the treemap data model to the Chart JS library. It gives control over almost all elements of a traditional treemap whereas the only exception is the shape of the treemap boxes. This can be altered by changing the source code but this requires deeper knowledge of how canvases and the library works. To display the data in a treemap it is only required to give a list of objects that have one property each, the property is used to determine the size of the box in the treemap. If it is needed, Treemap JS also makes it possible to group objects into any number of subgroups. The colour of the boxes can also be changed to either be a static colour or call upon a function to determine its finalised colour. [13] The treemap was not implemented in Chart JS and therefore since one of the prototypes was a treemap this packet was used.

2.1.4 REST API
Representational state transfer application programming interface (REST API) [10] is used to enable access of resources outside of an application. Most commonly the application getting the data is called client and the application or service that is containing the data is called server. The REST API allows developers to connect their application to other applications and then make requests between those. These requests are called CRUD, short for Create, Read, Update and Delete. Alot of browsers today have implemented security features around the REST API e.g. we can't let our web application access files on our local machine without using some form of HTTP request which in turn requires the local machine to run some sort of server application. This is called a CORS request which stands for Cross-Origin Resource Sharing and we can't make these requests when the URL isn't of the HTTP or HTTPS type. The reason this technology was used was to retrieve the information from the files used as input for the data plots.

2.3 UX Design

UX is short for User Experience and is the definition for how people are interacting with our product. For example when a user clicks a button to go to the next page of a website, the way the button looks, animates and transitions us to the next page may impact how the user feels about that interaction. Nick Babich [1] claims that when an user evaluates our product they usually evaluate their experience according to the following criteria:

1. *Value. Does this product give me value?*
2. *Function. Does this product work?*
3. *Usability. Is it easy to use?*
4. *General impression. Is it pleasant to use?*

UX design then refers to the act of creating products that are usable and practical. Peter Morville's UX honeycomb [1][2] breaks down the ideal characteristics even further:

1. *Usable: A product needs to be simple, easy to use, and familiar.*
2. *Useful: A product must fill a need. If the product isn't filling a perceived gap in the users' lives, then there is no real reason for them to use it.*
3. *Desirable: The visual aesthetics of the product need to be attractive and evoke positive emotions.*
4. *Findable: If the user has a problem with a product, they should be able to quickly find a solution.*
5. *Accessible: The product or service needs to be accessible to everyone, including those with disabilities.*
6. *Credible: The company and its products need to be trustworthy.*

These characteristics will be used during the thesis as a guideline for the student when evaluating the interface.

2.4 Visualisation guidelines

When designing UX there are a few practices that are good to follow to create usable designs. A few of those can be found in the sections below. These two guidelines have been chosen based on this thesis needs, when researching for this interface these were the two guidelines found to be most relevant. A lot more guidelines surely have been used given some afterthought, but these were the ones thought of before the interface was created. The first subsection 2.4.1 came from a need to explain and understand how data can be visualised in multiple ways. The second subsection 2.4.2 was relevant because of previous use of color in CityScapes and a need to understand how to utilise it.

2.4.1 Dimensions

Utilising all your dimensions when visualising data is important, it is wise to not use dimensions for things other than showing the data the user wants to see. In most cases this comes down to a choice between 3D and 2D visualisation. One of the last chapters in Claus O. Wilke's book *Fundamentals of Data Visualization* [3] is titled *"Don't go 3D"*. Claus explains that going for a 3D visualisation creates an extra dimension of data that is not used to visualise what is important. In extreme examples like CityScapes the user even has to navigate this new 3D space to get an overview of the data. Besides creating an extra

dimension for the sake of aesthetics the projection of the data gets distorted in most cases. As seen in figure 2.3.1 the 3D projection distorts the data in a way that causes the lowest 1st class bar to appear taller than the 2nd class bar which should be taller.

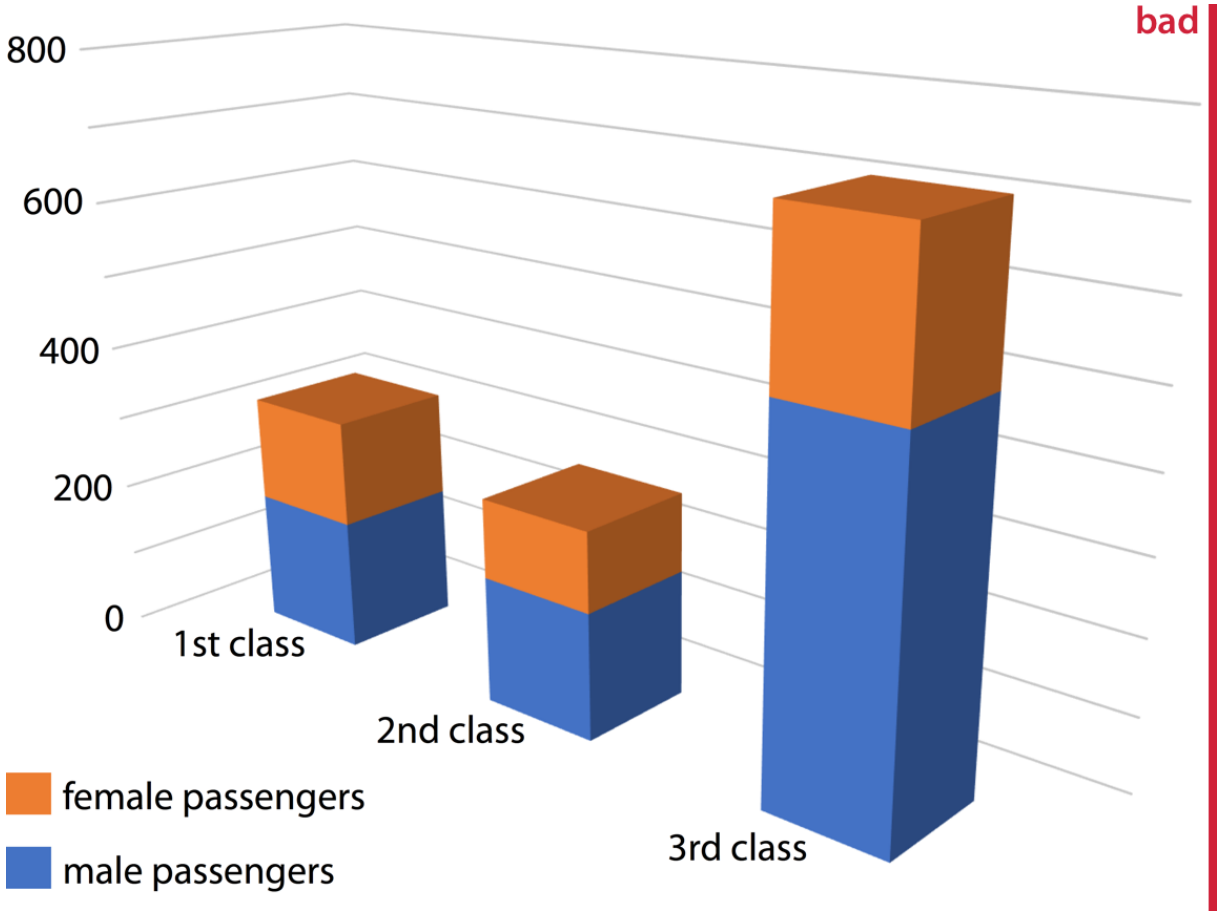

Figure 2.4.1 An example of bad usage of 3D in data visualisation. Image from "Fundamentals of Data Visualization" [3].

2.4.2 Color Theory

Color plays a large part in UX design, according to Wilke [3] there are three use cases for color in data visualisation. It can be used to distinguish groups of data from each other, represent data values or highlight data. Representing values with colors can be done with colors based on a single hue (e.g., from dark blue to light blue) or on multiple hues (e.g., from dark red to light yellow)(figure 2.4.2).

