SIGRAD 2012
A. Kerren and S. Seipel (Editors)

ESSAVis: A Framework to Visualize Safety Aspects in Embedded Systems

Ragaad AlTarawneh[1], Jens Bauer[1], Patric Keller [2], Achim Ebert[1], and Peter Liggesmeyer[2]

[1]Computer Graphics and HCI Group, University of Kaiserslautern, Germany, {tarawneh,j_bauer, ebert}@cs.uni-kl.de
[2]Software Engineering:Dependability Group, University of Kaiserslautern, Germany, {pkeller, liggesmeyer}@cs.uni-kl.de

Abstract

In this paper, we present a framework, called Embedded-System Safety Aspects Visualization (ESSAVis), that is a system prototype designed to analyze the safety aspects of embedded systems. The ESSAVis prototype provides a 3D environment that aids in detecting infected components in the hardware of the target embedded system. The prototype also provides an abstract representation for the failure mechanisms of the target embedded system, including the software structure and failure path information for the underlying safety scenarios at a certain moment in the system's life. The results indicate clearness of our proposed method over existing techniques and promise acceleration in performance of the failure detection process in embedded systems for critical applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—B.7.2 [Design Aids]: Graphics—Simulation D.2.6 [Programming Environments]: Graphical environments—

1. Introduction

Embedded systems are widely used in our daily activities. Some examples of such systems are control systems in cars, airplanes, rail-road crossings and even washing machines. Normally, embedded systems consist of both hardware and software components. Therefore, most of them have complex structures and are not centralized in one component but are distributed among a set of components, which represent the system parts. These components communicate with each other via a set of hardware and software interfaces. These features help in developing relatively complex systems since many small-embedded systems, although physically separated, can be grouped together to construct larger systems. [LS10].

For many embedded systems, the safety and reliability aspects are very important. Consequently, as the complexity of such systems increases, the task of detecting or analyzing failures of the system becomes increasingly difficult [KGF07, KLM03]. The process of analyzing failures is necessary, in order to trace the reasons that lead to a specific hazard of the system's life. Subsequently, many techniques have been proposed to trace the failure propagation paths amongst the set of cooperating components in the failure. The Fault Tree Analysis (FTA) technique is one of the most common modeling techniques, which helps in understanding failure mechanisms in embedded systems. FTA emphasizes the logical relations amongst the set of basic failures which could lead to a specific undesired state of the system. That is called the Top-Event (TE) state [KGF07, KLM03].

As a descendant of this technique, the Component Fault Tree (CFT) concept provides an extension of the FTA concept by introducing additional information about the system's structure. CFT is used to depict the failure scenario in complex embedded systems as a directed acyclic graph [KLM03, KGF07]. Usually, results of the CFT model of complex embedded systems could be very large. Therefore, the process of analyzing and tracing the failure paths in such complex models becomes a tedious task. In this work, we aim to give an overview of the set of components involved in a given failure scenario. Moreover, we aim to provide a mean for visually comparing the components with respect to the system safety aspects. This work also provides an insight into the ways in which components might contribute to an undesired system failure. To achieve the above mentioned goals for our framework, we map extracted information from the CFT to visual elements in our final layout. As

described above, the underlying embedded system consists of hardware parts, originally modeled as in a CAD model, which can be intuitively visualized in a 3D world.

We accomplish this by providing a Virtual Reality (VR) environment that allows exploring certain safety scenarios in the 3D model of the hardware components. The 3D model is integrated with an *abstract representation* that describes the hierarchy of failure relations among cooperating components in the current failure scenario. We visualize the abstract representation using the radial layout algorithm [Ead92], because the underlying CFT model is inherently depicted as a tree in most of the cases. We use the radial layout algorithm to help in utilizing the screen space more efficiently and for showing the hierarchical relations of the failure among the infected components. Our work attempts to provide an effective means of identifying and classifying system artifacts like software and hardware components, with respect to their participation in safety critical system outages. Of particular interest in this work is the development of visualizations, which are able to support the analysts in answering questions like:

- What are the critical system artifacts and how do they contribute to a system failure?
- How severe is their influence in terms of probability of occurrence?
- Which artifacts influence each other and how strong are those influences?
- Do the causes occur within SW or HW components or in both?

