We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

17

Quality Improvement Through Visualization of Software and Systems

Peter Liggesmeyer, Henning Barthel, Achim Ebert, Jens Heidrich, Patric Keller, Yi Yang and Axel Wickenkamp
Fraunhofer IESE / University of Kaiserslautern
Germany

1. Introduction

Many organizations still lack support for obtaining control over their system development processes and for determining the performance of their processes and the quality of the produced products. Systematic support for detecting and reacting to critical process and product states in order to achieve planned goals is often missing. As systems and software become bigger and more complex, classic approaches reach their limits, due to the difficulty of extracting relevant information from a large volume of measures. Here, suitable visualization and virtual reality solutions can offer a clear advantage by representing the relevant information in a more easily recognizable form. However, many resulting visualizations are still hard to understand, even for experts. This opens the door for researching modern, human-centered approaches that provide the user with visualization and interaction models for visually analyzing and understanding the underlying complex data. This chapter focuses on two main topics: system visualization and software visualization.

Software visualization continues a very successful direction of software engineering, namely the topic of "software measuring/software analysis". Software visualization in general does more than visualizing the product "software" itself. Visualization mechanisms are also typically applied to software development processes and to support the tracking of project progress (e.g., as a central component in project dashboards).

System visualization aims at a better understanding of system properties, e.g., safety. These properties are usually influenced by mechanical components, microelectronics, and software.

2. Software Visualization

The term "Software Visualization" refers to a broad range of visualization mechanisms that can be applied to many different issues relating to engineering-style software development. For instance, it may be used to visualize and analyze software development processes, different artifacts created as part of the development process (such as designs or code), or even aspects of the project itself (like the communication of the team or the information flow between development locations). This section focuses on providing an overview of the

variety of different visualization techniques that may be used. First, the visualization of product and process/project characteristics will be discussed. After that, some mechanisms for dealing with complexity and nesting issues in general will be highlighted.

2.1 Related work

Depending on the focus of a software development organization (e.g., safety-critical software or commercial off-the-shelf software), different characteristics of software products such as security, reliability, or maintainability may be important. ISO/IEC 25000 (see (ISO, 2011)) gives an extensive overview of different product characteristics and some recommendations on how to quantify and measure different attributes of the software. There are lots of measurement tools on the market for analyzing static and dynamic aspects of a software product. Examples are Metrics Eclipse Plugin (Metrics, 2011), CodePro Studio (Instantiations, 2011), JDepend (JDepend, 2011), or McCabe IQ (McCabe, 2011). Each of these tools has different capabilities in terms of the metrics supported and the visualization capabilities offered. A detailed description of such tools can be found in (Rech, 2005).

Typically standard visualization mechanisms are used for project control, such as simple line, bar, and pie charts, spider plots, tables and matrices, simple graphs (networks), Gantt charts, or box plots. The type of visualization used depends on the level of abstraction and the viewpoint for which the data is visualized. Different kinds of exploration capabilities are provided for project control, such as drill-down mechanisms aimed at getting a more detailed view on the data, or aggregation mechanisms aimed at getting an overview containing multiple, abstract pieces of information. Moreover, visualization techniques may provide basic interaction mechanisms for browsing the data, such as scrolling data along a time axis in order to browse the history and development of indicators over time. More advanced visualizations such as visualization metaphors are seldom used within project control frameworks. However, they can be found in special-purpose tools, e.g., visualizing software systems using 3D models of landscapes and urban structures (Balzer, 2004).

In software visualization, graph drawing techniques are used to describe and comprehend the complex structures of software systems, development processes, and software quality characteristics. The most popular graph drawing techniques used are node-link diagrams. The standard algorithm was proposed by Sugiyama (Sugiyama, 1981). Due to the inherent complexity of solving the sub-problems using this algorithm, many heuristics have been proposed, e.g., by Gansner et al. (Gansner, 1993) and Eiglsperger et al. (Eiglsperger, 2005). A detailed description of graph visualization techniques using node-link diagrams can be found in (Herman, 2000), (Battista, 1999), and (Kaufmann, 2001). Since the space requirements of node-link diagrams grows rapidly with the size of a graph, space-filling techniques like Treemaps (Johnson, 1991), Sunbursts (Stasko, 2000), or Icicle Plots (Kruskal, 1983) have been applied to realize a space-constrained visualization of large hierarchical data sets. However, in contrast to node-link diagrams, it might be more difficult to understand hierarchical relations using these space-filling techniques. Node-link diagrams might be combined with space-filling techniques to visualize both hierarchical and non-hierarchical relations within one view. Fekete (Fekete, 2003) uses a Treemap to visualize the hierarchy, while Bézier curves are used to show additional non-hierarchical relations connecting two Treemap regions. See (Fekete, 03) for more details on combining node-link diagrams with space-filling visualization techniques.

2.2 Software product characteristics

Fine-grained measurement of software product characteristics tends to result in a huge amount of measurement data. With simple high-level charting and diagram visualizations (Fig. 1), only condensed views on the measurement data can be provided.

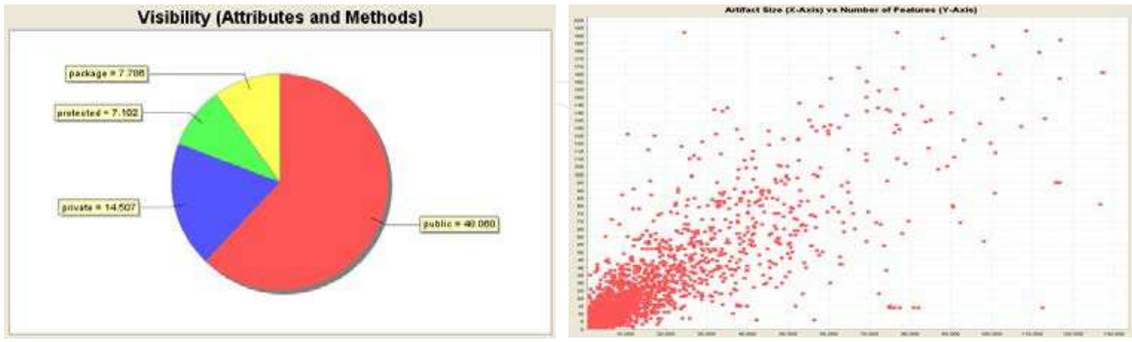


Fig. 1. High-level charting and diagram visualization techniques.

In order to provide a goal-oriented visual exploration and analysis of product characteristics, the structural information about the system needs to be utilized, preserving the user's mental map of the system architecture. Hence graph-based techniques and especially UML diagrams are often used as a base visualization technique, with relevant attributes of the software system being analyzed mapped onto the structural visualization using a set of additional graphical elements such as text, geometric shapes, and icons (Fig. 2). If product characteristics are to be analyzed on the source code level, image-based techniques combined with traditional brushing and shading mechanisms may be used. By utilizing appropriate scaling and interaction techniques, this approach allows the visualization of several levels of details of the source code within one picture: Some areas may show the original source code, while others may visualize several lines of code as a shaded region using color coding and shading to map measurement data onto the visual item (Fig. 3).

To get a deeper insight into the quality of a software product, it is not sufficient to use these visualization techniques with just simple interaction techniques only. Rather, the specification and selection of visualization techniques has to be tightly integrated with the specification and selection of metrics used to collect the measurement data. With such tight integration, the user may apply tailored metrics and visualization metaphors on different level of details when visually exploring the product characteristics.

At Fraunhofer IESE we have realized this approach with our scalable measurement tool *M-System*. We store an abstract syntax tree (AST) representation of the source code in a relational database. This database is then used to define metrics and collect measurement data by issuing appropriate SQL statements. The same SQL frontend is further used to filter the measurement data and to select and tailor the visualization technique for use on different levels of detail. Fig. 4 and Fig. 5 show example visualizations generated with the *M-System*, used to analyze the product characteristics of a software system on different levels of detail. The user starts with a 3D-Treemap overview visualization showing a condensed view of the product characteristics. The 3D-Treemap technique maps both the basic code structure as well as the quality data (e.g., component size and complexity measures) to cubes using different graphical properties (position, size, height, and color).

