CakES: Cake Metaphor for Analyzing Safety Issues of Embedded Systems *

Yasmin I. Al-Zokari, Taimur Khan, Daniel Schneider, Dirk Zeckzer, Hans Hagen[1]

1 **Department of Computer Science**
University of Kaiserslautern
Gottlieb-Daimler-Strasse
67663 Kaiserslautern, Germany
{alzokari, tkhan, d_schnei, zeckzer, hagen}@informatik.uni-kl.de

Abstract

Embedded systems are used everywhere. They are complex systems whose failure may cause death or injury to people or may damage the environment are required to be safety safe. Therefore, these systems need to be analyzed. Fault tree analysis is a common way for performing safety analysis. It generates a large amount of interconnected data that itself needs to be analyzed to help different domain experts (e.g., engineers and safety analysts) in their decisions for improving the system's safety. Additional difficulties occur for the experts in communication and in linking the data (e.g., information of basic events or minimal cut sets) to the actual parts of the system (model). Therefore, a large amount of time and effort is being spent on discussions, searching, and navigating through the data. To overcome this, we present a new metaphor called "CakES" consisting of multiple views visualizing the data generated by fault tree analysis and linking them to the actual parts of the model by intuitive interaction. Using the interaction techniques of CakES the user can directly explore the safety related data without navigating through the fault tree while retaining an overview of all critical aspects in the model.

1998 ACM Subject Classification I.3 [Computer Graphics]: Reliability, Testing, and Fault-Tolerance

Keywords and phrases Fault Tree Analysis, minimal cut sets, basic events, information visualization, scientific visualization, engineering, tiled-wall, multiple monitor system, color vision deficiency, embedded systems, safety critical systems

Digital Object Identifier 10.4230/DFU.Vol2.SciViz.2011.1

1 Introduction

The size and complexity of embedded system's increase steadily forcing users/experts of different domains (e.g., safety analysts and engineers) to analyze a large amount of data. However, not only the embedded systems grow but the data generated by safety analysis grows too. Thus, obtaining total knowledge about the data in the different domains becomes more and more important. In [29], the authors show that increasing human involvement in tasks (understanding, exploring, navigating) through large and/or complex data sets leads to slower and error prone results. Examples are: forgetting to explore a sub-tree, missing a BE, or incorrectly combining the events of a MCS. In addition, the users/experts need to find the interesting/useful data and link them to support in their decision making process (e.g., which

* This work was partially supported by DAAD, IRTG, ViERforES, BMBF.

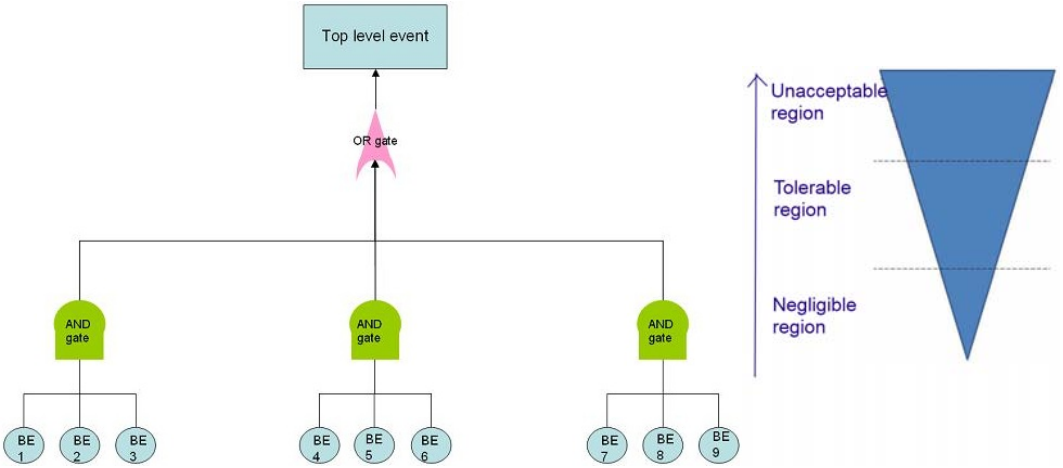
Scientific Visualization: Interactions, Features, Metaphors. *Dagstuhl Follow-Ups, Vol. 2.*
Editor: Hans Hagen; pp. 1–16

Dagstuhl Publishing
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany

parts of a model should be improved or replaced given limited time and/or cost). Because each user has a different domain background communication difficulties appear. Therefore, there is a need for methods/tools to support users/experts during their analysis, exploration, and comprehension of the data from both domains (generated from the safety analysis and given by the engineers). To be able to deal with enormous data, we propose reducing the complexity of the data being analyzed by first extracting the most important information of the fault tree analysis and by then providing dynamic filtering to the user. Filtering is used to ease and speedup understanding and exploration. We present a new metaphor, "CakES". CakES consists of multiple views visualizing the physical model (engineering data), minimal cut sets, and basic events' information generated from the fault tree analysis (safety data). The views are linked in an interactive way to support exchanging and combining the knowledge of the different domains. Providing different correlating environments for the users [12] enables the analysts to explore data in one glance by four views: three visualizing the data (minimal cut set, basic event, and model view) and one interaction view that also shows the exact values (menu view). We applied an informal expert evaluation that provided us with useful and productive feedback and criticism. The paper is structured as follows. Section 2 provides the definitions of terms used in this work. Section 3 describes the related work. Our approach, the visual metaphors and the interaction devices, are presented in Section 4. Section 5 illustrates the use of our approach with a real world data set and Section 6 describes our design decisions. The setup and outcome of an informal evaluation performed on this approach are described in Section 7. The future work is presented in Section 8. We close this paper with the conclusions in Section 9.

