CluE: An Algorithm for Expanding Clustered Graphs

Ragaad AlTarawneh* Johannes Schultz† Shah Rukh Humayoun‡

Computer Graphics and HCI Group
University of Kaiserslautern
Gottlieb-Daimler-Str., 67663 Kaiserslautern, Germany

ABSTRACT

In this paper, we present an algorithm, called **CluE** (The **Clu**ster **E**xpander of Compound Graphs), that expands cluster nodes in compound graphs. CluE was designed to work with multiple layout algorithms (e.g., orthogonal layout algorithm, Sugyima layout algorithm, etc.). It keeps the unexpanded nodes in their relative layers with slight changes in their original coordination. This helps the users in maintaining their mental map of the underlying graph. We applied the proposed algorithm for navigating through compound graphs representing the failure mechanisms in embedded systems. We also performed a brief user evaluation study in order to measure the users' acceptance level with regard to the application of CluE for compound graphs. Results show a high acceptance ratio from users having different backgrounds, which indicates the intuitiveness and feasibility of our approach.

Index Terms: Information visualization, graphs and networks, compound graphs visualization, clustered graphs exploration.

1 INTRODUCTION

Generally, creating abstract representations of large data for limited display sizes is considered a challenging task. Producing a graphical representation of the containment relationships between the graph nodes is one of the common ways to tackle this kind of challenging task in the state-of-the-art literature [7, 14]. Normally, diagrams of such abstract representations offer a hierarchical structure of the graph components, where nodes may contain other nodes and/or edges. These kinds of abstract representations are popular in many application domains [7] (e.g., UML diagrams of software systems bioinformatics, etc.). Besides keeping the adjacency relations between the graph nodes, one important factor in these kinds of abstract representations is that they also show at the same time the structural relations between the graph nodes. In such cases, this phenomena is called *hierarchy crossing edges*, especially when we talk about the *compound graphs* [14].

Hierarchal graph structures or compound graphs are widely used in information visualization due to their popularity in many application domains [16, 18]. Normally, they are common in those application domains where grouping of data elements, which belong to the same cluster or share the same attribute, is a natural part of the underlying data. This grouping process helps in producing an abstract representation of the underlying graphs, called as *clustered graphs* (see Figure 1 and Figure 2). These clustered graphs' representations offer many possibilities for trained users in understanding complex diagrams. For example, it speeds up the understanding process of the software systems architecture as it provides a logical grouping for the packages and classes. However, exploring these complex diagrams is considered a tedious task in the lack of proper interaction mechanisms. Therefore, it is important to provide an interactive navigation mechanism for exploring the clustered nodes in such representations, in order to get more insight about the adjacency relations between the graph nodes at different level of details in the graph. Expanding or contracting cluster nodes is one of the traditional mechanisms to explore such structures for increasing the comprehension level of the data.

In this paper, we present a novel algorithm, called **CluE** (The **Clu**ster **E**xpander of Compound Graphs), that aims at expanding the compound nodes in clustered graphs. The CluE algorithm works with any hierarchical layout algorithm (e.g., the orthogonal layout algorithm [8] or the Sugiyama algorithm [19]) or any hierarchal layout algorithm (e.g., the classical tree layout [17]) The proposed algorithm offers an intuitive approach for maintaining the user's mental-map for compound graphs. Also, it takes care of the bounded region of the underlying graph. Therefore, it produces an area-aware layout of the expanded cluster based on the number of children of that cluster [2]. All these features of the CluE algorithm facilitate the graph exploration and navigation in order to get more insight about the graph structure on-demand.

The remainder of the paper is structured as follows. In Section 2, we highlight some related work. In Section 3, we describe the background information and the motivation behind our approach. In Section 4, we explain the CluE algorithm. In Section 5, we provide few initial results to show the application of CluE algorithm as well as briefly describe results of the conducted user evaluation study. Finally, we conclude in Section 6.

