Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 44 (2015) 599 – 608
Procedia Computer Science

2015 Conference on Systems Engineering Research

Integrated Matrix-Based Fault Tree Generation and Evaluation

Michael Roth*[a], Moritz Wolf[a], Udo Lindemann[a]

a Technische Universität München, Institute of Product Development, Boltzmannstr. 15, 85748 Garching, Germany

Abstract

Increasing complexity of products and safety regulations combined with an increasing amount of variants complicates the process of safety analysis within systems engineering. Moreover, it is known that the early avoidance or prevention of failures saves costs and improves the quality. As methods of safety analysis, i.e. fault tree analyses require immense manual efforts and expert knowledge, the efficiency of these analyses has to be improved. Our paper thus presents an approach to generate and evaluate fault trees by the usage of matrix-based models. It is an approach tailored to the early phases of system design and provides a preliminary fault tree analysis. It automatically generates fault trees and evaluates them. Thus, it facilitates the efficient identification of safety critical elements and the assessment and comparison of alternative system architecture concepts. This paper provides a brief introduction to fault tree analysis and presents existing approaches to automate the generation or synthesis of fault trees. The limitations of these approaches during early stages of design are discussed and the need for a tailored approach is derived. The developed approach consists of four phases and six steps which each are explained in detail. The whole approach is validated within a small industrial case study and its benefits and limitations are discussed. The case study shows, that the approach successfully improves the efficiency of a preliminary fault tree analysis.

Peer-review under responsibility of the Stevens Institute of Technology.
Keywords: Fault Tree Analysis; Design Structure Matrix; Functional Modeling; Safety;

1. Introduction

Customers' expectations and requirements as well as their variance are steadily increasing. This is one reason, why the complexity of products and systems grows[1]. Thus, companies are under pressure to offer even more variants, which increases the size of their product portfolio. As a result we observe a hardly manageable amount of variants and evolutionary grown complex systems in the industry[2]. Combined with stricter safety regulations this leads to immense efforts for safety analysis and approval. For each new system variant, the process of safety analysis and approval has

* Michael Roth. Tel.: +49-89-289-15129; fax: +49-89-289-15144.
E-mail address: michael.roth@pe.mw.tum.de

Peer-review under responsibility of the Stevens Institute of Technology.
doi:10.1016/j.procs.2015.03.027

to be repeated. However, traditional deductive methods of safety analysis strongly rely on the experience of expert analysts[3] and focus on the design validation stage[4]. This limitations become more and more critical with increasing complexity of the system[3].

Recent publications in systems engineering acknowledge, that the consideration of safety aspects should be shifted to the early stages of the system design process and develop suitable methods and support (such as[3,4,5]). Safety analyses in early stages provide large value adding potential, but are difficult to perform due to the high level of uncertainty[6]. A challenge thus is, how classical methods like fault tree analysis (FTA) or failure mode and effect analysis (FMEA) can efficiently serve for a preliminary safety analysis in the early stages of design. To cope with uncertainty and a wide solution space, the required experience and manual efforts of these methods have to be reduced to facilitate the analysis and assessment of alternative concepts[7]. Thus, the objective of this paper is how the generation of preliminary fault trees and their evaluation can be automated to reduce the required efforts during the early stages of design.

In this paper we first provide a brief introduction to FTA and point out the interconnections between FTA and FMEA. We then discuss existing approaches to automate the generation of fault trees and their limitations. Based on that, we develop our approach which founds on established matrix-based models and methods. We apply and validate the approach in a simple case study on a cordless screwdriver. The paper concludes with the discussion of the findings and applicability of our approach as well as recommendations for further work.

2. Failure Analysis in Early Stage

2.1. Safety Analysis in Design – Classical Methods

The fault tree analysis (FTA) is a classical and standardized method applied to the safety analysis of systems (IEC61025[8]). It identifies conditions that may cause or contribute to the occurrence of an undesired top event and represents them in a graphical form[7,8]. Being a deductive method, it identifies an undesired top event and starting from there creates a tree structure of possible causes. The dependencies of multiple causes are modeled by Boolean logic gates. The tree's branches are followed down to the basic events. They represent failures in the system's components or elements. Thereby, the impact of a failure on multiple events, called common cause, can be identified[8]. Using analytic methods, all possible combinations of basic events that cause the top event can be determined. These combinations are called cut sets[8]. If the removal of one basic event will break the impact on the top event, a cut set is considered a minimal cut set[8]. As limitations, many researchers name the usually high manual efforts and experience, which are needed to perform the fault tree analysis[3,7,9].

Moreover, international norms recommend the combination of the deductive FTA with inductive methods like FMEA (IEC60812[10]) in order to ensure a comprehensive safety analysis[8]. The link between FMEA and FTA is provided by the basic events: each basic failure mode in the FMEA which causes a system failure has to be represented by a minimal cut set in the fault tree. Also each basic failure of the fault tree has to be considered in a complete FMEA[8]. Thus, the challenge is not only to automate the generation of the fault tree but also to derive the minimal cut sets and automatically evaluate them to identify the most relevant system elements and to improve safety.

2.2. Model-Based Generation of Fault Trees – Existing Approaches

Due to the large complexity of systems, the manual activity of generating fault trees usually requires immense efforts. During the last years various approaches to automate the generation of fault trees based on system models have been published. A condensed overview on the most relevant approaches can be found in Mhenni et al.[9] and Majdara et al.[7] give an extensive reference list of approx. 20 approaches. We thus just provide a short overview on the most relevant concepts in the context of complex systems engineering. In the following they are classified by their modeling approach and we point out their advantages and limitations from our point of view.

SYSML-based Approaches

SysML is one of the main modeling languages used in the field of systems engineering, yet only few approaches to generate fault trees from these models are published.

Xiang et al.[11] present an approach to derive fault trees form SysML. Based on use cases, the system and its functional dependencies stored in the internal block diagram (ibd) and sequence diagram (sd) are analyzed. To bridge the gap between system architecture and reliability analysis, they use a reliability configuration model and a static fault tree model. These models enable the automated fault tree generation and avoid problems through dynamic dependencies. Their approach provides a successful mechanism to generate fault trees from existing SysML models. But using specially defined stereotypes, it is strongly tailored to IT-systems. This fact limits the transfer to mechatronic systems from our point of view. Moreover, the usage of the reliability configuration model is not possible with the data available in early stages of design.

Mhenni et al.[9] follow a similar approach and translate the structural SysML diagrams into fault trees. They model components and their interfaces in an ibd. Standard flow ports are used to model the interactions between components. These diagrams are transformed into a directed graph and a graph traversal algorithm combined with recognition of characteristic patterns is used to automatically derive fault trees. This approach also generates fault trees of existing SysML models. Yet, this component-oriented approach requires knowledge, which might not be available in an early stage. Moreover, the manual modeling of dependencies between components in SysML still requires huge efforts. This can impede the exploration of alternative architectures in the solution space.

Simulink-based Approaches

Papadopoulos and Maruhn[12] propose a method to automatically generate fault trees from Simulink models. Using a hierarchical system structure and dependencies between components, their algorithm generates the system's fault trees. To identify failures, they use a form of computer HAZOP and examine each system element in detail. Thereby, all possible deviations in the output are determined. The approach improves the identification of hazardous dependencies between components and thus also facilitates the safety analysis. They claim, that their method is also applicable at the early design stage, when the system model is an abstract representation and functional description of the system. However, while they efficiently decrease the efforts to generate a fault tree, the modeling and preparation efforts still are very high. A detailed analysis of each element and profound knowledge is still required. Thus, the efficient application of this method for explorative examinations is limited.

Tajarrod and Latif-Shabgahi[13] propose a method to synthesize fault trees from Simulink models as well. They enhance the Simulink model with information on failure behavior and automatically derive a fault tree. The components are manually modeled and classified according to their impact on the top event. They moreover define "usual" and "redundant" subsystems in the model before the fault tree representation is created. Basing on components, their behavior and requiring enormous manual efforts, this approach is a suitable method to visualize the fault trees of Simulink models. Yet, the approach does not manage to reduce manual efforts during the analysis of dependencies and thus is not suitable for applications in the early phase of complex system design.

Approaches Based on Other Modeling Languages

Majdara and Wakabayashi[7] automatically generate fault trees form special models. They found their approach on a directed graph which represents the components and their dependencies (i.e. flows in-between). A trace-back algorithm is applied to generate the fault tree and the functional behavior of the components is modeled by functional tables. Thereby, they generate detailed component-based fault trees including specified failure conditions. However, its non-compatibility with common modeling languages in systems engineering and the high efforts needed for the modeling limit its applicability especially for early stages of system design.

Various approaches using other special modeling languages are published. For example Bieber et al.[14] present a method to derive fault trees from models built in the modeling language AltaRica. Yet, this language reaches its limitations when models of large and complex systems are built. Joshi et al.[15] generate fault trees from models in Architecture Analysis and Design Language (AADL), which is tailored for embedded systems and thus not fully suitable for complex systems engineering.

Approaches based on Design Structure Matrices

The Design Structure Matrix (DSM) and Domain Mapping Matrix (DMM) are a well-known tool to model the dependencies of system elements and are suitable for various applications[16,17]. Eppinger et al.[18] point out the advantages of these tools in modeling multilevel interactions. They expect large potential to improve systems

engineering processes and i.e. hazard and operability studies[18]. However, no approach to generate fault trees using DSM and DMM is known to us. Höfig and Guo[19] use DSMs to support the generation of fault trees by eliminating loops in the model, but they do not derive fault trees from the information stored in the DSM.

The previously presented approaches mainly focus on the generation of fault trees based on component data. Most of them do not allow to use abstract and functional concepts to assess potential safety. Moreover, most approaches require extensive models. Only if these models exist, the approaches can reduce the overall manual efforts for safety analysis. Only improved efficiency can facilitate the assessment and comparison of different system concepts. In this step, matrix-based models are commonly used to identify modules and evaluate alternative architectures[16]. Especially facing the increasing variance, this is an important step in system design. To realize an early integration of safety, methods of safety analysis should be integrated in that step. However, no suitable method using matrix-based models is published. Thus, the objective of our contribution is to develop an approach, which provides an automated preliminary FTA in order to assess the safety of modules and alternative architectures. It will be tailored to the early stages of design in order to identify safety critical system elements as early as possible.

3. Approach

Our approach founds on the methodology of Structural Complexity Management (StCM)[1]. It is a widely spread method to handle complex systems and structural dependencies. StCM combines the concepts of Design Structure Matrix (DSM)[16] and Domain Mapping Matrix (DMM)[17] by arranging them in the Multiple Domain Matrix (MDM) framework[1]. It provides a five phase procedure to analyze and optimize system structures. These phases are system definition, information acquisition, deduction of dependencies, structure analysis and application on product design[1].

As StCM is an established and proven approach to handle the challenges of structural complexity, we choose it as foundation for the analysis of structural relations between failures in complex systems. The generic procedure is adapted to our specific task. This adaption results in a four phase, six step procedure which is illustrated in Fig. 1. Starting with system definition and information acquisition, the system of interest is modeled. After deriving the indirect dependencies of failures, they are analyzed by generating fault trees identifying minimal cut sets. These results are evaluated and visualized. In the following paragraphs we discuss each of these six steps in detail.

3.1. System Definition – Setting up the Multiple Domain Matrix (MDM)

The main purpose of the system definition phase is to set up the MDM. This includes the definition of the domains and its relevant relations. In Fig. 2 the basic MDM of our approach is drawn. It includes the domains of functions, generic flows or states, failures, Boolean logic gates and minimal cut sets.

To match with the data available in the early phase of design, the approach models the system by its functions and generic flows. The domain of generic flows sets up the system's structural dependencies by two relations. It records which flows are produced or influenced by a function as well as which flows are input to or influence a function. In this context generic flows represent the abstraction of all information, energy and material flows. These two domains and their relations are recorded and modeled as direct dependencies.

The domain of failures can either be computed or recorded. Details on these alternatives will be discussed in the phase of information acquisition. Depending on the chosen source the dependencies of functions and failures either directly result from the computation or are recorded as direct dependencies.

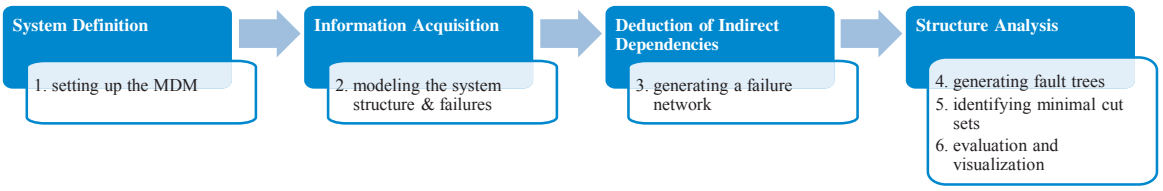

Fig. 1. Procedure of generating and evaluating fault trees

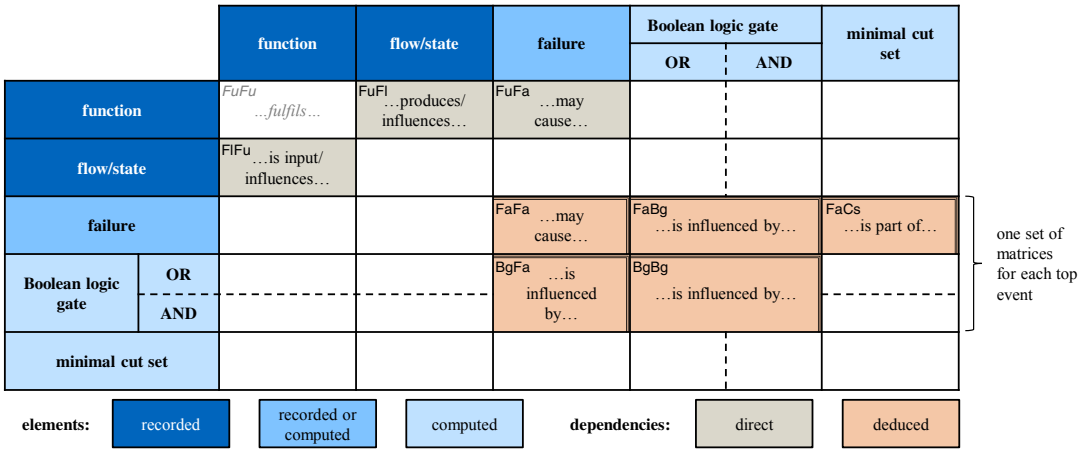

Fig. 2. Multiple Domain Matrix (MDM) with required domains and relations

While the paragraphs above describe the basic procedure, our approach has the advantage that the domains and relations can be adapted to the available data. Following for example the approach of modeling evolutionary grown systems[2], possible other domains are components, information and states. As long as the following three aspects are included in the MDM, the approach is applicable:

- domain of system elements: functions (basic procedure); alternatives for example are components or modules.
- domain and/or relations of structural dependencies: generic flows and their relation to functions (basic procedure); an alternative for example are direct relations between system elements.
- relation between system elements and failures

Besides these aspects, the basic MDM in Fig. 2 shows the domain of the Boolean logic gates, including “AND” and “OR” gates. Other Boolean logics as “XOR” etc. are not considered in this approach. Following the FTA methodology described in section 2.1., the Boolean logic gates represent the logical dependencies in the fault trees. The domain minimal cut sets represent all cut sets which can be considered as minimal according to the definition given in section 2.1. The relations “failure may cause failure” (DSM FaFa), “Boolean logic gate is influenced by failure/Boolean logic gate” (DMM BgFa/DSM BgBg) and the assignment of failures to minimal cut sets (DMM FaCs) will be derived during the phases three and four of our approach. Even though a regular fault tree does not have connections between Boolean logic gates, our approach has to include those in the DSM BgBg: Due to the automated generation of failures, not all combined events caused by a Boolean logic gate are modeled as a system element. However, the resulting propagation still is included in the model by structural dependencies. Thus, a fault tree created by approach can have direct connections between two Boolean logic gates.

During the definition of the MDM and its domains the scope of the failure analysis should be specified. Therefore, the system of interest’s borders and the aspired detail level of the failure analysis have to be defined.

3.2. Information Acquisition – Modeling the System Structure and its Failures

According to the detail level defined in phase one, this second phase models the structure of the system by recording the elements of the function and flow/state domain together with their direct dependencies in the DMMs FuFl and FlFu. To simplify the modeling and to improve model’s quality, we suggest to use a suitable support which can be chosen from the large amount of published approaches (e.g.[1,2,20]).

The essential task in this phase is the definition of failures and their relation to functions. Here our approach offers alternatives according to the chosen level of detail:

- high detail level: Following the definition of a failure according to ISO1901121, in a detailed analysis all requirements connected to a function are examined and every possible failure which can cause a noncompliance is recorded. In that case failures and their dependencies to functions are recorded directly.
- lower detail level: In the early phase of design, the definition of failures by the negation of each function can be sufficient[22]. This step can be automated to reduce the manual efforts. In that case, the elements of the domain of failures will be computed and the resulting DMM FuFa will have entries along the diagonal only. To improve the detail level, the negation can be extended to two types: (1) the function is not executed and (2) the function is not executed properly. Yet, it has to be mentioned, that this will double the number of failures in the model and the amount of data to be processed will strongly increase.

3.3. Deduction of Indirect Dependencies – Generating a Failure Network

Based on the basic MDM, in this step, the dependencies between failures (DSM FaFa) can be deduced:

$$FaFa = FuFa^T \times FuFl \times FlFu \times FuFa \quad (1)$$

These deduced dependencies can be deduced as follows: “one failure may cause another failure, if the function, which may cause the first failure, produces a flow, which is input to another function, which can have the second failure”. The DSM FaFa represents the failure network and includes all possible failure propagations. As the deduction bridges three domains, it might be useful in some applications to check the dependencies between functions on plausibility before deriving the dependencies between failures.

3.4. Structure Analysis – Generating Fault Trees

To extract the fault trees from the failure network in DSM FaFa, the top events have to be identified. The optional hierarchical function decomposition (DSM FuFu) can support this identification process. Functions on top hierarchical level are main functions and thus, failures in this functions can be considered as top events.

The failure network in DSM FaFa now includes all top events. Thus, for each top event and all its possible causes a new DSM $FaFa^T$ (failure chain) is extracted. At this stage it is important to transpose the DSM FaFa. Until now, the dependencies between failures due to the deduction via generic flows have an inductive character. The transposition converts those according to the deductive character of the FTA. Moreover the distance matrix[1] of the transposed DSM FaFa is derived. The transformation into the top event-specific DSMs then only incorporates the dependencies of elements which have a defined distance to the top event failure. Dependencies of elements without a defined distance to the top event are neglected. Fig. 3 illustrates this process with a simple example. Occurring cycles in the failure chain can be identified on the diagonal of the DSM FaFa and its products during the calculation of the distance matrix (for details see[1]). Our approach breaks them by removing the dependency between the two elements with the smallest and largest distance to the top event.

To transform the resulting failure chains into fault trees, at each output of the failures one Boolean logic gate has to be inserted. Thus, for each failure in the failure chain with an active sum larger than zero, an element in the domain of Boolean logic gates is created. As explained in Fig. 4, simultaneously a dependency between the failure and the Boolean logic gate is set, and the outgoing influences of the failure are transferred to the Boolean logic gate. This process is automated by using OR gates only. To explore the safety aspects in an early stage of system design, the exclusive use of OR gates represents a worst case scenario and helps to identify safety critical elements. If redundancies or events resulting from unique combinations of failures have to be modeled, the integration of AND gates still is a manual task. As described in section 3.1, this may result in dependencies in the DSM BgBg.

3.5. Structure Analysis – Identifying Minimal Cut Sets

To identify the minimal cut sets, we use the algorithm proposed by Hauptmanns et al.[23]. It requires the DMM BgFa and the DSM BgBg as input information, and only the basic events (active sum equal to zero) in the failure domain

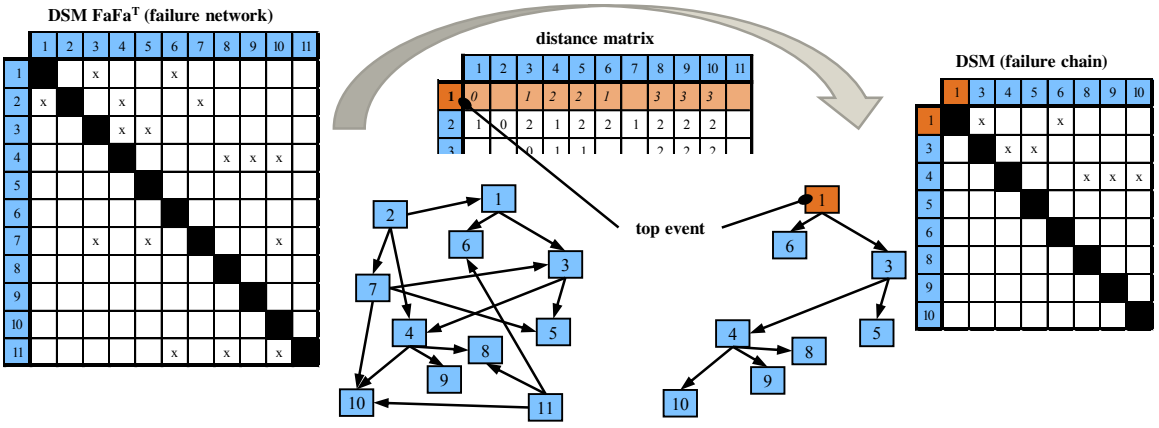

Fig. 3. Conversion of the failure network via the distance matrix to a failure chain

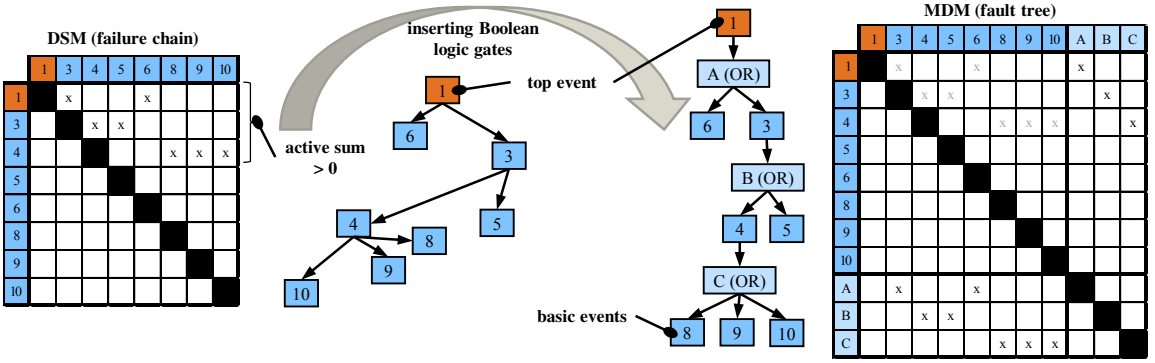

Fig. 4. Inserting Boolean logic gates into the failure chain to transform it into a fault tree

have to be considered. Starting from the Boolean logic gate which is closest to the top event, the iterative algorithm replaces the influencing gates by their own influences (primary failures or other logic gates). Each OR gate thus results in an additional cut set. By this all possible combinations which may cause the top event failure are identified and for each of them a new element in the domain of minimal cut sets is created. The basic events included in the minimal cut sets are recorded as dependencies in the DMM FaCs.

3.6. Structure Analysis – Evaluation and Visualization

Due to the abstract definition of failures in section 3.2., it is not possible to deduce probabilities and reliabilities of events. However, classical methods of failure analysis like FMEA define the criticality of a failure by its probability, its severity and its detection[10]. During the early stages of design at least the severity of a failure can be estimated. Therefore within our approach we consider a basic event as severe, if it has a strong individual impact on the main event and if it is often involved in minimal cut sets, causing the main failure.

A strong individual impact on abstract level is given, if the distance from basic to main event is small. To classify the basic events according to that, the distance matrix of the failure chain DSM is evaluated. For each basic event, the distance d is normalized to the interval [1; 10] considering the maximum occurring distance d_{max}:

$$d_n(failure) = (d-1)\frac{10-1}{d_{\max}-1}+1 \qquad (2)$$

The occurrence in minimal cut sets o is used for the second classification. Therefore, the active sum of each basic event in the DMM FaCs is evaluated. Same as in the first criteria, the values are normalized to [1;10]. The maximum number of occurrence o_{max} is normalized to 1, whilst the minimal occurrence o_{min} is represented by 10:

$$o_n(failure) = (o_{max} - o)\frac{10-1}{o_{max} - o_{min}} + 1 \quad (3)$$

Based on both classifications, the fault trees can be assessed and the impact of failures in alternative system concepts can be compared. To visualize the results we propose the arrangement of basic events in the portfolio shown in Fig. 5. To support the interpretation a scale of combined criticality is inserted. Based on the assumption, that a failure which can directly cause the top event and the failure with maximum occurrence in the minimal cut sets are most critical, hyperbolic curves can enhance the portfolio. They for example can be used to describe the severity of a basic event in a scale from 1 to 10, as it is often used within a FMEA.

4. Evaluation

The evaluation of a safety analysis method in the early stage of design is challenging. Therefore we use an existing product (cordless screwdriver) to test and evaluate our approach. We build a functional model as it is used in an early stage of design, even though the knowledge on the components is given. This enables us to compare the results of the approach with the failure analyses done after the detailed design and in field experience. In a second approach, we test the adaptability of our approach and model the system with knowledge of the components and final product design. Both results are discussed with an expert involved in the safety analysis and product validation of the screwdriver.

The abstract functional modeling results in approx. 30 failures, while the second modeling o results in a failure network consisting of 179 failures. We define the top events and derive fault trees following our approach. In the abstract case, no redundancies occur, so the manual insertion of AND gates is not necessary. In the larger network, redundancies occur and we have to manually perform the process of adding the AND gates. In both cases, minimal cut sets are identified and we evaluate the results using our proposed visualization. Fig. 6 depicts the results of the component oriented modeling for the top event “torque is not transferred to screw”. In the visualization the failures are clustered in classes according to their occurrence and individual contribution.

The discussion of the results with the expert shows: The component oriented approach identifies critical failures which in the actual safety analysis also have been identified as critical. This verifies the general procedure of the approach. The abstract modeling also is able to identify critical functions (e.g. the superposition of torque and

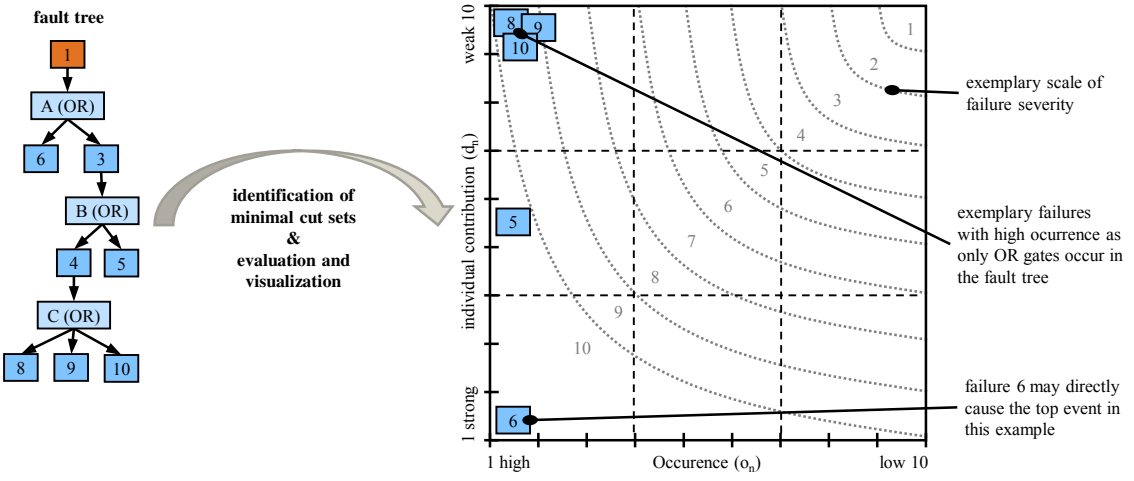

Fig. 5. Proposed portfolio to visualize the evaluation of the fault tree

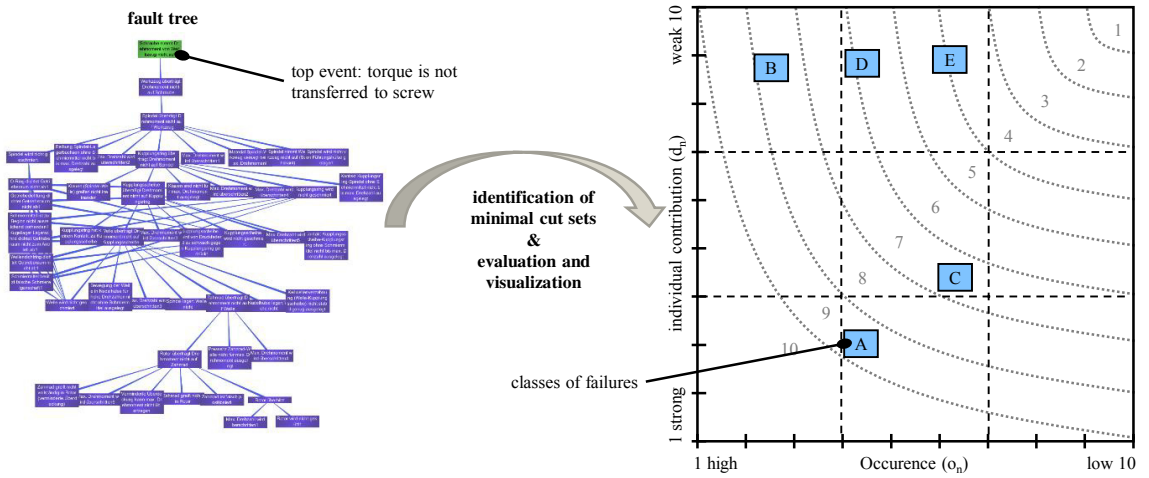

Fig. 6. Visualization of the failure chain and the visualization for a top event of the evaluation

axial force). This preliminary results help to identify the critical functions so that during component design, the critical influence of the components can be considered for dimensioning. However, the two applications show, a too abstract modeling will limit the quality of the results.

5. Discussion and Outlook

This paper presents and validates a matrix-based approach to generate preliminary fault trees during the early stages of design. As shown in the case study, even a small system can result in a huge amount of data which can hardly be analyzed manually. The approach helps to handle this amount of data and to identify critical failures or elements from a structural point of view. Its high grade of automation successfully reduces manual efforts and required experience. It, thus allows the assessment and comparison of alternative architecture concepts. Moreover, as the structural dependencies can be modeled flexibly, the approach can integrate deductive and inductive methods. In comparison to the works discussed in section 2, our approach has the following advantages:

- integration in matrix-based methods which are commonly used in early phases and during the evaluation of alternative architectures
- ability to handle alternative input data
- automation of failure definition
- transformation of the analysis' results to a simple and handy visualization

Even though our approach helps to manage the huge amount of information, the resulting fault trees consist of a large amount of elements and are hard to interpret for a human. The human cognitive process of neglecting unessential aspects to reduce complexity cannot be incorporated in the automated approach. In this context, it also has to be mentioned, that the manual integration of AND gates still requires much efforts. However, the automated fault trees can structure the considerations and support the manual process. In ongoing research we search for a solutions to automate this process as well.

The hierarchical dependencies of functions are inherently considered within our approach. Yet, the modeling of flows via multiple hierarchical levels is very challenging. The intuitive approach would be to model flows within one hierarchical level. In that case, our approach will only generate an inductive failure propagation network on one hierarchical level. This fact complicates the application of our approach and is one major limitation. Our future research will incorporate the functional decomposition into the approach to simplify the modeling of multi-level flows.

In further work, the validity of the derived metrics and visualization has to be examined. Also the approach will be evaluated with larger and more complex systems. Finally, one tool supporting and conducting all phases of the approach needs to be developed, as in current stage still multiple tools are involved.

For the application, the case study shows, that the abstract definition of failures limits the quality of results. Yet, this quality in an early stage of design still can be sufficient. Thus, before each individual application of the approach, the tradeoff between efforts, quality and uncertainty needs to be evaluated.

References

1. Lindemann U, Maurer MS, Braun T. Structural Complexity Management. Berlin: Springer; 2008.
2. Roth M, Kasperek D, Lindemann U. Functional Analysis and Modeling of Complex, Evolutionary Grown, Mechatronic Products. 2013 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) 2013; 346–50.
3. Sierla S, Tumer I, Papakonstantinou N, Koskinen K, Jensen D. Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework. Mechatronics 2012; 22(2):137–51.
4. Jensen DC, Tumer IY. Modeling and Analysis of Safety in Early Design. 2013 Conference on Systems Engineering Research 2013; 16(0):824–33.
5. Leveson N. Engineering a safer world: Systems thinking applied to safety. Cambridge, Mass.: The MIT Press; 2012.
6. Würtenberger J, Kloberdanz H, Lotz J, Ahsen A von. Application of the FMEA During the Product Development Process - Dependencies Between Level of Information and Quality of Result. In: Marjanović D, Storga M, Pavković N, Bojcetić N, (eds). Proceedings of the DESIGN 2014 13th International Design Conference. Glasgow: Design Society; 2014, p. 417–426.
7. Majdara A, Wakabayashi T. Component-based modeling of systems for automated fault tree generation. Reliability Engineering & System Safety 2009; 94(6):1076–86.
8. IEC. Fault tree analysis (FTA). 2nd ed. (61025). Geneva: International Electrotechnical Commission; 2006.
9. Mhenni F, Nguyen N, Choley J. Automatic fault tree generation from SysML system models. In: IEEE/ASME, (ed). International Conference on Advanced Intelligent Mechatronics (AIM); 2014, p. 715–720.
10. IEC. Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA). 2nd ed. (60812). Geneva: International Electrotechnical Commission; 2006.
11. Xiang J, Yanoo K, Maeno Y, Tadano K. Automatic Synthesis of Static Fault Trees from System Models. In: 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement. Piscataway: IEEE; 2011, p. 127–136.
12. Papadopoulos Y, Maruhn M. Model-based Synthesis of Fault Trees from Matlab-Simulink Models. In: International Conference on Dependable Systems and Networks (DSN 2001). Los Alamitos: IEEE Computer Society; 2001, p. 77–82.
13. Tajarrod F, Latif-Shabgahi G. A Novel Methodology for Synthesis of Fault Trees from MATLAB-Simulink Model. World Academy of Science, Engineering and Technology 2008; 17(5):1234–40.
14. Bieber P, Castel C, Seguin C. Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System. In: Goos G, Hartmanis J, van Leeuwen J, Bondavalli A, Thevenod-Fosse P, (eds). Dependable Computing EDCC-4. Berlin, Heidelberg: Springer Berlin Heidelberg; 2002, p. 19–31.
15. Joshi A, Vestal S, Binns P. Automatic generation of static fault trees from aadl models.
16. Browning TR. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. IEEE Transactions on Engineering Management 2001; 48(3):292–306.
17. Danilovic M, Browning TR. Managing Complex Product Development Projects with Design Structure Matrices and Domain Mapping Matrices. International Journal of Project Management 2007; 25(3):300–14.
18. Eppinger SD, Joglekar NR, Olechowski A, Teo T. Improving the systems engineering process with multilevel analysis of interactions. AIEDAM 2014; 28(04):323–37.
19. Höfig K, Guo JZ, Kazeminia A. Streamlining architectures for integrated safety analysis using Design Structure Matrices (DSMs). In: Nowakowski T, Młyńczak M, Jodejko-Pietruczuk, Webińska-Wojciechowska S, (eds). Safety and reliability: Methodology and applications, proceedings of the European Safety and Reliability Conference, ESREL 2014, Wrocław, Poland, 14-18 September 2014. London: Taylor & Francis; 2014, p. 299–302.
20. Becerril L, Kasperek D, Roth M, Lindemann U. Visualization of Interdisciplinary Functional Relations in Complex Systems. In: Marjanović D, Storga M, Pavković N, Bojcetić N, (eds). Proceedings of the DESIGN 2014 13th International Design Conference. Glasgow: Design Society; 2014, p. 1239–1248.
21. ISO. Guidelines for auditing management systems. 2nd ed.(19011): Beuth; 2011.
22. Lindemann U. Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden. 3rd ed. Berlin: Springer; 2009.
23. Hauptmanns U, Knetsch T, Marx M. Gefährdungsbäume zur Analyse von Unfällen und Gefährdungen. Bremerhaven: Wirtschaftsverlag NW; 2004.