UNIVERSITY OF TWENTE

CREATIVE TECHNOLOGY

BACHELOR THESIS

An Interactive Visualization Explaining Probabilistic Fault Tree Computations

Author

J.L. VAN DEN BERG

July 5, 2020

Abstract

The design of fault trees is standardized and not using its full potential in conveying important information. Fault trees are commonly used in reliability engineering. This model is used to investigate possible failures in systems such as airplanes, software or nuclear power engineering. In this study we propose a new design for this visualization which is more intuitive and understandable. The design will not only be able to describe the qualitative relationships between subsets, but also its quantitative properties. This new design will be used in explaining fault tree mechanics to non-experts.

Based on a literature review of visualization techniques and an inspection of existing fault tree designs, a new design will first be designed in Adobe Illustrator. This design will subsequently be realized in the form of an interactive web application written with the p5.js library. Expert opinions are used throughout the whole designing process to ensure that all the information of the regular fault tree is used in a correct way in the new design. And to ensure that the tool is useful to the experts when they need to explain fault tree mechanics. This process resulted in an innovative fault tree design that, according to the consulted experts, puts fault trees in a new and more understandable perspective.

Acknowledgement

First of all I would like to give thanks to Prof. Dr. Marielle Stoelinga for supervising me in this project. Even though we had to shift to online education in the middle of the project, due to the pandemic, the supervising was not much affected. Additionally, I want to thank Dr. Arnd Hartmanns for being the critical observer. I would also express my gratitude to the expert reviewers of this graduation project. They gave me useful insight in the world of reliability engineering. I would like to thank Karlijn Wiggers and Ahn Tuan Nguyen for cooperating, as they also have a graduation project about fault tree design. Last but not least, I want to thank my friends with whom I shared many ideas and who provided me with the necessary distraction. And of course my parents for always supporting me.

J.L. van den Berg, 02/07/2020

Contents

1 Introduction

Understanding risk is vital for the engineering process of i.a. nuclear power plants, airplanes or digital infrastructure. Without elaborate reliability evaluation, it is simply impossible to deliver a safe product. There are numerous ways these complex systems can fail. Especially combinations of failing elements in systems can become complicated to understand. E.g., the defect of part A in an airplane does not cause the airplane to crash, but a combination of a defect in part A, unfavourable weather, a software bug and a defect in part B, will cause the airplane to crash. Because there was no method to organise all these cluttering possible failure combinations, the US Nuclear Regulatory Commission introduced in 1981 a handbook for so-called fault tree analysis (FTA)[1].

Fault trees are a type of decision trees that provide insight in the causes of a system failure. Fault trees consist of both events and Boolean gates. The top event represents the failure of the whole system and the bottom events are events that will occur spontaneously. With this diagram, it is possible to intuitively examine how a single failure can lead to the failure of the system. It is also possible to add probabilities of failure to this diagram, this enables a wide range of statistical analysis methods. Fault trees, and all its variations, are currently widely adopted in reliability engineering [2][3]. An example of a standard fault tree is shown in figure 1. The top-event of figure 1 fails if either there is a Common cause seal failure or an independent seal failure.

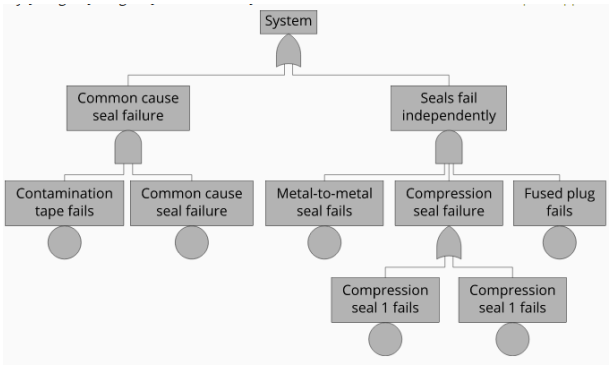

Figure 1: An example of a fault tree [4]

The fault tree is a visualization of the failure relationships between elements of a system. It is also possible to describe fault trees in plain text, in contrast to a visual diagram. The following text

represents the tree of figure 1 in Galileo format.

```
toplevel␣"System";
"System"␣or␣"CommonCauseSealFailure"␣"SealsFailIndependently";
"CommonCauseSealFailure"␣and␣"ContaminationTapeFails"␣"CommonCauseSealFailureBE";
"SealsFailIndependently"␣and␣"Metal-To-MetalSealFails"␣"CompressionSealFailure"
␣"FusedPlugFails";
"CompressionSealFailure"␣and␣"CompressionSeal1Fails"␣"CompressionSeal2Fails";
```

The information in figure 1 is faster processed by the human brain than the textual description, even though their semantics are similar. The human brain is very sensitive to visual cues, the visualization of the example mentioned above is, not only much faster processed, but it is also easier to recognize patterns. Given the positive effects of visualization in the previous example, it is worth to investigate if the standardized design of figure 1 can be improved even more. As the current design is focused only on experts, a new design may enable non-experts to use fault trees more. When the FT design is reviewed, it may be possible to increase understanding of the fault tree and subsequently increase the reliability engineering process. An insufficient reliability engineering process can have drastic consequences, such as unsafe computer software, malfunctioning bridges or the crash of a Space Shuttle [5]. No research has been conducted on the current design of fault trees.

The aim of this thesis is to create an innovative, interactive, fault tree design. The end product will be presented on a website to enable interactivity and be open-access. First a literature research is conducted, this part includes relevant background information and a state-of-the-art survey. Secondly, a design process is completed with the help of experts and a product is realized.

2 Background

2.1 Fault tree analysis (FTA)

In order to create an improved, innovative fault tree (FT) design, first the mechanics of FTs must be examined. Fault trees are widely adopted in reliability engineering [2][3]. The goal of these diagrams is to identify insight in the combinations of events that can cause a system to fail. In FTs, three event types can be identified; *basic events*, these have an own probability of occurrence in a certain time frame. *Intermediate events* are the consequence of (multiple) basic events. The *top event* represents the failure of the entire system the FT describes [3].

Events are connected with *logic gates* to describe which combinations cause a failure to propagate towards the top event. The most used gates in FTs are *OR* and *AND* gates. Next to these, common gates are *k/N* gates, these are basically OR gates with a number (N) of inputs and a threshold number (k). If *k* or more events happen, an output is produced and the failure propagates further towards the top event. The final gate is an INHIBIT gate. This gate behaves exactly like an AND gate, but is used in FTs to make the FT easier to understand. These gates are shown in figure 2. The gates have two or more inputs and one output [3].

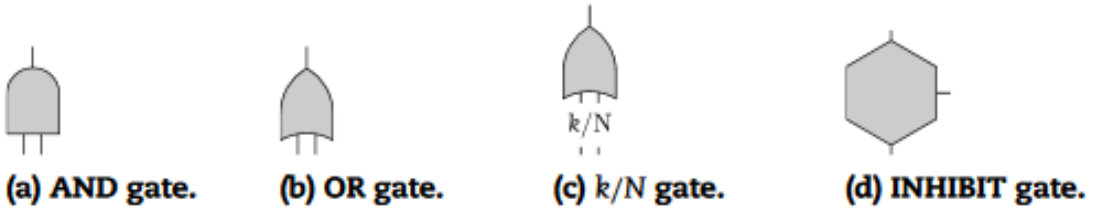

Figure 2: The most common gates in fault trees [3]

Two types of FTA can be distinguished, *qualitative* and *quantitative* FTA. The main goal of qualitative FTA is to determine the *cut sets* of a FT[6]. A cut set is a combination of components that can cause the whole system to fail [3]. Often, there are more ways that a system can fail, so there are also multiple cut-sets. The cut-set with the least amounts of components in it is called the *minimal cut-set*. Subsequently, a system with many components in its minimal cut set is generally more reliable than a system with only a few components in its cut set. There exist multiple algorithms and methods to find cut sets of a FT, but these are not discussed in this research.

Quantitative FTA focuses on the the probability of failure calculation of a FT [3]. Each basic event has its own probability of failure in a time frame. This method of evaluating FTs with numerical failure probabilities is useful for comparing design alternatives [7]. The quantitative approach to fault trees is useful to calculate if certain requirements will be met in certain design alternatives. There may be a minimal mean time between failures of availability. In addition to this gives quantitative analysis a more complete overview of a system's reliability compared with qualitative analysis. The components of the minimal cut-set could e.g. have an infinitesimal probability, while a larger cut-set could have components with a high probability of failure, making the latter more reliable.

Probability theory plays a big role in this. If the probability of failure of basic events is known, it is possible to calculate the probability of the intermediate events and, subsequently of the top event. For an AND gate, for example, holds that its probability of failure is equal to the product of all its inputs. This is a simple calculation as long as there are no shared subtrees. If these are present in the fault tree, it is not possible to calculate the probabilities of a fault tree from the bottom up [3]. When Basic events are shared there is dependency between the inputs and the bottom up approach does not hold.

Even though FTA is widely adopted, it has some drawbacks. The fault tree mechanics that are described above are for *static* fault tree analysis. These static models are not able to, in contrast to *dynamic* trees, adapt to a changing environment. Dynamic fault trees can additionally model time- and sequence-dependent failures [3]. In addition to this, multiple research papers argue that the fixed values of probability in quantitative fault trees is a drawback [2][8]. Despite these drawbacks, this research will focus on static fault tree, as they are relatively uncomplicated for non-experts.

2.2 Information Visualization

Graphics have been used for thousands of years to help people think. They function as a form of *external cognition* for humans. This external cognition extends a person's working memory, according to Card et al. and Ware [9][10]. A famous example is writing down calculations. Card et al. claim that the time to calculate a multiplication can be decreased by fivefold when using a piece of paper and pencil. Visualization is thus a powerful tool to aid cognition. This amplification of cognition by information visualization has six causes:

1. Visualization increases the memory and processing capabilities of the user
2. Visualizations help reduce search for information
3. Visualization uses visual representations to increase pattern detection
4. Visualizations enable the user to make assumptions very easily that were otherwise invisible
5. Visualizations utilize perception mechanism of the human brain
6. Visualizations can easily be modified

Card et al. define visualization as: *The use of computer-supported, interactive, visual representations of data to amplify cognition.* Ware defines it as: *a graphical representation of data or concepts to support decision-making.* The goals of visualization are discovery, decision making and explanation.

This effective tool can also be applied to reliability engineering. After reviewing literature about risk visualization, Martin Eppler and Markus Aeschimann [11] came to the following definition:

> *In the context of organizational risk management, risk visualization designates the systematic effort of using (interactive) images to augment the quality of risk analysis and communication along the entire risk management cycle. Risk visualization employs charts, conceptual diagrams, visual metaphors and mapping techniques to improve the understanding and subsequent management of risks.*

According to this definition, the fault tree is a subset of the risk visualization tools. Other tools are attack diagrams, decision diagrams or maps.

2.3 Risk communication

Next to the right medium for risk communication, the process of communicating is also of importance. In the 1970s this process was reserved solely for experts [12]. However, Roth argues that this view has shifted from expert based to an interactive process that involves a wide spectrum of stakeholders, policy makers, experts and the general public [13]. Bjögvinsson et al. [14] support this shift to a more participatory approach in design. The researchers call this the shift from designing "things" (objects) to "Things" (socio-material assemblies). In Nordic and Germanic societies, "Things" may refer to governing assemblies. The parliament of Norway is called *Stortinget*

from Stor *(great)* and ting *(in old Norse: an assembly of nearby clans and led by a jarl)* [15]. Even though risk communication has become more participatory, Roth claims that the process is still too hinged upon the experts, due to the complexity of the problems, technical jargon and lengthy reports. To support the suggested transition to "Things", the fault tree design must be revised as the current design is made for engineers and not the general public.