2 Definitions

For an introduction into the topic of safety analysis and visualization, the relevant concepts used in this paper are defined. In [14] pp. 74, risks are divided into three levels: Unacceptable, Tolerable, and Broadly Acceptable. Figure 1b shows these levels [31], [19]. For both tolerable and acceptable levels, risks have to be reduced until the cost is more than the risk "As Low As Reasonably Practicable"(ALARP). Risk acceptance depends on many factors such as applicable laws or public opinion [2]. A safety-critical system could cause different levels of harm (hazard)(i.e., delay, injury, death, etc.) to living beings or/and to its environment [21]. A risk is a combination of the severity of the consequences of a hazard and the probability of its occurrence [7]. *Safety analysis* is used to prove that the system's risk is negligible [20] and involves detecting hazards and their causes to reduce the likelihood of probable accidents and their consequence. *Fault Trees Analysis (FTA)* is a traditional methodology in safety analysis (hazard analysis in more detail) to asses systems' safety [13], [18], [7] pp. 8. Fault Trees (FTs) are usually constructed using conventional logic gate symbols. Figure 1a shows an example of a FT. *Basic Events (BEs)* are the lowest-level influence factors (system faults) in the FT that may cause (with a certain probability [11]) an undesired state in the system (top level event or called hazard) [7]. The top level event is the root of the FT. In Figure 1a the BEs are BE_1 through BE_9. *Minimal Cut Sets (MCSs)* are a unique combination of the BEs–so it contains BEs–, e.g., a system malfunction [4], [6]. When any BE of a MCS is removed, the remaining BEs no longer form a cut set [10]. "A fault tree can be represented as the collection of its minimal cut sets" [7]. To improve the safety of a system the MCSs should be specified and isolated or removed [7]. In Figure 1a, the MCSs are the inputs of the AND gates: $MCS_1 = \{BE_1, BE_2, BE_3\}$, $MCS_2 = \{BE_4, BE_5, BE_6\}$, and $MCS_3 = \{BE_7, BE_8, BE_9\}$. In safety analysis two factors are important [7]: First, the severity of the



(a) An example of a Fault Tree.

(b) The ALARP carrot diagram.

Figure 1 Foundations.

hazard's possible consequences. Second, the probability of the hazard occurrence, which is our concern in this paper. This is done usually by adding redundancy and/or improving the elements of the system.

3 Related Work

Visualization of safety and security data in embedded systems is an emerging research topic. With the growth of embedded systems and the growth of the data generated by fault tree analysis, visualization is becoming an important method to ease and speedup the developer's analysis. In this section a brief overview about the state of the art is given.

3.1 Fault Tree Analysis

We mention only some of safety analysis methods/tools, because others mainly follow the same principles. With the *ESSaRel tool* [10], users –usually safety analysts– can easily create FTs and generate MCSs analysis results (model generation and analysis). Here, component fault trees, gates, and BEs are visualized in 2D representations. However, the data generated by the MCS analysis –which is considered important for analysis– are not visualized and are represented in an external text file. This file includes some information of the MCSs such as their BEs, the BEs names, and their IDs (Figure 2). It is clear the user has to search for the FP of the BE from view (B) by clicking it to get view (D). Additionally, the names of the BEs in view (A) are not equal to the BEs names in views (C) and (D), which makes the exploration difficult. Further, there is no information about the MCSs' FPs the analysts can begin their search from. Therefore, the analyst has to find the other BEs of the examined MCS to compute its FP. As a result, the time and effort for exploring this small data set increases. In summary, the user has to navigate through the views to get simple informations such as which MCS is the most risky one, and still, no information about how the BEs look like or where they locate in the system is provided (especially if the naming of the BEs are inconsistent, which is usually the case when having different developers or engineers). Other tools/methods in safety analysis represent MCSs analysis data either in textual files

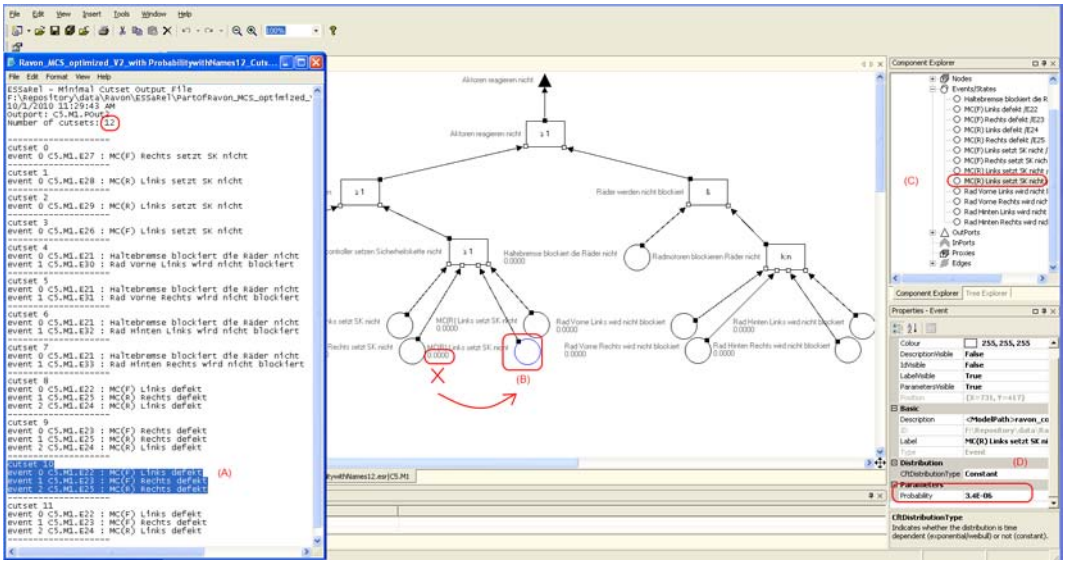


Figure 2 ESSaRel representations of a small sub-system containing 12 "minimal cut sets" with text (left). Fault tree and basic events (right).

or tables with or without their FPs depending on the methods, such as *FinPSA* [24], *RAM Commander* [1], *FSAP/NuSMV-SA Safety Analysis Platform* [6]. *Relex Fault Tree Analysis software* [17] has a feature for highlighting minimal cut sets. However, some other do not represent MCSs such as *ISAAC* [5] which is the new version of "Enhanced Safety Assessment for Complex Systems" (ESACS) [8]. In [15] the MCSs are represented as a subtree and in [26] as textual notation.