2 RELATED WORK

The clustering and interactive concept of large graphs has been described in many previous works. Feng et al. introduced the clustered graph model in [6], which is considered one of the first work in this area. They investigated how the graph model can be drawn with minimum edge crossings or without edge-region crossings. Also, they introduced efficient algorithms for testing C-planarity and for detecting C-planer embedding with this kind of graphs. Another important work in this domain was introduced by Eades et al. in [5]. Their proposed approach allows exploring and navigating large graphs based on the following order: first producing a hierarchy of the graph using the clustering graph algorithm, then making layout the current level of detail using the force directed layout algorithm, and finally providing animation to transform one graph view to another one. In [11], Huang and Eades described the DA-TU system that combines animated clusters with online force-directed animated graphs for visualizing large graphs. Another example is ASK-GraphView system, proposed by Abello et al. in [1], that allows clustering and navigating large node-links diagrams. Their proposed solution builds a hierarchy based on arbitrary weighted large graphs. Moreover, it suggests the navigation through top-down fashion in order to expand clusters on-demand. Archambault et al. proposed GrouseFlox approach in [3] that allows users interactively exploring large graphs using a cut operation of a superimposed hierarchy. This proposed approach creates hierarchy using a topological feature-based algorithm or any specific clustering algorithm. This solution was designed for exploring

*e-mail: tarawneh@cs.uni-kl.de
†e-mail:info@sagagames.de
‡e-mail:humayoun@cs.uni-klde.

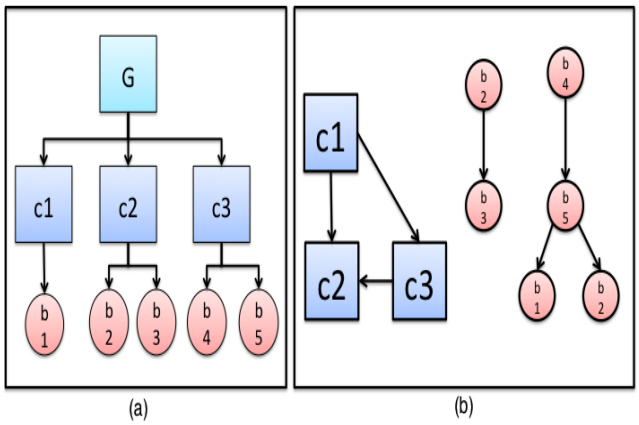

Figure 1: The original graph, showing two types of relations between the graph nodes: *(a)* the structural relations, and *(b)* the adjacency relations.

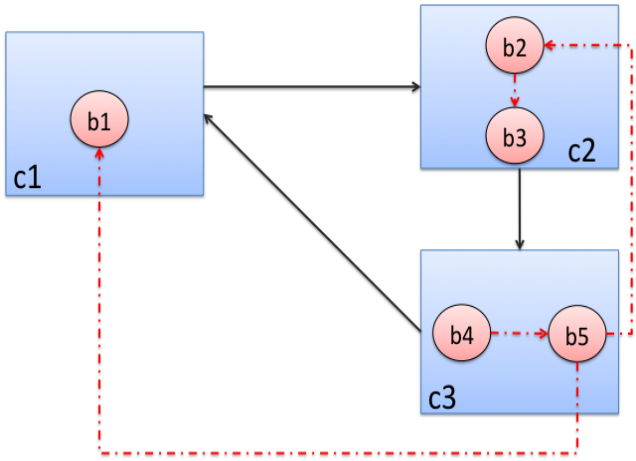

Figure 2: The compound graph representation of the graph presented in Figure 1.

a graph hierarchy space. This was achieved through allowing users to explore different possible hierarchies for the same graph.

3 Background and Motivation

The current embedded systems mostly consist of complex structures due to the tight integration between a number of hardware and software components. This tight integration between the hardware and software parts makes it difficult the task of tracing the system structure, if it is done manually. To help users in understanding and analyzing the underlying system structure, a decomposition phase is normally applied for defining the system main parts [15]. In hierarchical model-based systems, this process is done by refining recursively the sub-components till the desired component.

In current embedded systems, safety and reliability aspects are important for measuring the quality aspects of these systems. Many techniques have been proposed in the literature to support system developers for achieving these aspects. A few examples of these proposed techniques are [13, 12]: the Fault Tree Analysis (FTA), the Event Tree Analysis (ETA), and the Reliability Blocks Diagram (RBDs). FTA is one of the most common techniques for analyzing the safety aspects in embedded systems [4]. FTA depicts the chain, which leads to a specific failure, as a tree structure where to be examined system failure is treated as the root of the tree while the main reasons of this failure are treated as the leaves of the tree. This graphical representation of the failure describes a hierarchical breakdown, with regard to the hierarchy of the failure influences rather than the system structure perspective. In cases of big trees, FTA replaces the sub-tree with a container, called *module*, which represents an independent sub-tree. However, FTA suffers from many problems, e.g., sometimes the tree structure itself is insufficient to model the failure propagation path for such kind of system's hazard that is caused by more than one paths. Consequently, these paths are split into several repeated events in order to preserve the underlying tree structure [13].