Research of Slovic clearly indicates that there is a great difference between the perception of risk between experts and non experts [16]. From 30 activities and technologies, such as hunting, smoking or food coloring, non-experts rank nuclear power as the most risky activity or technology. However, experts rank it on place 20. That is even less risky than e.g. food preservatives. This misjudgement of risk can be solved by better informing non-experts.

Research suggests that informing people about risks using graphics instead of text is recommended [17][18]. Likpus also points out that graphic displays in risk visualization can be extremely helpful in conveying this kind of information. E.g. by enabling interaction and mathematical operations. Likpus recommends using graphic displays if the goal is to promote quantitative information, which is the case in this thesis.

However, the researcher points out some complications with using visualization in risk communication:

1. Patterns in data may discourage people from looking at the details
2. Some graphs are not well understood because of faulty design
3. The skill level required to read the graph may be too high for some users.
4. Interactive visualizations may require technology that is not widespread available
5. Visualizations may be challenging to create
6. Visualizations can be misleading.

These complications are avoidable, but should be taken seriously.

A pitfall in the communication of information to other people is the so-called *Curse of Knowledge*. Xiong et al. [19] define this psychological phenomenon as: *Well-informed decision makers fail to predict the judgements of less-informed decision makers, implicitly allowing their own knowledge to guide those predictions.* They show in their research that this phenomenon is also applicable with

information communication through visualization. This means that when an expert or designer observes a pattern in data, the expert assumes that it will also be observed by non-experts. In the experiment, the users were deliberately made biased about a dataset. Subsequently, they were asked to interpret graphs. Based on the extra knowledge that they gained before the interpretation, they adopted a certain perspective about the graph. This misinterpretation must be avoided when designing a visualization. The authors presume: *the inability to separate one's own knowledge and expertise from that of their audience can make visual data communication more difficult and less clear than presenters realize.* Therefore it is crucial for the expert or designer to determine the level of knowledge of the user in order to convey the right message.

3 Research Questions

The aim of this thesis is to create an innovative and interactive fault tree design that can be used to explain the workings of quantitative fault trees. The corresponding research question is:

> *How can innovative visualization methods make quantitative fault tree analysis more understandable?*

To aid the answering of this question, the following sub question is formulated:

> *What visualization methods improve attractiveness in information visualization?*

The sub question is answered in the State-of-the-Art section since this answer is of importance before the development phase of a new fault tree design. The main research question is answered in the conclusion of this thesis.

4 State of the Art

In this section the state of the art of data visualization are discussed. Afterwards, these concepts are applied to existing interactive fault tree designs to evaluate existing fault tree designs.

4.1 Data visualization

In this section, the basis of graphic design is discussed. Starting with two foundation books written by Ward and Ware. Subsequently, we elaborate on a psychological phenomenon about communication between experts and non-experts. Finally, a risk visualization framework is discussed.

4.1.1 Interactive Data Visualization - Visual variables

The book *'Interactive Data Visualization'* is a guidebook for creating digital (interactive) data visualizations [20]. Ward specifies *eight* ways of how graphical objects can represent information. These ways are called *visual variables*. The eight variables are: position, shape, size, brightness, color, orientation, texture and motion. In the next part, each variable will be discussed.

1. *Position* is the most powerful of all according to Ward. He mentions that the spatial arrangement of graphics is the first step when reading a visualization. A visualization that uses position in a highly insightful way is the scatter plot, see figure 3. The relationship between two variables is easy to distinguish with this plot. Other trends and clusters can also be discovered using scatter plots.

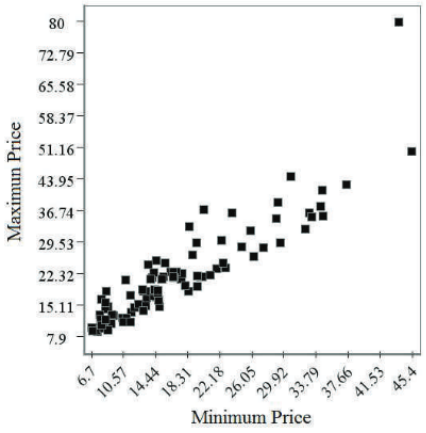

Figure 3: Using position to express the relationship between two variables [20]

2. Any graphical object can be used as a *shape* variable in visualizations. Ward emphasizes that the designer should use the same size of the shape when working with multiple shapes, as size is another variable. In addition, the shapes should be easy to distinguish from each other to make the visualization efficient. Finally, shapes should have a similar design and complexity to avoid accidentally highlighting shapes. Figure 4 shows an example of using shape in a scatter plot.

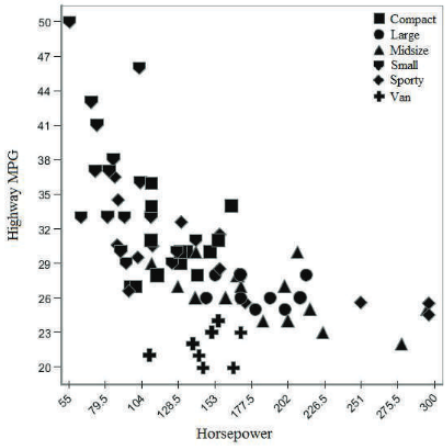

Figure 4: The shape variable is used in combination with position to distinguish car types, their fuel consumption and top speed [20]

3. *Size* can be applied to three dimensions: length, area and volume. In the first dimension, length, size can work excellent. The bar chart is a classic example where length can represent data well. Area and volume, on the other hand, are less useful for data visualization because it is difficult for humans to perceive a value for an area or volume, an example of this is shown in figure 5.

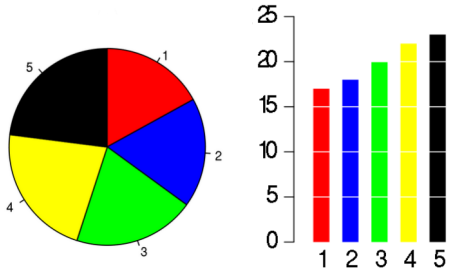

Figure 5: The human brain has difficulties measuring the quantitative data of the pie chart. Whereas a bar chart is easily interpreted [21]

4. *Brightness* has a limited use as visual variable. The reason for this is that it is difficult to discriminate levels of brightness. It is advisable to use a linear brightness scale to maximize the differences between classes. Figure 6 shows an example where brightness is used in a scatter plot.

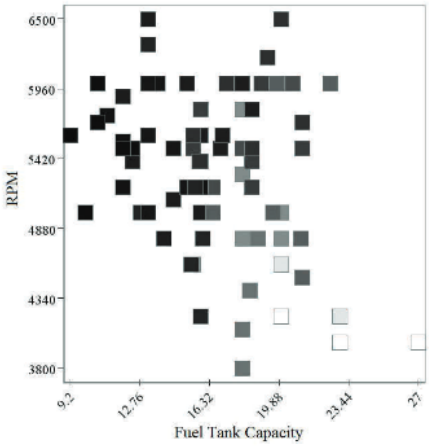

Figure 6: The brightness variable is used to indicate the width of cars, the darker the square, the wider the vehicle [20]

5. *Color* can be used as a visual variable in two ways. First, the dominant wavelength of the color (*hue*), this is what most people think of as a color, can be used as a variable. Secondly, the *saturation*, is the relative distance of hue to grey. Besides using color as a variable, color contains emotional associations. Clarke et al. [22] conducted researched the emotional connotations of color. Colors like blue were associated with calmness, neutral and coldness. Red was associated with love and anger. Another often used color association is the color red with failure, and green with success. These associations must be taken into account when using colors for graphic design.

6. The use of *orientation* of an object can serve as a visual variable. An important condition for this is that the object should have a natural single axis. I.e., having symmetry on only one axis, see figure 7.

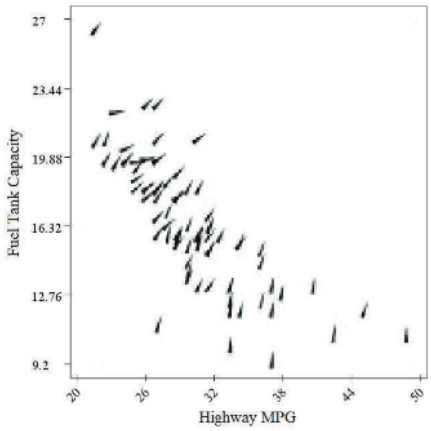

Figure 7: Orientation is used together with position. The orientation of the mark represents the price.

7. *Texture* is a combination of multiple, previously mentioned, variables. These are orientation, the size of the texture and the density of the texture (color), see figure 8.

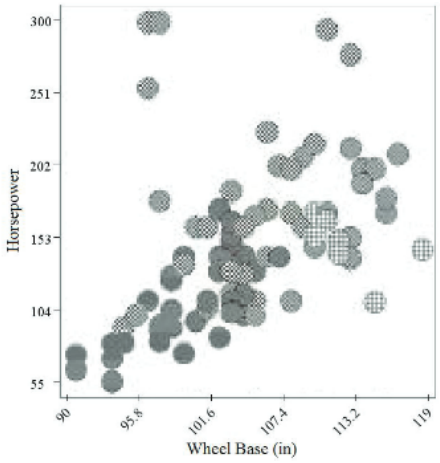

Figure 8: A scatter plot with a texture variable. The texture of the mark represents the class of the vehicle.

8. The final visual variable, *motion*, is difficult to express on a static page. But it is powerful if used correctly in digital visualizations. It combines the previously mentioned variables with a change over time.

4.1.2 Interactive Data Visualization - Visualization Techniques for Trees, Graphs and Networks

In the same book as in section 4.1.1 Ward specifies design rules for graphs and tree diagrams. As fault trees are a type of tree diagrams, this is relevant for this thesis. These rules are:

- Minimize line crossings
- Maintain a pleasing aspect ratio
- Minimize the total area of the drawing
- Minimize the total length of the edges
- Minimize the number of bends in the edges
- Minimize the number of distinct angles or curvatures used
- Strive for a symmetric structure

4.1.3 Information visualization perception for design

Ware's Information visualization perception for design covers a wide scope of fundamental information visualization concepts [10]. These are important to grasp before the design phase of this thesis starts. The most important concepts are summarized below.

- Gray scales are perceptually altered by background lightness. They are in addition unreliable when they are used to communicate quantitative information.
- There must be enough contrast between the foreground and background. Otherwise, the foreground information is difficult to distinguish from the background. Black or white borders can help increase contrast.
- Use high saturation colors for small color-coded objects and use low saturation colors for large color-coded objects.
- Use images and words together. This combination increases understanding and are remembered better.

- An interface for and interactive visualization should not increase the cognitive load. The interface must be able to be used rapidly by a newcomer.

In addition to these concepts, Ware elaborates on the so-called *Gestalt laws*. The Gestalt laws are principles that describe how the human brain recognizes patterns when viewing a visual. Although some principles may seem straightforward, they can have a huge negative impact on visualizations if they are not used correctly. The Gestalt laws are listed below.

- *Proximity*, components that are close together are grouped together by the mind.
- *Similarity*, components that have a similar shape are grouped together.
- *Correctness*, components that are connected by lines are grouped together.
- *Continuity*, components that are smooth and continuous are more likely to be constructed as a visual entity than components that have sharp corners.
- *Symmetry*, symmetrically arranged lines are noticed to be more strongly. A condition for this law to apply is that the object should not have too many features, otherwise the Symmetry is hard to recognize.
- *Closure*, components that have closed contours are likely to been seen as an object.
- *Direction*, in order to give direction to an object, the link between object must be made asymmetrical (an arrow).

4.1.4 Framework for risk visualization

Eppler and Aeschimann proposed a framework to aid the risk visualization design [11]. The framework consists of five elements: *why, what, for whom, when* and *how*. The framework is shown in figure 9. It is a circular framework as the sequence of the questions matter. A benefit of this circular design approach is that it is possible to reiterate after a cycle is completed. One may generate interesting ideas during a cycle, so a possibility to reiterate is desirable.