3.2 Visualization

A *3D Carousel View design* is presented in [23]. It takes the basic carousel model and elaborates it to hold an arbitrary number of 2D items. Here, the interactions used are spinning, zooming, and selection. Animation is included. Many video games use carousel views for displaying menus. They are usually used to select an item from a small set of items. The rotation begins slowly and the speed increases when the cursor gets close to the edge of the screen. The placement of the items is given by logarithmic equations. To avoid the problems caused by the clipping area, which causes difficulties for the users, they made the far items transparent. This method is applied when performance efficiency is not crucial. The *Hierarchy based 3D Visualization of Large Software Structures* ([3, 25]) presents a 3D visualization technique for the static structure of object-oriented software using distributions of 3D objects on a 2D plane. They adjust the transparency of object surfaces to the distance of the viewpoint to reduce the visual complexity. This method is used for hierarchical data (software packages). These data are represented as nested hemispheres. The *TrustNeighborhoods: Visualizing Trust in Distributed File Sharing Systems* in [9] is a tool for visualizing document trust relationships in large scale distributed file sharing systems. It uses a metaphor of a multilayered city to represent trust as geographic relations. It uses a radial space-filling layout, with a 3D mode for exploration. The interactions applied are dragging and dropping documents, overview and navigation, and smooth zooming (also called "fly to").

3.3 Summary

In summary: almost all safety analysis methods/tools are powerful in creating FTs and in generating the safety results (e.g., MCSs). However, analyzing the generated data needs: -Navigation through the textual files (upwards and downwards) for comparisons, and exploration. -Navigations through other views to compute or link the data together whenever the information is not complete in the textual reports (e.g., ESSaRel). Additionally, there is no information about the shapes and locations of the BEs. This implies that the user will have difficulties finding the real world relation between the physical hardware components and the fault trees and the MCS text representation. Finally, filtering these data according to Failure Probabilities is cumbersome if not impossible. Therefore, we propose to visualize the generated safety data to support the analysis.

4 Our Approach

In our approach both the benefits of the "Carousel View" [23] and the "Hierarchy 3D view" [25] (Section 3) and a "simplified circle packing algorithm" [34] are combined and extended to support our goal. In this work, the items are not only placed at the border of the table as in [23] but also in the inner area of the circle for space efficiency. The transparency of the containers (MCSs, Figure 5c in Section 4.3) depends on the user's selection, not on his distance as in [25]. Here, even though all the transformations are available, the users do not need to navigate forward and backwards to explore the contents of the containers as in [25]. Additionally, the containers are placed in a symmetric fashion in the inner area of each level, because the author in [28] pp. 192 shows that humans perceive symmetry as being important and thus it provides a powerful organizing principle. In our case, a vertical view (2D view) is sufficient to illustrate the MCSs that are the best candidates to be chosen first for analysis. Additionally, a horizontal view (3D view) is used when the user selects a MCS to display the physical parts related to the BEs in this MCS. Thus, the user is directly involved in the data analysis process without the need to navigate through the system's fault trees and text files. The fault tree information is hierarchically structured, whereas the model information is flat. Thus, we modified this data to get a hierarchical structure for the model information, because naturally the model parts are in a hierarchical form. Then, we mapped the relationships of both data domains, to get completely related data to work with. CakES is a multi-window, or multi-screen system supporting different levels of focus and context at the same time.

4.1 Overview of CakES

In this section, the views and the interactions of CakES are be presented. There are four views, one for interaction and switching the window focus and three for visualizing the data:

1. Model View: displays the system's model being analyzed and its parts.
2. MCS View: displays the collection of MCSs, their probabilities, and the physical parts related to their BEs, so this view contains the safety related data. Therefore, this view contains the information of both the "text file" and the "Properties-Event" view of ESSaRel as shown in Figure 2. Additionally, the shapes of the physical parts related to the BEs and the overall FP of each MCS are shown. This information comes from the model and is computed from the FT information. Finally, it also provides the overall system's safety.
3. Menu View: displays a menu and the safety data in values.
4. BE View: displays the physical parts related to the BEs that are contained in the selected MCS, this is an auxiliary view for future usage.

(a) RAVON robot in the model view.

(b) Outside the model.

(c) Inside the model.

Figure 3 Part of RAVON robot displayed in the Model view.

Table 1 Keyboard interaction.

Keyboard's buttons	
Key	Function
y, Y	Rotate in +y, -y
x, X	Rotate in +x, -x
z, Z	Rotate in +z, -z
→	Move right
←	Move left
↑	Move up
↓	Move down
PgUp	Zoom out
PgDn	Zoom in
b, B	Translucent, Opaque
a, A	Animate, Stop Animation
o and O	Return to default position
s, S	Spotlight on, off
PrntScrn	Snapshot of the model view
ctrl + 1	Switching between standard and stereo view

4.2 Model View

In this view the model of the system being analyzed is displayed. For example, we have a robot model called RAVON (Robust Autonomous Vehicle for Off-road Navigation) shown in Fig 3a. This model is given in OpenInventor format [30]. OpenInventor is a C++ Object Oriented 3D graphics API providing higher layer programming of OpenGL that helps in efficient and convenient programming [22], [32]. The user can interact with the model using keyboard, mouse, and space-mouse. The interactions provided are rotation and translation such as zooming(z axis) and panning (x, y axis). These interactions are shown in the Tables 1 and 2. Further, the user has the ability to explore the model not only from the outside but also from the inside by placing the near clipping plane in a suitable position. Figure 3 shows an example.

4.3 MCS View

The *Cake* is our metaphor to visualize the failure probability (FP) of MCSs and the shape of the physical parts related to their BEs. Figure 5c shows the MCS view. All examples in

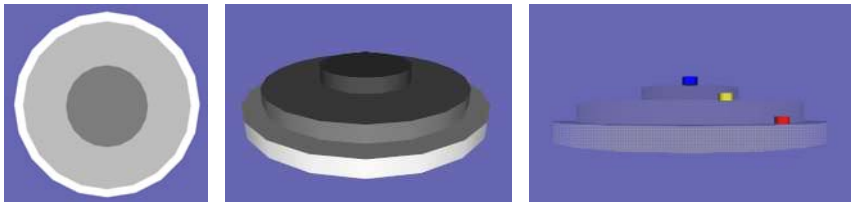
Table 2 Mouse and space mouse interaction.

