ESSAVis: A 2Dplus3D Visual Platform for Speeding Up the Maintenance Process of Embedded Systems

Ragaad AlTarawneh[1], Jens Bauer[1], Patric Keller[2], Achim Ebert[1]
[1]Computer Graphics & HCI Group, [2]Software Engineering: Dependability Group
University of Kaiserslautern
Gottlieb-Daimler-Str. 67663 – Kaiserslautern, Germany.
{tarawneh, j_bauer, pkeller, ebert}@cs.uni-kl.de

Modern embedded systems contain complex structures, as they are composed of many subsystems. Maintaining such systems requires collaboration between the engineers who designed them and the engineers who analyzed the failures in them. In this paper we present our proposed visual platform ESSAVis, a 2Dplus3D environment, to help both kinds of engineers in understanding the failure mechanisms of such systems. The goal of the proposed platform is to improve the understanding of the failure mechanisms in these systems and to reduce the communication gap between the system engineers and safety engineers. We describe the design process and the implementation of the 2Dplus3D visual platform, which was accomplished through a continuous end-user feedback. We designed a detailed evaluation study, in which we aim to measure the usability of our tool. The plan of the evaluation is designed with the help of the end-users to ensure that ESSAVis fulfills the expected goals in speeding up the analyzing process of safety aspects of embedded systems.

Embedded systems, Safety Aspects Visualization, Stereoscopic displays, Immersive environments

1. INTRODUCTION

Embedded systems are widely used in our daily life activities. Some examples of such systems are control systems in cars, airplanes, railroad crossings and even washing machines. Normally, embedded systems consist of both hardware and software components. Therefore, most of them have complex structures. Generally, they 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 [Lee and Seshia 2010].

In embedded systems, safety and reliability aspects are essential. Consequently, as the complexity of such systems increases, the task of detecting or analyzing failures of the system becomes increasingly difficult [Kaiser et al. 2003]. The process of analyzing the failure mechanisms is necessary in order to trace the reasons that lead to a specific hazard of the system-life. Consequently, many techniques have been proposed to trace the failure propagation paths amongst the set of cooperating components in the failure. Normally, embedded system engineers design the structural relations between the components, and the collaboration of components to achieve a specific task. On the other side, safety engineers design a Fault Tree (FT) model to trace failure propagation paths (see Figure 1), as they care about the failure relations between the system's components. Their goal is to find the set of critical components in the system, and the set of basic events that stimulate the top-event to occur. Hence, they are interested only in the failure relationships perspective between the system parts.

The maintenance process of embedded systems requires collaboration between the two engineering groups. The required collaboration is a multi-step process, as explained in Figure 2(b). However, one obstacle arises often in this collaboration is the difference of perspectives of the system by both groups. This hinders the maintenance of system, which is more costly and time consuming. The goal of this work is to reduce the communication gap between these two engineering groups by providing a common 2Dplus3D visual platform, where both groups can communicate and interact with each other through an interchangeable visual view. Moreover, the provided visualization helps the viewers in understanding the system failure

mechanism more clearly. For this, our solution provides a list of options to query about the infected components, the list of the basic events and the list of the minimal cut sets. Moreover, it maps between the safety information and their location in the 3D model of the targeted system. This paper is organized as the following, in Section 2 we present a list of some related work. While in Section 3 we discuss the design process of the ESSAVis toolkit. Moreover, in Section 4 we present the ESSAVis options, finally, we conclude in Section 5.

2. RELATED WORK

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 [Software Engineering Research Group: Dependability, Essarel Tool 2012], UWG3 in [Kaiser et al. 2003], and Cecilia OCAS in [Bieber et al. 2004]. In these tools the node-link diagram is used to show the relations between the infected system parts. 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. Some other tools, that analyze the safety aspects of software system is the PLFaultCAT [Dehlinger and Lutz 2006], which 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. In [Yang et al. 2012], 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 that 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 [Kumar et al. 2009] 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.

