# Interactive Visualization of Fault Trees

Robert Maaskant
University of Twente
P.O. Box 217, 7500 AE, Enschede
The Netherlands
r.maaskant@student.utwente.nl

## ABSTRACT

In this paper we present a case study of using interactive visualization of fault trees and fault tree analysis to improve understanding of the concept of fault trees and the exploration of the results of their analysis. The fault tree model is widely used by experts to analyze risks in many different fields. We believe that the current way of interacting with these fault tree models and the results of their analysis is prohibitive to many non-technical users and unnecessarily complicated for technical users. By creating a more intuitive method of interaction we hope to spread their use and increase their effectiveness to lower risks in fields that practice risk analysis. To create this interactive visualization we first examined the fundamental visualization principles and basic interaction techniques. We then combined these principles with our knowledge of fault trees to create a set of requirements for the visualization. These requirements were then used to choose a web framework and design a prototype visualization to validate our hypothesis.

## Keywords

Interactive Visualization, Fault Trees, Risk Analysis

## 1. INTRODUCTION

Risk analysis is important in many commercial fields. The analysis is used to increase understanding of the risks, determine the most effective strategy to decrease overall risk, and prove compliance with legislation. One of the tools in the toolkit of risk analysis is the fault tree (FT) model. FTs enable the analysis of the occurrence of a specific undesired top level event in terms of basic system events of which the risk profile is known. The constructed fault tree can be analyzed in a variety of ways. The major distinction between the different types of analyses is that they are either qualitative or quantitative [13]. Qualitative analysis gives insight into the causes of failure, where quantitative analysis gives insight into the probability of failure.

Currently the results of the analyses are usually visualized using simple static graphics like tables, bar charts, and scatter plots. This form of presentation can already offer a lot of valuable insights into the risks modeled, but we believe the presentation can be improved. This improvement is important, because sight is "not only the fastest and most nuanced sensory portal to the world, it is also the one most intimately connected with cognition" [6]. Thus improved visualization could lead to better understanding. To enable knowledge discovery from the analyses the visualization is made interactive. Interactivity is important because it allows users to quickly adjust the visualization to show the information required to validate or discredit hypotheses they have.

To test the hypothesis that an interactive visualization of FTs and FTAs can help in the understanding of the concept of FTs and the exploration of the results of the FTA, we will build a prototype application. The prototype will be an interactive visualization built as a website. This is in contrast to many current tools which are mainly command line based or a graphical desktop application. If the software runs on a website it can be easily accessed by anyone with the right permissions at any time, at any location on any Internet connected device. This further helps lower the threshold to widespread use of the fault tree modeling method, which hopefully leads to lowered risks, which is the ultimate goal of fault trees to begin with.

This paper will start by explaining in some more detail all concepts involved, followed by a brief overview of related work on this topic. It then goes on to describe the set of requirements for the prototype, the prototype itself and the results of a small user test. To finish off, conclusions are drawn and recommendations for future work are provided.

## 2. BACKGROUND

### 2.1 Fault Trees and Fault Tree Analysis

Fault trees model the occurrence of an undesired top level event by, as the name implies, using a tree structured model. In this model the root node represents the undesired event and the leaves the basic system events. The top level event is broken down into it's causes using a logic gate. These intermediate events are then broken down further into their immediate causes using more logic gates, until all the leaves of the constructed tree are basic system events.

Figure 1 shows an example fault tree. The top most node in the tree represents the modeled undesired event. All events, except basic events, are modeled as rectangles. Basic events are the leafs of the tree and are modeled as circles. All events are linked using logic gates. In standard fault trees (SFTs) the gates are limited to AND, OR and K/N voting gates, which fail when all, one or k child events fail respectively. Dynamic fault trees add priority AND (PAND), functional dependency (FDEP) and spare (SPARE) gates. A PAND fails when all child events


*24$^{th}$ Twente Student Conference on IT* January 22$^{th}$, 2016, Enschede, The Netherlands.