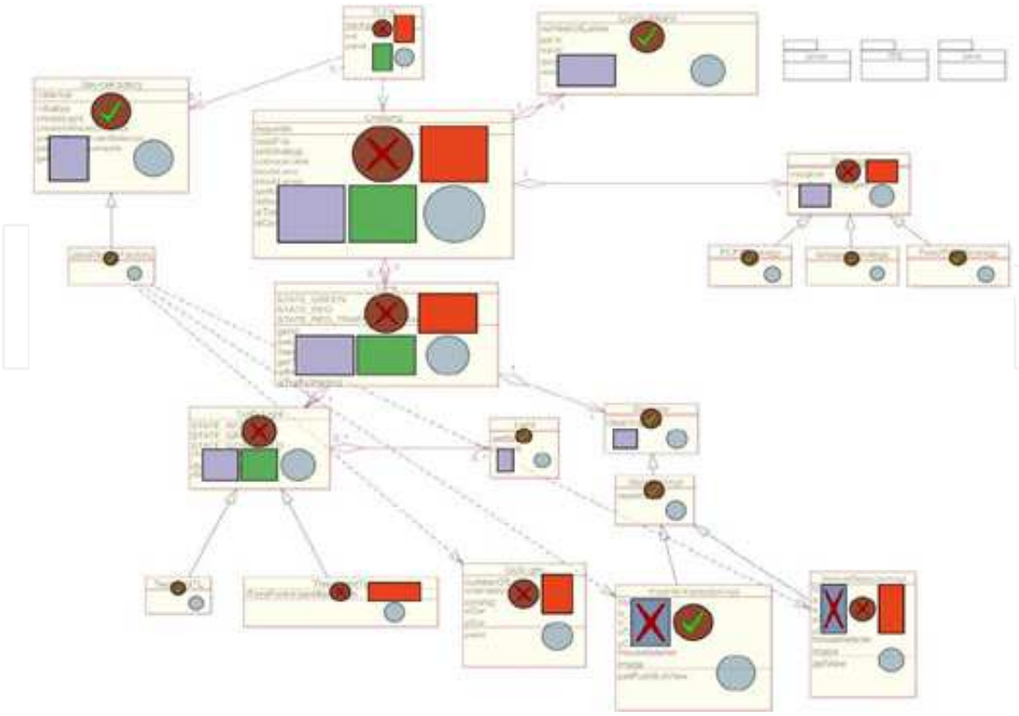


Fig. 2. Graph-based techniques (Termeer, 2005) using UML diagrams.

Fig. 3. Image-based techniques (Ball, 1996) and brushing (Lommerse, 2005) on the source code level.

Critical components (e.g., a red cube in the 3D-Treemap) might then be further analyzed by selecting such components within the visualization and adapting the measurement data and

the visualization technique to be used in the subsequent analysis process. For example, in Fig. 5, we use a node-link graph visualization to highlight critical components. We take the same measures, but simply adjust the visualization to color code the data according to a given threshold value. This simplifies the identification of critical parts of the component being analyzed.

Using the SQL frontend to define metrics and collect measurement as well as to filter the measurement data and to select and tailor the visualization technique has proven to be a very flexible and powerful approach. Nevertheless, making this frontend more user-friendly as well as integrating further visualization techniques into the *M-System* is ongoing work at Fraunhofer IESE.

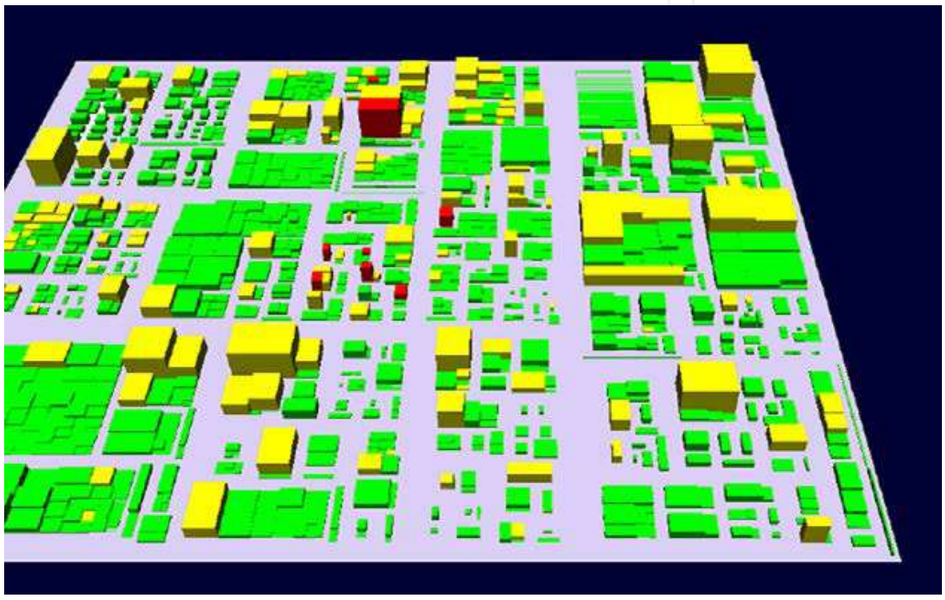


Fig. 4. 3D-Treemap visualization of software quality on different levels of detail using Fraunhofer IESE's M-System.

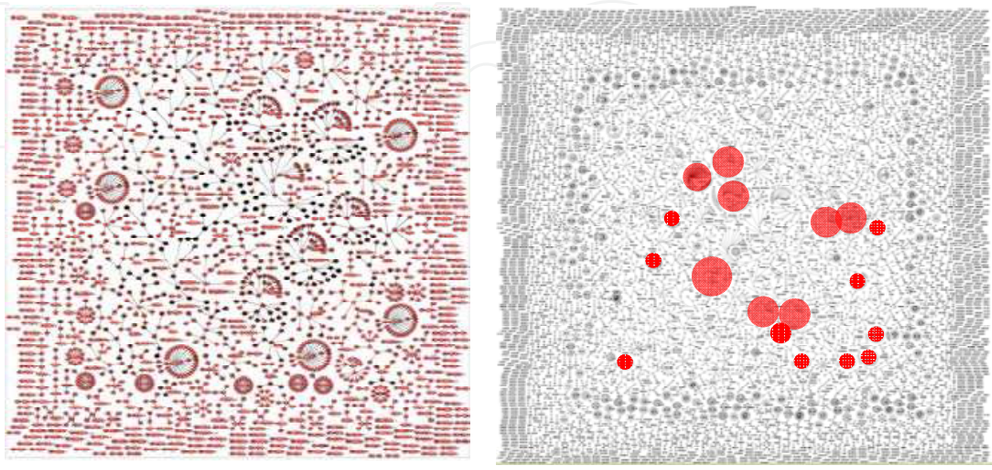


Fig. 5. Node-link graph visualization for highlighting critical components.

2.3 Software process and project characteristics

Visualization techniques may also be used for analyzing complex software development processes. For instance, the German V-Modell®XT has more than 1000 entities with different relationships among them. Analyzing the commonalities between the different versions of such a process and identifying inconsistencies and bottlenecks is a non-trivial task that requires advanced visualization techniques. Fig. 6 presents two analysis views for getting an overview of the evolution of the V-Modell®XT based on the DeltaP approach developed at Fraunhofer IESE (Soto, 2008). The left-hand side demonstrates the increase of entities (activities, artifacts, roles, etc.) over different versions of the V-Modell®XT. The vertical lines indicate official releases of an internal version. From version 1 to version 600, more than 200 entities were added. The right-hand side shows the number of changes in different modules of the V-Modell®XT over time. The size of the circle reflects the number of changes (additions, deletions, modifications) of a certain module. In order to obtain a simple footprint of the evolution, the changes over time are mapped to a bar code. What can be observed from the chart is, e.g., that the major work before release 1.0 was completed at the end of 2005, but it took more than one month with close to zero changes before the actual release took place. A second observation is, e.g., that a huge number of changes were postponed after release 1.1. As soon as the main development branch was released, some major changes were implemented into release 1.2.