Our goal is to support common analysis by providing representations integrating different views about safety critical embedded systems. The rest of the paper is organized as following: Section 2 provides a list of the related work. Section 3 provides the background information about the application domain. Section 4 highlights the proposed ESSAVis framework in details. While in Section 5, we present the results of the brief evaluation for our proposed framework. Finally, we conclude the work in Section 6.

2. Related Work

The visualization step is considered as one of the most important steps increasing the cognitive level of users. This is due to the facilities and the techniques which are offered to support the user in getting the correct insight about the current situation of the data only by watching or interacting with it. As mentioned earlier, this work's goal is the diagnosis of system status faster and more accurate by providing two linked-views of the CFT model and the 3D CAD model, where they are attached to each other, in a 3D environment. Therefore, we categorize the related work into two categories: Sub-section 2.1 presents the related work with regard to visualizing the FT and the CFT in general while Sub-section 2.2 presents the related work with regard to visualizing the data into the 3D world.

2.1. Visualization in Safety Domain

Depicting the failure relations among the system parts is critical to understand the failure mechanism. Many existing tools try to visualize this by showing a tree structure of the failure. Most of these tools visualize the fault tree in 2D representations like ESSAREL [Sof12], UWG3 in [KLM03], and Cecilia OCAS in [BBC*04]. In these tools the node-link diagram is used to show the relations between the infected system parts. While the simple primitive shapes are used to show the components of the Fault Tree (FT) e.g. small circles to represent basic events and a small rectangle to show the gate (the logical connector) between two basic events. These tools also use color or/and the text to depict other information such as the gate type. These kinds of tools are useful to model the failure relations between the system parts, but they do not provide options for analyzing the failure path or the set of the critical parts in the underlying system. In spite of the facility of editing and modifying the FT structure in these tools, generally they lack the abilities of analyzing the FT itself and the presentation of an overall view of the current failure mechanism. However, amongst them the ESSAREL tool provides a textual description of some safety aspects of the FT (like, the set of the minimal-cut-set), which is unreadable in most of the cases due to the data size and the file format.

There are some other tools that analyze the safety aspects of software systems. For example PLFaultCAT [DL06] has been used to reduce the effort needed to safely reuse the software requirements and to customize the product-line for software fault tree analysis (SFTA) during product-line engineering. [YKL*12] a visualization system has been proposed to support the engineers in identifying proper solutions for the system visually. The proposed visualization integrates the fault tree and a plot which represents the cause-effect relationship between the solutions of the system failures and the resulting risk reduction of the system. Moreover, the authors tried to associate the component fault tree view with the plot diagram to allow maintaining helpful context information about the current state of the system.

An interesting visualization system, called SViT (Safety Visualization Technique), was proposed by [KSZ09] showing the status of the digital home. SVit helps the homeowner to know the current safety level of the home and the reasons behind this level. It provides multiple interfaces for each device at home. It also automates the safety computation process of the safety information for each device. This helps homeowners to determine the required safety levels of their homes.

2.2. 3D Visualization Approaches

The 3rd dimension has been utilized in many information visualization techniques for many purposes. For example, the hyperbolic layout has been proposed by [Lam96,LRP95,

Mun97, Mun98, MB95] to show the information structure in the 3D world, while [CM00] presented the cone tree layout method as a pure 3D layout algorithm for hierarchical structures. Another example is semNet, which can be found in [FPF88]. These examples were based on using the animated 3D visualization and the lightening to show the depth perception. Collins et al. [CC07] provided many examples for showing the usability of the third dimension in encoding different aspects of the data. They achieved it by isolating a subset of a 2D graph representation to a separate layer, thus a 2.5D would be created. In this case, they proved that the third dimension could be used to encode some aspects of the data instead of the relations between nodes. The third dimension in [BDS03] has been used to show the evolution of the graph over time where each separated layer is used to represent the graph at a certain time step. Eades et al. [EF96] also used the third dimension to depict the hierarchical nesting of clusters in the graph using a set of transparent layers.

The current state of the art in graph visualization shows some interest in using the 3D layout in few cases. For a general introduction to graph visualization techniques, we refer the reader to the work of Herman et al. [HMM00]. In our work, we use the third dimension to represent the hardware components of the real model of the underlying embedded system, linked with the abstract representation of the safety scenario, which shows the important safety information of the current Top-Event. Initially, the abstract representation is laid out in a 2D plane that is integrated in a 3D world with the 3D model of the underlying embedded system.