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. In this work we discuss the design process of ESSAVis, where we propose a new approach to integrate between the fault trees and the component parts of the system in one view. The goal is to give more information about the system status such that both targeted engineers may collaborate together.

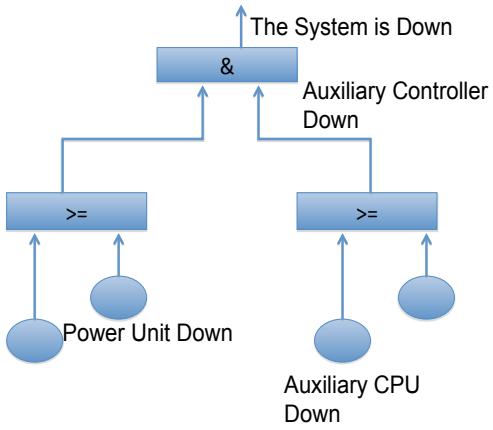

Figure 1: *The Fault Tree (FT) representation.*

3. DESIGN PROCESS

The first step was to understand what are the questions that both engineering groups wanted us to answer and in which form. The user-requirement can be listed as the following:

- 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?

Safety engineers normally deal only with the FT model (as shown in Figure 1), which is a 2D representation of the failure mechanism in the system at a certain time. They request to provide a 2D representation of the FT model, alongside the visual cues of important safety information embedded in the 2D representation. However, the system engineers' demands were completely different. Normally, they deal with the hardware model of the system; hence, they care only about the structural information of the system components. More precisely, they required to map the safety information with the infected component(s) in the 3D model and to provide the interaction facility with the 3D model of the system directly.

For our case study, we used safety scenarios of the RAVON robot [Proetzsch 2010] that can detect obstacles on the road. The safety engineers were responsible to provide the safety scenarios of the RAVON according to the system engineers' feedback. We concluded from both meetings that an integrated solution (2Dplus3D) would be a better solution for both groups. Our idea was to provide both groups' perspectives in one platform in order to give them the ability to collaborate in a more natural way.

3.1 Preliminary Ideas

The first solution was to provide a pure 3D representation, integrating both the safety information and the system information (as shown in Figure 3). This pure 3D representation gave the system engineers the ability to find out the infected component(s) easier and faster, as it highlights them and gives a textual description about them directly. However, the safety engineers didn't show any interest in this solution, as it does not provide them the relevant safety information and the failure relationships between the components. Moreover, it does not show the failure path between the components, as it is hard to visualize this in a 3D model due to the cluttering problem that might appear. Therefore, we arranged another focus group meeting with both groups to discuss the possible solution that would satisfy both of them. We came up with an integrated-views solution, which was about providing two views one for each data set, both views are arranged side-by-side in the initial setting.

The integrated-views solution offers two views, as shown in Figure 5, where the first one provides the 3D model while the second one provides an abstract representation of the failure mechanism in the system at the specified time. This solution also offers the possibility to switch between the two views on users' demand. This gives both groups the ability to collaborate through a common platform to achieve the required tasks together. We presented this solution and a small mock-up to both groups. Both groups agreed on it as it answers their demands in the required application. After the idea was finalized, we specified the work steps and the set of goals for the proposed 2Dplus3D integrated visual platform. In the following subsections, we briefly explain each step and the corresponding goals.