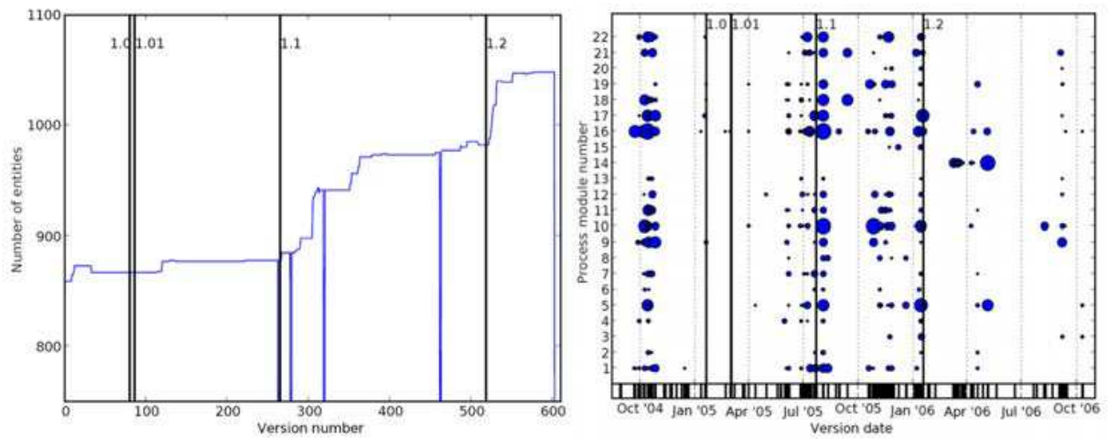


Fig. 6. Visual analysis of the V-Modell®XT evolution (Soto, 2008).

Another very common area for software visualization is project monitoring and control. Many different aspects of a development project have to be visualized on different levels of abstraction, depending on the goals and characteristics of the project. This typically includes the schedule of the project as well as effort/cost aspects, but also quality-related measures, such as the defect density, the requirements already implemented, or the test coverage. One challenge for the visualization of this data is how to show trends for identifying potentially critical project states in time and how to initiate corresponding countermeasures. Another challenge is how to provide visualization mechanisms for condensing and aggregating the data (e.g., making use of corresponding quality models). Fig. 7 presents a Software Project Control Center (SPCC) prototype created at Fraunhofer IESE. The Specula approach is able

to dynamically create an SPCC interface from project control building blocks for data collection, data processing, and data visualization. The screenshot on the right shows on overview of available control charts and displays the selected Gantt charts for monitoring the project's schedule and progress. The screenshot on the left side aggregates all data into a simple matrix structure for analyzing the trend with respect to core project measurement objects and properties.

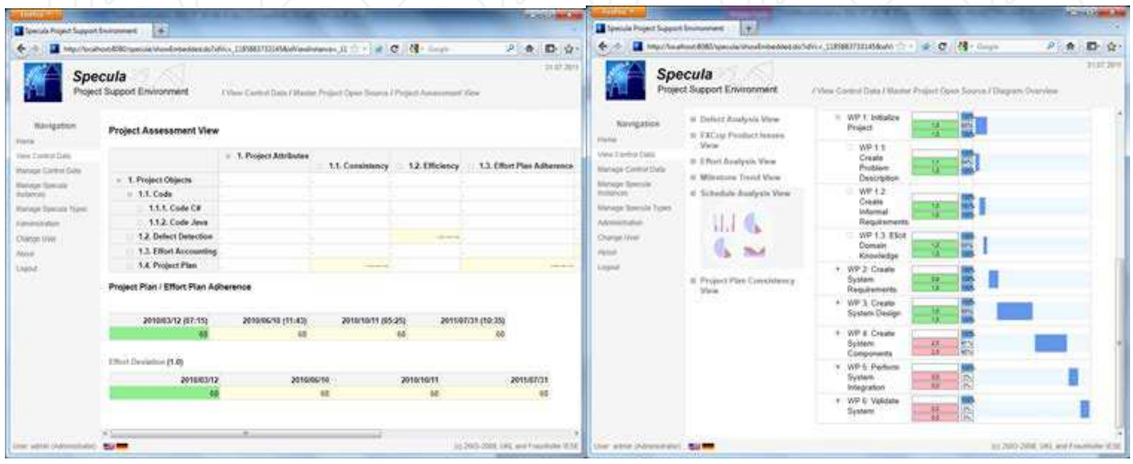


Fig. 7. Visualizing software project control data (Heidrich, 2010).

2.4 Complexity and nesting

A very general underlying problem of software visualization is how to deal with highly complex structures and nesting. Typically, graph representations are used for describing these structures, and graph visualization techniques are applied to improve the comprehension of software products, projects, and processes or to facilitate decision support for software architects and project managers.

However, using off-the-shelf graph visualization tools may be inadequate in the software engineering domain because they often do not reflect the user's context and role-specific visualization and interaction requirements in such distinct application scenarios as Quality Measurement, Architectural Analysis, Reengineering, Process Management, Process/Product Evolution, Risk Management, or Security and Safety. Rather, specially engineered tool support is required, providing domain-centered visualization and interaction techniques that support the interactive visual exploration of highly complex data structures within each application scenario.

At Fraunhofer IESE we align our tool support according to this strategy (see sections 2.2, 2.3, 3.1 and 3.2 targeted at relevant application scenarios). Based on the user's need, we combine traditional visualization techniques with suitable interaction and navigation techniques on different levels of detail. The user might start with a 2D or 3D visualization providing an overview of the whole data set, which acts as the starting point for the visual exploration (see Fig. 8). However, the user can also use traditional node-link diagrams or

a combination of node-link diagrams and space-filling techniques if he is used to that (see Fig. 9).

The user might then decide to further investigate parts of the dataset using more appropriate visualization and interaction techniques. For example, if the user performs an architectural analysis and needs to know more details about the relations between components of a subsystem, he might select the subsystem in the overview visualization and switch to a more suitable visualization like that in Fig. 10.

Practical examples from industry indicate that it is reasonable to realize domain-specific, user-centered visual exploration tools in the software engineering domain that are tailored to the user's context and role-specific visualization and interaction requirements. Realizing this tool support is ongoing work and an interesting research topic at Fraunhofer IESE.