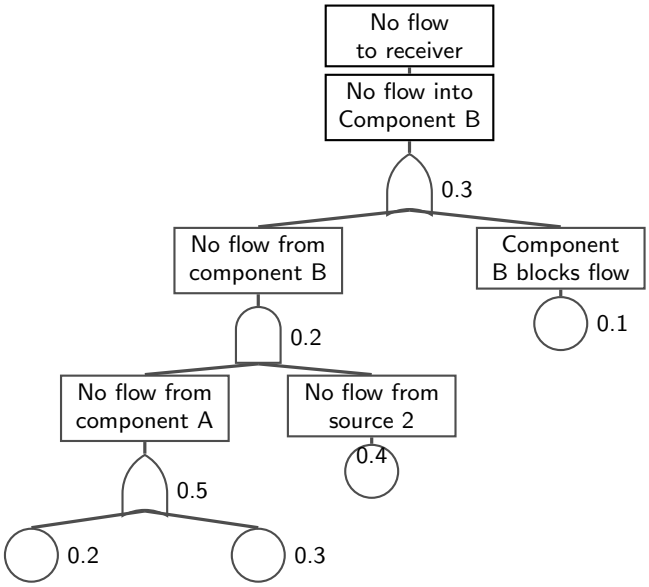


**Figure 1. Example fault tree.**

fail in the correct sequence, a FDEP indicates that if the first child event fails, the other children fail as well, and a SPARE indicates that if one of the children fails a spare is activated and only when that spare also fails, the SPARE fails.

Fault tree analysis can be divided into two major categories: qualitative and quantitative. Qualitative analyses include determining minimal cut sets (MCS) and minimal path sets (MPS). Minimal cut sets is a minimal set of basic events that if they all fail can create a top level event failure. Minimal path sets are the opposite: they are the minimum set of basic events that prevent the possibility of a failure. Quantitative analyses can be divided into discrete and continuous time analyses. Discrete time analyses include calculating failure probabilities and computing the expected number of failures for a certain mission time. Continuous time analyses include calculating the availability, mean time to failure and expected number of failures.

## 2.2 Information Visualization

We look at interactive visualizations to increase understanding of FTs and FTAs because according to Colin Ware "the human visual system is a pattern seeker of enormous power and subtlety. The eye and the visual cortex of the brain form a massive parallel processor that provides the highest-bandwidth channel into human cognitive centers. At higher levels of processing, perception and cognition are closely interrelated, which is the reason why the words 'understanding' and 'seeing' are synonymous." [16]. From this fact, the conclusion can be drawn that to increase understanding of a new concept, it should be visualized. Visualizing data in a graphic this way is called an infographic.

Before we can create infographics we must understand the workings of perception. Without a basic understanding of perception we would risk creating incomprehensible or misleading visualizations. The insights necessary to understand the design choices we made further on, are explained below.

According to Stephen Few [6] there are three important basic aspects of perception worth noting. First, we do not attend to everything that we see. Visual perception is selective, as it must be, for awareness of everything would overwhelm us. Our attention is often drawn to contrasts to the norm. Second, our eyes are drawn to familiar patterns. We see what we know and expect. Third, memory plays an important role in human cognition, but working memory is extremely limited.

Furthermore there is a limited set of attributes, called the pre-attentive attributes, that determine what draws attention [6]. Only two of these are perceived quantitatively with high degree of precision: length and 2-d position. The most important elements should thus be modeled using these properties.

In order to display quantitative information visually the following rules should be adhered to according to Edward Tufte [15]:

1. Enforce graphical integrity
2. Optimize Data-ink $= \frac{\text{Data-ink}}{\text{Total ink used to print graphic}}$
3. Avoid chart junk
4. Achieve high data density
5. Use small multiples

Enforce graphical integrity means data should be represented fairly. For instance, an y-axis on charts should not start on anything other than zero, otherwise it becomes hard to compare values of y. This could happen for example if the y-axis starts at 8 and end at 12. Say A has a value of 9, and B a value of 10. A would then seem to be half the value of B, where in reality it is only one tenth less than B. Optimizing data-ink ensures no unnecessary visual information needs to be processed in order to see the same thing. Avoid chart junk means that one should refrain from adding artistic details that distract from the actual data. This stems from the belief that the data is beautiful and should be interesting enough of its own accord. Achieving high data density means we should attempt to cleanly incorporate as much data as we can into one visualization, allowing us to explore more data at once. Small multiples is a method in which data is partitioned and each partition is shown in the same manner and on the same scale to enable comparison between partitions. For example displaying the average temperature by time of day partitioned into the months of the year would enable us to quickly compare the difference in temperature by time of day between months.