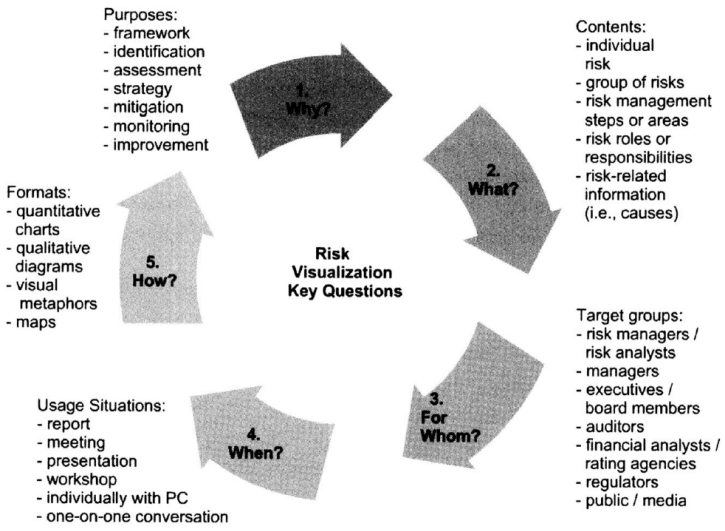

Figure 9: The risk visualization framework

In addition to the framework, the authors derived a set of guidelines that managers or risk analysts can follow in the visualization of risks. Even though the writers admit these guidelines are not definitive or comprehensive, they can be of use for designers. The guidelines are:

- Do not precipitate the use of a visualization
- Consider the application context and its constraints
- Make sure that risk visualization respects the basic rules of visualization and perception
- Avoid decorative visualization without added benefit
- Think visualizing, not visualization. The process of creating and modifying a risk visualization is as important (if not more) as the final result
- Pre-test the risk visualization

4.2 Evaluation of current FT graphic designs

In this section, the design of three existing fault tree tools is evaluated. Faulttree+, a professional software package and two non professional web applications. The evaluation is based on the visual rules that have been discussed in the previous section. After each design is briefly discussed, the designs will be rated on several criteria using a five-level Likert scale. These criteria are:

1. *Color and contrast* How is the use of color? Do large areas have a low saturation color and small areas high saturation color? Is there enough contrast so that everything is readable and clear.

2. *Adaptability* Is the arrangement of the fault tree adaptable? Is styling adaptable?

3. *Use of visual cues* Is the design correctly using visual cues to guide the user? Is the attention of the user guided properly? E.g. by using arrows.

4. *Compliance with design rules* Are there other design rules that are not adhered to? Is the design unambiguous? Is there miscommunication possible.

Tool	Color and contrast	Adaptability	Use of visual cues	Compliance with design rules
Faulttree+	o	+	+	+
DFT Visualization	–	-	-	o
Visual Paradigm Online	++	++	o	+

Table 1: The scores of the three evaluated fault tree tools

4.2.1 Faulttree+

Faulttree+ is a fault tree analysis software package by Isograph. With Faulttree+, it is possible to compute minimal cut sets and predict failures. It has a wide support for libraries and is also used for ISO (International Organization for Standardization) safety assessment [23]. The user-interface is shown in figure 10.

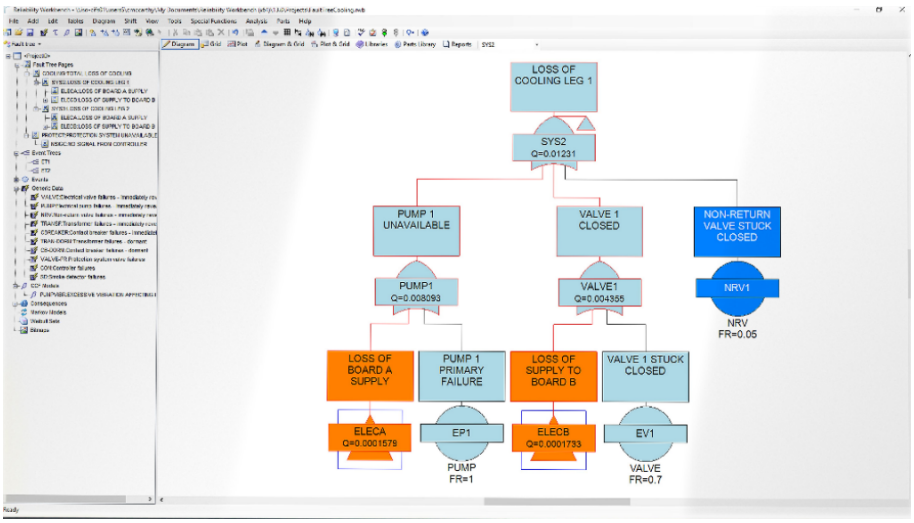

Figure 10: The UI of Faulttree+

The standard color of the elements is grey blue, no matter which gate or element is created by the user. But the colors are adaptable afterwards. The contrast between the background and the fault tree is good, however the lines connecting the elements could be a pixel thicker. Symmetry is not always generated automatically and there are some unnecessary angles in the links between elements. When a cut-set is shown, the line between the elements becomes red. More visual cues can be added by the user. There is a strong uniformity in fault trees in Faulttree+ because textboxes are of similar size. In the table below the scores for each criteria is shown.

Color and contrast	Adaptability	Use of visual cues	Compliance with design rules
o	+	+	+

4.2.2 DFT Visualization

Jungen and Volk made, using the Javascript library 'Cytoscape', a graphical user interface where one can create their own fault tree [24]. It supports both static fault trees as dynamic fault trees. This is a sufficient environment to build your own fault tree, but it does not have any analysis features. The UI is depicted in figure 11.

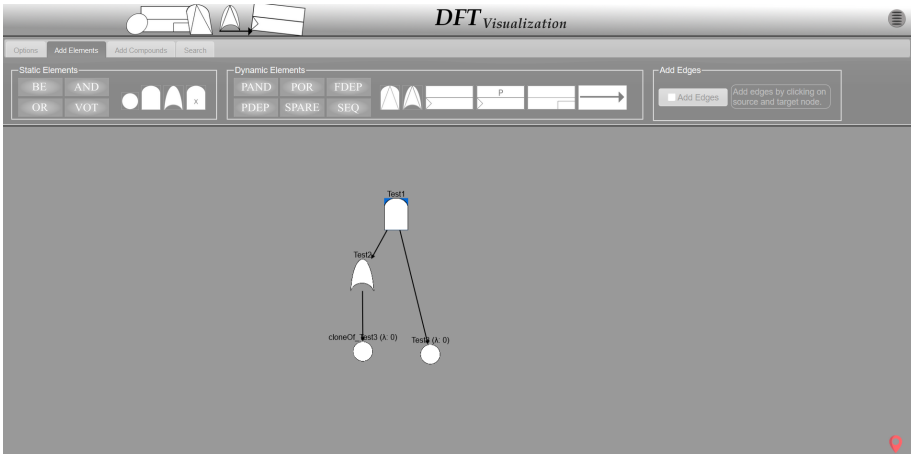

Figure 11: The UI of the tool

The interface solely consists of white and shades of grey, this makes it unattractive to use the built fault trees for publication to the general public. In addition to this, all the lines between elements are straight. When creating a FT with this tool, the elements are not lined up automatically so that a hierarchy between the top event and other events is made. This reduces insight in the fault tree because it will be more difficult to intuitively recognize cut-sets. The lines between the elements are actually arrows. This visual cue is unusual in FTA since a fault tree can be analysed by a user in both directions. Below, the scores for each criteria is shown.

Color and contrast	Adaptability	Use of visual cues	Compliance with design rules
- -	-	-	o

4.2.3 Visual Paradigm Online

Visual Paradigm Online (VPO) is an intuitive, web based, graphic editor [25]. It could be used to create block diagrams, flow charts and fault trees. Figure 12 shows a screenshot of the UI.

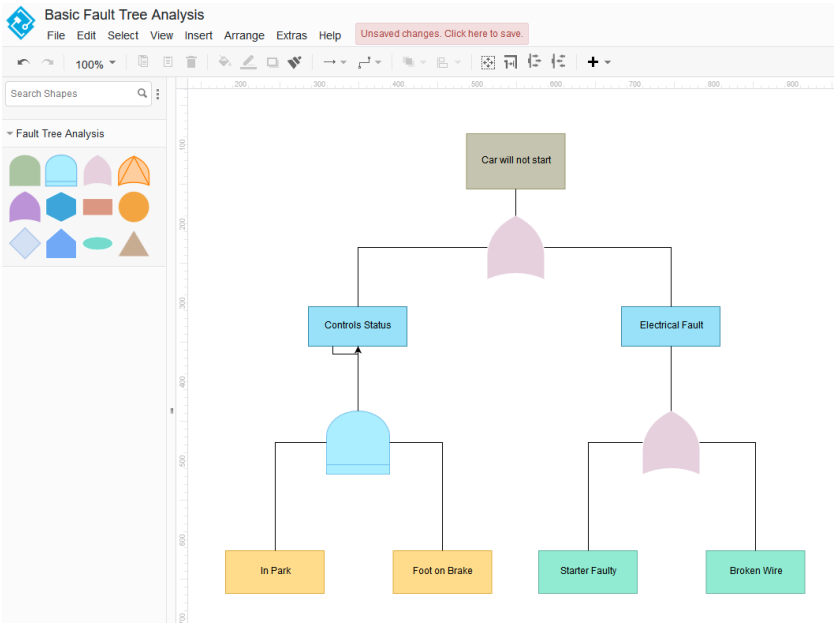

Figure 12: The user interface of Visual Paradigm Online

It is possible to drag and drop all the fault tree elements to the editor window. Connections between elements can be made by drawing lines between elements. Additionally, it is possible to change the color of each individual component. The only functionality of VPO is to create a visual representation of a FT. It is not possible further analyze the FT using algorithms. Each element has its own, customizable, low saturation default color, which is desirable. The logic gates do not have a textbox with their function on it, this could be helpful for non-experts. Lines are either straight, curved or straight with 90 degree angles. Another advantage is that the gates are easily lined up on the y-axis as, VPO includes snapping to position. There are no visual cues present. The scores for this product are shown below.

Color and contrast	Adaptability	Use of visual cues	Compliance with design rules
+ +	+ +	o	+

5 Ideation

The ideation phase will be supported by the framework of Eppler and Aeschimann [11] as discussed in section 4.1.4. Figure 9 depicts the framework. Subsequently, a use case scenario for the product is given and the user requirements are listed. Finally, an idea is presented.

5.1 Risk visualization framework

5.1.1 Why

In the State of the Art section it became clear that existing fault tree designs do not comply with all visualization rules. There was confusing color use, asymmetric design and too many bends in the edges. Additionally, there was no use of visual variables. The use of an extra visual variable and a well designed tree will make the fault tree design more attractive and understandable, leading to an improved risk assessment. In addition to this can a more attractive FT lead to more engagement of non-experts in the risk assessment process.

5.1.2 What

Following the main research question, the probability of failure of elements and the failure dependencies between system elements must be visualized by this fault tree. In addition to this, the structure of the design should be like the fault tree. The fault tree that we will use is depicted in figure 13.

5.1.3 For Whom

The target audience are students that have some basic knowledge of probability theory and logic gates. This fault tree must engage the students and generate initial insight in fault tree models.

5.1.4 When

The visualization must be used by an individual from a web browser. But it can also be used in presentations to demonstrate risks to a large audience of an organisation.

5.1.5 How

The format of the visualization is an innovative, quantitative fault tree. The semiotics may deviate from those described in section 2.1, as long as the design can be used to aid the explanation of the quantitative features of fault trees.