To handle the limitations of FTA, Kaiser et al. proposed a new technique [13], called the Component Fault Tree (CFT), that extends the traditional FTs. In CFT, the notation of *component* is used instead of the notation of *module* and these components are connected with each other via ports. It is important to note that in CFT model it is *not necessary* for a component to represent an independent sub-tree, which implies the existence of adjacent relations between the underlying component and other components at the same level or at different levels. Each component in a CFT model represents a real component of the underlying system. The whole CFT model is normally considered as a Directed Acyclic Graph (DAG) of the involved components in the safety scenario [13].

From the above explanation, we can conclude that CFT graphs can easily be modeled as compound graphs, which are one of the most general graph models having the hierarchical structure. A compound graph C normally consists of a set V of nodes, a set E of inclusion edges, and a set F of adjacency edges. The inclusion digraph T can be defined as the rooted tree and consists of a set of tuples, where each tuple is (V,E). The leaves of this inclusion graph T are simple vertices, while the remaining are compound vertices. A *cluster node* represents the sub-tree $T(v)$, rooted in vertex v. While the adjacency graph G is a directed graph and consists of a set of tuples, where each tuple is (V,F). In an adjacency graph, any adjacency edge connects a node to one of its descendants or ancestors in T. Note that the graph G also contains the cluster as vertices [7].

In a CFT model: the set of vertices V contains the list of components, the set of inclusion edges E contains the structural relations between these components, while the set of adjacency edges F contains the failure relations between the system elements. Moreover, a component may contain other components or leaves, where leaves in this context are basic events in the system. As the complexity of an embedded system is increased the corresponding CFT model size is also increased, which makes it difficult to handle the failure detection process. The information visualization field can play an important role in speeding up the system developing process, because it eases the steps of finding important information from both the system and the user perspectives. However, as the size of the graph is increased, the traditional layout algorithm fails in providing a readable visualization. Therefore, there is a need of abstracting the graph to reduce the visual cluttering and to enhance the information extracting process. In addition to the abstraction step, expanding and contraction operations over the produced cluster are required in order to navigate the hierarchy and to get more insight about the deeper level of details. In this work, our proposed CluE algorithm expands the compound nodes in the CFT model in order to help the users (e.g., safety engineers) for getting a better understanding of the underlying system's failure mechanism. Moreover, we provide a pleasant animation for showing the changing process of graph, which is useful for maintaining the user's mental map of the graph.

4 The CluE Algorithm

In this section, we explain the working of our proposed **CluE** (The **Clu**ster **E**xpander of Compound Graphs) algorithm that aims at

Procedure CluE
Input: All nodes and edges with their current positions and sizes, and the current node *N*
Output: New nodes' position and sizes
NL = Empty node list
EL = Empty edge list
for *all child nodes C of the current node N* **do**
  Add *C* to the node list *NL*;
  if *C is expandable* **then**
    Apply algorithm to C (i.e., to its all child nodes);
    Set size of C to the minimal bounding box of its children;
    Save this bounding box;
  end
end
for *all edges E* **do**
  if *E is connected to any two children of the current node N* **then**
    Add E to the edge list EL;
  end
end
Apply *layout algorithm* to the node list *NL* and the edge list *EL*
for *all child nodes C of the current node N* **do**
  if *C is expandable* **then**
    Move children of C by the difference between the saved bounding box of C and the current position (which is the result of the layout algorithm);
  end
end

Algorithm 1: The CluE pseudo code.

expanding the compound nodes in clustered graphs. Algorithm 1 provides the pseudo code of the CluE algorithm. We assume that the information about the underlying graph's vertices and edges is available, in a pre-stage before the execution of CluE. However, CluE cares only about the current node in the graph that is responsible about the current level of details. CluE takes this information and then it calculates the required size of the current level of details based on the number of children of the current node. In the beginning of execution, CluE takes the graph root and its children for calculating the required size. Then CluE calls the layout algorithm in order to calculate the coordinates of child nodes in the current level of details. CluE is called recursively based on the user's request. It is called when a node is triggered by the user to be expanded. After this, CluE performs the following steps.