## 2.3 Interactivity

Static infographics are limited in their use because one image can not be designed to answer all the questions that it provokes. To explore the information further and answer our questions we need to introduce interactivity. According to Stephen Few there are 13 elementary operations for interactivity that enable analysis. They are: comparing, sorting, adding variables, filtering, highlighting, aggregating, re-expressing, re-visualizing, zooming and panning, re-scaling, accessing details on demand, annotating and bookmarking. If these operations are supported, a user can adjust the displayed information to his or her needs accordingly.

# 3. RELATED WORK

A lot of research has been done regarding fault trees [12] and interactive visualizations. However to our knowledge no research exists that combines these two fields of research directly. We suspect this is due to the fact that a lot of fault tree experts are comfortable with numbers and

likely not many are literate in reading or creating interactive visualizations. Likewise, it seems likely that people who are literate in reading and creating interactive visualizations have a limited understanding of fault trees.

We can however learn from research being done in the TREsPASS project. Deliverable 4.2.1 [4] presents an approach to visualize information security risks. A few key take aways are that they also built a web application, work with a tool chain instead of one monolithic program, and used D3.js as a framework. For their user interface they chose to provide a generic structure in which all tools could be integrated. The menu is on the left in a collapsible sidebar, the contextual information is in a pane on the right and the main content is in the center.

More generic research has also been conducted on how to display trees. We chose to use the algorithm presented by E.M. Reingold [10] because it produces tidy trees as the title indicates.

# 4. REQUIREMENTS

The goal of the prototype is to test the hypothesis that an interactive visualization of FTs and FTAs can improve the understanding of the concept of FTs and the exploration of the results of the FTA. To do this we want to create a minimum viable product [2]. The MVP consists of being able to display an FT and the basic aspects of qualitative and quantitative FTAs. Furthermore we should support as many operations as described by Few as makes sense to display these aspects. This will ensure the prototype is actually interactive. The prototype should of course be constructed in such a fashion that it is technically sound and on par with expectations of users of modern software. Below the exact list of requirements that followed from the above stated philosophy.

1. Aspects of FT and FTA
   (a) Support gates from SFTs and DFTs
   (b) Show MCS, MPS
   (c) Show propagation of aspects through tree
   (d) Show probability over time (discrete)
   (e) Show event details
2. Possible Operations
   (a) Show event details on demand
   (b) Possibility to highlight events, MCS, and MPS
   (c) Aggregating by collapsing/expanding event
   (d) Zooming and panning on tree
   (e) Filter time to specified range
   (f) Show/hide parts of menu
3. Technical
   (a) Easily extensible
   (b) Clean separation of data and visual
   (c) Use modern standards
   (d) Design for resolution 1280x800 and higher
   (e) Static input (no direct coupling with FTA software)
   (f) Responsive (adjust to browser window size)
   (g) No calculation, only displaying visuals

# 5. ARCHITECTURE

Based on the requirements and the research goal, we researched what the best approach to building the prototype was. We divided this task into four separate steps that we will describe below. The first step was selecting a library or framework that would help us to create the visual dynamically from the data. The second step was obtaining the data to visualize. The third step was integrating the created visual in a web page layout. The last step was to serve this page and the associated data from a back end.