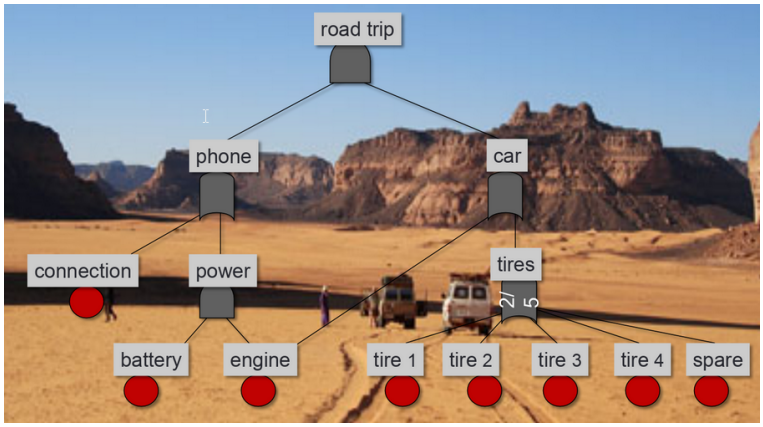

Figure 13: The design to be improved

5.2 User requirements

To categorize user requirements, both the *MoSCoW* [26] and *Kano* [27] method are used. The MoSCoW method stands for *Must*, *Should*, *Could* and *Would*. These categories enable a developer to prioritize requirements. Must has the highest priority while Would has the lowest priority. It is essential for a project to fulfill at least all the Must requirements to meet the minimal accepted needs. The listed requirement categories are sorted on priority.

The Kano method describes requirements in three categories: *must be (M)*, *performance (P)* and *attractive (A)*. A must-be is an essential feature, without this feature, the product is incomplete, for example water in a hotel room. A performance feature describes a feature where the more you have of it, the higher the satisfaction, e.g. internet speed in a hotel room. The final class, attractiveness, are features that are unexpected and cause a positive reaction. When these two methods are combined, it is more easy to recognize and prioritize important features. In the following table, features are listed based on the MoSCoW (rows) and Kano (columns).

Kano/ MoSCoW	Must be	Performance	Attractive
Must	The product must be displayed on a web browser	The product must contain more insight than a standard quantitative fault tree	The product must be made attractive by following all the visualization rules as mentioned in section 6.1
Should		The user should be able to interact with the product	
Could		The user could be able to view the product on a mobile device	The product could have animations
Would			The user would like to be able to create their own fault trees in the tool

1. The product must be displayed on a web browser
   This enables for easy presentation of the product and enables interaction. This is useful when the product is being used for educational purposes. The teacher can demonstrate it and students can experiment with it afterwards.

2. The product must be made attractive by following all the visualization rules as mentioned in section 4.1
   This is a must since this increases user engagement and an intuitive design can be more accessible for non-experts.

3. The product must contain more insight than a standard quantitative fault tree
   Required because this enables the target group to grasp the concepts of quantitative fault trees faster.

4. The user should be able to interact with the product
   This enhances both the user involvement and the demonstration of the fault tree mechanics.

5. The user could be able to view the product on a mobile device

This is useful when one want to explain the quantitative fault tree on a mobile device.

6. The product could have animations

   This can enhance the user engagement.

7. The user would like to be able to create own fault trees in the tool

   This feature is useful if users want to evaluate their own quantitative fault trees.

5.3 Generated ideas

A few ideas were generated in the brainstorm sessions that followed after listing the iteration of the Risk Visualization Framework and user requirements.

One idea was to make the fault tree circular, comparable with the structure of an onion. The top event would be in the centre and the events of its subtrees on the rings. This circular design is very beneficial when the fault tree is very extensive. If a fault tree is circular, it is not possible to read everything at the same time, as some part of the tree will be upside down. Therefore, a web application would allow this 'onion' to rotate like a wheel of fortune. If a person wants to study a certain part of the fault tree, he simply rotates it until that part is on the bottom. A simple sketch is shown in figure 14. Even though this design can be very helpful when trying to get an overview over large fault trees, it has not been chosen to realize as a product. The reason for this being that its potential for visualizing quantitative FT information is not great and the fault tree to be visualized is not extensive. There is additionally, no extra visual variable added in this design.

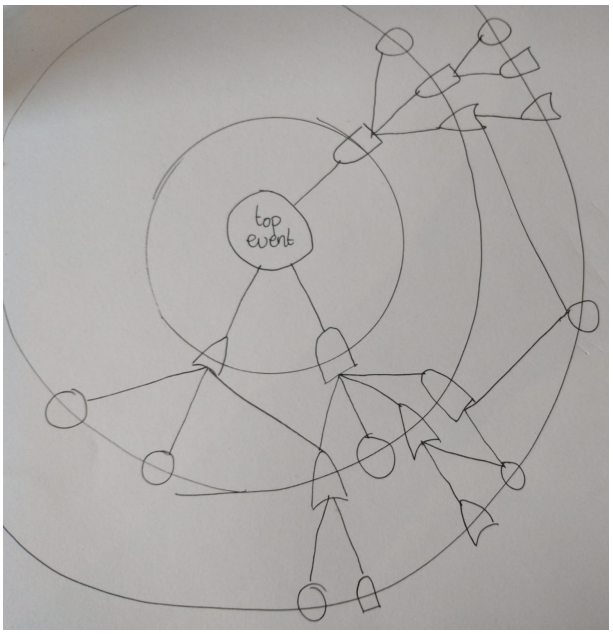

Figure 14: The sketch of the 'onion' fault tree

5.3.1 FT Visualization with Sankey Diagrams

Taking the user-requirements, the Risk Visualization Framework and the findings in the State of the Art section into consideration, the idea of integrating a Sankey diagram in a fault tree was chosen:

The Sankey diagram was introduced in 1898 in energy engineering and is used to visualize the flow of quantitative information [28]. An example of a Sankey diagram in energy context is shown in figure 15. Next to this domain, they have also been used to visualize monetary flow or the relation between college majors and the job one ends up with [29]. One of the most famous visualizations is Charles Minard's visualization of Napoleon's Russian campaign of 1812. However, this visualization is not a Sankey diagram, but a flow diagram since it is an overlay on a map. But it is relevant for this idea as it demonstrates excellently how the width of each edge can be used to visualize quantitative information and generate insight. The probabilistic properties of fault trees can be displayed as a Sankey diagram in a fault tree.

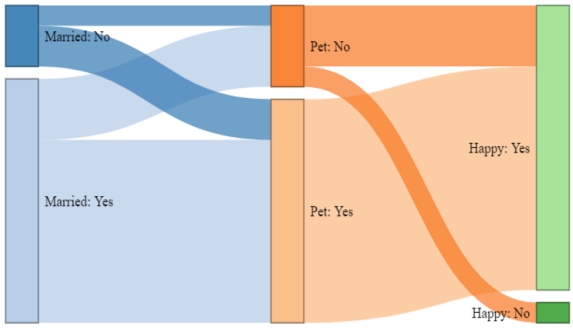

Figure 15: A Sankey diagram

Figure 16 shows a basic event of a FT in the hybrid form with a probability of failure of 0.8. This design decision is made so that the user can more quickly process the probabilities of a FT and is supported by the recommendations of Likpus [18]. The visual variable used here is size, the greater the probability of failure, the larger the width of the edge between nodes. However, size can be a tricky variable to use as its quantitative properties can fade if used incorrectly [20], see figure 5. Despite this, the design in figure 16 still holds its quantitative properties for the most part. It is difficult to recognize the difference between a probability of 0.8 and 0.82 in the output, but

these small differences are not relevant for this project. To aid the perception of these probabilities, text is added in the BE with the value of the probability of failure.

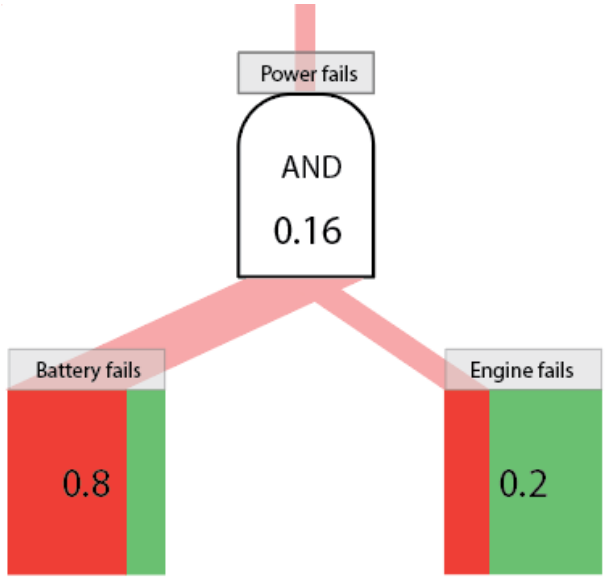

Figure 16: AND gate with Sankey diagram integration

As the main purpose this Sankey diagram is to show the relationship between failures and non-failures, the hue of the Sankey diagram is red for failure and green for non-failure. This corresponds to our findings on the use of color in section 4.1 in the State of the Art. A low saturation red and green color are chosen in the Sankey diagram. The reason for this low saturation being that the areas of it are relatively large. A high saturation color would be too distractive, while a low saturation color would not [10]. For this reason the edges were given a color with an even lower saturation. The edges between the nodes are straight, as Ward's design rules for graphs [20], section 4.1.2, state that the number of bends and the total length of the edges must be minimized. In addition to this, the tree should be symmetrical. Therefore the top-event must be placed in the middle of the window.

Figure 16 is the building block of the new fault tree design. With this design, the whole fault tree looks as follows:

With this design, it is also possible to fulfill the interaction requirement. When sliders are placed on the bottom of each basic event, the user can set their own probability of failure for each

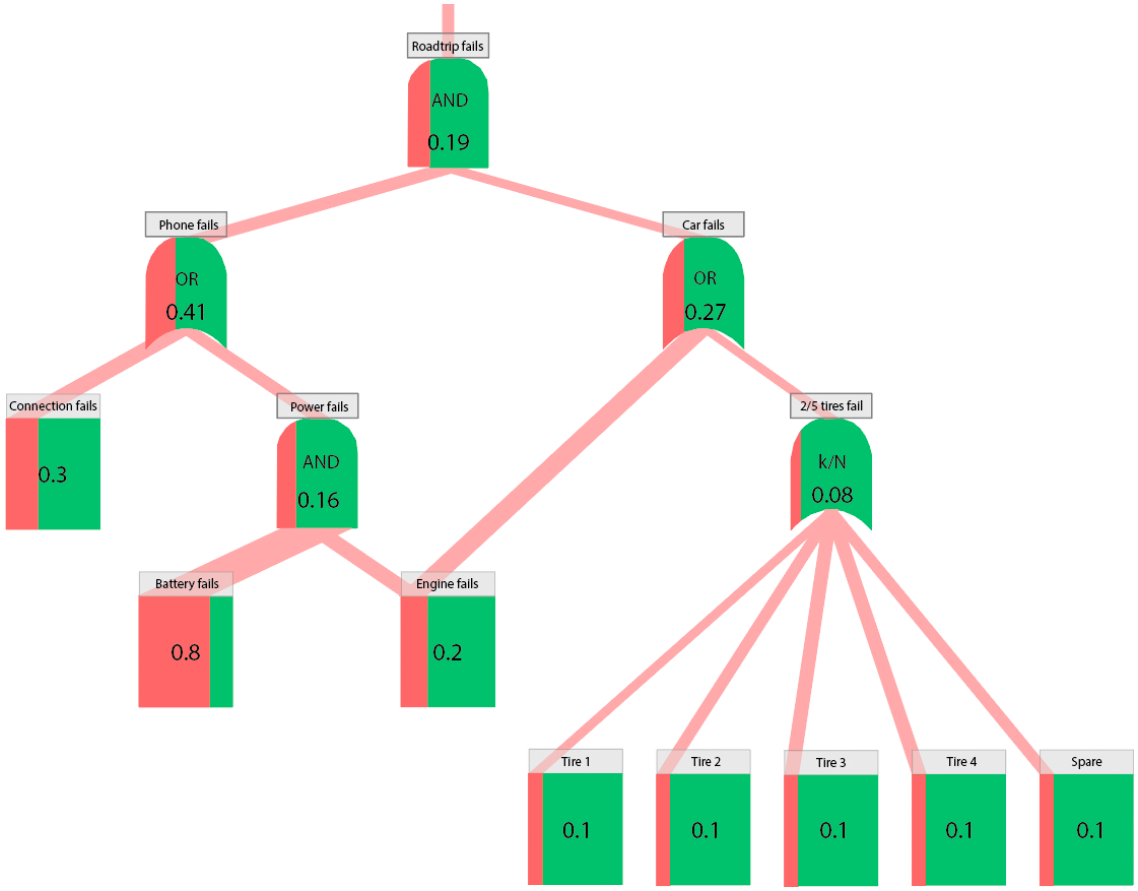

Figure 17: The hybrid fault tree concept

BE. Subsequently, the probability of each intermediate event can automatically be calculated using probability theory.