(a) Mouse.

Mouse	
Button	Function
Left	rotate
Right	moving left & right
Scroll	zoom in/out
Middle	Return to default position

(b) Space mouse.

Space Mouse	
Action	Function
Roll, Pitch, Yaw	Rotate in (x, y, z)
Move	in x, y, z directions
Left button	Return to default position



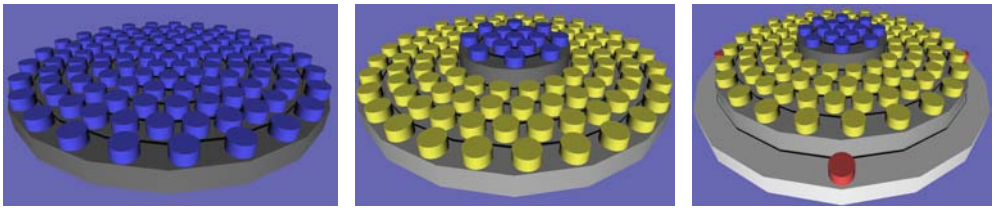
(a) A 2D view (top view).

(b) A 3D view.

(c) Placing MCSs depending on their failure probability.

Figure 4 Levels background and MCSs placement in the MCS view.

this section (only for the MCS view) are generated from artificial data for demonstration purposes. Two examples with real data will be presented in Section 5. The cake consists of a maximum of three levels that reflects FPs ranges shown in Section 2. There are four FP values that influence the number of levels displayed and the placement of MCSs: minimum, border between lowest and middle range (lower border), border between middle and highest range (upper border), and maximum. The first level, the one in the center, corresponds to the lowest range of FPs, those between the minimum and the lower border. This level includes MCSs with small FPs (could be considered as low risk MCSs). The second level corresponds to the middle range between lower and upper border. This level includes MCSs with higher FPs (could be considered as risky MCSs). The third level, the outer most one, corresponds to the highest range of FPs, those between the upper border and the maximum. This level includes MCSs with the highest FPs (could be considered as very risky MCSs). The first level has the darkest gray, the second level has a lighter gray, and the outer level has the lightest gray background. Figures 4a and 4b show the background coloring of the levels. The reason for giving the outer level the lighter gray color is that the user would then be able to directly concentrate on the most important level, which has the highest FPs [28] p. 143. Each level contains a number of MCSs (represented as cylinders) that have a FP within its range (Figure 4c). If there are no MCSs within the boundaries, then the level is empty and will not be displayed. Examples having only one or two levels are shown in Figure 5. Each MCS contains a certain number of BEs. The FP of a MCS is computed as the product of the FPs of its BEs. The user can manipulate the levels and the MCSs placements by changing the FP values using the menu sliders (Figure 9) described in Section 4.4. We computed the placements of MCSs using the following equations: The radius of the MCSs plus an empty distance around each MCS R_s is computed as: $R_s = 0.5 + \frac{1}{8}$. The radius of the ring holding the MCS R_c is $R_c = R_s \cdot RingNo$. The level radius R_h is $R_h = R_s + R_c$. The number of the MCSs in a ring is given by $m_r = \max(NoMCS) = \frac{\pi}{\arcsin \frac{R_s}{R_c}}$. The number of MCSs in the



(a) One level. **(b)** Two levels. **(c)** Three levels.

Figure 5 3D MCS view while interacting with the menu sliders.

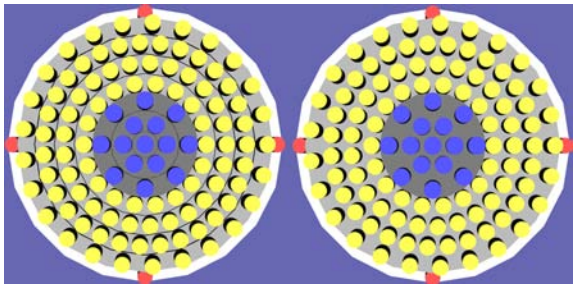


Figure 6 CakES: the MCS view. MCS view in 2D: With rings (left), With no rings (right).

level is $m_h = \sum_{r \in ring} m_r$. The number of MCSs in the cake is $m = \sum_{h \in levels} m_h$ Around each group of MCSs a black ring is placed to improve the distinguishability and to reduce any errors that could be caused by color contrast [28] p. 143. Figure 6 shows the view with and without rings, respectively. Table 3 shows the coloring of the MCSs in each level. The color vision deficiency entries work for Protanopia, Deuteranopia, and Tritanopia. Since the cyan has a level of green and blue, the blue-yellow problem of the Tritanopia type is avoided. Further, the red-green problem of the Protanopia and Deuteranopia type is solved, because red and green are not neighbors in the visualization (Figure 7). We added two different saturation levels to the MCSs colors, to provide more visualized details about the FPs of the MCSs to the users. These saturation levels are computed using the logarithmic mean of the range. High saturation is assigned to the upper part and low saturation is assigned to the lower part of this division. Figure 8 (right) shows MCSs of the second level of the cake (yellow) having different saturation values, where the MCSs having less saturation are the ones with less importance than the others in the same level. The MCSs of the first level are fully saturated blue and the ones of the third level (that contains red MCSs) are less saturated. The *BEs* of a MCS relate to both hardware and software parts. The physical parts related to the BEs of the fault tree are visualized inside each MCS to give the user the relationship between the cake's view and the BE view. Figure 8 shows two parts of the control box of RAVON in the selected MCS. The positioning of the BEs in the MCSs depends on the number of the BEs in a MCS. From our safety experts we know that the most important MCSs are the ones having 1-3 BEs. For example, for a single point of failure there will be a MCS with only one BE. This case is the most risky one, because the possibility for this MCS to occur is higher than the MCS that has two or more BEs, etc. However, we visualized a maximum of six BEs of MCSs that have six BEs or above to provide tolerability. Let x, y, z be the coordinates, h be the height of the MCS's cylinder in y direction, and n be the number of the BEs in a MCS. Then, the following equations are used for the placement of BEs depending of their number of BEs. If $n = 1$, then it will be placed in the center of the

Table 3 Coloring.