## 5.1 Visual

For realizing the visual, we sought a library that would do most of the heavy lifting for us, without compromising control on what the end result would look like. To find a fitting library, we first compiled a list of frameworks that are in existence and discarded all frameworks that did not fit the requirements. Our main source for frameworks was the website Datavisualization.ch [14]. Very quickly D3.js [3] emerged as a clear winner. Its name, Data Driven Documents, gives a clear indication of what it is. It is a JavaScript library that provides a facility to bind data to web page elements. The properties of the element can then be set based upon the bound data. For example: if you bind the array of [10, 5] to a rectangle, you could set the height and width to 10 and 5 respectively. In addition to realizing the data binding, D3.js also provides many helper methods to quickly solve standard visualization needs. For example it can render the axis of a graph for you in Scalable Vector Graphics (svg). All one has to do is provide the right options. Two other important reasons for choosing for D3.js was that it is backed by a large community and it is used as a base for many other visualization tools. This gave the confidence it could do what we wanted to achieve.

## 5.2 Obtaining data

In order to visualize an FT and FTA results, we need an FT and some FTA results. For the SFTs we choose to use example trees from the Open-PSA initiative [8]. SCRAM [9] was used to calculate the FTA results for these trees. For the DFTs we used examples as provided by the web version of the tool DFTCalc [5] and also used DFTCalc [1] to calculate the FTA results for these trees.

## 5.3 Layout

For the layout, the framework Foundation for Apps [7] by ZURB Foundation was chosen. It is described by ZURB Foundation as "the first front-end framework created for developing fully responsive web apps.". This is exactly what we were looking for. We mainly use it to give us a solid base for setting up the layout and making sure it is responsive and works across devices. We planned on using it for communicating with our back end as well, but the prototype never reached that stage.

## 5.4 Back end

To serve the page and the necessary data to generate the visual we planned on using Ruby on Rails. It is a proven and solid foundation [11] for creating the back end of websites. Sadly, this step was never realized due to lack of time.

# 6. PROTOTYPE

In this section we give a detailed overview of the design of the prototype.

## 6.1 User Interface Layout

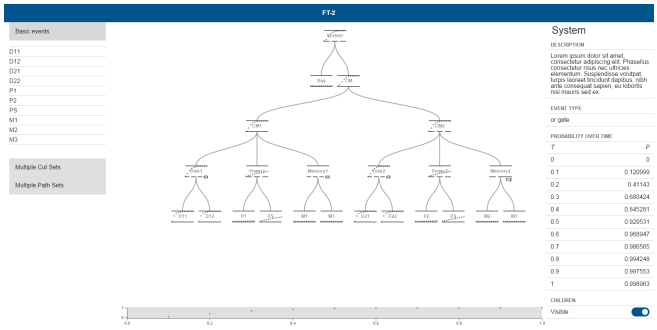

**Figure 2. User Interface Layout of the prototype**

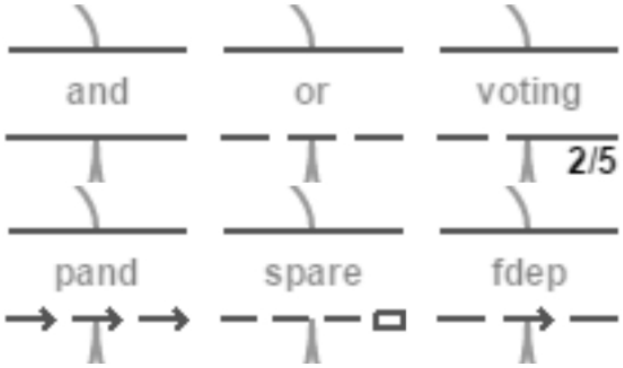


**Figure 3. Representation of the Gates**

For the layout we follow the general approach taken by many applications today: a title bar at the top, a menu on the left, the main content in the middle and extra information on the right. This can be seen in figure 2. The title bar contains the name of the currently displayed FT. The menu on the left consists of an accordion that gives access to all the features of the FT and FTA: list of basic events, list of MCS, and a list of the MPS. The main content area contains a zoom-able and drag-able FT and a time frame for filtering the time. The details of these elements are explained in the sections below.

## 6.2 Gates

The gates are represented as a shaped line at the bottom of each node as shown in figure 3. The idea is to minimize the amount of ink needed to convey the information and reduce the visual clutter by not using the standard gate symbols as described in the background section.