6 Realization

To realize the design as proposed in the previous chapter, a programming language must be chosen. As specified in section 5.2, the product must be viewed in a web browser. Therefore JavaScript is chosen as the main programming language. JavaScript is very well suited for web based interaction and web applications. Another advantage of JavaScript is the wide availability of libraries and frameworks. In this project we make use of the library p5.js [30]. This library is made for creative coders. It shares many features with Processing, but p5.js can be fully integrated in a web page. The complete code of the project is to be found in Appendix 1.

6.1 Classes

The fault tree consists of a collection of both *nodes* and *edges*, where edges show the relationships between nodes. In this program, a superclass Node is used to facilitate the inheritance of the constructor to the subclasses. The four subclasses are AND, OR, Basic Event (BE) or k/N. The following variables will be inherited from to superclass to each subclass, see figure 18 for the UML diagram:

1. x (number): the x position of the node on the screen.
2. y (number): the y position of the node on the screen.
3. children (array): the id of the children of the node, this is an array as a node can have multiple children.
4. parents (array): the id of the parents of the node, this is an array.
5. id (number): the id of the node. This id will be used in the parents and children to describe the relations that nodes have with each other.
6. columnPos (string): the horizontal position of the node in the grid system. It is necessary to have this extra variable to enable for a responsive.
7. rowPos (string): the vertical position of the node in the grid system.
8. output (number): the probability of failure of the node.

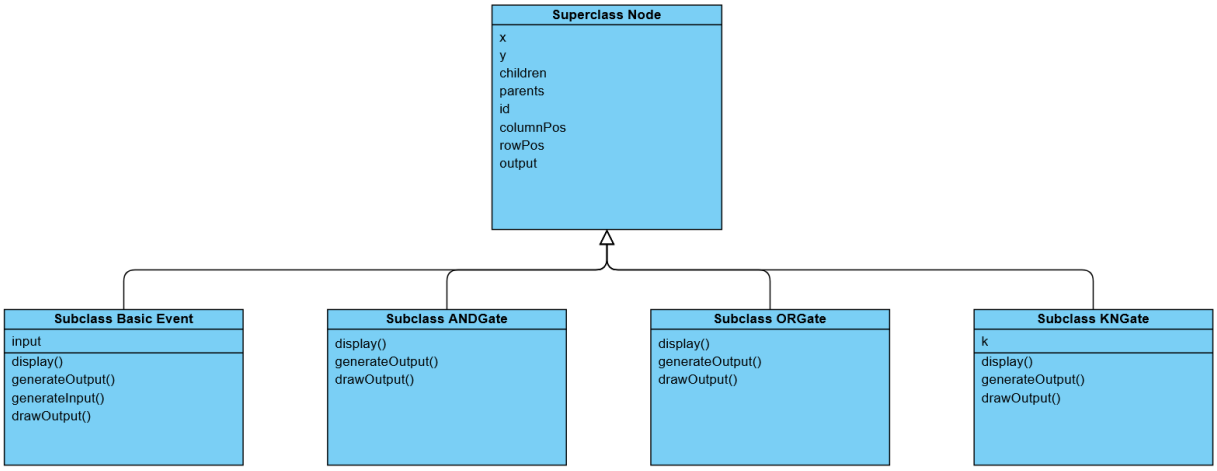

Figure 18: The UML class diagram

6.1.1 Basic Event

The Basic Event class has **4** methods: *display()*, *generateInput()*, *generateOutput()* and *drawOutput()*. The display function displays the node itself. The width of the Sankey diagram is defined by the the position of a slider positioned below the gate. The user can set a unique set of probability of failures for each BE. For simplicity reasons, it is assumed that each tire has the same probability of failure. The function generateInput maps the output of the corresponding slider to a number between 0 and 80, as the max with of the BE is 80 pixels and the slider ranges from 0 to 100. GenerateOutput calculates the probability of failure from the value of the slider so that this number will subsequently be used as an input of its parents. Finally, drawOutput draws the edge(s) between the BE and its parents.

6.1.2 AND Gate

3 methods can be associated with the AND Gate class. Again a display function which displays the object and its text. A drawOutput function, this draws the edges between the node and its parents. The calculateOutput method calculates the probability that all the children of the AND gate fail. This is done with the following equation:

$$AND_{\text{output}} = \prod_{n=1}^{AND_{\text{children.length}}} p_n \tag{1}$$

```
1
2 this.output = 1;
3 for (let i = 0; i < this.children.length; i++) {
4     this.output = this.output * nodes[this.children[i]].output }
```

Listing 1: The probability of failure calculation for the AND gate in JavaScript

6.1.3 OR Gate

The OR class has the same 3 methods as the AND class mentioned above. The only difference being that the output equation is different, namely:

$$OR_{\text{output}} = 1 - \prod_{n=1}^{OR_{\text{children.length}}} (1 - p_n) \tag{2}$$

```
1 let sum = 1;
2 for (let i = 0; i < this.children.length; i++) {
3     sum = sum * (1 - nodes[this.children[i]].output) }
4 this.output = 1 - sum;
```

Listing 2: The probability of failure calculation for the OR gate in JavaScript

6.1.4 k/N Gate

The output of the k/N node is calculated with the following equation. In this fault tree, it is assumed that the probability of failure of each tire (pt) is equal.

$$k/N_{\text{output}} = 1 + (\sum_{n=0}^{k} - \binom{k/N_{\text{children.length}}}{k/N_{\text{children.length - n}}} p_t^n (1 - p_t)^{\text{k/Nchildren.length-n}}) \tag{3}$$

```
1 let sum = 0;
2     for (let i = 0; i < this.k; i++) {
3     sum = sum - combinations(this.children.length, this.children.length - i) * Math.
      pow(nodes[5].output, i) * Math.pow((1 - nodes[5].output), this.children.length -
       i);  }
4 this.output = 1 + sum;
```

Listing 3: The probability of failure calculation for the KN gate in JavaScript

6.2 Dependency

Equations 1 to 3 hold when subtrees are independent of each other. This is not the case for the Roadtrip fails AND gate, the basic event Engine fails occurs in its subtrees. To calculate the output of the topevent, the law of total probability can be used, see equation 4.

$$P[Roadtrip] = P[Roadtrip|Engine]P[Engine] + P[Roadtrip|EngineNotFails]P[EngineNotFails] \tag{4}$$

$$P[Roadtrip|Engine] = P[Phone|Engine]P[Car|Engine] \tag{5}$$

$$P[Roadtrip|Engine] = P[Phone|EngineNotFails]P[Car|EngineNotFails] \tag{6}$$

First, in the case that the engine fails equation 5. The probability of the car failing is equal to 1, as the OR gate only needs 1 positive input to generate an output. P[Phone | Engine] is defined by the following expression:

$$P[Phone|Engine] = P[Connection] + P[Battery] - P[Connection]P[Battery] \tag{7}$$

In the case that the engine does not fail, the probability of P[Phone] is equal to P[Connection], as the Power fails gate can never fail if the engine does not fail. The probability of P[Car | EngineNotFails] is equal to P[Tires]. This results in formula 8 when these findings are substituted in equation 4.

$$P[Roadtrip] = (P_{\text{Con}} + P_{\text{Bat}}P_{\text{Eng}} - P_{\text{Con}}P_{\text{Bat}}P_{\text{Eng}})P_{\text{Eng}} + 1 - (1 - P_{\text{Eng}})(1 - P_T)^5 - 5P_t(1 - P_t)^4(1 - P_{\text{Eng}}) \tag{8}$$

6.3 Grid system

In order to guarantee that the fault tree will be displayed correctly on every resolution, the design must be made *responsive*. Next to this, the fault tree should still be centered when the browser window is being resized by the user. To achieve this, a grid system will be created. In Appendix 2 is a visualization of the grid system depicted. This grid system is calculated using the function *windowResized()* (see Appendix 1) and is called for each time the grid is resized by the user and in the setup function. A node can be displayed in each grid cell, this is defined by the *columnPos* and *rowPos* variable in the constructor of the node.

6.4 Final product

The final product is shown in figure 19. A low saturation, road trip themed, background image is chosen to add a layer of depth. Finally, an image of a stranded roadtrip is added at the top event to emphasize that it is the undesirable event of the tree. The sliders can be used to change the probability of the BEs. Subsequently, the Sankey diagrams in the gates will also change width based on equation 1 to 3 and 8.

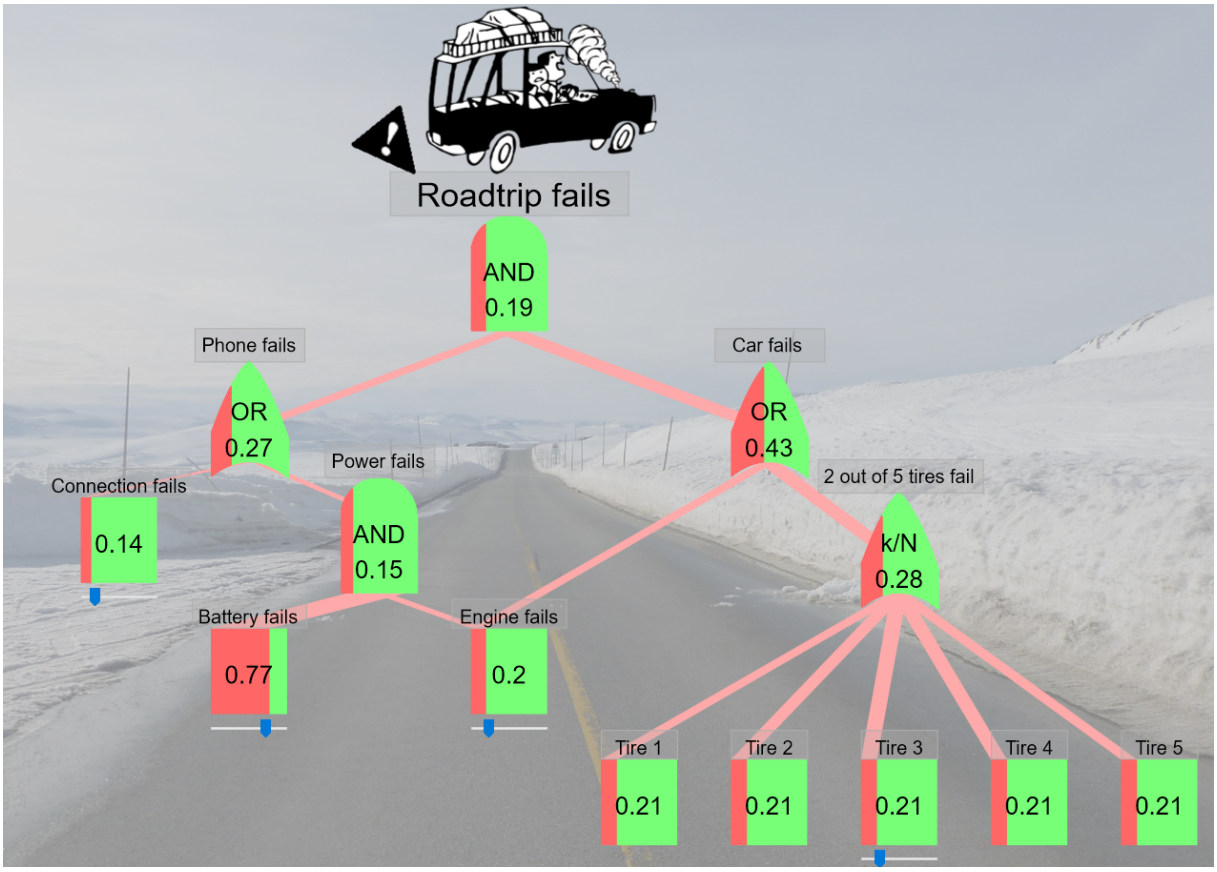

Figure 19: The final design of the FT tool

7 Evaluation

7.1 Interviews

Throughout the whole design phase, experts were consulted to enable a co-design process, this mode of designing has many benefits [31]. The first expert was consulted when the earliest version of a working prototype was realized and the last one when the product was almost final. In total 4 experts from research facilities, infrastructure and engineering companies have been consulted to participate in the design process. This has been done in the form of a semi-structured interview. The protocol of these interviews is as follows:

1. Introduction, information about the research, signing of consent form.
2. The participant introduces him/herself. Elaborates on the role fault trees play in his/her career.
3. Scenario 1: How would the participant explain the workings of FTs to a non-expert without any technical background? For example to a family member.
4. Scenario 2: How would the participant explain the workings of FTs to a non-expert who has some technical background? For example a computer science or physics student.
5. Are there any aids, tools or analogies the participant uses?
6. Show the prototype of the design

The most important results of the interviews are listed in Appendix 3. The initial reaction of the experts was very positive. At the first glance most of them were slightly confused, but when they discovered the sliders they understood it immediately. All experts mentioned that they use an everyday example whenever they need to explain fault tree mechanics to non-experts. This supports the choice for the roadtrip fault tree. One expert mentioned that the basic events should be circular, as this is also the case in the standard fault tree visualization. However, if the basic events are circular in combination with a Sankey diagram, it will become difficult to estimate the probability of the BE. Because the area of the BE will be distorted, see figure 20. A doubling of the probability in the circular BE has the illusion of being a larger increase than it actually is, therefore the BEs are still rectangles. Another remark experts made was the position of the name of the

BEs. First it was positioned under the basic event, but it was moved to the top because this is more akin to the standard design. The remainder of the remarks of the design was more targeted at new features. Expert mentioned adding a slider where one can change the time period for the probabilities. This can be useful when one wants to see the effect of a probability of failure over one day versus one week. Other experts suggested adding more gates and expanding this to a tool where one can drag and drop elements to create an own fault tree.

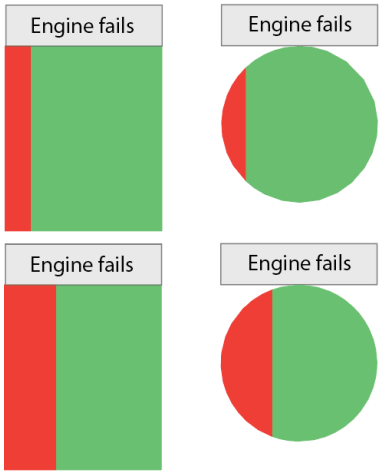

Figure 20: Top: Sankey diagram BEs with a probability of failure of 0.1. Bottom: Sankey diagram BEs with a probability of failure of 0.2.

7.2 Design evaluation

In this section, the design of the new design will be evaluated in the same way as the three other designs in section 4.2. The use of color is fully according to the design rules. The larger areas have a low saturation, while the smaller areas have a higher saturation. In addition to this is the failure color red and the non-failure color green. Adaptability has a lower score due to the fact that it is not yet possible to create an own fault tree in the tool. However, the roadtrip fault tree is adaptable in a sense that it is responsive to the screen size of the browser window. Visual cues are used in a correct manner, the sliders are located directly below the basic events, this corresponds with the proximity Gestalt law (section 4.1.3). Finally, the new design does not violate other design rules and is in accordance with Ward's guidelines for the visualization of trees, graphs and networks (section 4.1.2).

Tool	Color and contrast	Adaptability	Use of visual cues	Compliance with design rules
Faulttree+	o	+	+	+
DFT Visualization	–	-	-	o
Visual Paradigm Online	++	++	o	+
New design	++	o	++	++

Table 2: The scores of the three evaluated fault tree tools and the design as proposed in this thesis

7.3 User Requirements

In section 5.2, user requirements were formulated with the MoSCoW and Kano method. When considering the result, it can be concluded that the most important (Must and Should) requirements have been met. The user requirement table of the Ideation section is repeated below:

Kano/ MoSCoW	Must be	Performance	Attractive
Must	The product must be displayed on a web browser	The product must contain more insight than a standard quantitative fault tree	The product must be made attractive by following all the visualization rules as mentioned in section 6.1
Should		The user should be able to interact with the product	
Could		The user could be able to view the product on a mobile device	The product could have animations
Would			The user would like to be able to create their own fault trees in the tool

The explanation of why the user requirements are met is listen below:

1. The product must be displayed on a web browser.

   The product has been realized in the form of a web application. Therefore it can be viewed with any computer. The tree is responsive, meaning that it adapts to the resolution of the screen of the user.

2. The product must be made attractive by following all the visualization rules as mentioned in section 4.1.

   After evaluation the product in the same manner as the other fault tree designs (section 7.2), it can be concluded that the design is in accordance with the information visualization guidelines.

3. The product must contain more insight than a standard visualization of a quantitative fault tree.

   The Sankey diagram integration adds a new dimension to the fault tree. This hybrid fault tree is more insightful, as the results of changes in the failure probability of BEs are immediately seen in the rest of the tree. The Sankey diagram in the gates is also faster to interpret than regular numbers.

4. The user should be able to interact with the product.

   This requirement is met by adding sliders which the user can use to choose a unique set of probabilities.

The remaining requirements have not been met, but they are of a low priority, as they fall in the Should and Would category in the MoSCoW method. These requirements are:

1. The user could be able to view the product on a mobile device.

   The p5.js library does not support mobile devices well. But it is possible to elaborate the sourcecode to enable mobile support. However this can be a very time consuming process

2. The product could have animations.

   This user requirement has not been met due to its low priority and difficult process.

3. The user would like to be able to create their own fault trees in the tool

   This feature has not been implemented due to the difficult mathematics involved when there is dependency. But if an algorithm for the probabilistic calculations of shared subtrees is available, implementation in this tool is not difficult.

8 Conclusion

This thesis focused on creating an innovative fault tree design to aid the understanding of its quantitative mechanisms. To achieve this, first a literature review on visualization techniques and risk perception was conducted. In the review it became clear that current fault tree designs do not fully comply with the visualization guidelines and that there are still many opportunities for innovative visualization in risk management. After the review, brainstorm sessions were conducted to explore possible designs that are more attractive and insightful than current FT designs. Subsequently, Sankey diagrams were used instead of regular logic gates and basic events. The addition of this area visual variable resulted in a more insightful fault tree, as it is possible to perceive in a glance what effects a change in a single BE has to the whole fault tree. In the phase between the initial prototype and the final product, experts offered useful suggestions for the design. Lastly, the final product was evaluated similarly to existing designs. It can be concluded that the most important user requirements have been met for this thesis.

The main research question *How can innovative visualization methods make quantitative fault tree analysis more understandable?* can now be answered:
Even widely adopted visualizations, such as fault trees do not comply with all the design rules of the State-of-the-Art section. They often have asymmetry, the edges between the nodes have strange bends, there is confusing use of color or lack of color. By simply following these rules, a more attractive fault tree can be made. When an extra visual variable is added, area in our case, the visualization can become more insightful. Interaction can also help in making a visualization more attractive, this enables for more engagement.

8.1 Future work

A next phase in the development of the web-application would be to convert it to a tool that people can use to make their own fault trees in this design, or to add support for more fault tree elements. For this thesis this was out of scope because of the complicated mathematics.

A user study could be relevant to conduct. In the introduction section, the difference in insight between a textual description of a fault tree and a FT visualization has been demonstrated. A next step could be adding the fault tree design of this project to the comparison. This way, the exact results on insight and effectiveness of the different designs can be determined.

In addition to this, interactive visualization still has a gigantic number of possible applications, not only in risk management, but also in other fields. Therefore, further research in this field is important.

References

[1] W. E. Vesely, F. F. Goldberg, N. H. Roberts, and D. F. Haasl, "Fault tree handbook", Nuclear Regulatory Commission Washington DC, Tech. Rep., 1981.

[2] P. Badida, Y. Balasubramaniam, and J. Jayaprakash, "Risk evaluation of oil and natural gas pipelines due to natural hazards using fuzzy fault tree analysis", *Journal of Natural Gas Science and Engineering*, vol. 66, pp. 284–292, 2019.

[3] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools", *Computer Science Review*, vol. 15-16, pp. 29–62, 2015.

[4] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick, and J. Railsback, "Fault tree handbook with aerospace applications", 2002.

[5] D. Vaughan, "Autonomy, interdependence, and social control: Nasa and the space shuttle challenger", *Administrative Science Quarterly*, vol. 35, no. 2, 1990.

[6] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie, "Fault tree analysis, methods, and applications a review", *IEEE Transactions on Reliability*, vol. R-34, no. 3, pp. 194–203, Aug. 1985.

[7] M. Stoelinga and E. Ruijters, "Fault tree analysis, an introduction",

[8] J. H. Purba, J. Lu, G. Zhang, and W. Pedrycz, "A fuzzy reliability assessment of basic events of fault trees through qualitative data processing", *Fuzzy Sets and Systems*, vol. 243, pp. 50–69, 2014.

[9] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision To Think*. Jan. 1999.

[10] C. Ware, *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[11] M. J. Eppler and M. Aeschimann, "A systematic framework for risk visualization in risk management and communication", *Risk Management*, vol. 11, no. 2, pp. 67–89, 2009.

[12] F. Roth, "Visualizing risk: The use of graphical elements in risk analysis and communications", *CSS Risk and Resilience Reports*, Jul. 2012.

[13] P. C. Stern and H. V. Fineberg, *Understanding Risk: Informing Decisions in a Democratic Society*. 1996, ISBN: 978-0-309-08956-2.

[14] E. Björgvinsson, P. Ehn, and P.-A. Hillgren, "Design things and design thinking: Contemporary participatory design challenges", *Design Issues*, vol. 28, no. 3, pp. 101–116, 2012.

[15] *Definition of 'ting'*. [Online]. Available: https://en.wiktionary.org/wiki/%C3%BEing#Old_Norse.

[16] P. Slovic, "Perception of risk", *Science*, vol. 236, no. 4799, pp. 280–285, 1987.

[17] B. L. Lindner, J. Johnson, F. Alsheimer, S. Duke, G. D. Miller, and R. Evsich, "Increasing risk perception and understanding of hurricane storm tides using an interactive, web-based visualization approach", *Journal of Coastal Research*, vol. 34, no. 6, pp. 1484–1498, 2018.

[18] I. M. Lipkus, "Numeric, verbal, and visual formats of conveying health risks: Suggested best practices and future recommendations", *Medical Decision Making*, vol. 27, no. 5, pp. 696–713, 2007.