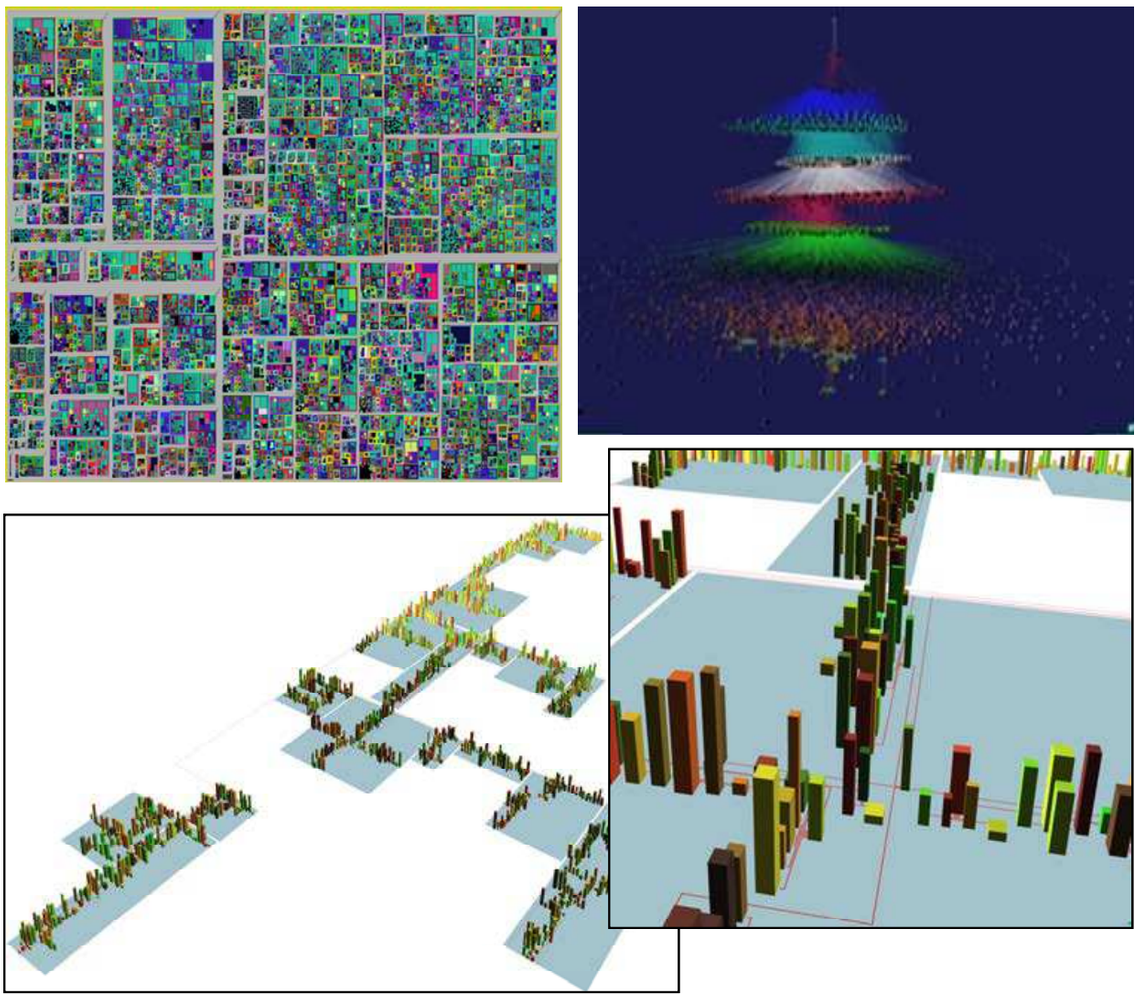


Fig. 8. Exemplary overview visualizations that might be used as a starting point for visual exploration.

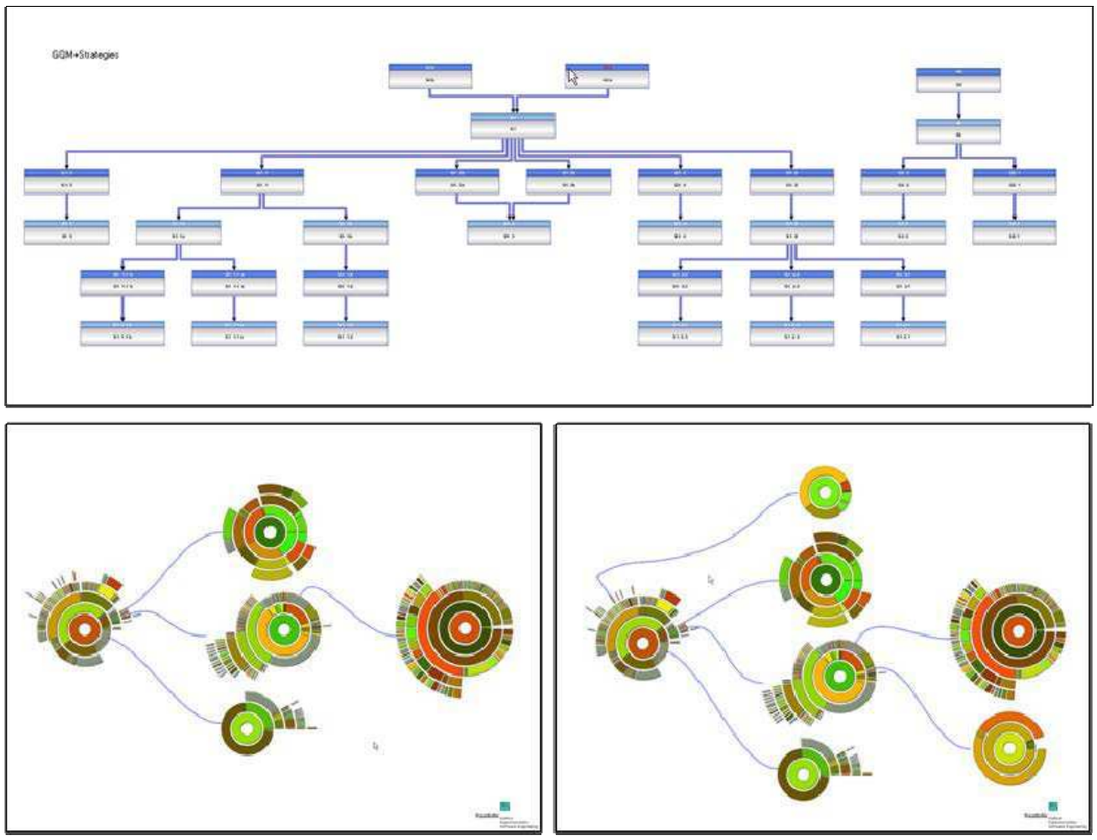


Fig. 9. Traditional node-link diagram visualization (top) and a combination with space-filling techniques (bottom) using sunbursts to visualize hierarchy nodes and Bezier-Curves to visualize links.

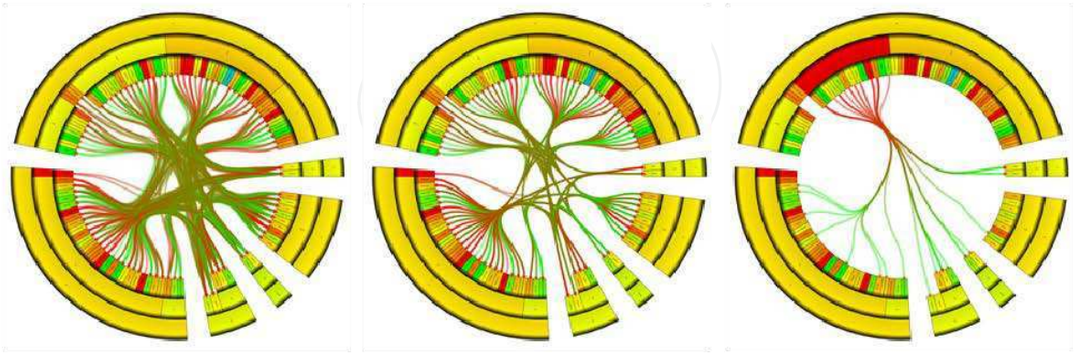


Fig. 10. Combined space-filling and node-link visualization. Edge bundling (Holten, 2006) is used to visualize non-hierarchical relations, based on the multivariate information taken from the original dataset.

3. System visualization

The importance of system visualization increases due to growing size, critical properties (e.g., safety, security), and new application domains (e.g., ubiquitous systems / ambient systems), which are characterized by complex behavior (e.g., dynamic aspects). Many safety-critical embedded systems need to be certified by certification authorities before being released for public use. This requires techniques that are capable of displaying detailed information about the safety properties of large systems and that can, at the same time, prevent critical information overload.

3.1 Safety visualization

Safety is a quality characteristic that is very important in many application domains, e.g., rail systems, avionics, or medical systems. According to the IEC 61508 standard, safety is the absence of unacceptable risks.

3.1.1 Safety analysis

The analysis of safety-critical systems is organized to follow certain rules. The corresponding techniques and methods tend to analyze the causes leading to critical system conditions that violate the specified safety objectives. One technique frequently used in this context is the so-called Fault Tree Analysis (FTA). It describes the relationship between low-level failures, e.g., failures on the component level, and system failures. Fault Trees (FT) use logical gates to model these cause-effect relationships. Low-level failures (Basic Events (BEs)) are represented as leaves of the FT, whereas the root represents a safety-critical system failure (Top Event (TE)). Besides being a qualitative analysis, FTA allows quantifying the probability of occurrence of the TE if the probabilities of the corresponding BEs are known.