## 6.3 Event details on demand

In the right side bar, the details of an event are shown on demand (See figure 4). This means that when a user clicks on an event in the tree or in the menu sidebar, the node is highlighted everywhere and the right side bar is updated to reflect the currently highlighted event. If the right side bar is closed due to the browser size, it is automatically opened. This feature is an implementation of the details on demand operation as described by Stephen Few as discussed in the background section.

## 6.4 Probability over time

One of the requirements was to show the probability over time of events. This was realized by making every node a graph with on the x-axis time and on the y-axis probability (see figure 5). The axis scale from 0 to the maximum value present in the tree. The idea is that by displaying each

System

DESCRIPTION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus consectetur risus nec ultricies elementum. Suspendisse volutpat, turpis laoreet tincidunt dapibus, nibh ante consequat sapien, eu lobortis nisl mauris sed ex.

EVENT TYPE

or gate

PROBABILITY OVER TIME

| *T* | *P* |
|---|---|
| 0 | 0 |
| 0.1 | 0.120999 |
| 0.2 | 0.41143 |
| 0.3 | 0.680424 |
| 0.4 | 0.845281 |
| 0.5 | 0.929531 |
| 0.6 | 0.968947 |
| 0.7 | 0.986565 |
| 0.8 | 0.994248 |
| 0.9 | 0.997553 |
| 1 | 0.998963 |

CHILDREN

Visible

**Figure 4. Event details on demand in the right sidebar**

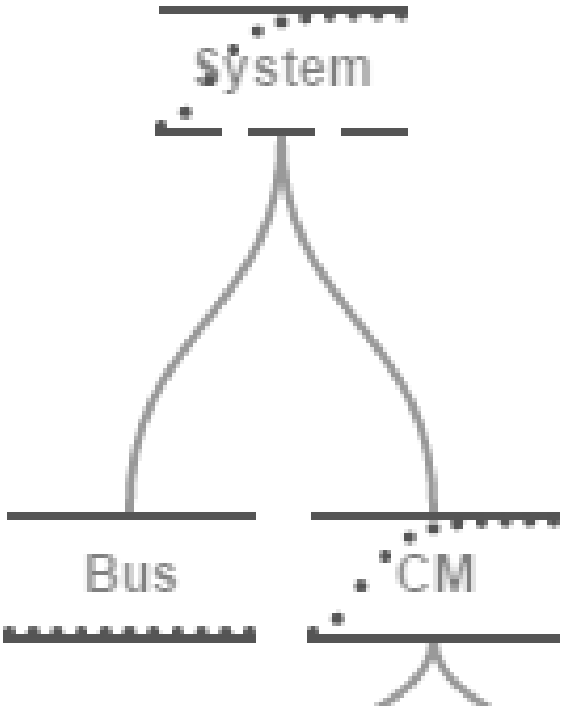


**Figure 5. Displaying probability over time**

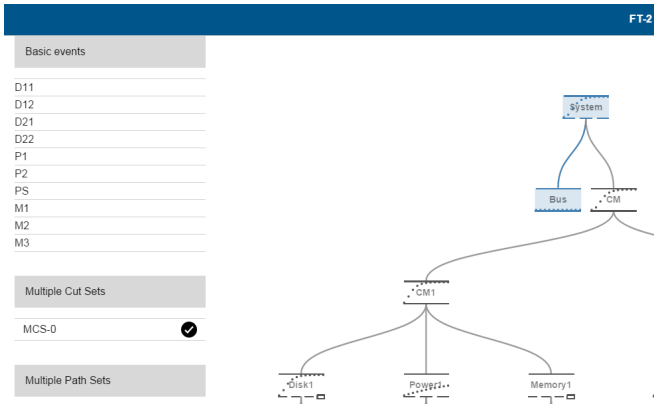


**Figure 6. MCS highlighted in FT**

event node as a graph one can easily see the behavior over time whilst at the same time compare it to other events. This also allows one to clearly see how events relate and probabilities propagate through the FT. It is form of small multiples.

If one wants to access the exact values of the probability at a certain time, they can use the details on demand feature.