Figure 2.4.2 Image from "Fundamentals of Data Visualization" [3].

A multi hue color sequence should be using colors seen in the natural world such as green, or blue since the opposite , e.g. dark yellow to light blue, looks unnatural. When choosing colors the result needs to be able to answer these two points, (i) which values are larger or smaller and (ii) how distant are two values from each other.

2.5 Prototype fidelity

A good practice used when creating new products is producing prototypes at regular intervals to establish a good UX design. A popular term used when creating prototypes is fidelity which refers to the level of detail for these three areas, visual design, content and interactivity. It is therefore referred to as a low fidelity prototype when the level of detail in those three areas are low. The opposite is called high fidelity prototypes, which is a high level of detail in those three areas.

A low fidelity prototype (lo-fi) [9] is a prototype that deliberately is created with low visual design, less content and low interactivity. Using lo-fi prototypes is a way to get the design process started since it's fast and inexpensive, stimulates collaboration and clarifies the idea of the project. It should be noted that lo-fi prototypes demand more from its test participants, since there's a drawback of interactivity and visual design; it can be hard for the participants to imagine what the end product would look and feel like.

A high fidelity prototype (hi-fi) is a prototype that is created with high visual design, more content and high interactivity. This prototype will be a closer representation to the end product and is therefore usually made later in the production phases when the team has a solid understanding of what they are going to build. It's easier to receive constructive feedback from the test participants as the participants can interact and visualise the product more closely to that of the end product. Hi-fi prototypes can be used to test new functionalities and visual elements before shipping a full product.

Sometimes neither a lo-fi or hi-fi prototype describes the fidelity correctly, those times some designers like to use the term "mid-fi" [17]. Mid-fi means the fidelity of the prototype lands somewhere between an lo-fi prototype or hi-fi prototype. Usually this is because the prototype

is that of a lo-fi one but has more interactivity, or it has a high level of detail but no interaction. Mid-fi prototypes can be useful when the team needs to be resourceful and not waste resources on aspects of the prototype that have less of an impact on the final product.

2.6 Mosaic plot

Mosaic plots [3] are a type of stacked bar chart with both the height and width areas varying. It is used when there's categories of data that overlap in our dataset and therefore is needed to show how those categories relate. As seen in figure 2.6 a mosaic plot is created by dividing the x axis into relative proportions by its categorical value and then doing the equivalent thing for the y axis. Whilst at the same time within each category of the x axis, subdividing the y axis into relative proportions.

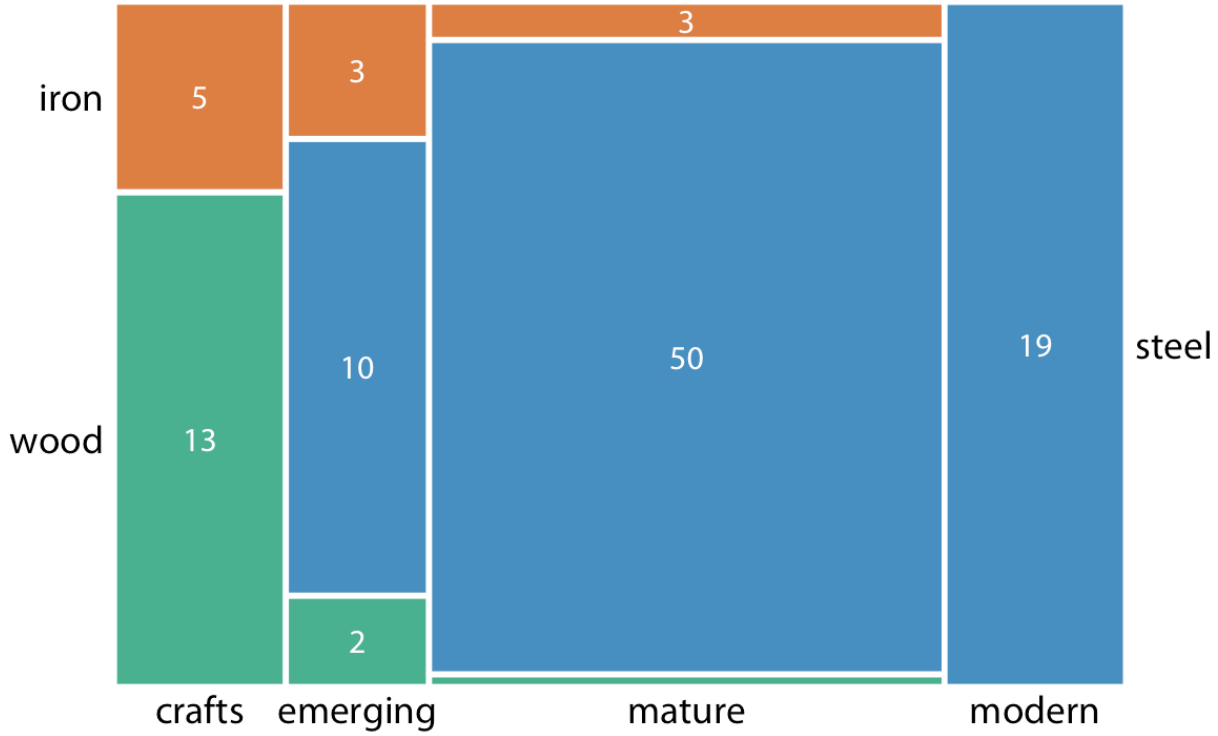

Figure 2.6 An example of a mosaic plot showing the amount of bridges in Pittsburgh and what construction material has been used. Image from "Fundamentals of Data Visualization" [3].

2.7 Treemap

Treemaps [3] are used to visualise hierarchical data using nested figures, the most commonly used shape to create the treemap is rectangles. A treemap requires some form of tree-structured data to be built upon, where each node becomes a rectangle. If there sub-nodes this can be chosen to be displayed as grouped rectangles. The grouping of rectangles is usually done by creating one big rectangle for the parent node then creating smaller rectangles inside that rectangle for each of the sub-nodes.

To create a treemap a tiling algorithm needs to be defined which divides a region into sub-regions for the treemap branches. Convex treemaps are the most commonly used but there are other treemaps like Voronoi treemaps based on Voronoi diagrams [6] or even treemaps based on Gosper curves [7]. The size of each rectangle is decided by a selected property from the current node in the data tree and the tiling-algorithm used to create the treemap tries to match its size to be proportional to its value. Therefore the size of each rectangle can't be measured for an exact value but is instead used to visualise the data values proportions in relation to each other.

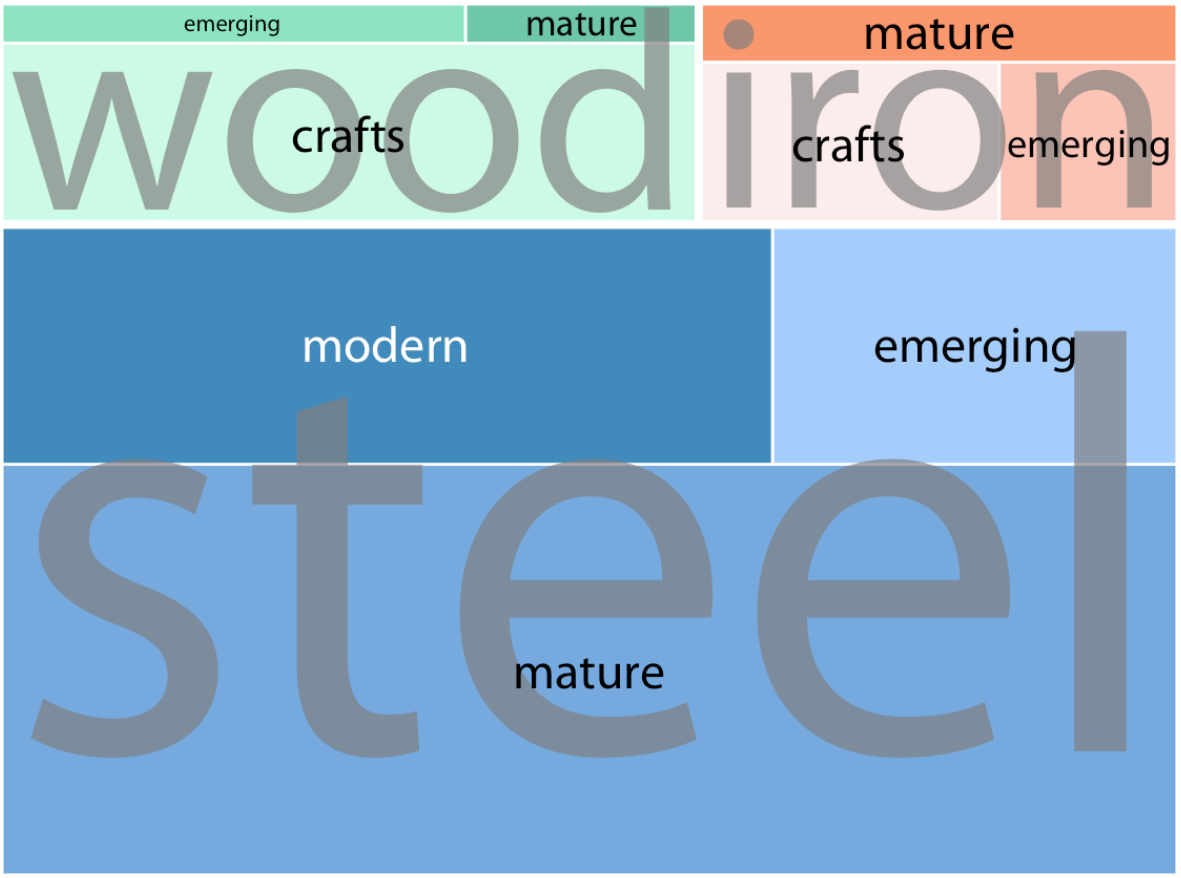

Figure 2.7 An example of a treemap showing the amount of bridges in Pittsburgh and what construction material has been used. Image from "Fundamentals of Data Visualization" [3].

2.8 Pie Chart

A pie chart [3] is a circular plot of data, it can be used to show data values and its proportions to a dataset as a whole. The reason it's called a "pie" chart is because of the way slices are made into the circular plot. Each variable in the dataset gets a slice of the circle in the shape of a pie slice proportional to its total fraction in the dataset. Therefore to be able to create a pie chart there needs to be a calculated total value of all the variables to be able to determine each variable's percentage of the total. The slices are then sorted from biggest to smallest in an effort to make readability easier since it's hard to compare small slices to each other.

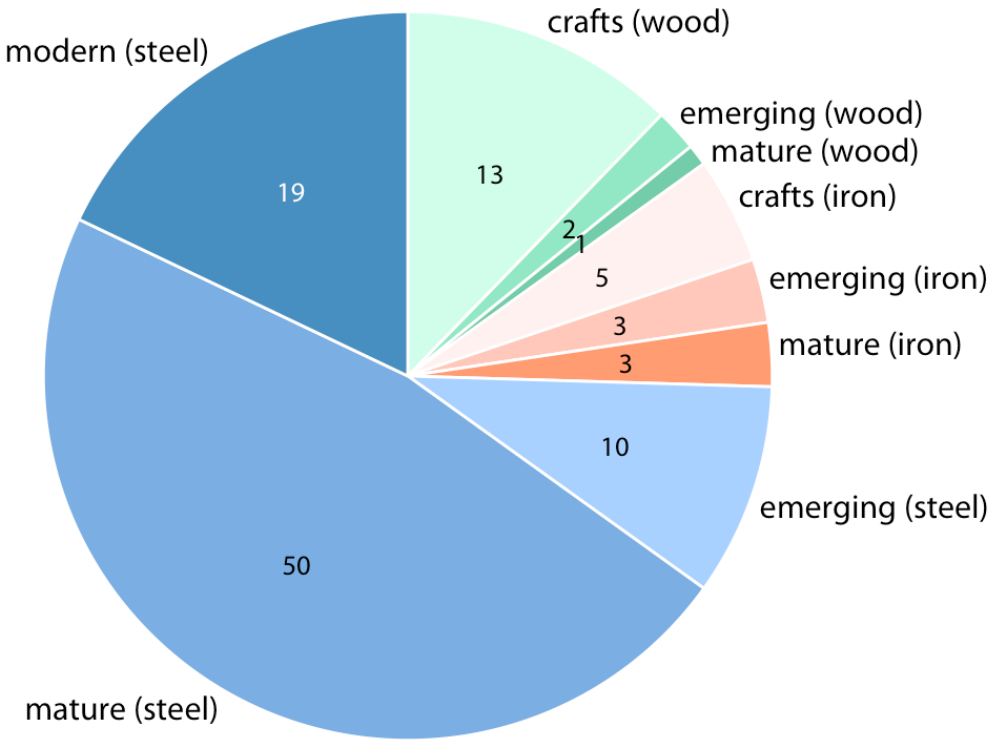

Figure 2.8 An example of a pie chart showing the amount of bridges built with different materials in Pittsburgh. Image from "Fundamentals of Data Visualization" [3].

Chapter 3
Process

When designing this interface the approach was mainly that of an iterative and agile one. At the start of the design process prototypes were made with extensive research on the subject. Later a prototype was chosen based on the feedback received from interviews and testing. It would be developed further with continued testing and user feedback until a satisfying result had been met. The testing and interviews were made on a weekly basis consisting of meetings in-office and a short presentation. After the meeting a list was made of the current feedback and the next week consisted of further development based on the feedback received previous week.

The working process of the project was inspired by that of the agile manifesto [15]. With the exception of working in a team. Every week a list of requested features was sent from Cadence in an email, points from that list could then be picked and worked on. It is a modified version of weekly sprints and a backlog with very loose deadlines.

3.1 Work Environment

Cadence provided a work computer and a virtual machine to aid in developing the project. The virtual machine had access to the Cadence filesystem where the data files from PinDown could be found. The virtual machine also allowed us to host a local web server for fast and continuous testing. The virtual machine was accessed through TigerVLC and the web server was hosted by running a Python script. A web server was mandatory at the beginning of the project since newer web browsers has implemented security features that prevents a website from accessing files that is not on the same domain, hence it was required to use the REST API to access the data files from PinDown or else the web browser would only give the CORS error [8].

3.1.1 Weekly meetings

Once a week a meeting was scheduled with Cadence to discuss the current development. A presentation was made of the current work and Cadence gave its feedback which was summarised in a weekly email. The next week of work would be based on the feedback to further improve the product and then the process would be repeated. This was the catalyst for pushing the project forward as proper feedback was given and a sort of deadline was created for next week's presentation. The meeting was most of the time hosted in-office with a few exceptions, the face to face interaction made discussion easier between Cadence and the student.

3.2 Method

The start of the project began with a planning phase. An estimation of the total hours of work was made based on the fact that the bachelor thesis is a course with 22.5hp and 1.5hp equals a full work week (40 hours). Based on that there are 600 hours to spare in a 15 week timespan. A timeplan was then made in Google Sheets and would also later be used as a time reporting tool.

Initially information was gathered about the task from the client to determine what the end product would look like. Cityscapes, a program made by a previous student was shown as an example of how Cadence wanted this new program to work. They pointed out that having this program in a 3D space made it overly complicated to use and wanted something simpler in terms of user experience. Further information was gathered on PinDown and the data it outputs to help write down an initial document for the project.

3.2.1 Prototypes

Several lo-fi and mid-fi prototypes (figure 4.2, figure 4.3 and figure 4.4) were made at the start of the project to explore options for what data model was going to be used in the program. The goal was to find out if the treemap model Cadence proposed was inferior to any other data model, by doing this we could know that either the treemap model was the right fit or we would find a better model suited for the project. We would also get our first user feedback on what works for this particular project and what doesn't. When the prototypes had all been made one was chosen based on the best feedback to continue development using that particular data model. The design choices hereafter were made based on available knowledge and research mostly from the book *Fundamentals of Data Visualization*.

Figure 3.2.1. First prototype of an treemap model

3.2.2 User testing

The user testing and interviews were conducted every week at the same time as the weekly meetings referred to in 3.1.1. Due to some limitations the number of users were a total of four and consisted mostly of the employees at Cadence's Lund office. Occasionally user feedback would come from one of Cadence's partners, but this was on rare occasions since there was a time zone difference between the offices and general time constraints. There were no specific questions asked during the testing, rather a task was given to the user on whatever new feature or design was being tested. Then the student evaluated how easy it was for the user to complete said task and noted what worked and what didn't work. Some follow up questions were asked changing depending on the situation but no set formula was created for those questions.

3.3 Lo-fi prototypes

The first phase to be introduced in the thesis was the lo-fi phase, it was a short phase to find out what data model would work best for the presented problem of visualising the data from PinDown. A few requirements were introduced for the data model, it would need to be interactable in some way to allow the user to click on the model. The idea behind the interaction was to allow the user to navigate a file system built upon the model. The second requirement was multiple ways to visualise data, some ideas were to use color and size but more ways of visualising data were explored during this phase. The last requirement was to be able to show file and folder names in the data model so the user could understand that it was in fact a file system they were looking at. This phase went on for about 1 week.

3.4 Mid-fi prototypes

After the lo-fi phase a decision was made during a weekly meeting to continue development with the treemap model. The features from CityScapes were implemented one by one and tested by Cadence. As mentioned in 3.1.1 every week there was a meeting where Cadence updated the list of functions they wished to have in the interface. Then the following week as many of those functions as possible were implemented. The key point in this phase was to implement as many functions as possible and not care too much about the UX design of the interface. The mid-fi phase was the longest one and took about 7-8 weeks to complete.

3.5 Hi-fi prototypes

When all the futures requested by Cadence had been implemented the Hi-fi phase started. During this phase mostly design changes were made and tested to see what kind of designs worked and what didn't. The goal was to satisfy Cadence and make an easy to use interface, to achieve that regular presentations of the interface were made to the test users which included Cadence personnel. The hi-fi phase was also short since most of the features and design were already in place and took about 2 weeks to complete.

3.6 Source criticism

Source [1] is from Adobe Inc. own website, with years of experience in the UX field and developing a multitude of programs used everyday by designers alike. The writer of the article draws a lot of inspiration and information from source [2] which is from an equally credible source. Source [2] is by Peter Morville, a designer and author since 1994. He has several books published in the topic of user experience and information architecture. With his work experience and accomplishments in mind it's safe to say he's a credible source.

Source [3] is written by Claus O. Wilke has a lot of experience teaching and practising what he's trying to teach in his book "Fundamentals of Data Visualization". The book was also recommended by Kirsten Rassmus-Gröhn from the Certec department at Lunds University. Kirsten also recommended source [17] meaning it has been reviewed by a trusted source. Sources [4][5][8][13][14] are pages containing documentation or information from the developer or company itself, since the documentation and companies are in active development the date of information retrieval has been noted in the source list.

Source [9] were used in the MAMF40 course at Lund University and would therefore be considered as a valid source. Whilst Source [6][7][11][16] are Wikipedia pages used as an demonstration of various technical aspects where the information has been investigated thoroughly based on the students own experience.

Source [12] is a popular page amongst developers to learn web development and is used by a lot of people. The information can be tested and checked validating its legitimacy. Source [10] is made by one of the largest IT companies in the world and has a great reputation surrounding their research and work in the IT space.

Source [15] is written by a lot of credible authors. They all have a lot of experience in their field and could each be cited as a credible source.

Chapter 4
Analysis

This chapter describes the decisions that were made from the information gathered during this thesis. The chapter will also discuss different problems that arose during the thesis and how they were solved.

4.1 Prototypes

Three different prototypes were made during the first development phase. They were created with low fidelity (lo-fi) and therefore had a lack of interactivity. Since the prototypes were mainly just static data models it demanded more from the users when they were giving feedback.

4.1.1 First lo-fi prototypes

To begin the first development phase three lo-fi prototypes were made. Extensive research was made on different kinds of data models and three models were found in *Fundamentals of Data Visualization* under the chapter *Visualizing nested proportions* [3]. Since our goal is to visualise four dimensions of data where the fourth dimension is our nested folder/file groups. The best idea for traversing the file system was proposed as a simple mouse interaction with the folder the user wants to jump into, therefore one of the criteria for a fitting data model would be big enough elements for the folders so that the user is able to click on them. The other dimensions of data we need to visualise are two columns from our mlData3 file which contains all the statistics of every commit and the last dimension is the name of the folder/file. Test data was given by Cadence to get an unified look of the models, the data can be seen in table 4.1.1 in an spreadsheet format.

file/folder name	bug_pszz	bug_pszz_all	commit_msg_fix_x	predictions
9	3	9	45.6	8.52
1	1	4	5	4.62
21	1	2	5	1.64
10913	1	3	0	3.37
7	0	0	3	1.22
667	0	1	2	0.87
19.txt	1	3	0	3.35
1889	0	0	1	0.77
27	0	0	2	0.36
24	0	0	2	0.49
70	0	0	1	0.26
1992	0	0	0	0.02
839	0	0	0	0.02
4	0	0	0	0.02

Table 4.1.1. Testing data for the lo-fi prototypes

4.2 Mosaic plot prototype

The first data model prototyped (figure 4.2) was the mosaic plot, we can visualise the files and folders as groups on the y-axis and categories the data values as colors. The size of each coloured bar would be the value of the mlData3 data. Because this is a lo-fi prototype it was only made as a static image.

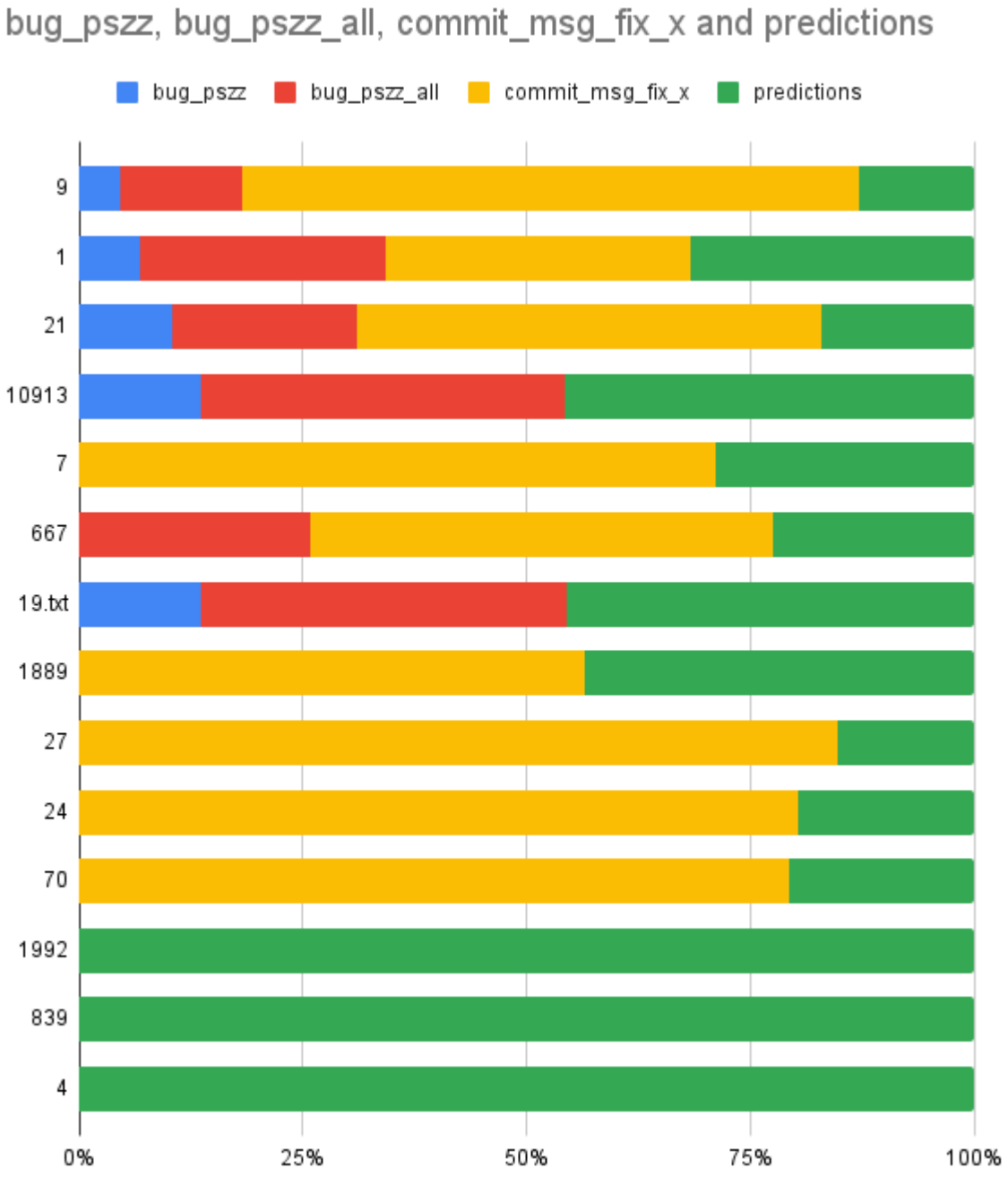

Figure 4.2. Lo-fi prototype of a mosaic plot showing the test data from table 4.1.1

4.2.2 Mosaic plot analysis

The downside to this model is the intuitiveness, during user interviews it was hard for participants to understand that they could click on a vertical bar to jump into that folder. So the idea that this was a visualisation of a filesystem was missed and users also pointed out that it was hard to gauge the values of our data. This was largely because the vertical size of each coloured bar is scaled to fill up the entire width of the plot and therefore miss presenting the actual values. The upside to this data model is the fact that we can show multiple data values at once, e.g figure 4.2 is showing us four different data values at once. The only problem with this is when data values are too far apart, e.g if bug_pszz is equal to 1 and predictions is equal to 100 the coloured bar for bug_pszz would become so small it is impossible to see and if we have multiple of these value differences a lot of the plot becomes unreadable. We can also see in the last three rows of data the prediction bar shows us 100% because the other three values are equal to zero, if the user were to compare these rows with the others they would get the idea that the prediction value in the last rows are bigger than it is in the rest of the plot. When in actuality the last three rows have the lowest value of all the rows. It is also worth noting that the lo-fi prototype used when presenting this model to the participating interviews was not a true mosaic plot as it lacked the varying height.

4.3 Treemap prototype

The second prototype (figure 4.3) was based on the treemap model, this model was requested by Cadence and therefore a sort of mid-fi prototype was made instead. In this case a mid-fi prototype would refer to an actual code implementation instead of a premade model or picture. The difference from a hi-fi prototype would be that there's limited interaction with the treemap itself and is mostly just a visual implementation.

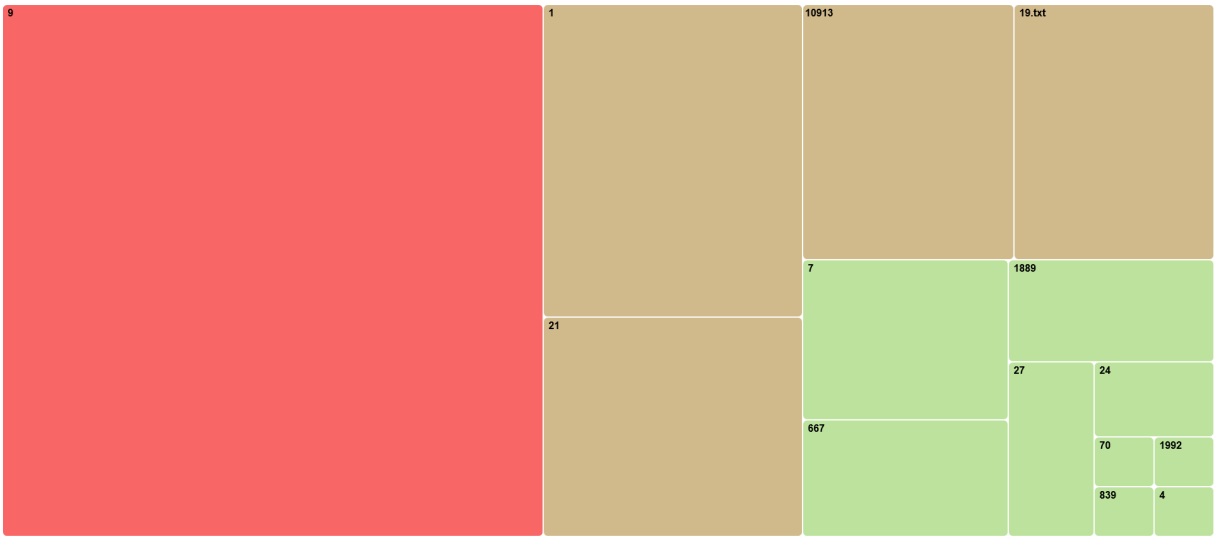

Figure 4.3. Final hi-fi version of the treemap showing the test data from table 4.1.1

4.3.1 Treemap analysis

The pros of a treemap is the fact that the size of the boxes is 99% accurate in relation to its value. The algorithm tries to fit all the boxes in a fixed size so it can take some liberties creating the sizes for each box. A treemap is also more intuitive for visualising file systems since alot of modern applications created to get an overview of your computers filesystem uses some sort of treemap in most cases, e.g WinDirStat [4]. The color of the boxes can also

be used to visualise some sort of data relation. The only downside to the treemap approach is when dealing with a lot of folders/files the boxes can become too small for a user to interact with them.

4.4 Pie chart prototype

The third prototype (figure 4.3) created was based on the pie chart data model, the folders and files were presented as different colors and the size of the pie fraction would represent the data value from mlData3. As in 4.2 a static image was created for the lo-fi prototype.

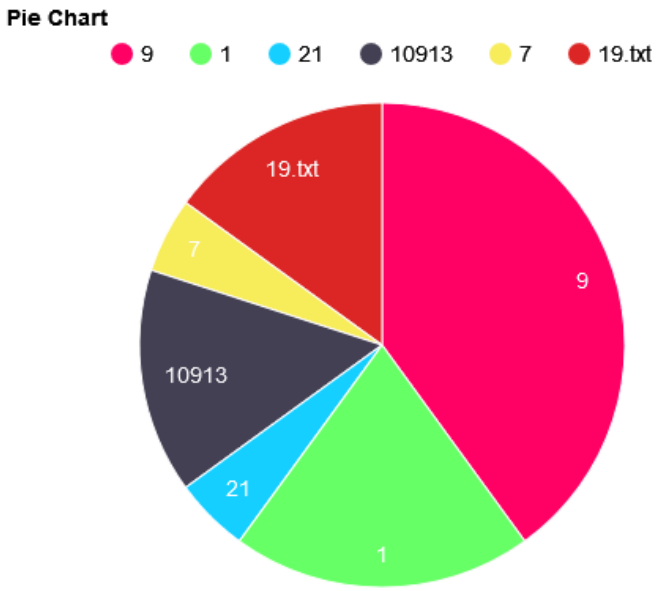

Figure 4.4. Lo-fi prototype of a pie chart showing the test data from table 4.1.1

4.4.1 Pie chart analysis

The first problem found with this model was when our data values had a big integer difference between them causing some pie fractions to be incredibly small whilst others took over most of the pie area. This would cause problems later in development since this chart only shows one layer of our entire file system that we're trying to visualise and when an user tries to interact with an incredibly small pie fraction it would most likely cause a lot of frustration.

4.5 Lo-fi conclusion

Since the treemap had the most positive user feedback and was easy to implement the decision was made to use the treemap going further into development. The mosaic plot had a lot of similarities to the treemap but didn't quite fit our use case and the pie chart had too many limitations because of its model.

4.6 Visualising the data

In our program we have three potential dimensions to explore data in, color, box size and layers. The layers are used to navigate the file system, each page is a folder and when you click on a box it takes you to a new page displaying all the files in the clicked folder.

Since almost all of the columns in the data input can be categorised as either bad or good the decided color to visualise this became an color scale of green to red colors. Just a quick look at a real world example would be stop signals. Red means stop and green means go, negative and positive. But as we can see in figure 4.6 if we use high contrast colors the text can be hard to read and therefore needs adjustment.

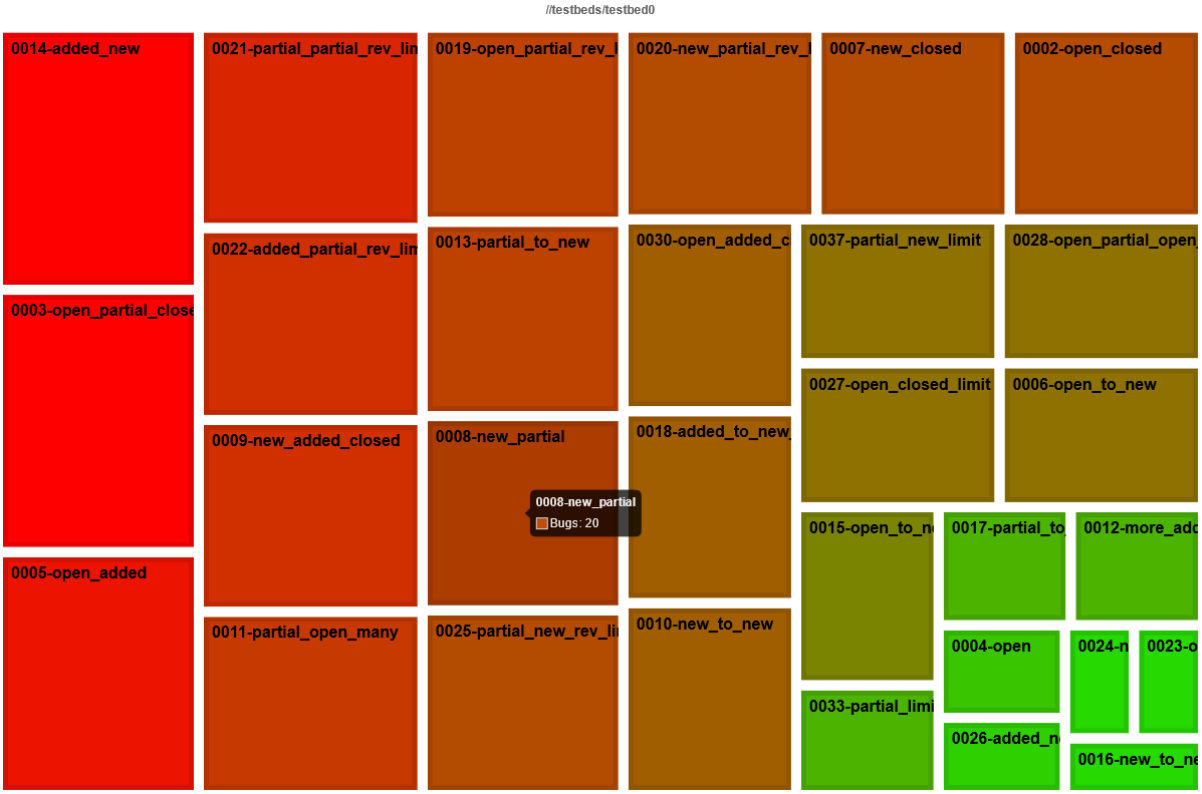

Figure 4.6 Early mid-fi prototype of the treemap model.

By lowering the contrast we can make the text more readable and yet keep our red and green colors, the only problem now is that a person with red-green color blindness will not be able to differentiate the colors at all. To solve this the colors could be changed to something else, Claus O. Wilke [3] suggests a few color scales in his book that are colorblind friendly (figure 2.4.2).

The problem with these scales is the fact that it's harder for an average person to instinctively understand what the colors mean. The simplest solution to this would be to use a button for activating the colorblind mode and using a colorblind friendly color scale when it’s activated.

4.7 Build script

A build script was made in Python to help Cadence with further development implementing this interface into their tool stack. The Python script imports all the csv data from PinDown

into the JavaScript file where this interface code is run from. The JavaScript file can be seen in figure 4.7 named *index.js* and contains the functional code for the interface. The Python script loads data from two csv files into the JavaScript file as two variables by writing two lines of text into *index.js* and then creating a copy of the JavaScript file and placing it in the same folder named *packedIndex.js*. If this isn't done the JavaScript file can't read the csv data without a web server because of the CORS policy. This solution solves that problem by creating a static website that doesn't need a webserver and can be run locally.

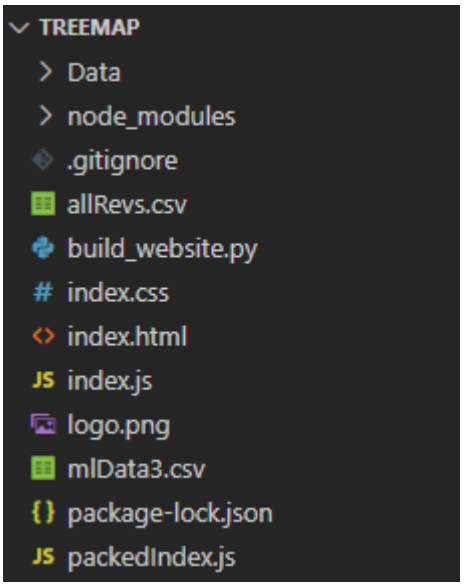

Figure 4.7 An overview of the interface programs files.

Chapter 5
Results

The result of this thesis has become an visualisation interface for Cadence own debug program PinDown, made to be an static website with JavaScript/HTML/CSS.

5.1 Interface

The interface is built upon a treemap, visualising the selected data by the user from PinDown. It only shows the user two sets of data categories at a time by changing the size and color of the treemaps elements. The data selection can be done in a small menu at the topmost part of the interface. It has two dropdown menus for changing color and size of the treemap elements (as seen in figure 5.1.1), there are also two menus for filtering out commits depending on their dates (e.g figure 5.1.2).

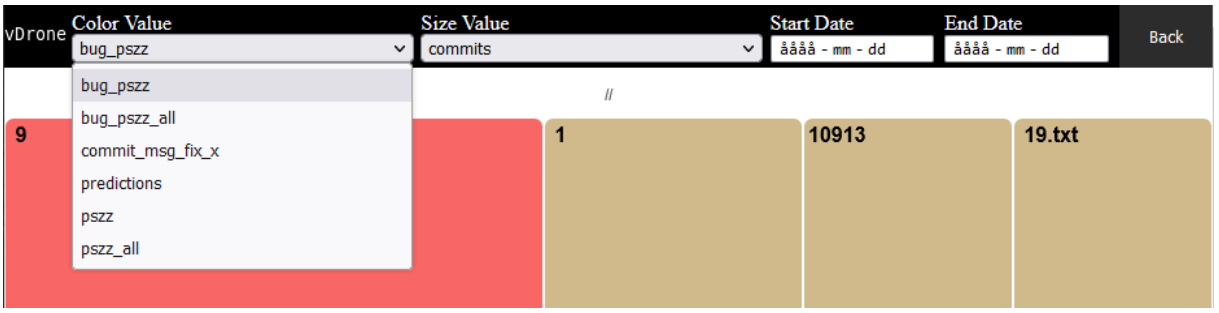

Figure 5.1.1 Showing the color value dropdown menu.

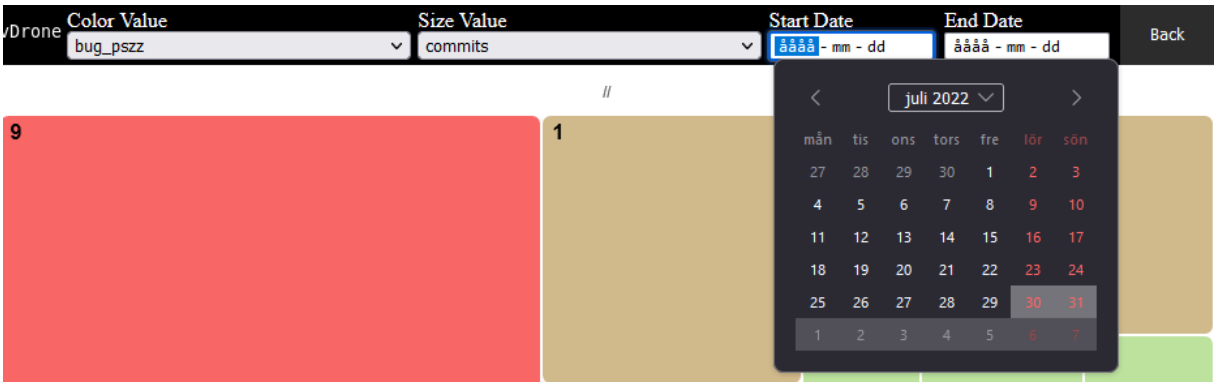

Figure 5.1.2 The date filter menu opened up.

Each treemap element has a label in the top-left corner which displays either a folder name or filename. If the element represents a file its file extension will be written into the label or else, if the element represents a folder only the folders name will be shown. The user can then traverse the filesystem by left clicking their mouse on a folder to traverse into the next layer of the filesystem or they can use the back button in the top menu to traverse out from a layer.

When the user hovers its mouse pointer over a treemap element a small menu will be displayed and show a more detailed view of the element's data. As seen in figure 5.1.3 the label of the element is shown first and then the full file path can be seen on the second row, the third row shows how many commits have changed something in that folder or file. The next two lines change depending on what the user has chosen in the size and color value menu, the user can see the exact value for the chosen categories in this menu. The last two lines changed depending on if the element is a file or folder. If the element is a folder the user will be able to see how many subfolders and files that folder contains, if it’s a file that information won’t be prevalent.

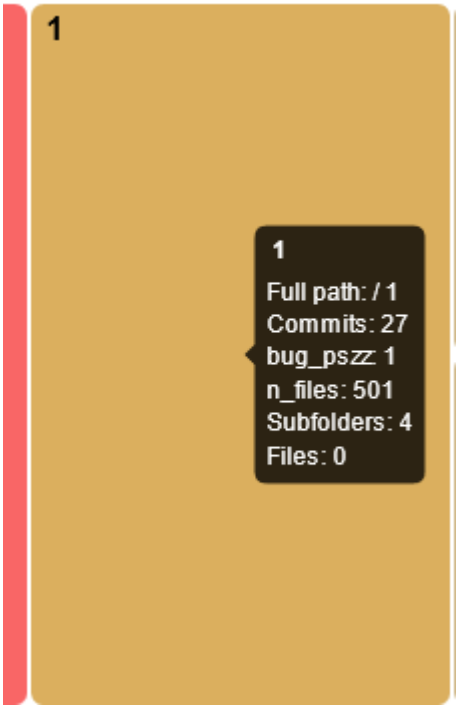

Figure 5.1.3 Detailed view of a folder element.

When a user right clicks a treemap element a context menu appears (e.g figure 5.1.4) and is given three choices. The exclude folder option adds that folder to a list of excluded folder paths and the interface will update its view to not contain any file or folder containing that file path. The commit message option will open up a new menu containing all the commit messages of the commits that have changed the targeted folder or file (e.g figure 5.1.5). Lastly the file filters options will open up the filter menu allowing the user to manage the applied file/folder filters.

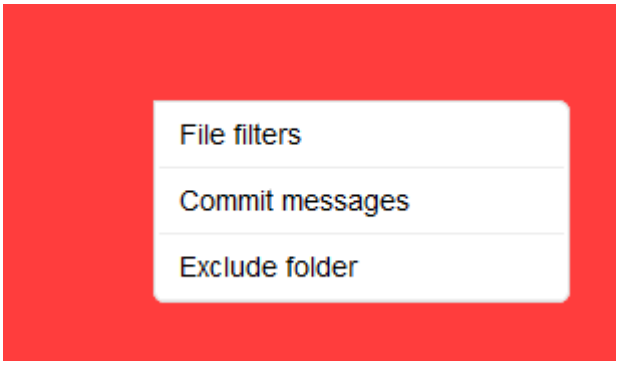

Figure 5.1.4 Right click context menu of a treemap element.

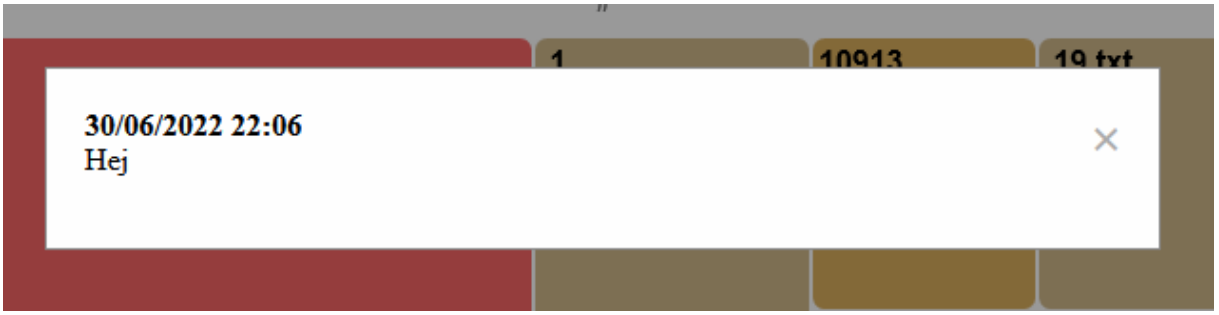

Figure 5.1.5 Commit message menu.

The file filter menu as seen in figure 5.1.6 has three functions, reset folder filters, reset file filters and apply new file filters. The user can see a list of applied folder or file filters in the text boxes below its corresponding header, folder filters to the left and file filters to the right. When the user applies a new file filter by pressing the apply button it will update the treemap view accordingly and add it to the file filters list above.

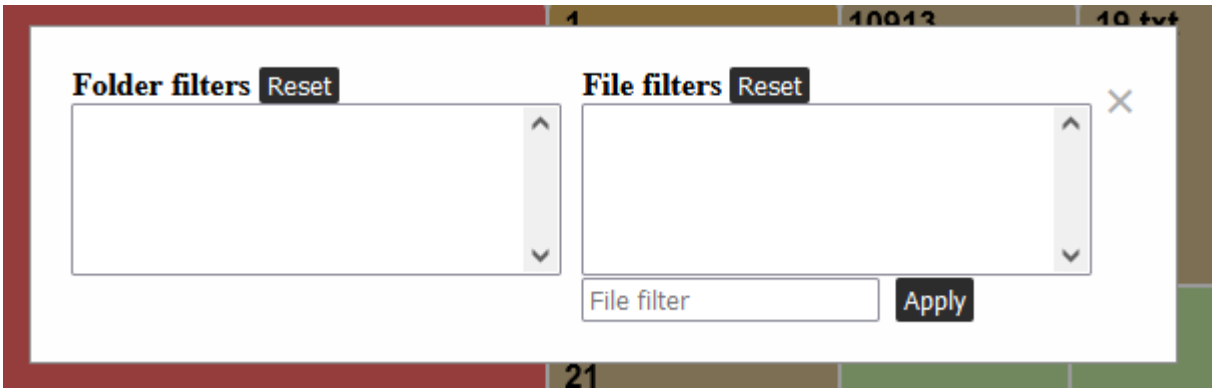

Figure 5.1.6 File filter menu.

Chapter 6
Conclusion

This thesis had two goals, review different data models and find one that fits PinDowns output and has the best user feedback. Then implement that data model into a static website.

At the start of this thesis the student received a reference program made by a previous student called CityScapes. It was used as a template for what functions Cadence wanted from this new program created in this thesis. The finished product as seen in chapter 5 is therefore a reconstructed version of CityScapes created with a different data model and spatial view.

6.1 Answers to problems

Below the answer for the presented problems in chapter 1.4 will be answered.

6.1.1 What type of data structure will be used for visualisation?

During the lo-fi prototyping phase in chapter 3, three different data models were analysed. Out of those three models the treemap had the best feedback from the test users including Cadence themself. Therefore an unanimous decision was made by the student and Cadence to continue further development using the treemap data model. During the hi-fi prototyping phase no issues were found regarding the model and would therefore continue into production.

6.1.2 How do we display the bugs that have occurred in the commits?

The user will be able to choose what type of data they want to show in the interface. The data values will be represented as either color differences in the treemap elements or size differences. The data values shown is a summarised value of all the commits that have changed the current file the user is looking at. Each file or folder will be shown as a treemap element.

6.1.3 What type of programming language and framework will suit us best?

Cadence stated early on in the project that it was necessary for the interface to be able to run on a web application. Besides needing the interface to be compatible with a web application it was also necessary for the interface to not have any build dependencies. The best solution found was to make a static website with JavaScript and HTML with no additional build dependencies.

6.1.4 How do we read data from files and use it in the interface?

A Python script was created that will read the data from PinDowns csv files and then create a copy of the JavaScript file with the csv data printed as an JavaScript variable. As seen in figure 6.1.4 the main JavaScript file contains the code to run the interface.

```
var csvData

function main() {
    print("hello world!")
    print(csvData)
}
```

Figure 6.1.4 Example main JavaScript file with no csv data.

The Python script will then create a copy of that JavaScript file and write two new lines. These new lines will contain the csv data in a one line string format as seen in figure 6.1.5. (Note that in figure 6.1.5 there's only one variable for demonstration purposes.)

```
var csvData = "file,date,commitmessage\n98,12/17,commit
message\n231,08/2,another commit message\n"

function main() {
    print("hello world!")
    print(csvData)
}
```

Figure 6.1.5 Example of the final JavaScript file created by the Python script.

The Python script was made so the interface could be run as a static website, so it could be later implemented into Cadence's Verisium tool stack. If it wasn't a static website there would be a need for a webserver to run to allow the interface to access the csv files because of the CORS policy [8].

6.2 Future development

The interface created can take a long time to start depending on the size of the data it has to read. There aren't a lot of solutions to this that wouldn't create other problems because it's a hardware limitation and not a software limitation. One solution is to optimise the data reading functions, instead of looping through every line of data the interface could instead figure out what lines it needs and disregard the rest of the data.

Besides optimising the interface speed improvements can be made to the design itself as well. The commit message menu seen in figure 5.1.5 doesn't have a scroll bar, which means that when there's a lot of messages the window will expand beyond the size of the treemap window. This results in a lot of whitespace below the treemap that could be avoided if the message window was contained inside a text box with a scroll bar.

6.3 Reflection of ethical aspects

This interface was made using confidential data from Cadences partners, but the data had been anonymised. Because the data had been anonymised there wasn't a big concern for breaching any confidentiality policies such that there was no information in the used data that could reveal any secrets. As for the people using the interface this will have a net positive outcome for them since they don't have to inspect hard to read data dumps in massive text

files to decipher their data. Hopefully they will feel this is a more relaxed approach to their research and work.

References

[1] N. Babich. “What is UX Design? User Experience Definition”. Adobe.com. [Online] https://xd.adobe.com/ideas/career-tips/what-is-ux-design/ (accessed Dec. 19, 2022)

[2] P. Morville. “User Experience Design”. SemanticStudios.com. [Online] http://semanticstudios.com/user_experience_design/ (accessed Dec. 19, 2022)

[3] Claus O. Wilke. *Fundamentals of Data Visualization*. O’Reilly Media, Inc. Accessed Dec. 19, 2022 [Online]. Available: https://clauswilke.com/dataviz/

[4] Website for windirstat, (accessed Dec. 19, 2022) https://windirstat.net/

[5] Website for Cadence, (accessed Dec. 19, 2022) https://www.cadence.com/en_US/home/company.html

[6] Wikipedia article for Voronoi Diagrams, (accessed Jan. 02, 2023) https://en.wikipedia.org/wiki/Voronoi_diagram

[7] Wikipedia article for Gosper Curves, (accessed Jan. 02, 2023) https://en.wikipedia.org/wiki/Gosper_curve

[8] Website for MDN Web Docs, (accessed Jan. 02, 2023) https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS/Errors/CORSRequestNotHttp

[9] Laura Bushe. “The Skeptic’s Guide To low-Fidelity Prototyping”. SmashingMagazine.com. (accessed Jan. 11, 2023) https://www.smashingmagazine.com/2014/10/the-skeptics-guide-to-low-fidelity-prototyping/

[10] Website for IBM, (accessed Jan. 16, 2023) https://www.ibm.com/cloud/learn/rest-apis

[11] Wikipedia article for Comma separated values, (accessed Jan. 02, 2023) https://en.wikipedia.org/wiki/Comma-separated_values

[12] Website for W3Schools, (accessed Jan. 16, 2023) https://www.w3schools.com/html/html5_canvas.asp

[13] Website for Treemap extension to Chart .JS, (accessed Jan. 02, 2023) https://chartjs-chart-treemap.pages.dev/

[14] Website for Chart .JS, (accessed Jan. 02, 2023) https://www.chartjs.org/

[15] M. Beedle et al. “Manifesto for Agile Software Development”. AgileManifesto.org. [Online] https://agilemanifesto.org/iso/en/principles.html (accessed Jan. 16, 2023)

[16] Wikipedia article for Akademiska poäng, (accessed Jan. 02, 2023)
https://sv.wikipedia.org/wiki/Akademiska_po%C3%A4ng

[17] “What is prototyping”. Interaction Design Foundation. (accessed Jan. 23, 2023)
https://www.interaction-design.org/literature/topics/prototyping