CluE starts from a specified node *N* in the graph, which represents the current node to be expanded or the graph root in the case of initial stage. Then it finds all the child nodes of this node. First, it adds the child nodes to a list *NL* (Node List), that is empty at the initial stage. All the nodes in this list represent the current level of details and are connected with each other via the set of structural relations of *F* edges. It also keeps the adjacency relations between these nodes in a list data structure *EL* (Edge-List), that is initially empty. CluE works iteratively, as it layouts the graph nodes level by level. It checks if one of the child nodes in *N* is needed to be expanded, then it calls itself to add the child nodes of this targeted node in the node list *NL*. After that, it estimates the size of this expanded node based on the number of its children and then it saves this size for later usage. At this stage, CluE adds the adjacent relations related with the internal structure of the expanded node to the edge list *EL*.

In the next step, CluE calls the layout algorithm. In fact, CluE does not rely on any specific layout algorithm, as it requires only the coordinates of nodes in the graph. Therefore, it works perfectly

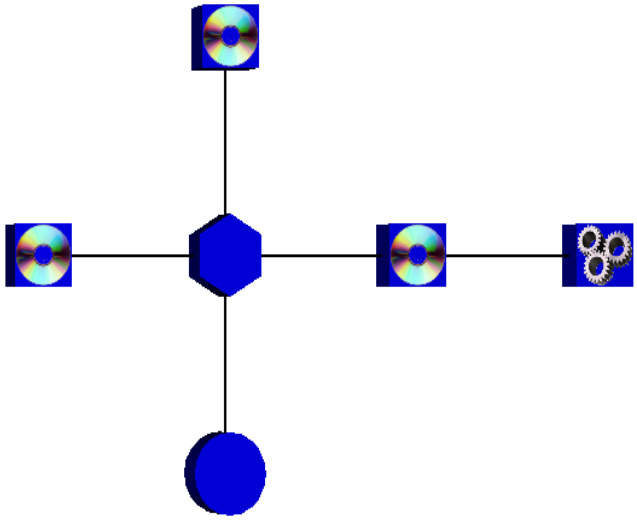

Figure 3: The highest level of details of an CFT model without expanding any node.

with many different layout algorithms (e.g., the orthogonal layout, the Sugiyama algorithm, the grid layout algorithm, or any multi-level layout algorithm). We tested CluE with all these mentioned layout algorithms and it performed well in all cases. After this step, CluE takes the coordinates of graph nodes and then it shifts these nodes in two directions according to the size of the bounding box of the expanded node, which was saved in the previous stage of CluE.

The main limitation of CluE is that it calls the layout algorithm whenever there is a need to expand a new node. This requires calculating new positions each time. However, CluE keeps fixed the layer of the nodes. This affects on maintaining the user's mental-map of the layout [9]. In order to help users in tracking the nodes at previous positions, we added a slow animation to show the slight changing in nodes' positions. As a future work, we aim to optimize the used layout algorithms for keeping the coordinates of nodes in order to maintain a better mental-map. Also, we intend to analyze the CluE performance on large compound graphs.

5 Results

In this section, we discuss results of our proposed CluE algorithm when it is called on the abstract representation of a CFT model. Figure 3 shows the most abstracted level of a CFT model. In this figure, the presented graph shows those direct components that trigger the critical safety situation in the underlying system. The root of this model represents a critical safety situation, which is called the "TopEvent" in the safety analysis domain. As a part of the safety analysis process, it is important to find those safety critical components that could lead to this unpleasant system situation. Therefore, CluE offers the facility to expand the targeted component and allows the users to go deeper in the graph hierarchy for showing the in depth details on-demand. For example in Figure 3, we show the highest level of details in the current CFT model for describing one possible safety situation in the underlying system. In this figure, the Hexagon represents the TopEvent, the blue circles represent the Basic Events (i.e., the simplest structure in a CFT model for representing leaves of the graph), the blue rectangles represent the system components, and the texture on each rectangle represents the type of this component (i.e., either software or hardware). As described earlier, a component may contain many sub-components or Basic Events; therefore, these structures are expandable.

When a user expands the targeted component a new view of the graph is displayed, which shows another level of details by provid-

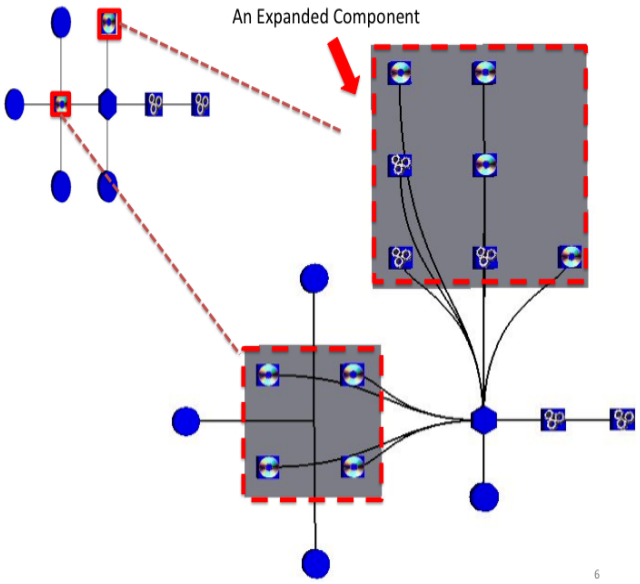

Figure 4: The process of expanding the required nodes using the CluE algorithm. The abstract view is shown at the top left side of the figure with smaller size, while the expanded view is shown at the center of the figure.

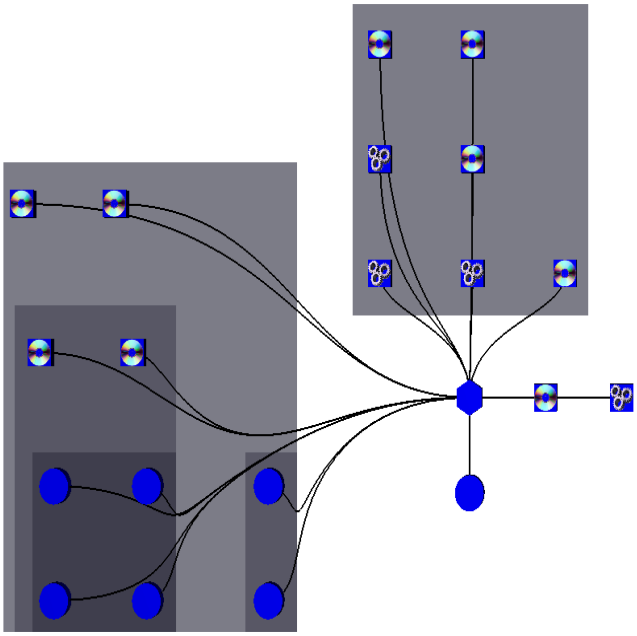

Figure 5: An illustration of the graph representation after expanding three compound nodes. The color saturation is used to show the parent-child relationships between the components and their sub-components and the failure relations are visualized using explicit edges between the graph nodes.

ing the internal structure of this targeted component (see Figure 4). Figure 5 shows the visual representation of structural relations between the different components, which are visualized implicitly using the containment relation representation such that the parent node contains the child nodes. Moreover, after expanding, extra failure relations that were not included in the previous view are then displayed. These failure relations are appeared on the visual representation, because CluE adds the adjacency relations upon their presence in the graph during the expanding process. CluE scales well with different number of levels in the graph hierarchy, as we show in Figure 5 the three levels of details of the graph after expansion. We designed the visualization in such a way that the color saturation of the node depends on its level, e.g., a higher level of details looks more transparent than the lower level of details. This difference in color saturation helps in defining the structural relations between the components at different levels. We also use the spline curves for showing the adjacent relations between nodes at different level of details. This design decision was made because it helps in reducing the crossings rate between the adjacent edges [10].

In order to evaluate the expanding and contracting facility through our proposed CluE algorithm, we conducted a brief user evaluation study. We invited 25 subjects from different backgrounds for performing exploration tasks over the graph representation of a CFT model. The tasks mainly were focused on extracting some safety information about the corresponding components at different levels.

First, we trained each subject that how to use the tool and then we provided the required background they needed to achieve the given tasks. We explained them the tasks and then asked them to provide us their feedback. After it, we also asked the subjects to give us their feedback through a closed-ended questioners form. Subjects were asked to select a number between 1 and 5 (1 meant strongly disagree while 5 meant strongly agree) based on the likert scale to rate their performance. 21 subjects found the expanding and contracting node operation using CluE algorithm as intuitive and easy to follow. Their rating of their own performance was either 4 (i.e., agree) or 5 (i.e., strongly agree). While the remaining 4 subjects rate themselves either 2 (i.e., disagree) or 3 (i.e., neither agree nor disagree), as they found some difficulties in tracing the transitions of nodes. Regarding the trust level of the tool, 19 subjects assigned their trust level to be either 4 or 5, while the remaining 6 subjects requested more time in order to judge their trust level of the tool. Overall, the results of this user evaluation study show the intuitiveness of our approach and indicate a high acceptance ratio from the users' perspective. However, there is a need of extended detailed evaluation studies to generalize the findings.