A Minimal Cut Set (MCS) is defined as a minimal set of BEs whose simultaneous occurrences cause the TE. Traditional approaches for analyzing MCSs primarily use text-based ((ESSaREL, 2011), (RELIA, 2011), (Fig. 11 (a)) or table-based ((RelexArchitect, 2011), (SYNC, 2011), (Fig. 11 (b)) representations. The information listed there usually includes an ID for identifying the individual MCSs and the BEs contained together with their corresponding failure probabilities. In many cases, table-based representations rely on regular interaction techniques such as filtering and sorting. However, the possibilities of these manipulation techniques are limited. They neither provide appropriate graphical assistance facilitating better understanding nor do they support the analysis of correlations between BE and MCS within large data sets. This hampers efficient handling and consequently might lead to situations in which information intended to contribute to safety improvements may become lost. To reflect the way the effects of the current MCS propagate along the FT (Fig. 11 (c)), some approaches ((ESSaREL, 2011), (ISOGRAPH, 2011)) represent MCS via highlighted paths between BEs and the TE.

The idea of modeling cause-effect relationships was extended to so-called Component Fault Trees (CFT). CFTs permit encapsulating sub-trees that correspond to technical components and thus provide the possibility to deal with different levels of abstraction.

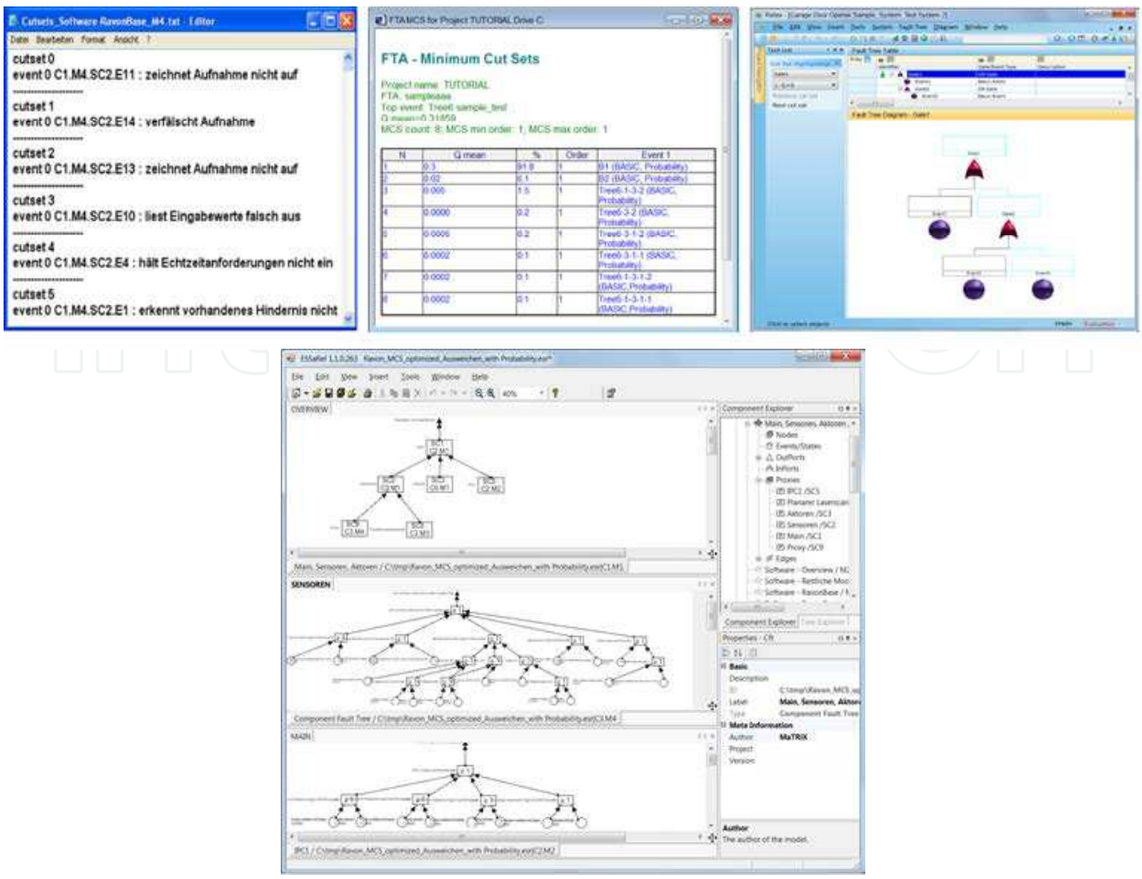


Fig. 11.Common methods for representing MCSs: (a) Plain text (ESSaREL, 2011). (b) Table format (ALD, 2011). (c) Associated paths highlighting (RelexArchitect, 2011). (d) Component Fault Tree (ESSaREL, 2011).

The tool ESSaREL (ESSaREL, 2011) supports the use of CFTs (Fig. 11 (d)). Additional MCS information in the form of plain text is provided in separate views (Fig. 11 (a)).

3.1.2 ViSSaAn

A new visualization tool called ViSSaAn (Visual Support for Safety Analysis) developed at the University of Kaiserslautern (Yang, 2011) uses a matrix-based visualization to efficiently represent the information associated with MCSs of FTs and CFTs, respectively.

The matrix assigns the MCSs to the rows, whereas the failure probabilities of the individual MCSs, the MCS IDs, the orders of the MCSs (Fig. 12, area 1), and the BEs (Fig. 12, area 2) are assigned to the columns. The tool classifies the MCSs into three different safety categories according to their failure probabilities: high, moderate, and acceptable; the thresholds for this classification can be set according to the specific needs of the analysis process. The color scheme applied for visualizing these relations uses red to indicate high, yellow to indicate moderate, and green for acceptable probabilities. In this way the user is able to perceive the criticalities of the individual MCSs very intuitively. The three-level categorization concept is also applied in the same way for characterizing and visually highlighting the BEs.

