Towards Standardizing the Generation of Component Fault Trees through the Engineering Life Cycle

AXEL BERRES, TIM BITTNER

German Aerospace Center, Braunschweig, Germany. E-Mail: Axel.Berres@dlr.de, Tim.Bittner@dlr.de

MARC ZELLER

Siemens AG, München, Germany E-Mails: Marc.Zeller@siemens.com

Today, fault trees are still created and maintained manually by experts. But there are various approaches for the automatic generation of fault trees. These procedures are very different and have been strongly influenced by their application domain. Therefore, no standard procedure exists yet. In order to define a standardized procedure, it is necessary to understand the generation process and to implement a pragmatic solution. In this work, we describe an approach to generate fault trees automatically by using automatically created component fault trees. The aims of this approach are to simplify the creation of fault trees and enable the continuous maintenance.

Keywords: Automatic Generation, Component Fault Trees, CFT, Fault Tree Analysis, FTA

1. Introduction

The development of safety-critical systems requires more effort, since these systems also need to prove safety. Standards such as the ARP 4761 [SA96] show that the system development and safety assessments should take place simultaneously. However, [AB14] shows safety assessments often start too late. Using the model-based safety assessment (MBSA) approach can help to reduce this problem, because MBSA eases access to the required data for all developers at any time. Moreover, the recurring, time-consuming tasks of creating fault trees can be automated. In this way, inconsistencies in the system description can be avoided, since manual synchronization of the system model and the developed fault trees are no longer necessary. In addition, it is possible to provide fault trees for joint discussions in early design phases and continuously during the whole design process [ZK16].

Currently no standardized procedure exists for the automatic generation of fault trees. Fault trees are created and maintained manually, which can lead to problems. The created fault trees are refined or reviewed by safety engineers, who require the current system information to perform their tasks. Maintaining consistency between systems and safety engineering is complex and if this is carried out manually, it is time-consuming and error-prone [PM98]. Therefore, MBSA proposes the automatic creation of fault trees from the system model. Various existing approaches for automatic generation were investigated in [AB16]. In summary, the automatic generation of fault trees: (1) can be done with very different approaches which are strongly influenced by their application domain and (2) these approaches are not derived from a standardized method. To cope with these challenges, we present a generic approach for an automatic generation of fault trees by using component fault trees during the development life cycle of safety-critical systems.

Thus, the main contributions of this paper are: (1) Fully automatic generation of safety analysis models (in form of CFTs) in early development phases based on a logical or functional architecture. (2) Seamless refinement of the safety analysis models during the whole development life cycle. (3) Support of the systematic development of safe system architectures. The applicability of the approach is shown by a case study using CFT's.

2. Related Work

Various previous approaches deal with the generation of fault trees. In [AB99], [HG04] and [MM08] fault tree models are generated from UML models to perform safety analysis during design. Other approaches such as [MB04], [LG05], [AJ07], [NM11], [YP01], [AR04] and [GS09] use specific system models (e.g. MATLAB, AADL, etc.). A review and classification of the generation procedures was carried out in [AB16]. It was shown, that the generation can be based on system structure or behavior. In addition to the system structure and relations among system components, a failure model is also required. This model contains the component faults and failure conditions which lead to a failure and is used to analyze the failure propagation, which leads to system failures. In case of a state-driven approach, all system states need to be determined. Hence, the resulting state model

Proceedings of the 29th European Safety and Reliability Conference.
Edited by Michael Beer and Enrico Zio

Published by Research Publishing, Singapore.
ISBN: 978-981-11-2724-3; doi:10.3850/978-981-11-2724-3_0043-cd

contains the failure model. Various representative methods and their understanding of a failure model are briefly discussed below.

In AltaRica a model consists of a component hierarchy. Components are related as nodes and include flows, states, events, state transitions, and assumptions. The model is described as a text document using a domain-specific language. The algorithm to generate a fault tree is described in [AR02]. HiP-HOPS [PM99] can analyze systems that have identifiable components and material, energy or dataflow transactions among them. In HiP-HOPS, a system model is annotated with safety-related information. This information includes component failure modes and expressions for output deviations, which describe how a component can fail and how it responds to failures occurring in other parts of the system. In [FM14] an approach using SysML for the automatic generation of fault trees is shown. System components and their interfaces are modeled using an internal block diagram (IBD). Flow ports are used to model the interactions among the components. The IBD is converted into a directed graph and by performing an FMEA, component faults are identified. The graph incorporating component faults corresponds to the failure model. It is transformed into a fault tree by using patters on the graph.

Each of the described approaches uses its own system description as well as a failure model. In collaborative engineering additional efforts are necessary to synchronize the different system models. Unfortunately this is usually the case, because in companies different tools are used and therefore document-driven engineering is often applied. Regardless of the previously described approaches, the generation creates complete, system-wide fault trees. However, the system has to be described almost completely before the generation can be performed. To cope with this challenge, an approach using CFT to enable the automatic generation of fault trees is required.

2.1 *CFT- Component Fault Trees*

A component fault tree is a Boolean model associated with system development elements such as components [BK03]. In CFTs, a separate Component Fault Tree Element is attached to a component. Failures that are visible at the component output are modeled using Output Failure Modes which are related to the specific outport. To model how specific failures propagate from a component input to the component failures, Input Failure Modes are used. The internal failure conditions that influence the failures are modeled using Boolean expressions. Using the system structure and the identified top-level events, fault trees can then be created. The basic events used by several components are identified and marked as input of those components. Then the top-level elements of the components are identified. By analyzing the failure conditions of each component the corresponding fault tree is created. Since a fault tree must be free of cycles, cycles must be broken.

CFT methodology corresponds to the "divide and conquer" strategy, which is already intuitively applied as the "best practice" when creating fault trees. This has advantages for team collaboration: Fault trees can be developed iteratively and in parallel. Due to the resulting decoupling of the system, it does not need to be traversed entirely during the fault tree generation. Moreover, inconsistencies between system description and safety analysis models as well as the effort required for fault tree analysis can be reduced due to simplified information exchange [ZK16]. The creation of CFTs allows an automated synchronization between systems and safety engineering [ZK16] and eases the reference of system components in safety assessment. This approach simplifies the development of systems as both system and safety engineers can work simultaneously on individual components. However, CFT's are typically manually created for each component. A procedure or tools to enable an automatic generation are still missing.

3. Towards, a Generic Automatic Generation

For the generation of fault trees, component fault trees are integrated into a lean system model and used throughout the whole development process.

In our understanding a function or their implementing system can be considered as a component which may fail. By allocating system functions to components the transformation from a function-based CFT to a system-based CFT is done. Thus, the allocation of functions enables the full integration of our generation approach out of components fault trees into the development process (see Fig. 1).