3. Background and Motivation

The Fault Tree Analysis (FTA) technique is a top-down approach. It supports modeling the safety scenarios of complex systems [KGF07, KLM03] in which the relations between the set of faults (basic events) leading to a specific undesired event (Top-Event) are depicted graphically. The basic event is defined as one of the reasons causing the current Top-Event with a certain failure probability to occur [KLM03]. In the Fault Tree model, sets of basic events are represented as the tree leaves. The FTA depicts logical relations between sets of basic events that lead to a specific Top-Event, as shown in Figure 1. The Component Fault Tree (CFT) technique is a safety and reliability modeling technique that supports a hierarchical decomposition of large systems. The difference between CFT and FTA is that FTA offers a decomposition of the system into modules, which is a breakdown process of the system regarding the hierarchy of failure influences rather than the system architecture. Whereas, CFT represents each component by an extended fault tree that contains input and output ports besides basic events and gates. By connecting the set of components via set of ports, components can be integrated into a higher-level system model [KGF07, KLM03].

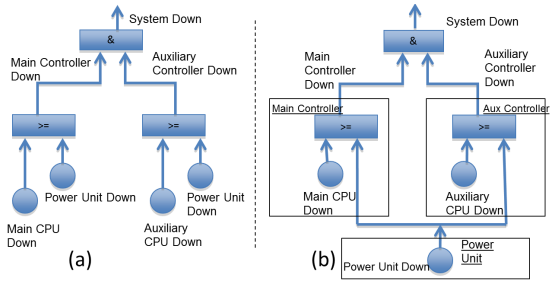


Figure 1: Fault Tree Concept vs. Component Fault Tree Concept.

The CFT modeling technique is useful for relatively large systems because all components can be developed independently and then can be saved in separate files, which provides a way to integrate the components and reuse them independently. The CFT extends the tree structure of the FTA into a Directed Acyclic Graph (DAG). This feature gives the ability of producing compact representation for large failure scenarios [KGF07, KLM03]. Although the safety scenario in our framework can be modeled as a DAG structure; currently, our prototype provides only the basic functionalities of the system. Therefore, at the moment we model only the tree structures as they are often used to model safety scenarios in such applications. However, for future work we would like to visualize more realistic models that can be visualized as a DAG structure to reflect the CFT concept.

Visualizing the safety aspects of embedded systems is comparatively a new field. As the complexity of the underlying embedded system is increased the size of corresponding fault tree is also increased, which makes it difficult to handle the failure detection process. Moreover, it makes maintenance of the underlying system more costly and time consuming. The information visualization can play an important role in speeding up the system developing process because it eases the step of finding the important information from both the system and the user's perspectives. Also, it helps in reducing the errors made by human in searching the relevant information. For example; a study, conducted by IBM and Industry Studies [BV10], shows that 30% of people time is wasted on looking for the important information. Moreover, the visualization helps in interacting with the data and automates the steps of information extracting, which helps in detecting the relevant information more accurately. Furthermore, the visualization can provide the integration between different parts of the complex system. Like in our case, it helps in integrating the hardware part and the abstract safety view, which is useful in conveying the whole story to the end users.

Due to the interaction between safety analysts and systems engineers, the task of analyzing the system is an iterative task. We summarize this process in the following steps: first, system engineers and safety engineers built the FT for

a specific Top-Event in a way that reflects the system design; then the relevant information is analyzed from the safety engineers' perspective. Whereas in our case, we are interested in showing the most critical components of the system from both the hardware and the software perspectives; then system engineers react to fix the infected components based on the analyzed output from the previous step; and finally the corresponding FT is updated accordingly as it is shown in Figure 2.

4. The Embedded System Safety Visualization (ESSAVis) Framework

The proposed ESSAVis framework provides a two-views layout where the first view shows a 3D model of the hardware part while the second view shows an abstract representation of the underlying safety scenario as shown in Figure 3. To achieve this, we analyze the safety scenario that represents a critical status of the underlying embedded system. In this work, we analyze the critical situations of a robot, called RAVON [Pro10]. The safety scenario is modeled using the Component Fault Tree (CFT) technique. We analyze this safety scenario to produce the required data structure we need in computing the set of critical components for the scenario. This set of critical components consists of all those components that have a high probability to fail. For each component, we compute the criticality based on many factors such as the number of basic events and their probabilities or the number of failure paths that pass through each component. We visualize the abstract information using the Radial Layout algorithm [Ead92]. The main task of the abstract representation is to reveal the failure relations between the collaborating components in a specific hazard of the system at a certain time. In this abstract representation, the set of nodes represents components set while the set of edges reflects logical relationships among components of the CFT model (as shown in the left-side of the Figure 3).

ESSAVis also provides a 3D model of the hardware part of the underlying embedded system. For example, the 3D model of RAVON is dedicated to convey right impression of the real physical model of hardware components. To do this, ESSAVis parses VRML files of physical components of RAVON and assembles them together to produce the full model of RAVON. Then the two views are rendered in a 3D world. The ESSAVis framework synchronizes between the two-views to let users trace the set of changes in both views faster. For example, we highlight infected components in both views in red color according to the underlying safety scenario (as shown in both views of Figure 3).

4.1. The ESSAVis Pipeline

To achieve the desired goals, we developed a tool that reads a safety scenario of the underlying system at a certain time. Then we extract the relevant safety information to help

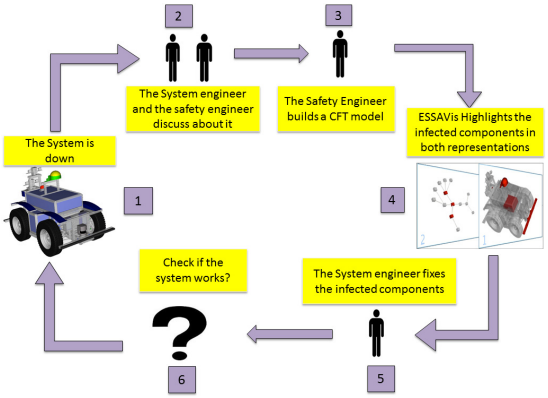


Figure 2: The embedded systems safety analysis cycle using ESSAVis.

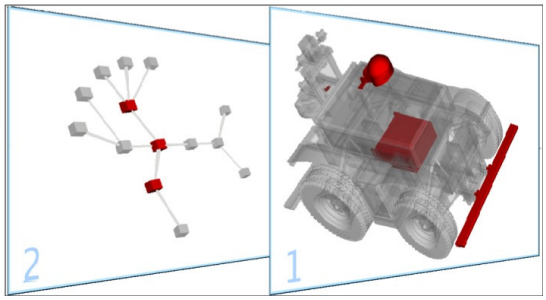


Figure 3: Integrated view showing infected hardware components (right) and the corresponding failure path in the abstract layout (left).

safety engineers in defining the set of critical components. As mentioned earlier, the criticality of components is based on many factors, such as the number of failure paths passing through each component or the accumulated probabilities of the contained basic events [KGF07]. We generate the required safety scenario using a specialized safety-modeling tool, called Essarel [Sof12]. The Essarel tool supports quantitative analysis of safety, availability, and reliability of embedded systems. Also, it provides the feature of exporting the model into an xml file, which contains a description of the safety scenario, or into the component fault tree model and corresponding required safety attributes. For example, the set of minimal-cut-set in the current scenario is included together with the information about the related set of components and failure connections among them. This xml file is the input of our framework, where we extract the safety data related to each component and then arrange the set of components to be ready for the visualization process.

For visualizing data, we use Vrui package [Kre12], which is a toolkit to provide a virtual reality development environment. Vrui package shields the application developing process from the particular configuration of the VR environ-

ment. Moreover, it works in many different hardware platforms. Therefore, it is possible to render the application in many different environments such as 3D displays, PowerWall displays, or CAVE systems. This feature increases the scalability and portability of our framework into different environments, which also helps in showing the depth cue in the 3D world more precisely. We render the 3D model of the RAVON robot using the Vrui package as it supports the construction of 3D world using a scene graph data structure. Then we link the safety information, generated using the Essarel tool, with the 3D model of RAVON. This allows us to highlight directly infected components in the hardware model, which helps system engineers to identify faults quicker than the traditional way of the textual description. Figure 4 shows how the different toolkits in our framework are interacting together.

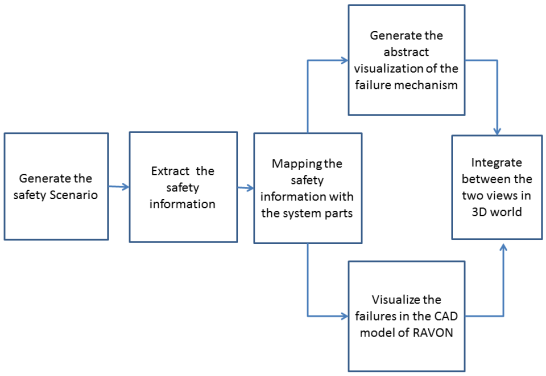


Figure 4: The ESSAVis pipeline.

4.2. The Two-Linked View

In this subsection, we describe the proposed integration model between the two data representations. As mentioned earlier, ESSAVis synchronizes between the abstract layout and the hardware model in a 3D world. Initially, both representations are visible side by side as shown in Figure 5. ESSAVis gives the possibility to toggle between different views on-demand. It provides three different arranged views to link between the two layouts. The initial one is the *side-by-side view*, in which both representations are displayed next to each other with equaled sizes for both and at the same distance from the viewer. Users can change this by selecting the toggle view option from the ESSAVis main menu. The available options are the *big-small view* option and the *layered view* option.

The big-small option provides the facility to users scaling up of one of the representations while scaling down the other one, as it is shown in Figure 7. This option is useful in few cases, e.g. safety engineers, who are interested in knowing more about some failures in the system, can decide to view the abstract view larger than the 3D model (as shown in Figure 7) or system engineers can go for the opposite case. While in the third view, the depth is used to show importance degree of the representation. Initially, the two-views are arranged in the same depth value that is zero in our case (as shown in Figure 5). However, this arrangement can be modified by changing the view through the layered-view option. For example, if user is interested in the hardware model, the view of hardware model pops up to a layer closer to the viewer point and the second view will be stacked behind the current active view in the first layer as it is shown in Figure 3. In this example case, the second layer represents the abstract representation while the first layer represents the 3D model of RAVON robot respectively. The user can toggle between the different views on demand. If he/she is interested in the hardware model then this view will be the dominant in the screen. The user then can change the visual cues of this view accordingly, e.g. changing the transparency, size, or orientation. The toggle view option is provided via one of the main menu options. Currently, users can interact directly with the prototype using a pop-up menu that includes the basic functionalities of the framework. This menu has a set of sub-menus that are used to list the extracted information from the current safety scenario. These submenus allow users to map the needed information with the actual 3D hardware components and components in the abstract visualization (as shown in Figure 6).

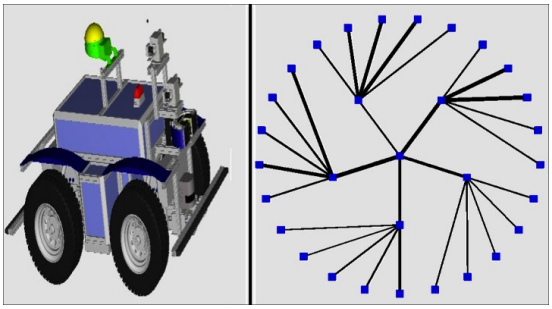


Figure 5: The initial integrated layout of hardware model and the abstract representation.

The menu in the prototype is triggered by clicking the right mouse button at any free space in the 3D world. ESSAVis also supports the basic interaction techniques in the 3D world, like zooming, panning or rotation. One of the options that are provided by the Vrui itself is scale bar option, which helps user to indicate the size of the object in the 3D scene and quickly adjusts the display's zooming factor to a standard ratio, like 1:10 or 50:1. The zooming ratio is between the physical space length and the navigational space length of the object [Kre12]. As mentioned earlier, the abstract visualization is used to encode relations between the infected components in the corresponding safety scenario. For example, in Figure 8, we show an abstract representa-

tion of a component fault tree consisting of 30 components. The top-event node represents the main hazard in the system that is depicted as the root of the tree, which is the central node in the radial tree layout. The leaf nodes are those components that have the set of basic events, which contribute in a specific Top-Event.

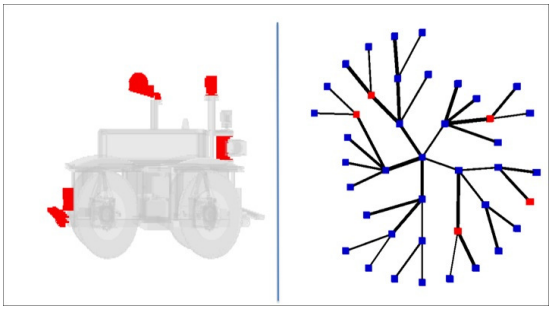


Figure 6: An integrated *side-by-side view* showing the correlation between the infected components in the hardware model and the abstract representation.

ESSAVis provides a list of options for analyzing the safety aspect of the corresponding safety scenario. This helps in answering the set of safety engineers' questions, already mentioned in Section 1. For example, users can highlight infected components in both representations by selecting the component name from the submenu. The current prototype also allows users to get information about each involved component in the scenario such as basic events for each component. Figure 8a shows mapping between the selected basic events and the related components. This function is crucial in helping safety engineers in finding the criticality of the component according to the probability of basic events that it has. It is worth mentioning that each basic event has a probability value, which indicates the occurrence probability of such an event. The current prototype provides information about the basic event probability via a label containing the value that is attached to the basic event name in the list. We intend to use other visual cues to encode the probability value to help users getting more insight about such an important metric from one glance.

One of the important aspects about our framework is showing of the Minimal-Cut-Set (MCS) information. The MCS indicates the least number of Basic Events that are required to cause a top-event in the system. Showing the MCS is a strong indicator of the infected components influence in the current scenario. ESSAVis provides a mechanism to show the relations between lists of MCSs in the current scenario with correspondence components in the 3D hardware model. This facility aids safety engineers in diagnosing the critical components in the current scenario. We give users the ability to interact directly with the list of MCSs and to select one of the entries of the underlying scenario. Due to user selection, the set of infected components in the current

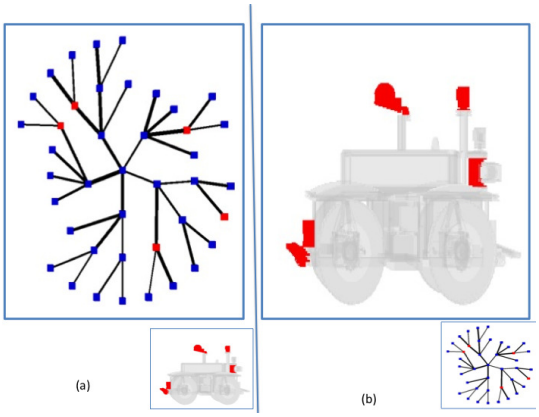


Figure 7: A big-small view to integrate the different views.

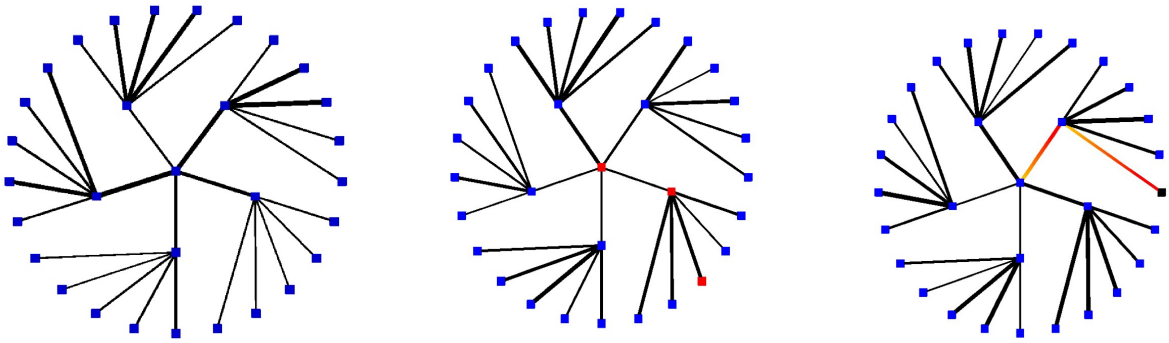
scenario are highlighted in red color. This is applied on both views, the hardware view and the abstract view, as shown in Figure 8b. This helps system engineers to diagnose the system status quickly; hence, it speeds up the maintenance process of the underlying system.

Fault trees for large systems are considerably complicated and difficult to read. Therefore, it is quite harder to achieve in them the process of tracing the path failure between any component and the root component. Our prototype provides an option to show paths between the infected components that collaborate in a Top-Event in the system based on the current configuration. This feature provides the safety engineers the ability of finding out the set of influenced components through the selected ones. This option facilitates them an insight of those components that might fail next according to the current selected components. ESSAVis also provides the facility to animate the failure path among the infected components by selecting one component from the abstract layout to the top-event node. As a result, the set of edges, which contributes in the selected failure, are highlighted in orange color. We use a simple animation for showing the propagation of the failure path. This case is illustrated in Figure 8c. Beside the failure propagation path, edges between the set of components are also used to show the criticality of the path between two components. Different thickness are used to depict the path criticality, as shown in Figure 8a.

5. Framework Evaluation

A brief evaluation was conducted for the proof of the validity of our proposed ESSAVis solution.

- *Environment settings:* we carried out the evaluation experiment using a Zalman stereoscopic desktop display equipped with polarized 3D glasses. The participants in the evaluation were from two different user groups. The first group consisted of 6 safety experts while the second group consisted of 3 robotics experts. A safety scenario,



(a) (b) (c)

Figure 8: (a) Abstract view for a safety-scenario in a certain moment. (b) Red colored components represent the set of infected components by a specific MCS. (c) The failure path depiction between the selected component and the Top-Event and the edge thickness indicates the path criticality.

consisted of 40 components aligned side-by-side to the 3D model of RAVON, was presented to both groups.

- *Hypothesis:* we assumed that the ESSAVis solution increases the detection process speed of infected components of the current scenario. The detection speed is compared with the detection speed of the traditional tools like Essarel tool.
- *Tasks:* the participants were asked to answer the following questions:
  - Detect the infected components in both representations; the abstract representation and the 3D model representation of RAVON using our prototype.
  - What are the basic events that are related to a specific component.
  - What is the failure path between a specific component and the top-event node.

The preliminary results of the evaluation show the usability of our framework especially from the system engineers' side that work with the hardware model of RAVON robot. They agreed that our given framework could help them in defining and detecting the infected components in the current state faster than the textual description of the failure. We discuss the accuracy in terms of users' groups to accomplish the task successfully. The accuracy of system engineers' group answers was 94% with average time of 24 seconds. Also, it was easy for them to infer those components that have the possibility of failure based on the current configuration. While the safety group mentioned that our given prototype could help them in detecting the relations between the infected components beside in tracing the failure propagation paths among the infected components. 5 out of 6 safety group participants were able successfully to find out the set of most critical components approximately within 18 seconds with accuracy reached to 85%.

The results also indicate the need of training for both groups to read the new representation and the 3D environment that is used for the rendering process. Beside this, we found that some visualization aspects need to be optimized more, especially the path tracing option as shown in Figure 8c. In this regard, participants were not able to map the changes in the abstract view with respect to the changes in the 3D model. So, there is a need to provide a solution for tracing the changes between the different views of our framework. Moreover, one limitation of the current prototype is performance of the rendering speed of the 3D model of RAVON. It is comparatively large model and consists of 395 vrml files. Therefore, there is need to apply some filtering techniques to such a large model for speeding up the rendering in order to increase the response time of ESSAVis.

6. Conclusions

We presented a framework, called ESSAVis, to visualize the failure mechanism of embedded systems. ESSAVis shows the infected components in parts of any embedded system, the hardware parts or the software parts. Based on a specific safety scenario at a certain time step, it visualizes the hardware components as 3D objects and highlights the infected components in red color. For showing the failure mechanism among the different types of components in the underlying component fault tree, it provides an abstract visualization to depict the failure relations among the system parts. To prove the validity of the framework, a brief evaluation took place using a stereoscopic desktop display to show the real 3D impression of our work. Results show the intuitiveness and clearness of our technique over the traditional techniques that currently provide a textual description of the system's safety aspects.

As a future work, we intend to conduct a more extended version of the evaluation by comparing performance of our prototype with other available systems, which are currently used to analyze the safety aspects of embedded systems, with different scenarios and different data sizes. As mentioned earlier, the real data is more complex than the normal tree structures and has many different possible scenarios that are included in our current version of the prototype. Therefore, we intend to include those cases, such as DAG case, in our visualization. Moreover, as we mentioned the designing of an immersive environment for visualizing safety aspects of embedded systems; so, we also intend to evaluate the accuracy of the proposed framework in different environments, like using CAVE system, Power-Wall, and stereoscopic desktop display.

Acknowledgements: This work is part of ViERforES2 project and partially funded by IRTG 1131 (DFG) and BMBF. Many thanks go to the Software Engineering and Dependability Group at University of Kaiserslautern headed by Prof. Dr.-Ing. habil. Peter Liggesmeyer for their support. Also, we would like to thank the Robotics Research Lab at University of Kaiserslautern for their collaboration and support.