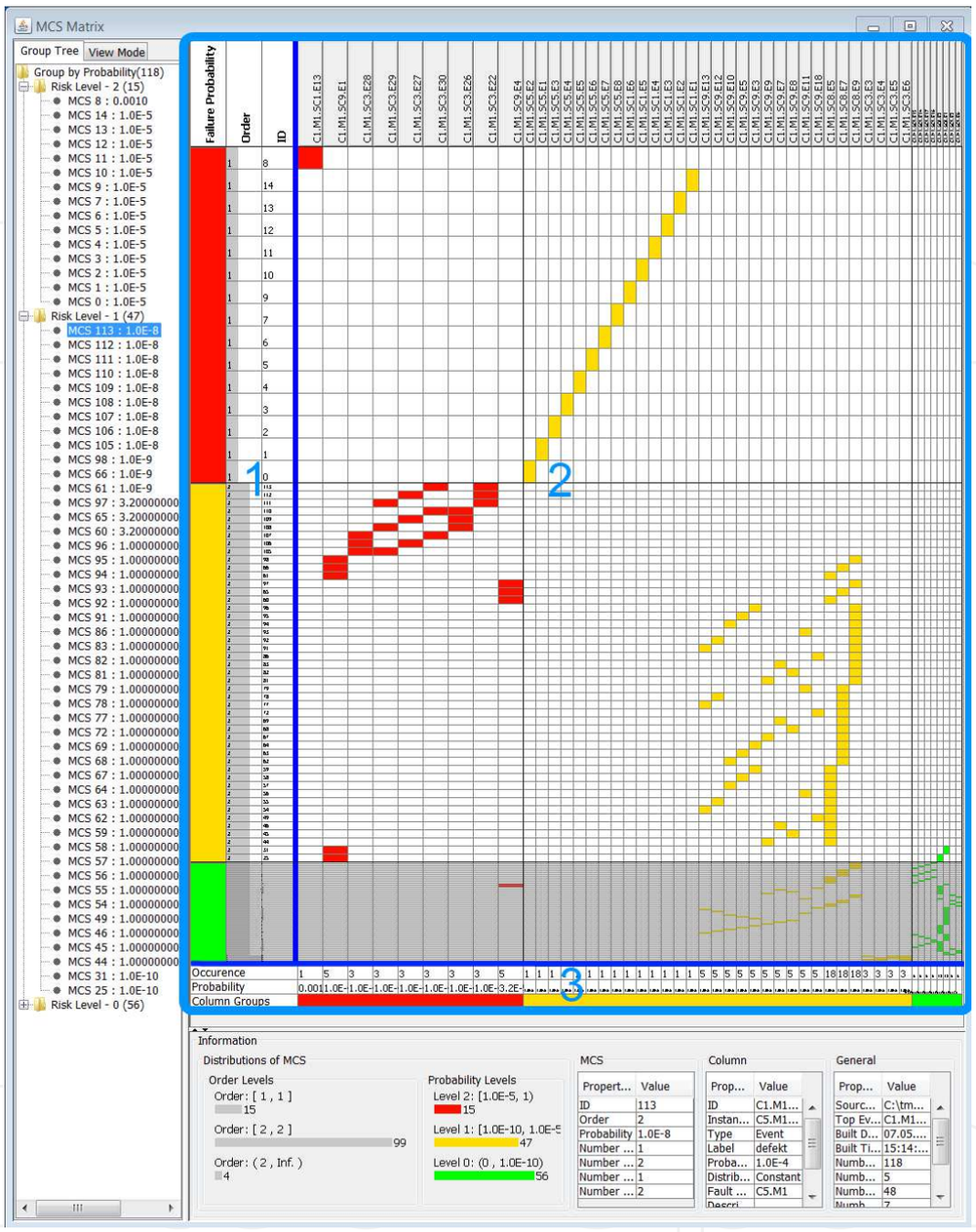


Fig. 12. Overview of ViSSaAn: matrix view (center), navigation tree (left), and information panel (bottom). In matrix view, area 1: properties of MCSs; area 2: containing relationships between MCSs (rows) and BEs (columns); area 3: properties of BEs and indicators of groups for BEs.

Furthermore, ViSSaAn uses translucent bar charts to represent the orders of the MCS. The order in this regard is defined by the number of BEs contained in an MCS. In general, this means: the smaller the order, the more critical the MCS.

Whenever a BE belongs to an MCS, the matrix indicates this dependency by coloring the cell at the location where the BE column intersects with the MCS row (see Fig. 12, area 2). The

cell color used at the intersection depends on the BE's safety category. In addition, ViSSaAn displays the exact occurrences and failure probabilities of the BEs at the bottom of matrix to support an extended quantitative analysis (Fig. 12, area 3).

In order to quickly identify essential dependencies within a large number of MCSs, ViSSaAn provides a range of sorting and grouping functionalities. The *sorting function,* for instance, allows sorting the MCSs according to their failure probabilities in descending and ascending order. The resulting groups of MCSs having high, moderate, and acceptable probabilities separating the matrix vertically into three distinct segments with different colors (Fig. 12, area 1 shows the three MCS groupings in red, yellow, and green). The sorting function automatically groups the MCSs according to their criticalities, so that the user can efficiently delimit the number of MCSs to be examined. The sorting function is also available for the columns in order to intuitively identify and delimit important BEs. The grouping in case of BEs is illustrated by the last row of the matrix view in Fig. 12 (area 3).

To satisfy the need for providing a suitable overview when addressing large-scale MCS data sets, ViSSaAn additionally provides flexible scaling interaction functionalities like *uniform scaling* and *scaling by groups*. *Uniform scaling* allows narrowing the row heights in order to depict as many MCSs as possible to exploit the limited display space. However, one problem immanent to this scaling approach concerns the reduction of the space used to depict other MCS-related information. To address this issue, ViSSaAn uses a scaling approach following the degree-of-interest (DOI) concept referred to as *scaling by groups*. It preserves important MCS information, while simultaneously maintaining a suitable overview (Fig.12, area 1). The heights of the rows are adapted to the safety criticality of its corresponding MCS: the more critical the MCSs, the larger the row heights. This way, the users can focus on the important information while maintaining the overview as their context. Both scaling approaches can be applied to rows as well as to columns in order to reduce information overhead by simultaneously maintaining all details necessary for an efficient analysis.

3.1.2.1 Embedded CFT structures

The FT logical structure supports better understanding of the influences of MCSs in a system. ViSSaAn integrates the structures of CFT components into the matrix view using focus+context techniques in combination with semantic zooming concepts. Whenever a color-filled cell is double-clicked, ViSSaAn enlarges the corresponding cell area and displays an embedded view (see Fig.13) showing the internal logical structure of the component associated with the BE. The leaf nodes are again colored according to their safety criticality levels. The color-filled nodes represent the BE corresponding to the selected column, whereas the nodes having a thick border represent all other BEs contained by the MCSs of the current row. Those leaf nodes or BEs not directly related to the MCS are marked with a thin border. Integrating such embedded logical structures allows ViSSaAn to provide additional information about how the BEs of an MCS influence the component of a CFT along the internal structure. This allows the users to focus on the detailed internal structures while maintaining the overall context. ViSSaAn also provides interactions for the internal structure inside the embedded view, such as smooth zooming and panning, which allow adapting the view to the current visualization needs. A more detailed description of the ViSSaAn tool and its possibilities is provided in (Yang, 2011).

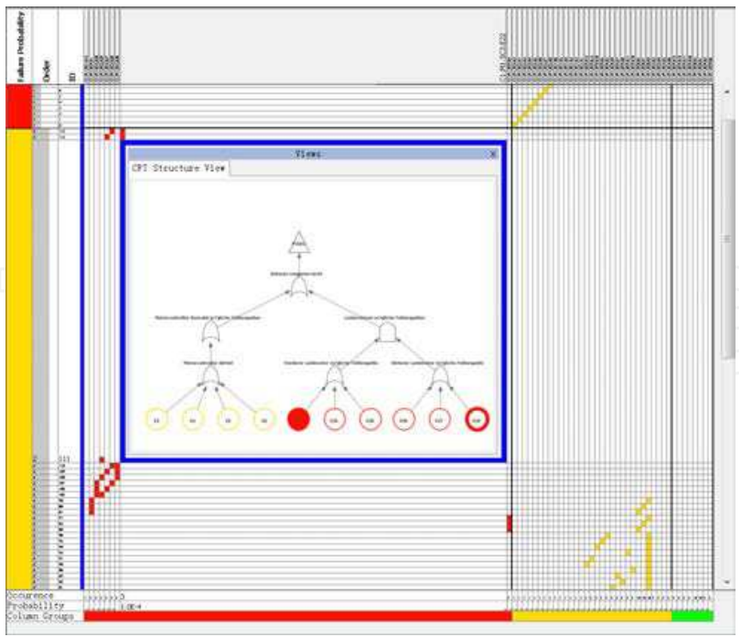


Fig. 13. Logical structure of the CFT component embedded in the matrix view. The color-filled node represents the BE in the current column. The nodes with a thick border are contained by the MCS in the current row. BEs not related to the current MCS are depicted with a thin border.

3.2 Virtual Reality

Virtual Reality (VR) is considered a technology for providing virtual mappings of real-world objects, which are allowed to be explored and manipulated by users. It combines 3D visualization techniques with human-centered interaction techniques. Depending on the display technology, VR is classified into four main categories: immersive (e.g., head-mounted displays), semi-immersive (e.g., car driving simulator), projected (e.g., CAVE – Cave Automated Virtual Environment, 3D Walls etc.), and desktop (e.g., monitor), each with its own advantages and disadvantages.