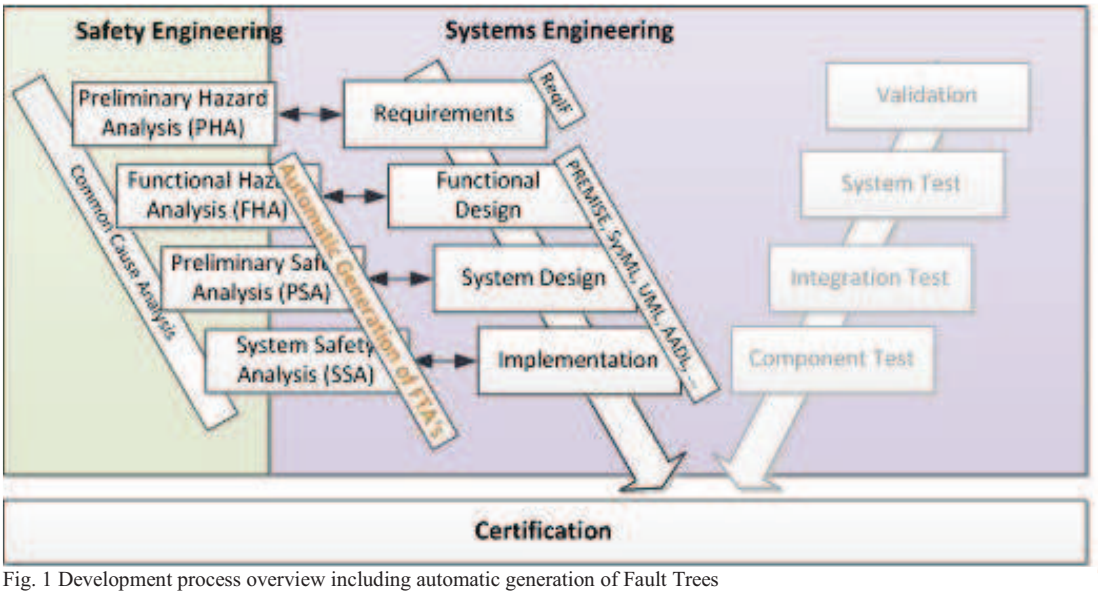

Fig. 1 Development process overview including automatic generation of Fault Trees

As shown in [AB16], a system description and a failure model are required to generate fault trees. The functional architecture and the system structure, after the functional allocation, can serve as a component-based system description. In addition, by adding a failure model to each component and using the component dependencies, a generic procedure can be applied to generate fault trees automatically.

3.1 Generalized Component System Description

The basis of the proposed approach is a safety component graph. This graph can be generated from a functional architecture (E3) or a technical system description (E6). If both are available then the resulting safety component graphs could be merged into one graph (E8).

The directed graph (E1) defines the functional architecture for the system functions, including the functions (F) and their connections (V_F) among them. Each function can be depicted as safety component which can fail. Therefore, by applying (E2) the safety component graph G_{FC} can be initialized with G_{FA}, i.e., by replacing each function with its associated component. To enable the automatic generation of fault trees G_{FC} (E3) then needs to be transformed into a tree by breaking any cycles.

$$G_{FA} = (F, V_F),\ V_F \subseteq F \times F \quad \text{(E1)}$$
$$g: F \rightarrow C \ \ bijective \quad \text{(E2)}$$
$$G_{FC} = (C, V_C),\ V_C \subseteq C \times C\ , V_C = g(V_F) \quad \text{(E3)}$$

During design the technical system G_{TA} (E4) is specified, i.e. a set of system components will be defined to fulfill the purpose of the system. If just the technical system is available, the replacement function (E5) can be used to create the safety component graph G_{SC1} (E6). During the creation of G_{SC1}, cycles have to be broken as well.

$$G_{TA} = (S, V_T),\ V_T \subseteq S \times S \quad \text{(E4)}$$
$$j: S \rightarrow C_1 \ \ bijective \quad \text{(E5)}$$
$$G_{SC1} = (C_1, V_{C1}),\ V_{C1} \subseteq C_1 \times C_1\ ,\ V_{C1} = j(V_T) \quad \text{(E6)}$$

If the functional architecture and a technical system are available then system components are allocated to functions (E7). Using the allocation and the replacement (E2) the system component graph (E8) can be created. Again, the cycles of the system component graph have to be broken.

$$i: S \rightarrow F \ \ surjective \quad \text{(E7)}$$
$$G_{SC2} = (C_2, V_{C2})\ ,\ V_{C2} \subseteq C_2 \times C_2\ ,\ V_{C2} = g\big(i(V_T)\big) \quad \text{(E8)}$$

If the graphs G_{FC} and G_{TA} were designed independently, it is possible to check if a system fulfils all required functions. For this purpose, the functions implemented by the system components must be assignable to the system functions. If the allocation (E5) was complete and correct, it is verified that $G_{SC1} = G_{SC2}$, the single responsibility principle is fulfilled. Otherwise system components need to be decomposed.