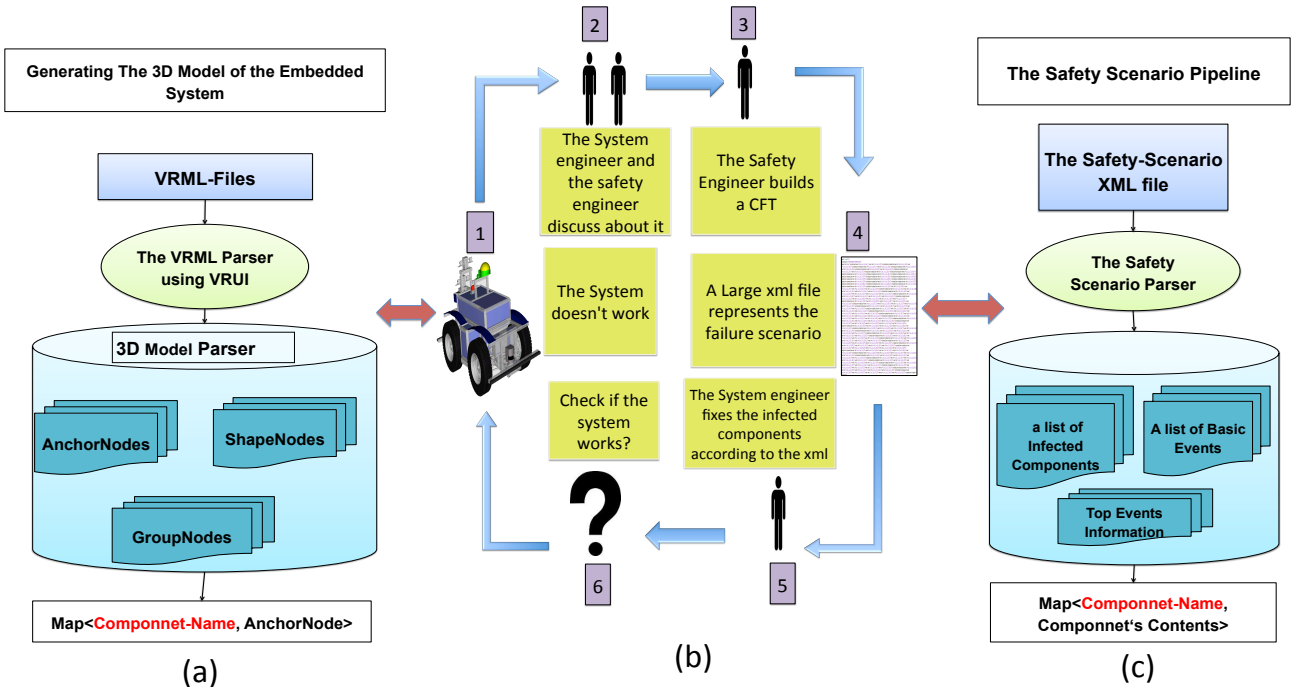

Figure 2: *(a) Generating the 3D model pipeline (b) The safety cycle analysis (c) The safety scenario pipeline, the two red arrows indicate the connections between the safety cycle and the two pipelines*

3.2 Formulating the Safety Scenario

The Fault Tree Analysis (FTA) technique [Kaiser et al. 2003] is one of the most common failure modeling techniques (see Figure 1), which helps in understanding the 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. This state is called the Top-Event (TE) state [Kaiser et al. 2003].

The first step was to find out the format of safety scenarios, designed by safety engineers, and the extraction of the relevant information from them. The safety engineers use the Essarel tool [Software Engineering Research Group, Essarel Tool 2012], an editor for the FTs. In general, they used it for designing these safety scenarios. They model the failure mechanism by dragging and dropping a set of geometrical shape representing the main part of the fault tree. The result of this process is a large xml file representing the failure mechanisms (see Figure 2(c)).

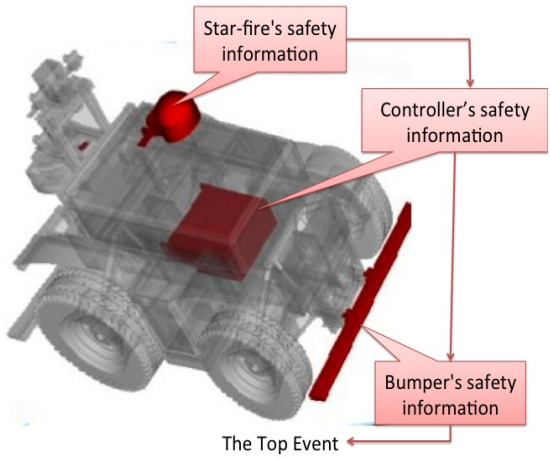

Figure 3: *The first idea model, where the safety information embedded with the 3D model of RAVON.*

3.3 Building the 3D Model of the System

The system engineers were interested in diagnosing those possible situations in which the RAVON could fail [Proetzsch 2010]. Their demand was a visualization of the infected components of a specific safety scenario within the RAVON's 3D model. We build the complete 3D model of the RAVON robot according to the system engineers' feedback through parsing the VRML (Virtual Reality Modeling Language) files that were representing the RAVON's components (see Figure 2(a)). This step was done through the help of VRUI framework [Kreylos 2012].

The 3D model of RAVON consists of 394 VRML files, where each file represents the geometry of one hardware component in RAVON. Also, it has one file to show the structure of the whole model. This file represent the root file that we used to parse the 3D model information using the Scene-Graph package in VRUI. From this root file we extracted the structural information between the hardware components in the system. The result of this step is a 3D representation of the hardware model of the underlying system, which is in our case is RAVON.

After we built the 3D model of RAVON, we mapped the safety information with the Real-Parts of the system to establish a link between the different types of data in our platform. As it mentioned that the output of the safety analysis process is an xml file that rep- resents a critical situation of RAVON. For visualizing the infected component within the 3D model of RAVON, we unified the components' names, used by safety engineers in the safety scenarios, with the anchor nodes' names in the corresponding VRML files. This step was done through a focus group meeting with both groups in order to ensure the compatibility of the safety scenario with the underlying 3D model.

To highlight the infected components in the RAVON's 3D model (see Figure 3), we traversed down the scene-graph data structure of RAVON. We found those anchor-nodes that represent the geometry of the infected components and highlight them using the red color and represent the non-infected components as transparent elements.

3.4 Visualizing the Safety Scenario

We used the orthogonal layout algorithm to visualize the failure mechanism; this layout utilizes the screen-space more efficiently and shows the failure structure more clearly. Moreover it reduces the edge-edge crossing in the final layout, for more details pleas refer to [Hermann et al, 200]. We call it the abstract layout (Figure 4). The main role of this layout is to show the failure relations between the collaborating components in the current safety scenario. Moreover, it also shows some safety information important to safety engineers such as the basic events for each component, the criticality degree of each component, and the failure propagation path between system components.

Users can interact directly with the abstract information to query about some relevant information of them. We provide a mouse interaction tool that allows users to select one node from the graph representation in order to show the corresponding failure path to it. Also, the texture was used to show the component type either software or hardware, and the animation was used to show the direction of the failure.

4. THE ESSAVIS TOOL

ESSAVis provides the integrated *2Dplus3D* view. This integration step was necessary because it synchronizes the two views, i.e. the 3D model and the abstract layout of the safety scenario (see Figure 4). The default case in ESSAVis is the integration of the two views in one screen and aligning them side-by-side.

This option gives both views the same degree of importance. However, this is not the ideal case in real life as both groups care more about their interested perspective rather than the other one. Therefore, the ESSAVis tool also provides two other options to arrange the views in the space according to the user's personal interest. The second option is called the Big-Small view, in which one of the views is bigger than the other (as shown in Figure 5). Users can toggle between them in the real time according to their interest. This option is the preferable one compared to the other two options, as it renders the two views in the space while it focuses on the bigger one and keeps the context as well by displaying the smaller view.

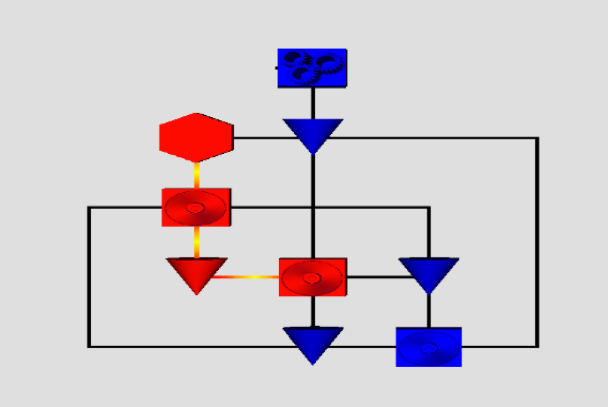

Figure 4: *The abstract representation of the safety scenario produced using the orthogonal layout algorithm.*

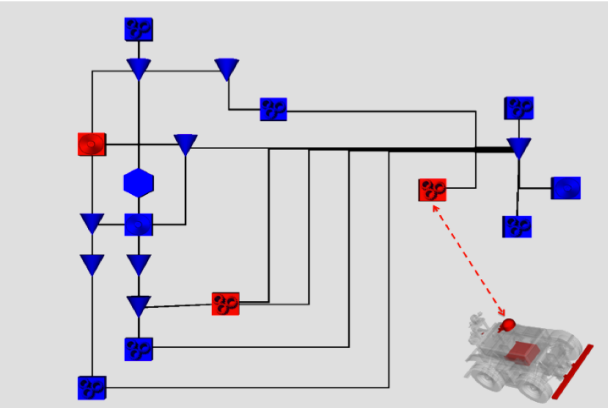

Figure 5: *The Big-Small view integrates the 3D model and the abstract representation of the safety scenario.*

The third option is inspired by the idea of having a 3D world for the ESSAVis. For this, ESSAVis provides a layered option in which both views are arranged into a stack of layers. In this case, the important view is displayed at the top of the stack and the other one is rendered at the background, i.e. at the bottom of the stack (as shown in Figure6). The layered-views option is provided based on the assumption of using a 3D display to present the result, as we use the depth cue to highlight the importance degree of the view. In this case, the closer view is more important for the viewer. Additionally, ESSAVis provides two possibilities for the users to interact with it. The main Menu, which pops up by clicking the right mouse button, provides the main options. While with a direct Mouse interaction, users can also interact directly with the abstract layout to observe the failure path between the required components.

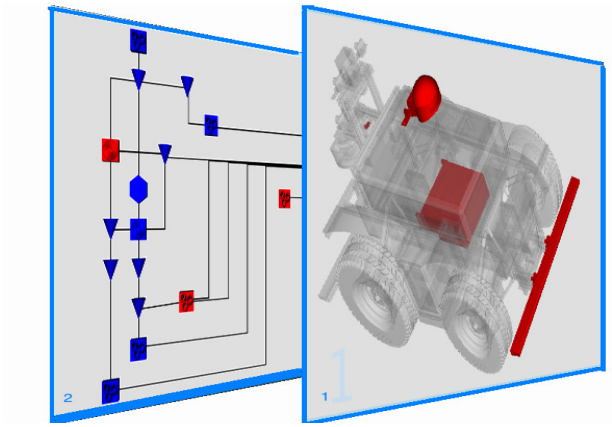

Figure 6: *The Layered view integrates the 3D model and the abstract representation of the safety scenario.*

The next step for ESSAVis is to test the feasibility of its use in practice. To do this we are planning a detailed controlled evaluation, where we will ask the targeted end-users (safety experts, robotic experts) to perform our designed study. The evaluation is designed in the collaboration with the safety engineers and the system engineers. The goal of this study is to find the possible situations where the ESSAVis tool is useful for those engineers to perform their tasks. The evaluation tasks are designed to evaluate the following aspects:

- The usability of the tool, to measure how much the ESSAVis tool is usable to achieve the normal tasks of the safety engineers to diagnose the system safety level. The ESSAVis is assumed to perform more effective and efficient in detecting the safety level of the underlying embedded system.
- The collaborative aspect of the ESSAVis, to measure this aspect we intend to render ESSAVis using a large display augmented with tablet devices so the targeted users can work together to analyze the safety scenario. Moreover, they can interact

directly with the system and query about the required information.

- The acceptance ratio of the tool, to measure this we tend to ask the users via a questioner or a likert-scale questioner to show the level of the satisfaction of the user after using the ESSAVis tool.

The current evaluation plan of the formal study is designed with the help of the end-users of the ESSAVis tool. The set of the study goals also were discussed with the end-users in a closed-focus meeting. We also discussed the possible expected tasks from the safety engineers' perspective.

We tend to perform the evaluation in two stages the first stage is a pilot case study to show if the designed planned is feasible and to measure our tool capacity in performing the expected tasks.

According to the pilot study results, we plan to perform a refinement of the tool then we conduct the second stage, in which we perform the formal evaluation study with a larger number of users to undergo the specified set of tasks.

5. CONCLUSIONS

In this work we presented a core of a tool called ESSAVis, which is a 2Dplus3D visual platform for visualizing the safety aspects of embedded systems. It provides a common environment for system engineers and safety engineers in order to reduce the communication gap between them and to improve their understanding of failure mechanism in the underlying embedded system.

Currently, the tool provides many options to analyze the safety scenarios, those options were designed in the help with the end-users of the tool. As a future work, we aim to conduct a formal evaluation study to measure the feasibility and the usability of the tool as it is described in the previous section.

The next step is to provide another visualization of the software architecture of the embedded system and to synchronize between it and the other two views. This step is required to increase the collaboration between the end-users, as they required this to understand the software structure of the system and to detect the responsible software components of the failure in the system, and to show their internal structures. Additionally, the software metrics of the software components will be added to give more insight about the criticality value of detected components.

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 their support.

6. REFERENCES

Bieber, P., Bougnol, C., Castel, C., P. Heckmann, J., Kehren, C., and Seguin, C. 2004. 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, 26.

Dehlinger, J., and Lutz, R. R. 2006. Plfaultcat: A product-line software fault tree analysis tool. Automated Software Engineering 13, 1, 169–193.

Herman, I., Melancon, G., and Marshall, M. S. 2000. Graph visualization and navigation in information visualization: A survey. IEEE Trans on Visualization and Computer Graphics 6, 1, 24–43.

Kaiser, B., Liggesmeyer, P., and Mäckel, O. 2003. A new component concept for fault trees. In Proceedings of the 8th Australian workshop on Safety critical systems and software - Volume 33 (SCS '03), Peter Lindsay and Tony Cant (Eds.), Vol. 33. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 37-46.

Kreylos, O., 2012. Vrui virtual reality toolkit, July. "http://idav.ucdavis.edu/~okreylos/ResDev /Vrui/index.html".

Kumar, P., Subramanian, N., and Zhang, K. 2009. Savit: Technique for visualization of digital home safety. ACIS International Conference on Computer and Information Science, 1120– 1125.

Lee, E. A., and Seshia, S. A. 2010. Introduction to Embedded Systems - A Cyber-Physical Systems Approach, 1 ed. Lee and Seshia 2010, 978-0-557-70857-4.

Proetzsch, M. 2010. Development Process for Complex Behavior-Based Robot Control Systems. RRLab Dissertations. Verlag Dr. Hut. ISBN: 978-3-86853-626-3.

Software Engineering Research Group: Dependability in Kaiserslautern University, ESSAREL Tool, 2012. Embedded systems safety and reliability analyzer, July, 2012. "http://essarel.de/index.phpsite=backgroundtext".

Yan, Y., Keller, P., Livnat, Y., Liggesmeyer , P., and Hagen, H. 2012. Improving safety-critical systems by visual analysis. Dagstuhl Follow-Up series. 2012