## 6.5 MCS and MPS

MCS and MPS are displayed by highlighting the relevant BEs that together respectively cause or prevent the top level event failure (see figure 6). For MCS the failure propagation is shown by highlighting all failed nodes when the BEs in the MCS fail. This is an implementation of the highlighting operation as described by Stephen Few.

## 6.6 Expand/collapse part of FT

Parts of the FT can be collapsed by using the switch at the bottom of the details on demand pane (see figure 7). This allows the user to filter out irrelevant parts of the tree

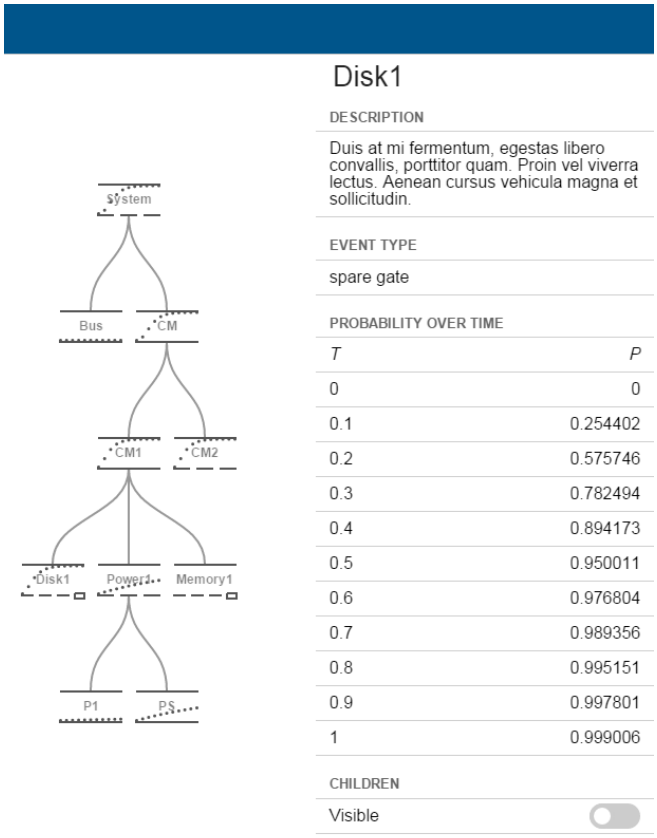


**Figure 7. Partly collapsed FT**

**Figure 8. Filtering time displayed in FT events**

not under study. A user gets a visual clue that a collapsed node is not expanded, due to the fact that all collapsed events have gates. These are represented by a line at the bottom, whereas BEs do not have this line. Thus if a line is present at the bottom of a node, it has hidden children.

## 6.7 Filtering time

The control in figure 8 is used to select the time under study. The top level events probability data points are plotted in a graph with axes. In this graph a shaded bar is drawn to represent the current range of time being displayed. The bar can be moved, resized at both ends or reselected by dragging.

## 6.8 Responsive

The layout is designed to be responsive. This means that when the screen becomes smaller, the sidebars are hidden automatically and can be pulled over the main content area when required. This is achieved by clicking on the arrows in the upper left and right corners. The right side bar is hidden by default on screens smaller than a 1200 pixels and the left side bar is hidden on screens smaller than a 1000 pixels. See figure 10 and 9.

# 7. VALIDATION

To validate or reject or hypothesis that an interactive visualization of FTs and FTAs can help in the understanding of the concept of FTs and the exploration of the results of the FTA we have done a small user test.