3.2 Failure Model

In order to implement a resource saving Safety Assessment, only safety-critical functions should

be evaluated in detail. From the Functional Hazard Analysis (FHA), which was already carried out, a set of safety-critical functions can be identified by using the risk index RI (E9). The risk index is usually defined in the domain specific safety standard.

$$SF = \{ f \in F \mid assess(f) \geq RI \} \quad \text{(E9)}$$

If the criticality of a system is at least "major" then the FHA and the Fault Tree Analysis (FTA) must be performed [SA96]. During the FHA all safety critical functions are identified and based on the findings the decision whether to conduct the FTA is made. It is also recommended to conduct a Preliminary Hazard Analysis (PHA). In the PHA the system and the environment are analyzed to identify a list of "generally" valid hazards. This list is called Preliminary Hazard List (PHL) and may include hazards like "exposure to fire" or "corrosion". In regard to [DO12] a hazard is "a real or potential condition that can cause injury, illness, or death to personnel; damage to or loss of a system, equipment or property; or damage to the environment". The PHL can be defined as a set HP including all previous identified hazards.

As mentioned in section 2, a failure model can be described in different ways. For example, AADL provides the error annex model, which describes the failures of software components by mathematical expressions. In SysML and Simulink, state machines can be used to describe a failure model. As shown in [BK06], CFTs can also be derived from event trees, which are derived from state machines. To enable a widespread usage of different system descriptions, we recommend the use of state machines as a base for a generalized failure model.

For the failure model the set (E10) is defined by incorporating a list of hazards (E11) and the failure conditions represented by a set of Boolean conditions B. Thus, the set FM describes the system behavior in case of failures.

$$FM = (C, B) \quad \text{(E 10)}$$
$$H = H_P \cup H_E \cup H_I \quad \text{(E 11)}$$

To define a hazard, we propose to use state machines. This state machine includes the states Normal and Failure as well as the transition between them. Part of the transition is a guard condition, which depicts the condition of the hazardous event. Furthermore, the failure model can be described as follows (E12). For each component c ∈ C a failure is defined by a list of hazards and a condition b ∈ B. The hazards can be taken from external hazards H_E or safety component hazards H_I. A condition is the combination of operators o ∈ O (AND, OR, PAND, …) and shows the dependency among hazards. The condition can be constructed by using the EBNF grammar (E12).

$$haz = h \in H \ , op = o \in O$$
$$cond = expr\ op\ expr;$$
$$expr = haz \mid "(" \ cond \ ")"; \quad \text{(E 12)}$$

As an example consider a component c_1 with three hazards $\{h_1, h_2, h_3\}$. The occurrence of h_1 or the combination of h_2 and h_3 leads to the failure of c_1. By applying (E12) the following failure condition b_1 can be created (E13).

$$b_1 = h_1 \vee (h_2 \wedge h_3) \quad \text{(E 13)}$$

If it is possible to show that the graphs $G_{SC1}$and G_{SC2} are isomorph and the related failure models FM_1 and FM_2 are identical a consistent and complete initial failure model exists.

4. Automatic Generation of CFTs

After the initial failure model was defined it is possible to generate CFTs automatically at any time. To show the initial generation of a CFT we use the function F1. We assume the PHL contains the two hazards {p1, p2}. Furthermore, F1 includes the two previous identified hazards {i1, i2} and depends on the function F2. The function F2 can fail by the two external hazards {e1, e2} which are propagated. The construction of the initial failure condition b1 can be done as follows. In the worst case, all hazards are independent and may lead to the failure of function F1. Thus, by using the OR gate the failure condition $b_1 = p_1 \vee p_2 \vee i_1 \vee i_2 \vee e_1 \vee e_2$ is automatically constructed. By using b_1 the CFT shown in Fig. 2 can be generated automatically.

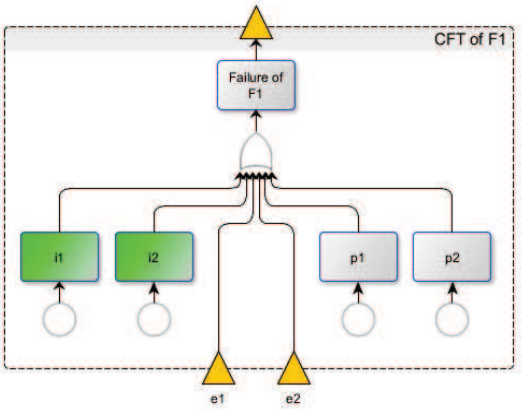

Fig. 2 Initial CFT of function F1

The constructed text notation of b_1 can be transformed into a tree using the shunting-yard algorithm [ED61]. Furthermore, the graphical representation can be generated out of this tree. With the described procedure, it is possible to create CFT's fully automatically at any time during the development life-cycle.

Without experience and manual improvements, the effort required to assess safety-critical functions is enormous. This is because the analysis is carried out for all possible failures and hazard and dependencies are considered as independent. This can lead to the fact that no solution can be found. To find a solution and to reduce the assessment effort the refinement of component failures, analysis of the failure conditions and review of the generated component fault tree needs to be done. After performing the three following steps, the CFT shown in Fig. 3 is generated.

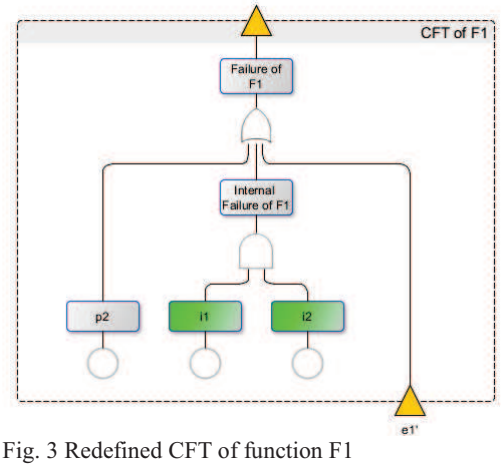

Fig. 3 Redefined CFT of function F1

Refinement of component failures: The identification of failures which may lead to the system component failure starts with the evaluation of the hazards initially added by the PHL and external hazards. For each failure, it is discussed whether the failure could occur or not. If hazards can be removed, it must be documented including the reasoning. The analysis of internal hazards then starts by identifying new hazards and reviewing existing ones. Such an analysis must be performed very thoroughly, as it contributes to a better understanding of system behavior. In addition, it can be described how the system should react in the event of failure. If common hazards can be identified, they should be included in a component hazard list. This enables the reuse of hazards for components.

Analysis of failure conditions: In the next step, the dependencies of the hazards are checked. Initially, the identified hazards are newly combined with the OR expression. Then, the hazards are analyzed separately. If hazards occur only in combination or depending on each other, then intermediate hazards are defined and combined with the corresponding logical expression. This step is complete when all hazards have been examined and documented properly.

For function F1, the list of hazards can be adapted as follows: The analysis showed that the hazard p1 did not apply and the hazards propagated from function F2 are combined into one hazard. Also the internal hazard i1 and i2 are independent of each other and this can be expressed by an AND operator.

Review of the CFT: The graphical representations of the CFTs are generated as previously described and can be evaluated qualitatively or quantitatively. It is recommended that systems as well as safety engineers review the generated trees together. The CFTs are checked for completeness, correctness and logical consistency. By committing the trees after the review, it is assumed that a common understanding exists and the quantitative analysis can be performed.

5. Case Study

As an example, we use the Wheel Brake System shown in Fig. 4 from [SA96]. Because, a functional architecture is missing, no functions were specified. But the system behavior can be described by system components.

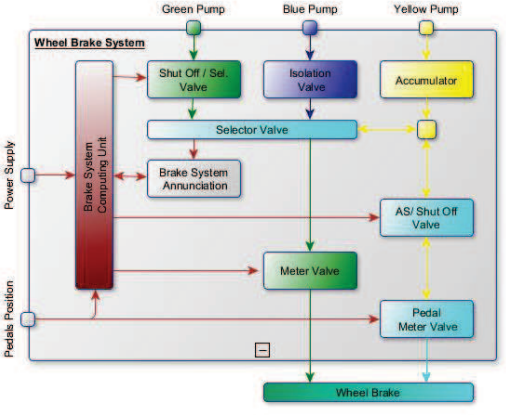

Fig. 4 Wheel Brake System example

Each system component is colored to ease a later identification of related component failures in the generated fault tree. We assume that a PHL exists filled with the common hazards such as corrosion, erosion and fire.

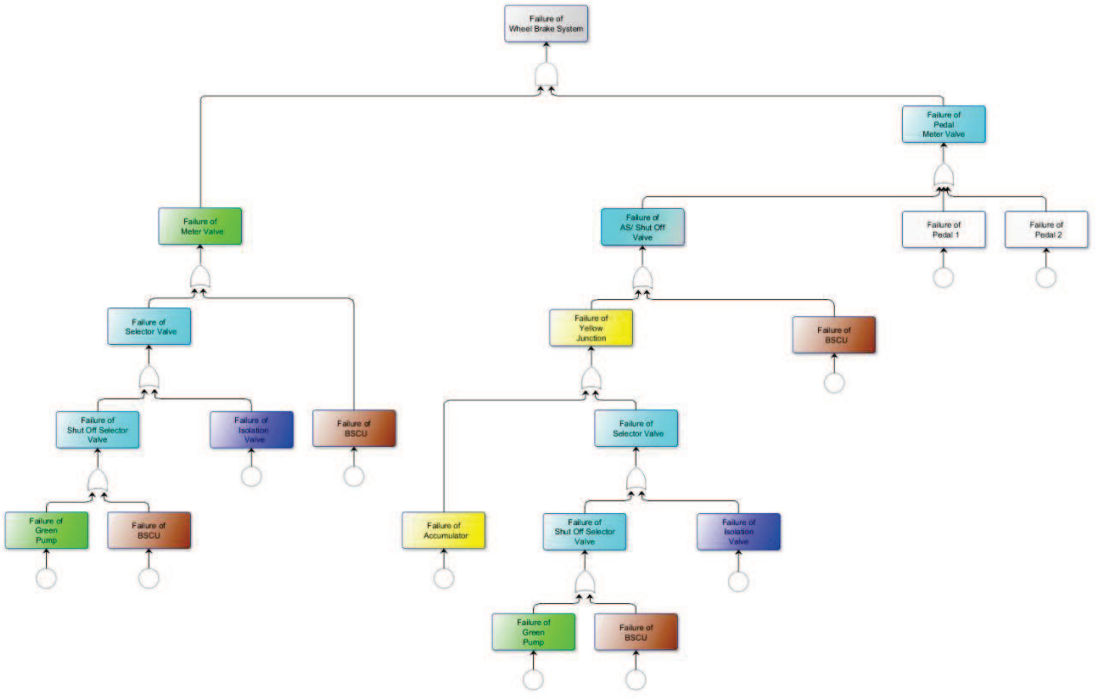

Fig. 5 Excerpt of the generated colored FTA

During the creation of the system model, standard hazards were automatically assigned to each component. Hence, the CFTs could be created automatically as described in section 4. After all connections among the system components were added, automatic fault tree generation was executed and the refinement of CFTs started as described. As shown in the example, three pumps and six valves are used. By using a generic CFT for the pumps, the effort to create a fault tree can be reduced. Several internal hazards of the pumps were added like cavitation, erosion and corrosion. In addition, generic hazards were also added to the six valves. In Fig. 5 a part of the automatically generated fault tree is shown. By manually extending the failure model, the generated CFT can be refined at any time.

6. Conclusion

By describing the creation of CT's in a generic way from a functional architecture, we enable a preliminary safety analysis in an early design phase and close the identified gap. Furthermore, by creating CT's from a technical system, we enable the automatic generation of FT's for the whole development cycle. Also the automatic generation allows an incremental, parallel and continuous creation and maintenance of a fault tree during design process. Hence, automated synchronization between systems and safety engineering is possible which enables the safety engineering in a controlled way.

In contrast to existing approaches which require specific system description including their own failure model, our approach is based on a graph and a failure model using logical expressions to describe the failure behavior. This description can be derived from any existing system modeling approach using model transformation. Moreover, the consistency and completeness of safety analyses can be increased while the effort needed to perform a FTA is decreased. By applying our approach, the analyst can concentrate on the important issues like hazards identification and dependencies, removing cycles etc. The "mechanical" part of the FT creation can be automated.

In addition, the validation of failure models is possible, by checking whether failures are assigned to functions or system components. Also the failure conditions can be checked for logical consistency from a system perspective. Furthermore, the integration of hazard taxonomies can be considered during generation of CFT's. In taxonomies hazards are pre-defined and classified. Out of general hazards, also domain specific hazards can be included. These hazards can then be assigned to the function or the implementing item.

Acknowledgments

Parts of the research have received funding by the Framework Program Horizon 2020 under grant agreement 732242 (DEIS) and under grant agreement 779576 (FLHYSAFE).

References

[AB14] Berres A., H. Schumann, and H. Spangenberg, European survey on safety methods application in aeronautic systems engineering, ESREL Wroclaw, Poland, 2014

[AB16] Berres A., H. Schumann, Automatic Generation of Fault Trees: A survey on methods and approaches, ESREL Glasgow, UK, 2016

[AB99] Bondavalli, A., I. Majzik, and I. Mura, Automated Dependability Analysis of UML Designs. In IEEE International Symposium on Object-Oriented Realtime distributed Computing, 1999

[AJ07] Anjali J., S. Vestal and P. Binns, Automatic generation of static fault trees from AADL models, DSN Workshop on Architecting Dependable Systems, Springer, 2007

[AR02] Rauzy A., Mode automata and their compilation into fault trees, Reliability Engineering System Safety Vol 78/1, Elsevier, 2002

[AR04] Rae A., P. Lindsay, A behaviour-based method for fault tree generation, In Proceedings of the 22nd International System Safety Conference, 2004

[BK03] Kaiser B., P. Liggesmeyer and O. Mäckel, A new component concept for fault trees, Proceedings of the 8th Australian workshop on Safety critical systems and software-Volume 33, 2003

[DO12] DOD, Deparment of Defence Standard Practice - System Safety, MIL-STD-882E, USA, 2012

[ED61] Dijkstra E.W., Algol 60 translation:An algol 60 translator for the x1 and making a translator for algol 60, Stichting Mathematisch Centrum; MR 34/61, 1961

[FM14] Mhenni F., N. Nguyen and J.Y. Choley, Automatic fault tree generation from SysML system models, International Conference on Advanced Intelligent Mechatronics, 2014

[GS09] Szabo, G. and G. Ternai, Automatic Fault Tree Generation as a Support for Safety Studies of Railway Interlocking Systems, IFAC Symposium on Control in Transportation Systems, 2009

[HG04] Giese, H. et al., Compositional hazard analysis of UML component and deployment models, LNCS Volume 3219. Springer, 2004

[LG05] Grunske, L. et al., Specification and evaluation of safety properties in a component-based software engineering process, LNCS Volume 3778 Springer, 2005

[MM08] de Miguel, M. A., J. F. Briones, J. P. Silva, and A. Alonso, Integration of safety analysis in model-driven software development. Software, IET 2(3), 260–280, 2008

[MB04] Bretschneider, M., H. J. Holberg, E. Bode, and I. Bruckner, Model-based safety analysis of a flap control system, In Proceeding of the 14th Annual INCOSE Symposium, 2004

[NM11] Mahmud, N. et al, Compositional synthesis of temporal fault trees from state machines, In Sixth International Conference on Availability, Reliability and Security ARES, pp. 429–435, 2011

[PM99] Papadopoulos Y., J. A. McDermid, Hierarchically Performed Hazard Origin and Propagation Studies, SAFECOMP, 1999

[SA96] SAE Aerospace, ARP4761 - Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, Warrendale, USA, 1996

[YP01] Papadopoulos, Y. and M. Maruhn, Model-Based Automated Synthesis of Fault Trees from Matlab Simulink Models, In International Conference on Dependable Systems and Networks, 2001

[ZK16] Zeller M., K. Hoefig, INSiDER: Incorporation of system and safety analysis models using a dedicated reference model, RAMS IEEE, 2016