6 Conclusion

In this work, we presented the CluE algorithm that expands clusters in compound graphs. The main contribution of this algorithm is its ability of expanding the clusters regardless of the used layout algorithm in the background. It produces an area-aware layout, as it takes care of the size of the expanded node before arranging the child nodes in their final positions. Additionally, it keeps the unexpanded nodes in their corresponding layers. However, a slight difference in the unexpanded nodes' coordinates might occur.

We added a slow animation for showing gradually the graph nodes' transition process in order to help the users in maintaining their mental-map of the underlying graph. Moreover, we applied the CluE algorithm in real graphs for navigating through compound graphs representing the failure mechanisms in embedded systems. The results of the brief user evaluation study show a high user-acceptance ratio, as they were able to distinguish between the structural relations and the failure relations between the nodes at different hierarchy levels. In future, we aim to test the presented algorithm in large-sized graphs with different layout algorithms.

Acknowledgment

We would like to thank Jens Bauer from the University of Kaiserslautern for the technical support during the implementation phase. Also, we would like to thank DFG (Deutsche Forschungsgemeinschaft) for the financial support of the project.

REFERENCES

[1] J. Abello, F. van Ham, and N. Krishnan. Ask-graphview: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[2] D. Archambault, T. Munzner, and D. Auber. Topolayout: Multilevel graph layout by topological features. *IEEE Transactions on Visualization and Computer Graphics*, 13:305–317, 2007.

[3] D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, July 2008.

[4] M. Bozzano and A. Villafiorita. *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, 2010.

[5] P. Eades and M. L. Huang. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms and Applications*, 4:157–181, 2000.

[6] Q.-W. Feng, R. F. Cohen, and P. Eades. Planarity for clustered graphs. In *ESA*, pages 213–226, 1995.

[7] I. M.-A. Fuhrmann. Layout of compound graphs. *Diploma thesis from Department of Computer Science Real-Time and Embedded Systems Group, Christian-Albrechts-Universitt zu Kiel*, 2012.

[8] N. Gelfand and R. Tamassia. Algorithmic patterns for orthogonal graph drawing. In S. Whitesides, editor, *Graph Drawing*, volume 1547 of *Lecture Notes in Computer Science*, pages 138–152. Springer, 1998.

[9] C. Helen, Purchase, E. Hoggan, and C. Gorg. *How Important Is the Mental Map? An Empirical Investigation of a Dynamic Graph Layout Algorithm*, volume 4372/2007, pages 184–195. Springer Berlin / Heidelberg, 2007.

[10] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006.

[11] M. L. Huang and P. Eades. A fully animated interactive system for clustering and navigating huge graphs, 1998. 10.1007/3-540-37623-2_29.

[12] B. Kaiser, C. Gramlich, and M. Forster. State/event fault treesa safety analysis model for software-controlled systems. *Reliability Engineering System Safety*, 92(11):1521–1537, 2007.

[13] B. Kaiser, P. Liggesmeyer, and O. Mckel. A new component concept for fault trees. *Reproduction*, 33:37–46, 2003.

[14] J. Katreniaková. Mental map preserving cluster expansion. In *SOFSEM (2)*, pages 58–69, 2008.

[15] E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. Lee and Seshia, 1 edition, 2010.

[16] M. Raitner. Visual navigation of compound graphs. In J. Pach, editor, *Graph Drawing*, volume 3383 of *Lecture Notes in Computer Science*, pages 403–413. Springer Berlin Heidelberg, 2005.

[17] E. M. Reingold and S. Tilford. Tidier drawing of trees. *IEEE Trans. Software Eng*, 1981.

[18] K. Sugiyama and K. Misue. Visualization of structural information: automatic drawing of compound digraphs. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(4):876–892, 1991.

[19] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *Ieee Transactions On Systems Man And Cybernetics*, 11(2):109–125, 1981.