References

[BBC*04] BIEBER P., BOUGNOL C., CASTEL C., P. HECKMANN J., KEHREN C., SEGUIN C.: Safety assessment with altarica - lessons learnt based on two aircraft system studies. In *18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification* (2004), p. 26. 60

[BDS03] BRANDES U., DWYER T., SCHREIBER F.: Visualizing related metabolic pathways in two and a half dimensions. In *Graph Drawing* (2003), pp. 111–122. 61

[BV10] BOZZANO M., VILLAFIORITA A.: *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, 2010. 61

[CC07] COLLINS C., CARPENDALE S.: Carpendale s: Vislink: revealing relationships amongst visualizations. *IEEE Trans Vis Comput Graph 2007* (2007). 61

[CM00] COCKBURN A., MCKENZIE B.: An evaluation of cone trees. In *University of Sunderland* (2000), Springer-Verlag, pp. 425–436. 61

[DL06] DEHLINGER J., LUTZ R. R.: Plfaultcat: A product-line software fault tree analysis tool. *Automated Software Engineering 13*, 1 (2006), 169–193. 60

[Ead92] EADES P.: Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications, pp. 10 -36* (1992). 60, 62

[EF96] EADES P., FENG Q.-W.: Multilevel Visualization of Clustered Graphs. In *Proc. Graph Drawing, GD* (Berlin, Germany, 1996), no. 1190, Springer-Verlag, pp. 101–112. 61

[FPF88] FAIRCHILD K. M., POUTROCK S. E., FURNAS G. W.: SemNet: Three-dimensional graphic representation of large knowledge bases. In *Cognitive Science and its Applications for Human-Computer Interaction, R. Guinon, Editor. 1988, Lawrence Erlbaum: Hillsdale NJ* (1988), pp. 201–233. 61

[HMM00] HERMAN I., MELANCON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Trans on Visualization and Computer Graphics 6*, 1 (2000), 24–43. 61

[KGF07] KAISER B., GRAMLICH C., FORSTER M.: State/event fault trees: A safety analysis model for software-controlled systems. *Reliability Engineering System Safety 92*, 11 (2007), 1521–1537. 59, 61, 62

[KLM03] KAISER B., LIGGESMEYER P., MÄCKEL O.: A new component concept for fault trees. *Reproduction 33* (2003), 37–46. 59, 60, 61

[Kre12] KREYLOS O.: Vrui virtual reality toolkit, July 2012. "`http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html`". 62, 63

[KSZ09] KUMAR P., SUBRAMANIAN N., ZHANG K.: Savit: Technique for visualization of digital home safety. *ACIS International Conference on Computer and Information Science* (2009), 1120–1125. 60

[Lam96] LAMPING J.: The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages Computing 7*, 1 (1996), 33–55. 60

[LRP95] LAMPING J., RAO R., PIROLLI P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. of CHI '95* (1995), ACM, pp. 401–408. 60

[LS10] LEE E. A., SESHIA S. A.: *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 1 ed. Lee and Seshia, 2010. 59

[MB95] MUNZNER T., BURCHARD P.: Visualizing the structure of the world wide web in 3d hyperbolic space. *Proc. of the first symposium on Virtual reality modeling language VRML 95* (1995), 33–38. 60

[Mun97] MUNZNER T.: H3: laying out large directed graphs in 3d hyperbolic space. In *Proc. of InfoVis '97* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 2–. 60

[Mun98] MUNZNER T.: Drawing large graphs with h3viewer and site manager (system demonstration). In *Proc. of Graph Drawing'98, number 1547 in Lecture Notes in Computer Science* (1998), Springer-Verlag, pp. 384–393. 60

[Pro10] PROETZSCH M.: *Development Process for Complex Behavior-Based Robot Control Systems*. RRLab Dissertations. Verlag Dr. Hut, 2010. ISBN: 978-3-86853-626-3. 62

[Sof12] SOFTWARE ENGINEERING RESEARCH GROUP: DEPENDABILITY KAISERSLAUTERN UNIVERSITY, ESSAREL TOOL: Embedded systems sadety and reliability analyer, July 2012. "`http://essarel.de/index.php?site=backgroundtext`". 60, 62

[YKL*12] YANG Y., KELLER P., LIVNAT Y., LIGGESMEYER P., HAGEN H.: Improving safety-critical systems by visual analysis. *Dagstuhl Follow-Up series* (2012). 60