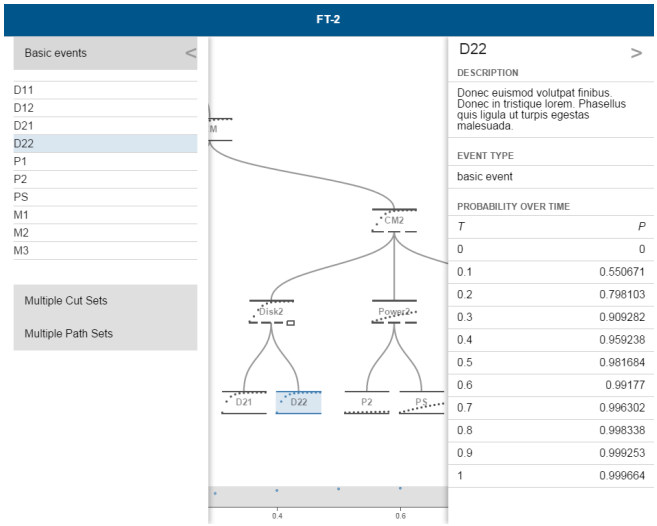


**Figure 9. Small screen - sidebars expanded**

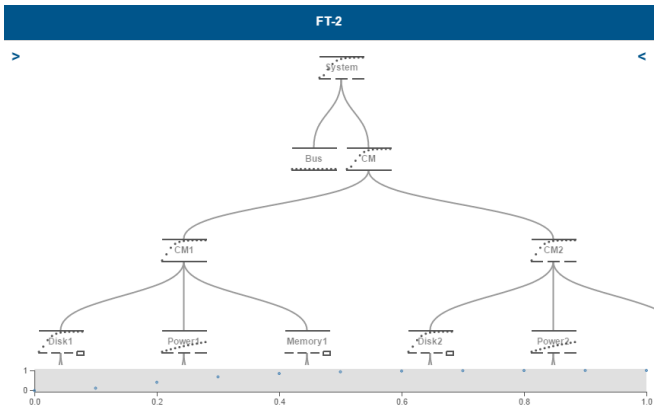


**Figure 10. Small screen - sidebars collapsed**

## 7.1 Method

We asked five different users to use our prototype and tell use what they were thinking whilst taking notes ourselves. The five users consisted of a FT expert, one familiar with FTs, a user experience professional and two users unfamiliar with FTs and user experience. They were each explained the goal of the project and, if necessary, the fundamentals of FTs. We then asked them to play with our prototype. First we would make observations about what they did or did not do. After this phase we told them about the parts they had not discovered and asked them for recommendations on making things clearer.

## 7.2 Results

This resulted in list of findings that can be used to give an indication of whether we should validate or reject the hypothesis. The findings, in no particular order, were as follows:

1. gray used for data points in small multiples too light (fixed)
2. hover on data point in time frame should give a clear indication of at which time a user is by letting this come back in the visual of the small multiples
3. make a distinction between SFT and DFT analyses: SFTs have minimal cut sets, whereas DFTs have minimal cut sequences
4. use standard symbols for gates instead of chosen gate lines (4x)
5. information overload for non-experts
6. use a color scheme to select complementing colors (fixed)
7. create association between the same elements by use of highlighting and colors (fixed)
8. repeat graph from tree node in event details pane to create association between components
9. enable whole node to be click-able (fixed)
10. scroll and zoom on the tree are useful (3x)
11. be list in menu should be click-able to ensure same behavior everywhere (fixed)
12. hide children visible toggle in leafs (fixed)
13. children visible toggle should also be present at the node itself
14. visually indicate that the menu accordion is expandable and collapsible
15. headings in the details on demand pane are clear
16. data points in tree node can go through the name
17. expand details on demand side bar if clicked on BE and it is not visible at that moment (fixed)
18. not directly clear after side bars disappear on resize how to get them back
19. in the details pane mention the unit associated with time
20. time frame not working as expected: it is clear what it does, but can not find the handles to get it to work

21. highlight current node (fixed)
22. exiting links transition to the wrong coordinates (fixed)
23. quickly grasped the concept of small multiples to display probability over time (3x)
24. quickly understood MCS and MPS
25. liked the responsive design (4x)
26. provide a legend for gate types (3x)

## 7.3 Discussion

These findings shows that user testing revealed a lot of minor usability problems, many of which have already been fixed. They also show that the gate type indicator was not understood and that a mistake was made in the correct naming of MCS on a DFT. However, the general layout with a menu on the left, content in the middle and a detail pane on the right proved intuitive. The associated responsive design as useful. And, in general users quickly grasped the concept of FTs and could use the visual to find answers to questions about the FTA results.