3.2.1 Current approaches In VR

Regarding the usage of VR technology, several fields in science and engineering have greatly benefitted from its application. This includes, among other things, that VR contributes to a better understanding of real-world systems and processes. VR can emphasize which information is important for the user and simultaneously allows blanking out other kinds of information. It also supports the exploration of complex structures, processes, and relations (Kreylos, 2003).

In addition, VR facilitates the process of developing large systems and environmental structures by providing means for efficient design and analysis. In this context, VR is being adopted in domains like automotive (e.g., the design of cars), aeronautics (e.g., in the development of airplanes and satellites), as well as in the construction industry to provide a first glance at final products, e.g., in the form of virtual prototypes. Following up on this

idea makes it possible, e.g., to perform a proof of concept even without any real system at hand. When we consider economical aspects, the application of VR techniques provides great benefits in terms of cost reduction and time savings during development. Particularly, in scenarios where the existence of expensive functional models is required, it may no longer be necessary to come up with a fully functional real-world system. The article by Purschke et al. (Purschke, 1998) gives an example of how VR techniques have recently been used with respect to developing vehicle systems at Volkswagen.

Another application area uses VR for the purpose of training and operator qualification. Virtual training systems are built with the intention to reduce expenses for education and training and to provide a safe way of preparing critical operations and procedures that otherwise would most likely endanger persons or equipment. This is possible since VR systems are, by their very nature, safe, controllable, and economical. VR applications were developed e.g., for education (Oliveria, 2007), preparation of medical surgeries (Heng, 2004), and others.

In addition, VR is preferably used in cases where safety-critical systems have to be tested with regard to dangerous situations. This may become necessary within the context of common certification procedures concerning a range of critical system properties. Not all situations can be covered by real-world testing – be it due to time or money restrictions or simply because of safety reasons. In (Barret, 2010), the authors introduce a virtual system allowing the design and exploration of safety-critical scenarios in the context of electrical engineering.

3.2.2 VR and safety

In the last few decades, the number of electronic and electronically programmable devices deployed within embedded systems has risen dramatically. This not only had effects on the size and complexity of those systems but also on their quality. Especially safety is a quality characteristic strongly influenced by this evolution.

One possible step towards "safer" systems is to take advantage of extended means provided by Virtual Reality (VR) technologies. VR is increasingly becoming a powerful tool for tackling the challenges found in quality assurance of embedded systems. It allows incorporating knowledge/information from different sources, ranging from physical domains like object shapes and materials to more imaginary quantities like system quality and usability, and allows presenting it in a way intuitively accessible for human perception. In addition, special interaction techniques facilitate the handling of the provided representations. The idea of supporting the analysis of complex embedded systems by using VR technology is pursued by the project ViERforES (ViERforES, 2011) and its successor ViERforES 2 (ViERforES, 2011). These projects especially aim at improving the quality of complex embedded systems by providing methods and techniques for virtually accessing and evaluating particular system characteristics such as safety, security, and reliability.

Regarding the analysis and evaluation of safety-related properties, the goal is to identify critical system parts, relating them to possible system failures, and providing quantitative measures concerning the probability of occurrence of hazardous situations. A system failure in this sense is an undesired deviation of the actual behavior from the specified one. Performing such an analysis necessitates the abstraction of certain system properties to yield

a view that represents a reduced image of the system. To accomplish this, most common techniques rely on graph- or text-based approaches (see section on safety analysis). These allow capturing specific safety-related aspects of the system under investigation. However, in general this entails the loss of context information, which may also indirectly contribute to system failures and hence is important with regard to the completeness of the analysis.

When considering embedded systems, the failing components at the most abstract stage can be hardware (HW) or software (SW). By nature, HW and SW components have a great variety of characteristics, modalities, and working conditions, and are generally not representable by a single model. For example, installation prerequisites, exposure to environmental interferences, as well as mutual interference between HW components are additional properties not reflected by the models used for safety analysis.

Let's assume that the safety of a control unit exposed to internal/external forces such as extreme heat, shock, and/or radiation has to be evaluated. Here, only a safety analysis is capable of making assumptions about the failure probability of the controller and how it may contribute to a system crash. However, it does not consider the ways the controller correlates with the aforementioned physical factors, i.e., it cannot answer the question whether or not there is an actual dependency between the way the controller is installed and the system's safety. Deriving cross-related knowledge from other models or system views is not possible from the safety models or is only provided up to a certain limit. To ascertain such extended coherences, the incorporation of knowledge/information from additional sources is required. Collecting all relevant information and representing them in a suitable way is necessary for the hidden relations to become apparent.

The above observations hold for HW as well as for SW, which has a variety of essential characteristics that indirectly contribute to safety. For example, important issues for assessing the criticality of a SW component can be its complexity (e.g., lines of code, number of dependencies between components, etc.), its state space situation, or its dependencies between workload and the system response time in critical scenarios. Again this represents context information not always considered in standard safety models.

Another important aspect concerns the distribution of the components of an embedded system. Generally, the sort of systems that belong to this category feature the property of being distributed. This means that communication takes place, which might also be affected either directly by intended attacks or indirectly by physical interferences. The connection between the vulnerability of communication channels in general and the way they affect the safety/security criticality are explored best if there is a combined virtual sight.

In the ViERforES 2 project, VR is used to combine safety-related information with such kind of "hidden" context information. This provides the chance to visually integrate safety-critical relationships with the corresponding static and dynamic physical dependencies of the investigated system. The related paper (Al-Zokari, 2010) proposes a way to link static safety-related information in the form of Minimal Cut Sets (MCS) (see section 3.1) with information about the physical system structure. A direct visual connection between the critical components and the ways they cause the system to fail is established. A new metaphor representing all possible constellations of cause-effect relationships allows the effective identification of most critical failure causes. The linkage between the MCS safety indicators and the physical context information in the form of a geometric representation of

the system components is based upon highlighting those components that contribute to a selected failure scenario. The tool allows to intuitively trace the causes of safety-critical system failures back to the components responsible for their origination. This way the users (safety engineers and system engineers) can explore relations that could help, for example, to efficiently identify critical system regions. Moreover, it provides a common view for system and safety engineers to support inter-connected collaboration.

The ongoing research could be seen as a starting point regarding further development aimed at the integration of multiple models from complex embedded systems. The possibilities of application of the VR technology are various and provide great potential concerning future research.

Virtual Reality (VR) provides promising possibilities for visually assessing the characteristics of large embedded systems. Analyzing functional and non-functional aspects generally necessitates uncovering hidden properties and relations between system parts that heavily affect overall system functioning and system quality. Particularly, properties related to software are not always visible at first glance. The research within the ViERforES (Virtual and Augmented Reality for Maximum Safety and Reliability of Embedded Systems) project (ViERforES, 2011) concentrates on capturing and visualizing qualitative aspects of safety-critical systems to support system analysis and improvement. The process of tracing back undesired system characteristics to their roots constitutes the first step in identifying critical parts of a system. Generally, this produces huge amounts of data that are strongly correlated and not accessible without the usage of the proper context information. The adoption of VR techniques allows the user to look at every aspect of the system in detail. Furthermore, it facilitates the exploration of hidden properties and supports the collaboration of domain experts such as system and safety engineers. It associates safety-related information with virtual models of the examined systems and provides interaction techniques to support the process of identifying safety-critical elements.

4. Conclusion

The visualization of software and systems is becoming increasingly important for many organizations. Well-known visualization and interaction techniques are applied to obtain better insight into and control over system development processes and the quality of produced products. Even though these methods have proven to be advantageous, more domain-specific, user-centered approaches are necessary to support users in recognizing and finding relevant information more easily. Based on practical examples from industry and the experience of the authors, this chapter presented an overview of such goal-oriented, domain-specific visual exploration tools in the software engineering domain.

We showed that tightly coupling the specification and selection of visualization techniques with the specification and selection of metrics used to collect measurement data better supports the visual recognition of software product, process, and project characteristics and their interrelations.