[19] C. Xiong, L. van Weelden, and S. Franconeri, "The curse of knowledge in visual data communication", *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019.

[20] M. Ward, G. Grinstein, and D. Keim, *Interactive Data Visualization: Foundations, Techniques, and Applications*. 2010.

[21] *Piechart*. [Online]. Available: https://www.businessinsider.com.au/pie-charts-are-the-worst-2013-6.

[22] T. Clarke and A. Costall, "The emotional connotations of color: A qualitative investigation", *Color Research & Application*, vol. 33, no. 5, pp. 406–410, 2008.

[23] *Isograph*. [Online]. Available: https://www.isograph.com/software/reliability-workbench/fault-tree-analysis-software/.

[24] *Dft*. [Online]. Available: https://github.com/moves-rwth/dft-gui.

[25] *Visual parardigm online*. [Online]. Available: https://online.visual-paradigm.com/diagrams/features/fault-tree-analysis-software/.

[26] *Moscow*. [Online]. Available: https://www.projectsmart.co.uk/moscow-method.php.

[27] *Kano*. [Online]. Available: https://foldingburritos.com/kano-model.

[28] M. Schmidt, "The sankey diagram in energy and material flow management: Part i: History", *Journal of industrial ecology*, vol. 12, no. 1, 2008.

[29] *Major - job diagram*. [Online]. Available: `http://benschmidt.org/jobs/`.

[30] *P5.js*. [Online]. Available: `https://p5js.org/`.

[31] M. Steen, M. Manschot, and N. De Koning, "Benefits of co-design in service design projects", *International Journal of Design*, vol. 5, no. 2, 2011.

9 Appendix 1: Javascript code

```
const greenColor = '#77ff77';
const failColor = '#ff6666';
const lightRed = '#ffaaaa';
const columns = 11;
const rows = 5;
let fontRoboto;
let nodes = [];
var column = [];
var row = [];
var slider = [];
let infoText = 'This tool is an interactive fault tree.\nThe width of the edges are in proportion with the probability of the gate or event.\nUse the sliders of the basic events to set a probability and notice the effects in the rest of the fault tree.\nFor education purposes only ';
var horizontalSpacing, verticalSpacing, margin, horizontalCentering, verticalCentering;
let img, backgroundImg;

function preload() {
    img = loadImage('roadtrip3.png');
    backgroundImg = loadImage('mountain.png')
    fontRoboto = loadFont('Roboto-Regular.ttf');
}

function generateRoadtrip() {
    nodes[0] = new BasicEvent(column[2] + horizontalCentering, row[2] + verticalCentering, [], [11], 0, 'column[2]', 'row[2]', 0, 0, 'Connection fails');
    nodes[1] = new BasicEvent(column[3] + horizontalCentering, row[3] + verticalCentering, [], [10], 1, 'column[3]', 'row[3]', 0, 0, 'Battery fails');
    nodes[2] = new BasicEvent(column[5] + horizontalCentering, row[3] + verticalCentering, [], [10, 9], 2, 'column[5]', 'row[3]', 0, 0, 'Engine fails');

    nodes[3] = new BasicEvent(column[6] + horizontalCentering, row[3] + verticalCentering, [], [8], 3, 'column[6]', 'row[4]', 0, 0, 'Tire 1');
    nodes[4] = new BasicEvent(column[7] + horizontalCentering, row[3] + verticalCentering, [], [8], 4, 'column[7]', 'row[4]', 0, 0, 'Tire 2');
```

```
    nodes[5] = new BasicEvent(column[8] + horizontalCentering, row[3] +
    verticalCentering, [], [8], 5, 'column[8]', 'row[4]', 0, 0, 'Tire 3');
    nodes[6] = new BasicEvent(column[9] + horizontalCentering, row[3] +
    verticalCentering, [], [8], 6, 'column[9]', 'row[4]', 0, 0, 'Tire 4');
    nodes[7] = new BasicEvent(column[10] + horizontalCentering, row[3] +
    verticalCentering, [], [8], 7, 'column[10]', 'row[4]', 0, 0, 'Tire 5');



    nodes[8] = new KNgate(column[8] + horizontalCentering, row[2] +
    verticalCentering, [3, 4, 5, 6, 7], [9], 8, 'column[8]', 'row[2]', 0, '2 out of
    5 tires fail', 2);
    nodes[9] = new ORgate(column[7] + horizontalCentering, row[1] +
    verticalCentering, [2, 8], [12], 9, 'column[7]', 'row[1]', 0, 'Car fails');
    nodes[10] = new ANDgate(column[4] + horizontalCentering, row[2] +
    verticalCentering, [1, 2], [11], 10, 'column[4]', 'row[2]', 0, 'Power fails');
    nodes[11] = new ORgate(column[3] + horizontalCentering, row[1] +
    verticalCentering, [0, 10], [12], 11, 'column[3]', 'row[1]', 0, 'Phone fails');
    nodes[12] = new ANDgate(column[5] + horizontalCentering, row[0] +
    verticalCentering, [11, 9], [], 12, 'column[5]', 'row[0]', 0, 'Roadtrip fails');

    for (let i = 0; i < nodes.length; i++) {
        if ((nodes[i].constructor.name == 'BasicEvent' && nodes[nodes[i].parents
    [0]].constructor.name != 'KNgate') || nodes[i].constructor.name == 'KNgate') {
            slider[i] = createSlider(0, 100, 50);
            slider[i].position(nodes[i].x + 5, nodes[i].y + 100);
            slider[i].style('width', '80px');
        }
    }
}

function windowResized() {
    resizeCanvas(windowWidth, windowHeight);
    margin = (windowWidth - 1500) / 2;
    if (margin < 25) {
        margin = 25;
    }
    horizontalSpacing = (windowWidth - margin * 2) / columns;
    if (horizontalSpacing < 100) {
```

```
        horizontalSpacing = 100;
    } else if (horizontalSpacing > 300) {
        horizontalSpacing = 300;
    }
    for (let i = 0; i < columns; i++) {
        column[i] = margin + horizontalSpacing * i;
    }
    verticalSpacing = (windowHeight - 250) / rows;
    if (verticalSpacing < 125) {
        verticalSpacing = 125;
    } else if (verticalSpacing > 300) {
        verticalSpacing = 300;
    }
    for (let i = 0; i < rows; i++) {
        row[i] = 225 + verticalSpacing * i;
    }
    for (let i = 0; i < nodes.length; i++) {
        nodes[i].x = eval(nodes[i].columnPos) + horizontalCentering - 23;
        nodes[i].y = eval(nodes[i].rowPos) + verticalCentering;
    }
    for (let i = 0; i < slider.length; i++) {
        if (nodes[i].constructor.name == 'BasicEvent' || nodes[i].constructor.name
    == 'KNgate') {
            slider[i].position(nodes[i].x + 5, nodes[i].y + 100);
        }
    }
    rect(column[0] - 125, row[0], 200, 500)

    horizontalCentering = (horizontalSpacing / 2 - 40);
    verticalCentering = (verticalSpacing / 2 - 50);
}

function nameText(name, x, y) {
    textSize(20);
    stroke(175);
    fill(0);
    text(name, x + (80 - textWidth(name)) / 2, y - 7);
}

```

```
function drawRect(name, x, y) {
    fill(150, 150, 150, 100);
    stroke(150);
    strokeWeight(1);
    rect(x + (80 - (textWidth(name) + 29)) / 2, y - 30, textWidth(name) + 29, 30);
}

function gateText(output, x, y) {
    fill(0);
    noStroke();
    if (typeof output != 'string') {
        output = output.toFixed(2);
    }
    let str = output.toString();

    if (str.charAt(str.length - 1) == '0') {
        str = str.slice(0, 3);
        if (str.charAt(str.length - 1) == '0') {
            str = str.slice(0, 1);
        }
    }
    textSize(26);
    text(str, 40 - (textWidth(str) / 2) + x, y + 65);
}


function setup() {
    var canvas = createCanvas(windowWidth, windowHeight);
    canvas.parent('sketch-div');
    textFont(fontRoboto);
    windowResized();
    generateRoadtrip();
}

function draw() {
    image(backgroundImg, 0, 0, windowWidth, windowHeight)
    for (let i = 0; i < slider.length; i++) {
        if (nodes[i].constructor.name == 'BasicEvent' && nodes[nodes[i].parents[0]].
    constructor.name != 'KNgate') {
```

```
            slider[i].position(nodes[i].x + 5, nodes[i].y + 100);
        }
        if (nodes[i].constructor.name == 'KNgate') {
            slider[i].position(nodes[5].x + 5, nodes[5].y + 100);
        }
    }
    fill(0);
    textSize(16);
    text(infoText, column[0] + 10, row[0] - 190, 400, 200)

    fill(100, 100, 100, 125);
    stroke(175);
    strokeWeight(1);
    rect(column[0], row[0] - 200, 400, 200);
    for (let i = 0; i < nodes.length; i++) {
        nodes[i].x = eval(nodes[i].columnPos) + horizontalCentering - 23;
        nodes[i].y = eval(nodes[i].rowPos) + verticalCentering;
        nodes[i].display();
        nodes[i].generateOutput();
    }
}

class Node {
    constructor(x, y, children, parents, id, columnPos, rowPos, output, name) {
        this.x = x;
        this.y = y;
        this.children = children;
        this.parents = parents
        this.id = id;
        this.columnPos = columnPos;
        this.rowPos = rowPos;
        this.output = output;
        this.name = name;
    }
}

class BasicEvent extends Node {
    constructor(x, y, children, parents, id, columnPos, rowPos, output, input, name)
     {
```

```
        super(x, y, children, parents, id, columnPos, rowPos, output, name);
        this.input = input;
        this.x = this.x - 130;
    }
    display() {
        noStroke();
        this.generateInput();
        fill(failColor);
        rect(this.x, this.y, this.input, 90);
        fill(greenColor);
        rect(this.x + this.input, this.y, 80 - this.input, 90);
        gateText(this.output, this.x, this.y - 7);

        line(this.x, this.y, this.x, this.y + 90);
        this.drawOutput();
        nameText(this.name, this.x, this.y);
        drawRect(this.name, this.x, this.y);
    }
    generateOutput() {
        if (slider[this.id] != undefined) {
            this.output = (slider[this.id].value()) / 100;
        } else {
            this.output = (slider[8].value()) / 100;
        }
    }
    generateInput() {
        if (slider[this.id] != undefined) {
            this.input = slider[this.id].value() * 0.8;
        } else {
            this.input = slider[8].value() * 0.8;
        }
    }
    drawOutput() {
        smallStroke(this.output);
        fill(lightRed);
        for (let i = 0; i < this.parents.length; i++) {
            quad(this.x, this.y, this.x + this.input, this.y, nodes[this.parents[i
]].x + (this.input / 2) + 40, nodes[this.parents[i]].y + 100, nodes[this.parents
[i]].x - this.input / 2 + 40, nodes[this.parents[i]].y + 100);
```

```
        }
    }
}

function drawANDgate(x, y, output) {
    for (let i = 81; i > 0; i--) {
        strokeWeight(2);
        if ((1 - output) * 80 > 80 - i) {
            stroke(greenColor);
        } else {
            stroke(failColor);
        }
        if (i < 40) {
            line(x + i, y + 100, x + i, y + 20 - Math.sqrt(1600 - Math.pow(40 -
    float(i), 2)));
        } else {
            line(x + i, y + 100, x + i, y + 20 - Math.sqrt(1600 - Math.pow(float(i)
    - 40, 2)));
        }
    }
}

function smallStroke(output) {
    if (output < 0.05) {
        strokeWeight(1);
        stroke(lightRed);
    } else {
        noStroke();
    }
}

class ANDgate extends Node {
    constructor(x, y, children, parents, id, columnPos, rowPos, output, name) {
        super(x, y, children, parents, id, columnPos, rowPos, output, name);
    }
    display() {
        this.drawOutput()
        drawANDgate(this.x, this.y, this.output);
        gateText(this.output, this.x, this.y + 20);
```

```
        gateText('AND', this.x, this.y - 17);

        if (this.id != 12) {
            textSize(20);
            nameText(this.name, this.x, this.y - 22);
            drawRect(this.name, this.x, this.y - 22);
        } else {
            textSize(35);
            fill(0);
            text(this.name, this.x + (80 - textWidth(this.name)) / 2, this.y - 31);
            fill(150, 150, 150, 75);
            stroke(150);
            strokeWeight(1);
            rect(this.x - 85, this.y - 67, 250, 46);
            image(img, this.x - 125, this.y - 240)

        }
    }

    generateOutput() {
        if (this.id == 12) {
            this.output = (nodes[0].output + nodes[1].output - nodes[0].output *
    nodes[1].output) * nodes[2].output + nodes[0].output * nodes[8].output * (1 -
    nodes[2].output);
        } else {
            this.output = 1;
            for (let i = 0; i < this.children.length; i++) {
                this.output = this.output * nodes[this.children[i]].output
            }
        }
    }

    drawOutput() {
        smallStroke(this.output);
        fill(lightRed);
        for (let i = 0; i < this.parents.length; i++) {
            quad(this.x + 40 + (this.output * 80) / 2, this.y + 18, this.x + 40 - (
    this.output * 80) / 2, this.y + 18, nodes[this.parents[i]].x + 40 - (this.output
     * 80) / 2, nodes[this.parents[i]].y + 97, nodes[this.parents[i]].x + 40 + (this
```

```
    .output * 80) / 2, nodes[this.parents[i]].y + 97);
        }
    }
}

function drawORgate(x, y, output) {
    strokeWeight(2);
    for (let i = 81; i > 0; i--) {
        if ((1 - output) * 80 > 80 - i) {
            stroke(greenColor);
        } else {
            stroke(failColor);
        }
        if (i < 40) {
            line(x + i, y + 100 + 0.25 * pow(float(i - 40) / 5.0, 2), x + i, y - 15
    + 0.6 * pow(float(i - 60) / 5.0, 2));
        } else {
            line(x + i, y + 100 + 0.25 * pow(float(i - 40) / 5.0, 2), x + i, y - 15
    + 0.6 * pow(float(i - 20) / 5.0, 2));
        }
    }
}

class ORgate extends Node {
    constructor(x, y, children, parents, id, columnPos, rowPos, output, name) {
        super(x, y, children, parents, id, columnPos, rowPos, output, name);
    }
    display() {
        this.drawOutput()
        drawORgate(this.x, this.y, this.output);
        nameText(this.name, this.x, this.y - 7)
        drawRect(this.name, this.x, this.y - 7)
    }

    generateOutput() {
        let sum = 1;
        for (let i = 0; i < this.children.length; i++) {
            sum = sum * (1 - nodes[this.children[i]].output)
        }
```

```
316         this.output = 1 - sum;
317         noStroke();
318         gateText(this.output, this.x, this.y + 30);
319         gateText('OR', this.x, this.y - 8)
320     }
321     drawOutput() {
322         smallStroke(this.output);
323         fill(lightRed);
324         for (let i = 0; i < this.parents.length; i++) {
325             quad(this.x + 40 + (this.output * 80) / 2, this.y + 66, this.x + 40 - (
    this.output * 80) / 2, this.y + 66, nodes[this.parents[i]].x + 40 - (this.output
     * 80) / 2, nodes[this.parents[i]].y + 101, nodes[this.parents[i]].x + 40 + (
    this.output * 80) / 2, nodes[this.parents[i]].y + 101);
326         }
327     }
328 }
329
330 function product_Range(a, b) {
331     var prd = a,
332         i = a;
333
334     while (i++ < b) {
335         prd *= i;
336     }
337     return prd;
338 }
339
340 function combinations(n, r) {
341     if (n == r) {
342         return 1;
343     } else {
344         r = (r < n - r) ? n - r : r;
345         return product_Range(r + 1, n) / product_Range(1, n - r);
346     }
347 }
348
349 class KNgate extends Node {
350     constructor(x, y, children, parents, id, columnPos, rowPos, output, name, k) {
351         super(x, y, children, parents, id, columnPos, rowPos, output, name);
```

```
352         this.k = k;
353     }
354     display() {
355         this.calculateSliderInput();
356         this.drawOutput();
357         drawORgate(this.x, this.y, this.output)
358         gateText(this.output, this.x, this.y + 30);
359         gateText('k/N', this.x, this.y - 8);
360         nameText(this.name, this.x, this.y - 7)
361         drawRect(this.name, this.x, this.y - 7)
362     }
363 
364     calculateSliderInput() {
365         this.slidervalue = (slider[this.id].value()) / 100;
366     }
367 
368     generateOutput() {
369         let sum = 0;
370         for (let i = 0; i < this.k; i++) {
371             sum = sum - combinations(this.children.length, this.children.length - i)
     * Math.pow(nodes[5].output, i) * Math.pow((1 - nodes[5].output), this.children.
    length - i);
372         }
373         this.output = 1 + sum;
374     }
375 
376     drawOutput() {
377         smallStroke(this.output);
378         fill(lightRed);
379         for (let i = 0; i < this.parents.length; i++) {
380             quad(this.x + 40 + (this.output * 80) / 2, this.y + 66, this.x + 40 - (
    this.output * 80) / 2, this.y + 66, nodes[this.parents[i]].x + 40 - (this.output
     * 80) / 2, nodes[this.parents[i]].y + 101, nodes[this.parents[i]].x + 40 + (
    this.output * 80) / 2, nodes[this.parents[i]].y + 101);
381         }
382     }
383 }
```

Listing 4: The Javascript code of the web application

10 Appendix 2: The grid system

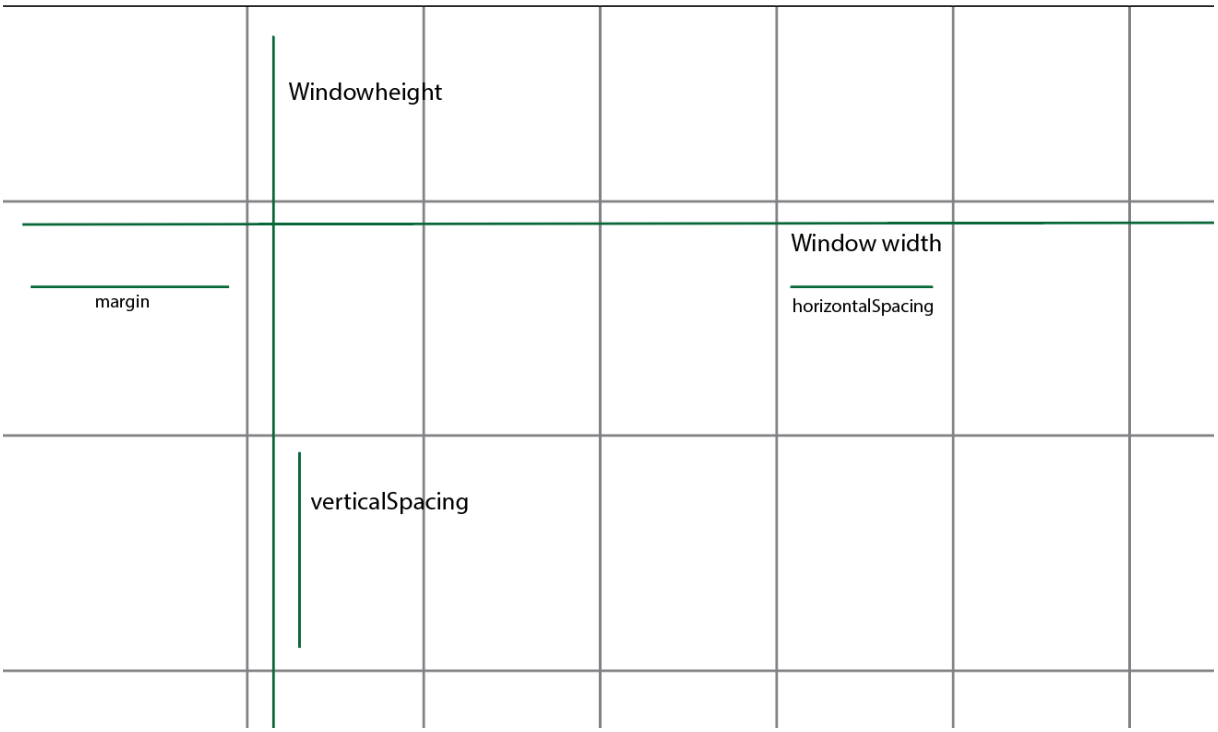

Figure 21: Visualization of the grid

11 Appendix 3: Summary of Expert interviews and Consent Form

In this appendix, the most important points of the expert interviews are given.

11.1 Interview 1

Job:

Researcher, statistical model checking

Relation to fault trees

Uses fault trees in his research

Explanation of fault trees to non-experts

Use a relatable example to explain FTs. Identify a top-event and break down how this could occur until the basic event level. With a pencil and paper it works better.

Feedback on prototype

"I have to say this is amazing, it conveys very clearly the idea" AND and OR shape is confusing, adding text would made it more clear. Time probabilities could also be interesting to add. Nice framework to add more gates too.

11.2 Interview 2:

Job:

Researcher, Civil Engineering background

Relation to fault trees

Improved of fault trees using machine learning and big data.

Explanation of fault trees to non-experts

Show a basic fault tree of an everyday object (e.g. a bike). Explain the workings of AND and OR gates. No mathematical formulations. Ask the person to point at the most risky event of the fault tree image. Little extra knowledge is necessity the fault tree.

Feedback on prototype

Very nice and interesting way to show the idea behind quantitative properties and propagation. Maybe also show change the time period of a basic event probability. Basic events should be circular according to the standardized design.

11.3 Interview 3:

Job:

Engineering company

Relation to fault trees

Risk management for railroads/bridges etc.

Explanation of fault trees to non-experts

Take a relatable example. Ask someone about the causes of a failure and relate this to a fault tree.

Feedback on prototype

Definition of roadtrip fails is a bit ambiguous. For the rest is the concept very clear.

11.4 Interview 4:

Job:

Infrastructure company, tunnel and fire safety

Relation to fault trees

Uses quantitative fault trees for RAMS in infrastructure.

Explanation of fault trees to non-experts

Use an infrastructure example, describe the importance of reliability. Decompose fault tree part by part.

Feedback on prototype

Very interesting, would be interesting to see if this could be used more widespread.

11.5 Consent Form

Informed consent form template for research with human participants

---

This research is meant to investigate the design of Fault tree diagrams. Participants in this research are asked to provide feedback on different designs of fault tree diagrams. This will help the researcher to make design decisions.

Audio will be recorded and deleted according to the GDPR. It is not necessary to provide personal information in this recording. Anonymous transcriptions of this recording may be used in the thesis.

If you have any questions, please contact Jan van den Berg j.l.vandenberg@student.utwente.nl
Or the project supervisor Prof. Dr. M.I.A. Stoelinga.

1. Consent Form for CreaTe

Please tick the appropriate boxes	Yes	No
Taking part in the study		
I have read and understood the study information dated [12/05/2020], or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
Use of the information in the study		
I understand that information I provide will be used for a thesis	○	○
I understand that personal information collected about me that can identify me, such as [e.g. my name or where I live], will not be shared beyond the study team.	○	○
I agree that my information can be quoted in research outputs	○	○
Consent to be Audio Recorded		
I agree to be audio recorded.	○	○
Future use and reuse of the information by others		
I give the researchers permission to keep my contact information and to contact me for future research projects.	○	○

Signatures

_____ _____ _____

Name of participant

Signature Date

I have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.