Risk level	Normal vision	Color vision deficiency
Low	Blue	Cyan
Medium	Yellow	Yellow
High	Red	Orange

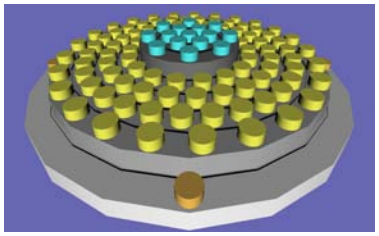


Figure 7 Cake using colors adapted to Protanopia, Deuteranopia, and Tritanopia.

MCS (x, y, z). If $n = 2$, they will be placed in a straight line, on points $(x, y \pm \frac{1}{4} \cdot h, z)$. If $n = 3$, they will be placed in a triangle, on points $(x - \frac{1}{2} \cdot r, y \pm \frac{1}{4} \cdot h, z)$ and $(x + \frac{1}{2} \cdot r, y, z)$. If $n = 4$, they will be placed on a polygon, on points $(x \mp \frac{1}{2} \cdot r, y \pm \frac{1}{4} \cdot h, z)$. If $n = 5$, they will be placed in the form of a cross, on points $(x \mp \frac{1}{2} \cdot r, y \pm \frac{1}{3} \cdot h, z)$ and (x, y, z). If $n = 6$, they will be placed on the six corner points of a polyhedron $(x \mp \frac{1}{2} \cdot r, y \pm \frac{1}{3} \cdot h, z)$ and $(x, y, z \pm \frac{1}{2} \cdot r)$. The most common approach for an analysis is to start with an overview and then to zoom into details as needed. The cake can be rotated using the same interactions as the model interaction. Two types of zooming are available in this view: spatial and semantic zooming. This approach is supported by providing the data at different levels of granularity. Using spatial zooming, the cake can be enlarged to pick the required MCS. Picking is done by (ctrl + left mouse button) and invokes a semantic zooming. When this is done, the MCS becomes transparent, the inner contents of the MCS will appear (BEs), the BEs are also displayed in the BE view, and the related parts in the model are highlighted.

4.4 Menu View

The menu view is primarily used for interaction. There are three interaction areas in the menu. The first area ((A) in Figure 9) consists of four buttons. Three of them can be used to select the views in focus and one terminates the application. The second area ((B) in Figure 9) consists of four range sliders with logarithmic scale. They determine the minimum, lower border, upper border, and maximum probabilities of the levels (Section 4.3). The sliders act as filters allowing to select FP ranges dynamically depending of the standard of the system being analyzed. Changing the values removes those MCSs from the view that are below the minimum or above the maximum FP. Further, MCSs are assigned to different levels when the borders are changed. Finally, two radio buttons in the third area determine whether to use normal coloring or colors adapted to a certain color vision deficiency type. We reduced the size of the menu in the figure for demonstration purposes. The second function of the menu is to display additional information. Statistical quantitative information of the safety file is provided, such as the number of MCSs and the number of BEs in the system. Further, if the user selects a MCS, its size (number of BEs it contains), its FP, and information about its BEs are displayed. The information about the BEs includes their ID, name, and FP.

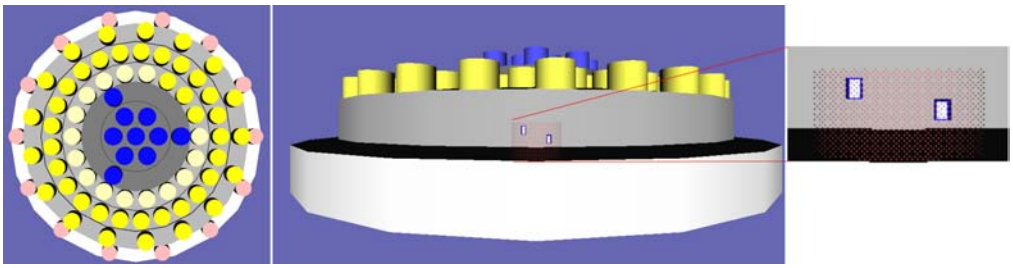


Figure 8 MCS view. (right) Saturation difference in the second level. (middle and left) Two parts of the controller box related to the BEs shown in the selected MCS.

4.5 BE View

The BE view has a direct relation to the cake and to the model view. In this view, the hardware parts related to the BEs of the selected MCS in the MCS view are displayed. This view shows these parts in more detail. As in the MCS view, a maximum of six BEs are displayed. This view is an auxiliary view for future requirements. The interaction is the same as in the MCS and the model view.

5 Example using Real World Data

We applied our method to a real model in the context of the ViERforES project [27] funded by the German Federal Ministry of Education and Research (BMBF). It aims at optimizing the analysis of safety, security, and reliability aspects of embedded systems. The Robotics Research Lab of the University of Kaiserslautern, Germany [33] provides the Robot RAVON (Robust Autonomous Vehicle for Off-road Navigation) [16]. This complex embedded system is analyzed and tested using fault tree analysis (FTA) by safety analysts in the software engineering department. The results were fault trees containing basic events and additionally, after applying a MCS analysis, a list of minimal cut sets (540 MCSs) and their BEs. We adapted the system to two different environments. The first consists of a standard monitor and a stereo monitor (Section 5.1). The second is a tiled wall display (Section 5.2).

5.1 Standard and Stereo Monitor

This configuration consists of a standard monitor with 1920×1200 pixels and a Zalman Trimon passive stereo monitor with a resolution of 1600×1050. The computer has an AMD Phenom™ Quad-Core Processor with 2.60GHz, 3.25 GB of RAM, and an NVIDA GeForce GTX 280 graphics card. A 3Dconnexion Space Navigator is used for interaction. The multiple window system is used in an environment with two monitors (two standard or one standard and one stereo). The model view is displayed on the second monitor that is either a standard or a stereo monitor, and the three other views are displayed side-by-side on the standard monitor as shown in Figure 10. Here, a yellow MCS is selected in the MCS view as shown in Figure 10a (right). The hardware components associated with these BEs are displayed in the BE view (middle). Information about the BEs in the selected MCS is displayed in the menu view (left). Additionally, the consequence of this selection is shown in the model view automatically (Figure 10b). Here, the Ravon model is displayed on the second monitor, either in normal or in stereoscopic view. The BEs from the MCS that was selected in the MCS view (Figure 10a) are opaque, while the rest of the model is transparent.

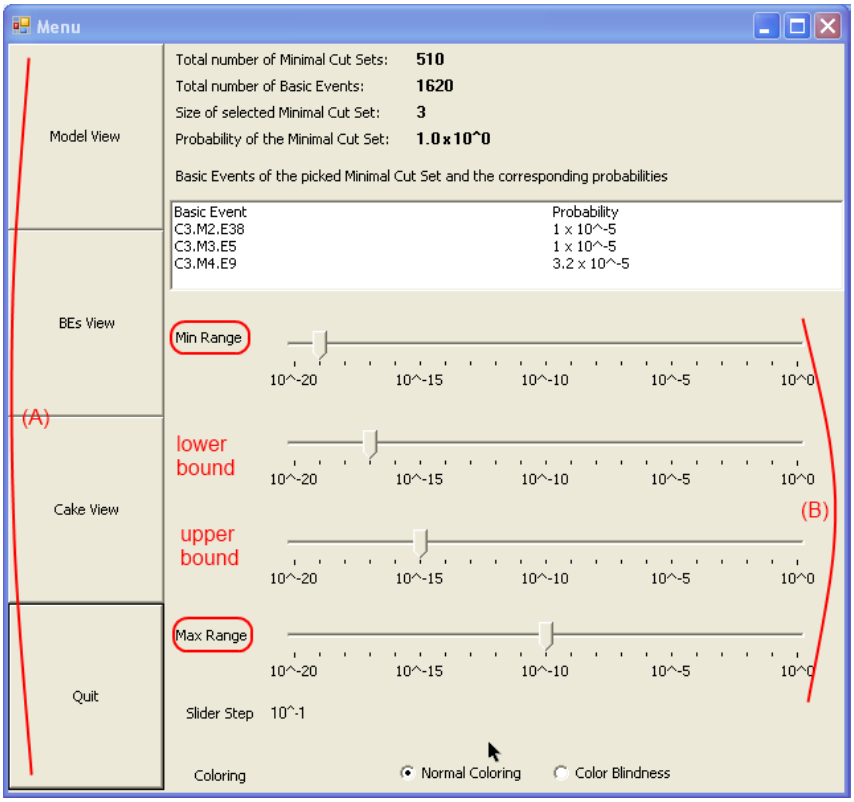


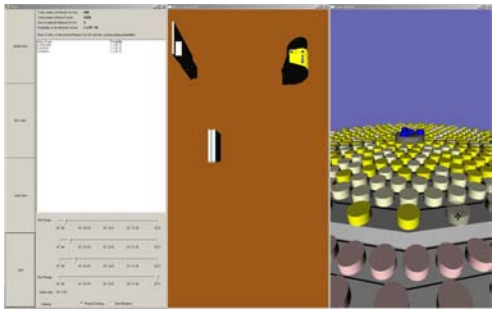
Figure 9 Menu view.

5.2 Tiled Wall

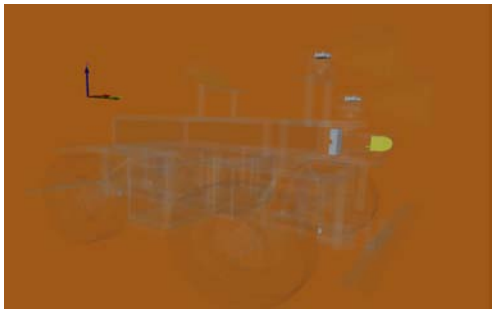
The 3×3 tiled wall consists of 9 displays that are connected to 5 computers. It is a multi-screen environment shown in Figure 11. The top 6 screens display the model view and the bottom 3 screens display the menu view, the BE view, and the MCS view (from left to right) as shown in Figure 11. As a result, this visualization can work in any environment having different views, windows, or screens.

6 Reasons for Some Choices

Our choices for the visualization metaphors are based on the following considerations. Having MCSs in different sizes, i.e., if a MCS has more BEs then it is larger, is better to be avoided. Large MCSs will have mostly a lower importance, since the probability of a MCS is calculated as the product of its BEs probabilities. Therefore, even though they have larger sizes, they are less critical. Assigning a small size to a MCS having a large number of BEs and vice versa causes a counter-intuitive visualization, that we wanted to avoid. Having only six BEs visualized in a MCS is sufficient, for the same reason described above. Whenever a MCS has more than six BEs, it is most probably not critical. In addition, when visualizing more than six BEs, the sizes of these elements will decrease too much and will no longer be distinguishable. Finally, MCSs having more than six BEs are crowded. Having MCSs visualized as cylinders avoids lines crossing on the surfaces of polygonal shapes that will partially occlude the shape of the BEs inside the selected MCS. Additionally, spherical shapes limit scalability and waste the surface area that could be used in the case of cylinders with the



(a) The menu (left), the BE (middle), and the MCS view (right).



(b) The Ravon model on a standard monitor.

Figure 10 CakES on standard monitors.

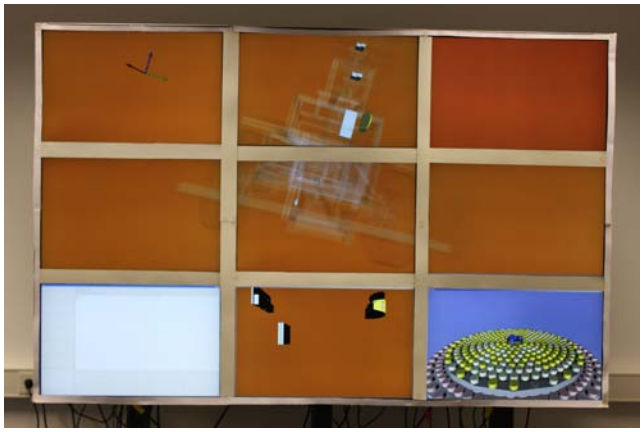


Figure 11 The CakES system on a tiled wall.

same radius. Having spaces between the cylinders enables the user to move freely inside the cake, reduces over crowdedness of MCSs, and eases the user's ability to distinguish between the BEs in the MCSs, when a MCS is selected. According to the risk acceptance diagram (Section 2, Figure 1b, [19]) there are three levels: Negligible, Tolerable, and Unacceptable. Therefore, we prefer having three levels of risks in the cake. Using different saturations for the same level increases the granularity inside each safety level. This helps the user to find the most critical MCSs. Neither 2D nor 3D text is used to avoid the drawbacks of text in 3D environments. For example, having text on top of the MCSs is only clearly visible to the user, when the scene is vertical, however, when the scene is not vertical, the text is less readable. Additionally, if we repeat the text many times or if we hang the text close to the MCSs, the scene will get crowded, even if the number of MCSs that are visualized is small. Finally, when zooming out, the text and textures become unreadable and the exploration will become difficult. As we will analyze systems having a large number of MCSs (normally, more than a hundred), having a unique color for each MCS is not feasible. According to [28], the amount of colors being easily distinguishable lies between five and ten different colors. For the same reason, textures are also not used.

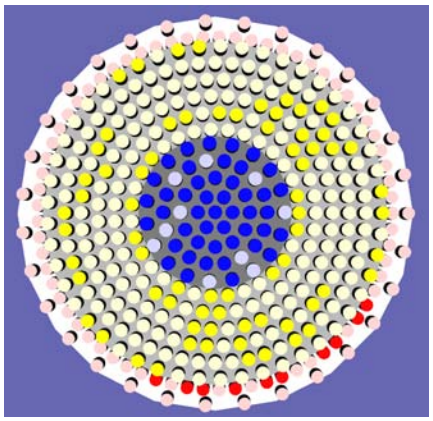


Figure 12 MCS view needs ordering.

7 Informal Expert Evaluation

An informal expert evaluation was performed on this approach by a group consisting of specialists from the domains software engineering, interaction, engineering (robotics department), and safety analysis, as well as information visualization. The aim of this evaluation was to get as much feedback as possible to improve our approach for the need and the questions of the experts of the different fields. The final feedback from this group was used to enhance the visualization in various aspects. They observed and suggested the following:

- The cylinders are not smooth.
- There are occlusions in the MCS view, because of the distortion.
- The most important MCSs should be in the first level, so the placement of the important and less important MCSs in the MCS view should be exchanged.
- It is better to have the MCSs in order from inside to outside, so the user can get a feeling about the importance degrees between them.
- The MCSs should be sorted depending on their probabilities.
- The color of the least important MCS should be green instead of blue in agreement with standard dependability requirements.
- Show clearly the connection between BEs and physical parts.
- Have a bigger, clearer text font in the menu.
- The interaction with the sliders are difficult.

As a conclusion, we should enhance our method based on this feedback.

8 Future Work

We plan to improve CaKES by taking the recommendations of the informal expert evaluation into account. Especially, ordering the MCSs by their probabilities in each level level would be beneficial. Figures 11 and 12 show the problem of unordered data. This would even increase more, if the data is large. Further, we would like to apply our approach to other display configurations, e.g., CAVE environments. Additionally, we will perform a user study and examine the usefulness of CakES. improve the sliders usability. Because the transparency caused delay in the interaction we need to change it to another function or by using a transparency node. Therefore, we changed to the *SORTED_OBJECT_BLEND*

transparency type. Finally, we want to test our approach using other examples from ViERforES and related projects.

9 Conclusion

From the related work, we found that it is necessary to come up with new approaches and/or visualizations to explore the results of safety analysis, because these results are complex and/or large in nature. Additionally, the information about the geometrical location of the events that cause failures to the system and that are most significant to the embedded system is important. The CakES metaphor provides the user with the ability to explore data with more than five different levels of focus and context: the model, a combination of the MCSs' Failure Probability, the components of each MCS, the BEs' Failure Probability, and the shape and the locations of the BEs in the model. Linking the views shows the relationship of the data in a realistic interactive approach. This work draws immediate attention to the most important MCSs of the embedded system that cause its failure. The user (a safety analyst and/or an engineer) can apply this solution in different environments such as standard monitor configurations, tiled wall, and the combination of standard and stereo monitors [12]. To our knowledge, we introduced the first 3D representation for safety analysis features, linking the existing knowledge of this domain with the engineering domain's knowledge reducing the time of searching for relevant information, and leading to a reduction of human errors, effort, and cost. Additionally, CakES provides the ability of collaboration between specialists in different domains (e.g., safety analyst and robotics engineer). Finally, after gaining experience in either domain the specialists could efficiently work alone exploring and judging how to choose the parts to maintain and the parts to replace depending on their importance.

10 Acknowledgments

This work is partially supported by the DAAD, the IRTG sponsored by the DFG, and by the BMBF project ViERforES. We thank all our colleagues for helpful discussions and suggestions, especially Martin Proetzsch, Yi Yang, and Guo Zhensheng.

References

1 ALD Reliability Software Advanced Logistics Development and Services Worldwide. Fault tree analysis (fta). `http://www.aldservice.com/en/reliability/fault-tree-analysis.html`; Online; accessed 6-August-2010.

2 Colin Atkinson, Christian Bunse, and Hans Gross. *Component-Based Software Development for Embedded Systems: An Overview of Current Research Trends*. Springer Berlin Heidelberg, 2005.

3 Michael Balzer and Oliver Deussen. Hierarchy based 3d visualization of large software structures. In *VIS '04: Proceedings of the conference on Visualization '04*, page 598.4, Washington, DC, USA, 2004. IEEE Computer Society.

4 Tim Bedford and Pieter van Gelder. *Safety and Reliability: Proceedings of the ESREL 2003 Conference, Maastricht the Netherlands, 15-18 June 2003*. Taylor & Francis, 2003.

5 Piergiorgio Bertoli. *Model Checking and Artificial Intelligence*. ERTS, 2006. `http://www.springerlink.com/content/rq430x354035p654`.

6 Marco Bozzano and Adolfo Villafiorita. The FSAP/NuSMV-SA Safety Analysis Platform. *International Journal on Software Tools for Technology Transfer (STTT)*, 9(1):5–24, 2007.

7 Marco Bozzano and Adolfo Villafiorita. *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, 2010.

8 Marco Bozzano, Adolfo Villafiorita, Ove Äkerlund, Pierre Bieber, Eckard Boede, Matthias Bretschneider, Antonella Cavallo, Charles Castel, Massimo Cifaldi, Alessandro Cimatti, Alain Griffault, Christophe Kehren, Benita Lawrence, Andreas Lüdtke, Silvayn Metge, Chris Papadopoulos, R. Passarello, Thomas Peikenkamp, Per Persson, Christel Seguin, Luigi Trotta, and Laura Valaccaand G. Zacco. ESACS: an integrated methodology for design and safety analysis of complex systems. In *In Proceedings of ESREL 2003*, pages 237–245, 2003. Maastricht, The Netherlands.

9 N. Elmqvist and P. Tsigas. Trustneighborhoods: Visualizing trust in distributed file systems. In *Proceedings of the Eurographics/IEEE VGTC Symposium on Visualization 2007*, pages 107–114, 2007.

10 ESSaRel. Background information — essarel, 2002. `http://www.essarel.de/background/background.html`; Online; accessed 6-August-2010.

11 Bernhard Kaiser, Catharina Gramlich, and Marc Förster. State/event fault trees - a safety analysis model for software-controlled systems. *Reliability engineering & systems safety*, 92:1521–1537, 2007. `http://www.ingentaconnect.com/content/els/09518320/1998/00000061/00000001/art00061`.

12 Taimur Khan, Daniel Schneider, Yasmin Al-Zokari, Dirk Zeckzer, and Hans Hagen. Framework for comprehensive size and resolution utilization of arbitrary displays. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, Dagstuhl Follow-Ups, Wadern, Germany, 2010. Schloss Dagstuhl–Leibniz Center for Informatics. accepted for publication.

13 Nikolaos Limnios. *Fault Trees (Control Systems, Robotics & Manufacturing Series)*. Wiley, John & Sons, 2007.

14 UK Ministry of Defence. Safety Management Requirements For Defence Systems, Defence Standard 00-56, 2007.

15 Frank Ortmeier, Wolfgang Reif, and Gerhard Schellhorn. Formal safety analysis of a radiobased railroad crossing using deductive cause-consequence analysis (DCCA). In *In Proceedings of 5th European Dependable Computing Conference EDCC, volume 3463 of LNCS*. Springer, 2005.

16 RAVON. AG Robotersysteme: Ravon, 2009. `http://agrosy.informatik.uni-kl.de/en/robots/ravon/`; Online; accessed 6-August-2010.

17 RELEX. Fault tree event tree. `http://www.relexsoftware.co.uk/products/ftaeta.htm`; Online; accessed 6-August-2010.

18 ReliaSoft Corporation. Fault tree analysis — weibull, reliability engineering resources, 2009. `http://www.weibull.com/basics/fault-tree/index.htm`; Online; accessed 6-August-2010.

19 Jørn Vatn. A discussion of the acceptable risk problem. *Reliability Engineering and System Safety*, 61:11–19(9), July 1998. `http://www.ingentaconnect.com/content/els/09518320/1998/00000061/00000001/art00061`.

20 Savive Pty Ltd. Risk analysis and assessment. `http://www.savive.com/inform/riskassessment.html` ; Online; accessed 6-August-2010.

21 Savive Pty Ltd. Safety-Critical. `http://www.savive.com/inform/safetycritical.html` ; Online; accessed 6-August-2010.

22 sgi. Open Inventor™, 2011. `http://oss.sgi.com/projects/inventor/`; Online; accessed 6-July-2011.

23 Wang Shuo, Poturalski Marcin, and Vronay David. Designing a generalized 3d carousel view. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 2017–2020, New York, NY, USA, 2005. ACM.

24 STUK. Finpsa - tool for professional living psa. `http://www.stuk.fi/ydinturvallisuus/ydinvoimalaitokset/en_GB/finpsa/`; Online; accessed 6-August-2010.

25 Alfredo Teyseyre and Marcelo Campo. An overview of 3d software visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):87–105, 2009.

26 Andreas Thums and Gerhard Schellhorn. Formal safety analysis in transportation control. In *Proceedings of the Workshop on Software Specification of Safety Relevant Transportation Control Tasks, VDI Verlag GmbH*, 2002.

27 ViERforES. Vierfores, 2009. `http://www.vierfores.de/`; Online; accessed 6-August-2010.

28 Colin Ware. *Information Visualization.* Morgan Kaufmann, 2004.

29 Markus Weber. A survey of semantic annotations for knowledge management, 2008. `http://www.mendeley.com/profiles/markus-weber/`; Online; accessed 6-August-2010.

30 Josie Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor™, Release 2, Chapter 9. Applying Actions.* Addison-Wesley, 1994.

31 Wikipedia. Alarp. `http://en.wikipedia.org/wiki/ALARP` ; Online; accessed 6-August-2010.

32 Wikipedia. Open Inventor, year = 2011, note = `http://en.wikipedia.org/wiki/open_inventor`; online; accessed 6-july-2011.

33 Wikipedia. Ravon — wikipedia, the free encyclopedia, 2009. `http://en.wikipedia.org/w/index.php?title=Ravon&oldid=278928053`; Online; accessed 23-November-2009.

34 Dirk Zeckzer, Fang Chen, and Hans Hagen. Computing an Optimal Layout for Cone Trees. In Hans Hagen, editor, *Scientific Visualization: Advanced Concepts*, volume 1 of *Dagstuhl Follow-Ups*, pages 11–29. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2010.