# 8. CONCLUSION

The small user test of the created prototype suggests that there is potential for a web based interactive visualization of FTs and the result of FTAs. We draw this conclusion, because in general the test users quickly grasped the concept of FTs and were able to explore the FTA results. The small multiple was perceived as useful. The gate type indicator was not understood by any users and should thus be changed. Furthermore the user testing was really useful in finding strengths and weaknesses of the prototype.

# 9. FUTURE WORK

Future research could expand upon the created visualization to display more aspects of FTs and FTAs. It would also be interesting to see the effect of directly coupling a visualization tool with FTA tools. We expect that if one can easily alter the FT and request more FTA results on demand, that would allow for better understanding of the concept of FTs and easier exploration of FTA results. The end goal being a seamless experience. In order to make this chaining of tools achievable, work also has to be done into further developing a universal data exchange format for FTs and FTAs. The only accurately documented standard at the time of writing is the Open-PSA Model Exchange Format, which lacks support for DFTs. The widely used Galileo format only specifies FTs and is not uniformly implemented.

# References


[1] Florian Arnold et al. *DFTCalc: a tool for efficient fault tree analysis (extended version)*. info:eu-repo/semantics/report TR-CTIT-13-13. Enschede, the Netherlands: University of Twente, Centre for Telematics and Information Technology (CTIT), June 2013. URL: `http://doc.utwente.nl/86711/` (visited on 01/13/2016).

[2] Steve Blank. *Minimum Viable Product | SyncDev*. Jan. 2016. URL: `http://www.syncdev.com/minimum-viable-product/` (visited on 01/13/2016).

[3] Mike Bostock. *D3.js - Data-Driven Documents*. Jan. 2016. URL: `http://d3js.org/` (visited on 01/13/2016).

[4] Lizzle Coles-Kamp et al. *Initial report on visualisations of information security risks*. Mar. 2015.

[5] *DFTCalc - Web-Tool*. Jan. 2016. URL: `http://fmt.ewi.utwente.nl/puptol/dftcalc/` (visited on 01/13/2016).

[6] Stephen Few. *Now You See It: Simple Visualization Techniques for Quantitative Analysis*. English. 1st edition. Oakland, Calif: Analytics Press, Apr. 2009. ISBN: 9780970601988.

[7] ZURB Foundation. *Foundation for Apps*. Jan. 2016. URL: `http://foundation.zurb.com/apps.html` (visited on 01/13/2016).

[8] *Open PSA Initiative*. URL: `http://www.open-psa.org/joomla1.5/index.php` (visited on 01/13/2016).

[9] Olzhas Rakhimov. *rakhimov/scram*. Jan. 2016. URL: `https://github.com/rakhimov/scram` (visited on 01/13/2016).

[10] Edward M. Reingold and J.S. Tilford. "Tidier Drawings of Trees". In: *IEEE Transactions on Software Engineering* SE-7.2 (Mar. 1981), pp. 223–228. ISSN: 0098-5589. DOI: `10.1109/TSE.1981.234519`.

[11] Sam Ruby, Dave Thomas, and David Heinemeier Hansson. *Agile Web Development with Rails 4*. English. 1 edition. Pragmatic Bookshelf, Oct. 2013. ISBN: 9781937785567.

[12] E. J. J. Ruijters and M. I. A. Stoelinga. "Fault Tree Analysis: A survey of the state-of-the-art in modeling, analysis and tools". In: (2014). URL: `http://eprints.eemcs.utwente.nl/25404/` (visited on 03/30/2015).

[13] Michael Stamatelatos et al. *Fault Tree Handbook with Aerospace Applications*. 1.1. NASA, Aug. 2002.

[14] Interactive Things. *Datavisualization - Selected Tools*. Mar. 2015. URL: `http://selection.datavisualization.ch/` (visited on 03/31/2015).

[15] Edward R. Tufte. *The Visual Display of Quantitative Information*. English. 2nd edition. Cheshire, Conn: Graphics Pr, Jan. 2001. ISBN: 9780961392147.

[16] Colin Ware. *Information Visualization, Third Edition: Perception for Design*. English. 3 edition. Waltham, MA: Morgan Kaufmann, June 2012. ISBN: 9780123814647.