In addition, we presented visualization techniques aimed at a better understanding of system properties such as safety and security in highly complex, dynamic embedded systems.

More sophisticated human-centered visual exploration techniques are needed to be able to analyze such system properties, which generally necessitates uncovering hidden properties and relations between system parts that heavily affect overall system functioning and system quality. The presented work allows interactively exploring information associated with the results of a safety analysis to maintain an overview of critical elements and relations. Approaches like the matrix-based representation presented here facilitate the understanding of the structural composition of failure causes and, in particular, allow to efficiently determine weak points of safety-critical systems.

5. References

ALD RAM Commander. (2011) http://www.aldservice.com.

Al-Zokari, Y., Khan, T., Schneider, D., Zeckzer, D., Hagen, H. (2010). *CaKES: Cake Metaphor for Analyzing Safety Issues of Embedded Systems.* Dagstuhl Follow-Ups.

Ball, T., Eick, S. (1996). *Software visualization in the large*. IEEE Computer, pp. 33–43.

Balzer, M., Noack, A., Deussen, O., Lewerentz, C. (2004). Software Landscapes: Visualizing the Structure of Large Software Systems. In: Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym), pp. 261-266.

Barret, M., Blackledge, J. Eugene, E. (2010). *Using Virtual Reality to Enhance Electrical Safety and Design in the Built Environment.* Dublin: Dublin Institute of Technology.

Battista, G. D., Eades, P., Tamassia, R., Tollis, I. G. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ, USA.

Eiglsperger, M., Siebenhaller, M., Kaufmann, M. (2005). *An efficient implementation of Sugiyama's algorithm for layered graph drawing*. InGraph Drawing (GD 04), vol. 3383 of Lecture Notes in Computer Science, Springer, pp. 155-166.

ESSaREL. (2011). http://www.essarel.de.Fekete, J.-D., Wang, D., Dang, N., Aris, A., Plaisant, C. (2003). *Overlaying Graph Links on Treemaps*. In Proc. InfoVis'03, Poster Compendium, IEEE Press.

Gansner, E. R., Koutsos, E., North, S. C., Vo, K. P. (1993). *A Technique for Drawing Directed Graphs*. IEEE Trans. Softw. Eng. 19, 3, pp. 214-230.

Heidrich, J. Münch, J. (2010). *Goal-oriented customization of software cockpits*. Journal of Software Maintenance and Evolution: Research and Practice, Volume 22 Issue 5.

Heng, P.A., Cheng, C.Y., Wong, T.T., Xu, Y., Chui, Y.P., Chan, K.M., Tso, S.K. (2008). *A virtual-reality training system for knee arthoscopic surgery*. IEEE Transaction on Information Technology in Biomedicine 8 (2), pp. 217-227.

Herman, I., Melancon, G., Marshall, M. S. (2000). *Graph Visualization and Navigation in Information Visualization: A Survey*. IEEE Transactions on Visualization and Computer Graphics 6 (1), pp. 24–43.

Holten, D. (2006). *Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data.* IEEE Transactions on Visualization and Computer Graphics (Proc. of INFOVIS'06) 12 (5), pp. 741–748.

Instantiations CodePro Studio. (2011). http://www.instantiations.com

ISOGRAPH FaultTree+. (2011). http://www.isograph-software.com.

ISO – International Organization for Standardization. (2011). http://www.iso.org

JDepend. (2011). http://clarkware.com

Johnson, B., Shneiderman, B. (1991). *Tree-maps: A spacefilling approach to the visualization of hierarchical information structures*. Proceedings of Visualization '91, (San Diego, California), pp. 284–291.

Kaufmann, M., Wagner, D. (2001). *Drawing Graphs, Methods and Models*. Lecture Notes in Computer Science, Springer Verlag.

Kreylos, O., Bethel, E.W., Ligocki, T.J., Hamann, B. (2003). *Virtual-Reality Based Interactive Exploration of Multiresolution Data*. In: G. Farin, H. Hagen and Bernd Hamann: Hierarchical Approximation and Geometrical Methods for Scientific Visualization. Springer Verlag, pp. 205-224.

Kruskal, J. B., Landwehr, J. M. (1983): *Icicle Plots: Better Displays for Hierarchical Clustering*. The American Statistician 37 (2), pp. 162–168.

Lommerse, G., Nossin, F., Voinea, L., Telea, A. (2005). *The Visual Code Navigator: An Interactive Toolset for Source Code Investigation.* Proceedings of the 2005 IEEE Symposium on Information Visualization, IEEE Computer Society.

McCabe IQ. (2011). http://www.mccabe.com

Metrics Eclipse Plugin. http://sourceforge.net/projects/metrics

Oliveria, D.M., Cao, S.C., Hermida, X.F., Rodríguez, F.M. (2007). *Virtual Reality System for Industrial Training*. IEEE International Symposium on Industrial Electronics, pp. 1715-1720.

Purschke, F., Schulze, M., Zimmermann, P. (1998). *Virtual Reality - New Methods for Improving and Accelerating the Development Process in Vehicle Styling and Design*. Computer Graphics International.

RelexArchitect. (2011). http://www.relexsoftware.co.uk.

Rech, J., Weber, S. (2005). *Werkzeuge zur Ermittlung von Software-Produktmetriken und Qualitätsdefekten- Studie zu Software-Messwerkzeugen*. Fraunhofer IESE.

RELIA BlockSim. http://www.reliasoft.com/BlockSim.

Stasko, J., Zhang, E. (2000). *Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations*. In INFOVIS '00: Proceedings of the IEEE Symposium on Information Visualization 2000, pp. 57-68.

Soto, M., Ocampo, A., Münch, J. (2008). *The Secret Life of a Process Description: A Look into the Evolution of a Large Process Model*. International Conference on Software Process (ICSP) - Proceedings. Berlin: Springer, pp. 257-268.

Sotograph. http://www.software-tomography.de

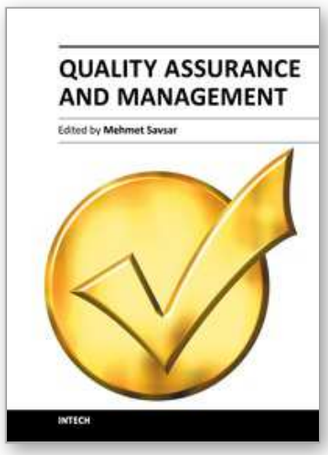
Sugiyama, K., Tagawa, S., Toda, M. (1981). *Methods for Visual Understanding of Hierarchical System Structures*. IEEE Transactions on Systems, Man, and Cybernetics SMC-11(2), pp. 109-125.

SYNC DPL-Faulttrees. (2011). http://www.syncopationsoftware.com/faulttree.html.

Termeer, M., Lange, C. F., Telea, A., Chaudron, M. R. (2005). *Visual Exploration of Combined Architectural and Metric Information*. VISSOFT '05 Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis, IEEE Computer Society.

ViERforES Project. (2011). http://www.vierfores.de.

Yang, Y., Zeckzer, D., Liggesmeyer, P., Hagen, H. (2011). *ViSSaAn: Visual Support for Safety Analysis*. Dagstuhl Follow-Ups, Vol. 2.



Quality Assurance and Management
Edited by Prof. Mehmet Savsar

ISBN 978-953-51-0378-3
Hard cover, 424 pages
Publisher InTech
Published online 23, March, 2012
Published in print edition March, 2012

The purpose of this book is to present new concepts, state-of-the-art techniques and advances in quality related research. Novel ideas and current developments in the field of quality assurance and related topics are presented in different chapters, which are organized according to application areas. Initial chapters present basic ideas and historical perspectives on quality, while subsequent chapters present quality assurance applications in education, healthcare, medicine, software development, service industry, and other technical areas. This book is a valuable contribution to the literature in the field of quality assurance and quality management. The primary target audience for the book includes students, researchers, quality engineers, production and process managers, and professionals who are interested in quality assurance and related areas.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen