Toward an MDD-based Analysis of Stateful and Variant-rich Automotive Functions

Michael Kässmeyer, Rüdiger Berndt, Peter Bazan, Reinhard German

Toward an MDD-based Analysis of Stateful and Variant-rich Automotive Functions

Michael Käßmeyer*, Rüdiger Berndt*, Peter Bazan†, Reinhard German†
*AUDI AG, Ingolstadt, Germany
†Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany
{michael.kaessmeyer,ruediger.berndt}@audi.de,
{peter.bazan,reinhard.german}@fau.de

***Abstract*—The high integration of (sub-) systems performing safety critical automotive functions characterizes the current development in the automotive industry. The development and analysis is challenged by an increasing complexity resulting from product customization and variance in implementations by software-hardware solutions. In order to save costs for such scenarios, a systematic analysis of the dependencies between functions, as well as the functional and technical variance, is required. In this paper we introduce a new approach which allows to compactly represent and analyze a function with its different configurations, states, hardware modules, and software variants—also: Product Line Fault Tree (PLFT)—in a unified data structure based on Multi-valued Decision Diagrams (MDDs). The methodology allows to represent the function´s architecture within an MDD and transfer it in Fault Tree (FT)- and Minimal Cut Sets (MCSs)-MDDs. Therefore, complete fault trees are analyzed in one step—opposed to stepwise analyzing FTs of all configurations, software variants, and states. Summing up, this article introduces a systematic approach allowing to analyze fault propagation in variant-rich and stateful functions.**

***Index Terms*—Fault Tree, Multi-Valued Decision Diagram, Dependability Analysis, States, Product Line Engineering, Minimal Cut Sets**

I. Introduction

The current challenge in the automotive industry is to enable piloted driving. Therefore a part to realize piloted driving is to combine and interact the state of the art assistant functions with each other. Moreover, additional assistant functions are developed to increase road safety or to offer new in-vehicle services.

Each assistant function and the interaction of these operate with respect to different contexts and contain a high number of sensors, actuators, and embedded software. This is why automotive functions are marked by variance and high complexity. Mostly all innovative functions are realized by software and many of them are safety related. Therefore functional safety is of considerable importance.

To manage the increasing complexity model-based development has been established in the automotive sector—mainly induced by highly-integrated Electronic Control Units (ECUs) and the growing number of variant-rich and stateful functions [1]. In this context an automotive safety standard is given by the ISO 26262 [2] postulating normative requirements to ensure functional safety within electrical and electronic vehicle systems. These include a hazard and risk analysis, a functional and technical safety concept, as well as safety analysis and safety case just to name a few. Consequently, the overall safety assessment is labor-intensive and time-consuming. As an example: available software tools that ought to support traceability or semi-automated consistency and completeness checks often do not meet specific user expectations.

Summing up, to enable piloted driving or develop a new innovative safety critical assistant function which contains all safety mechanisms is time-consuming with state-of-the-art safety analysis methods such as Fault Tree Analysis (FTA). This is because on the one hand, functions are marked by numerous configurations (cf. Table I), states (cf. Table II), and software variants and on the other hand due to the lack of appropriate methods and tools to compactly represent and efficiently analyze such safety-critical functions. In this paper the characterization of configurations, states, hardware modules, and software variants is also called Combined Automotive Properties (CAP).

Therefore, in this article a new approach is established allowing to compactly represent the function´s architecture with its CAP in one data structure. Given that, a CAP-Fault Tree (FT) is established using Multi-valued Decision Diagrams (MDDs), cf.: Product Line Fault Tree (PLFT)—also referred as 150% model. Based on such compact MDD-FT representations, the proposed method allows to generate an MDD representation of the Minimal Cut Sets (MCSs) of the system including all CAP. This data structure, accordingly, enables one to extract the MCSs of specific configurations, software variants and states. Moreover, for a given MCS, the proposed method allows to identify the variants and states where the corresponding MCS will cause the system to fail.

This article is structured as follows: Section II introduces related work. Section III presents a systematic approach to analyze safety-critical functions by means of MDDs. Finally, in Section IV, we summarize the article and suggest future research.

II. Related Work

The related work on the overlapping areas of the safety method FTA and Product Line Engineering (PLE) are introduced.

Dehlinger proposed how to attach commonality and variability attributes to the PLFT and managed it as a core asset [3].

Lu's work [4] extends Dehlinger's work and is another way to obtain the same result for product lines. The so-called Fault Contribution Tree (FCT) is a variability tree for the product line where nodes are features instead of events (or conditions) [5]. Feng's work [6] is another extension of Dehlinger's contribution constructing a software fault tree in a different manner: the method begins considering commonality and variability analysis as well as the product line architecture to obtain the so-called Extended Commonality and Variability Analysis (XCA). After that, the Software Fault Tree Analysis (SFTA) is carried out based on the XCA and the Software Failure Modes and Effects Analysis (SFMEA). Noda [7] proposes a method assuming the fault of features from the feature diagram. A feature is selected and turned into the root node of the FT, then this structure is analyzed to identify all paths to the root node. Based on that, countermeasures are identified—like adding optional features to the diagram.

In [8], the authors carry over product lines to the Component Fault Tree (CFT) approach. By doing so, the product line is steadily maintained over time and information on variability is considered w.r.t. FTAs. This method, for example, considers a component ("Ventilation System") to be reused from a previous Gas Turbine PL (SGT 500) within another Gas Turbine PL (SGT 400), the component is analyzed whether it is marked by the same software behavior, functions and structure. In the end, if the component is fully compatible with required characteristics, it is suited for the new product line [9].

In contrast to these articles and the corresponding modeling approaches, this paper will rather focus on a systematic approach with a specific data structure to analyze fault propagation of variant-rich and stateful safety-critical automotive functions using MDDs.

III. Systematic Approach to Analyze Safety-Critical Functions

In this section a systematic approach is presented to derive safety analysis concepts. The objective is to analyze and compare MCSs and identify fault propagations of different configurations and software (SW) variants of a specific safety-critical function in one data structure. These functions are highly distributed, incorporate commercial off-the-shell components, and are developed by different supplier and Original Equipment Manufacturer (OEM) business units. In this case, neglecting of safety-critical fault propagations—which could lead to harm people—may result in costly redesigns during later development phases. Therefore, a systematic data structure approach is introduced to analyze safety-critical functions on an early development phase (architecture concepts). The advantage of an Multi-valued Decision Diagram (MDD) structure is, to store all feasible function architecture possibilities (due to CAPs) and their corresponding Fault Trees (FTs) in one framework. Moreover, a Minimal Cut Set (MCS) analysis of the different stored FTs are possible in this data structure.

In Figure 1 the transfer and encoding of properties within MDDs is shown. Specific algorithm steps are necessary to evaluate the approach and achieve the results of the shown MDDs. For reasons of simplicity, the algorithm steps are

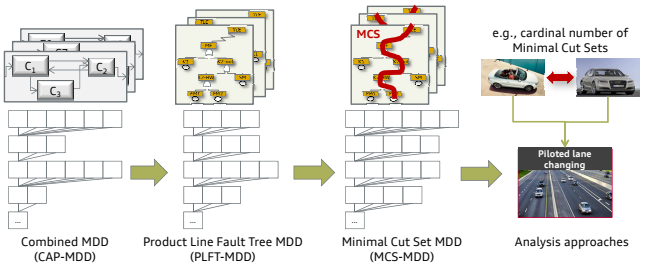

Figure 1. Systematic transfer of properties within different MDDs

illustrated with different MDDs. The starting MDD is the encoded Combined Automotive Properties (CAP)-MDD, where the properties of all feasible interconnection architectures of a function are contained. A given ordered and reduced CAP-MDD is the basis for the corresponding safety analysis. The next step is to generate a Product Line Fault Tree (PLFT)-MDD of the encoded CAP-MDD. The last MDD is the MCS-MDD, which is generated from the PLFT-MDD. With the MCS-MDD specific analysis methods can be applied, for example, a comparison of the MCSs of feasible configurations, states, and SW variants of a safety-critical function.

First, we outline the CAP-MDD generation. Secondly, we illustrate the properties which are transfered from the CAP-MDD to an PLFT-MDD. Thirdly, the concept of the construction of an MCS-MDD from an PLFT-MDD is explained.

A. Coding CAP to an MDD

Hence, safety analysis techniques on architecture concepts are increasingly applied at early design states [2]. State of the art methods are Failure Mode and Effects Analysis (FMEA) and Fault Tree Analysis (FTA). Techniques like code review, hardware (HW) review, and development audits are usually done at later design stages. To identify both single and multiple faults, it is common to use the FT technique due to the complex interaction of different systems (cf. assistant functions).

Nowadays, for each vehicle configuration, each state, and each SW variant an FT is developed. Therefore, plenty of product FTs with their corresponding MCSs are stored in different documents. A formal model has to be designed to contain all CAP in one framework and to allow the systematic transfer from a function to its safety analysis. Next, we describe the approach of encoding the set of CAP by using an MDD. This MDD, which is called $\overline{Q}$, shows the set of all possibilities to develop an architecture for a function without any constraints. An edge connects an attribute of the upper layer with a node on the lower layer and hence encodes all combinations–Cartesian product–of the attribute and the attributes of the node. This in turn means that each path of the structure–from the top layer to the bottom layer–represents one element of $\overline{Q}$.

The top layers represent the configuration domains. Each of those contains all corresponding attributes. Next, the architecture properties are given. First, the HW and SW variants are depicted, followed by the input and output of each possible component. Each layer of a HW domain contains all corresponding HW modules. The following layers represent

the SW variant domains, the input domains and output domains. The last layers of the MDD encode the state space. Therefore, each layer represents a state domain and every of those contains all corresponding attributes.

Furthermore, constraints are necessary to determine all feasible configurations, SW variants, HW modules, and states. For example, to offer piloted driving specific technologies (e.g., specific functions, sensors, cameras etc.) are needed and different safety critical states (vehicle accelerates, lane change etc.) are possible. All constraints of the configuration space, the state space, the architecture space, and the CAP-space, are constructed in a similar way (if-then paradigm). Based on the MDD-representations of $\overline{Q}$ and the constraints we successively apply the intersection operator to generate the MDD representation of Q named CAP-MDD. A pre-condition of the operation is that the orders of the layers match. During the intersection the constraints are verified with respect to self-consistency by checking if the resulting set is empty after each intersection task.

As a result of the intersection of $\overline{Q}$ with the constraints, the number if nodes increases. In order to keep the systematic approach practicable, it is important to investigate the impact of diverse variable ordering methods—cf. ([10]–[12])—upon the MDD-based representations of all, the CAP, the PLFT, and the MCS.

With the encoded Q the systematic transfer is initialized. In the following the task for the MDD transfers are introduced.

B. CAP-MDD to PLFT-MDD

The general starting point to construct a Binary Decision Diagram (BDD) representation of a single FT is the Top Level Event (TLE). Given that, a depth-first-search is carried out—by doing so, all gates and events are encoded by the BDD nodes. During this procedure all cyclic dependencies between gates have to be resolved. Finally, each component of the FT is represented by a variable (also: level) of the BDD.

In [13] an MDD-based analysis of FTs whose components have three states is described—this approach is extended to any number of states in [14]. This idea can also be carried over to systems that might be distinguished by several phases or different fault conditions [15], [16]. Moreover, we take this approach and carry it over to the automotive challenges with different CAP.

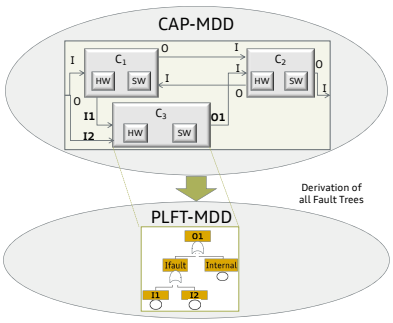

Figure 2. Derivation of FTs with an example of component C_3

For a specific architecture such as in Figure 2, the input and output properties as well as the corresponding SW variants and HW modules are contained in the CAP-MDD. To generate a FT the inputs of a component are used as basic events and the output as top level event (cf. Figure 2).

Note, a drawn connection can also represent more input and output signals which are transmitted between two components due to SW variants or different configuration strategies. Based on Figure 2 the parallel input signals ($I1$ and $I2$) are summarized with an OR gate which propagates to the intermediate event $Ifault$. The component by itself is seen as a black box where an internal fault ($Internal$) can occur. Due to the serial connection of $Ifault$ and $Internal$, the intermediate event $Ifault$ and the internal fault $Internal$ is summarized with an OR gate and propagate to the output event (top level event) $O1$. This FT generation process is done for all components with HW modules and many different SW variants in one (sub)-architecture. The complete FT of an architecture can be generated by composing the sub-FTs [9]. The PLFT-MDD stores all FTs of each feasible architecture of a function. With a valid PLFT-MDD a MCS-MDD can be generated.

C. PLFT-MDD to MCS-MDD

This approach is an extension of the common identification of the MCSs, which can be directly constructed from a product FT. This extension approach and its application represent a determination of all MCSs of a PLFT [17].

First, we introduce basic terms and definitions. The BDD representation of a Boolean function is based on Boole's expansion theorem (also: Shannon decomposition):

$$F(v_1, ..., v_n) = v_i.F_{v_i=1} + \overline{v}_i.F_{v_i=0} \tag{1}$$

Here $F_{v_i=1}$ and $F_{v_i=0}$ denote the function F with argument v_i set to 1 or 0, respectively. This theorem can be extended to the multi-valued case:

$$F : \{1, 2, \ldots, s\}^n \rightarrow \{0, 1\} \tag{2}$$

$$F(v_1, \ldots, v_n) = (v_i = 1).F_{v_i=1} + (v_i = 2).F_{v_i=2} + \ldots + (v_i = s).F_{v_i=s} \tag{3}$$

The assignment of values to F's variables can be written as *minterm*[1]. In the Boolean case, the *literal* v_i denotes the assignment $v_i = 1$ and the *literal* $\overline{v}_i$ denotes the assignment $v_i = 0$. Considering multi-valued functions, the literal $v_{i,j}$ corresponds to the variable assignment $v_i = j$. Obviously, minterms only contain one literal of each variable. A *conjunction term* is a set of literals which are exclusively connected by logical 'and' (also: conjunction)—accordingly minterms are special conjunction terms. Conjunction terms yielding $F = 1$ are called *implicants* of the function. Implicants that cannot be further reduced are called *prime terms*. According to [18] and based on a given set of literals L (also: 'literals of interest'), the set of MCSs of a function F is defined as the set of all prime terms given that all literals $l \notin L$ are removed. In static FTs no temporal sequences of faults are considered. Let a static FT

[1]A minterm is a product term in which each variable appears once. Boolean functions can be expressed as sum of minterms where each minterm corresponds to a row of the function's truth table whose final value—the function's output—is 1.

and the corresponding Boolean function F be given. According to [19] and based on the decomposition $F = v.F_1 + \overline{v}.F_0$, the MCS can be derived as follows:

$$\mathrm{MCS}[F] = \mathrm{MCS}_1 \cup \mathrm{MCS}_0 \quad (4)$$

$$\mathrm{MCS}_0 = \mathrm{MCS}[F_0] \quad (5)$$

$$\mathrm{MCS}_1 = \{v.\pi | \pi \in \mathrm{MCS}[F_1 + F_0] \setminus \mathrm{MCS}_0\} \quad (6)$$

The minimal cut set analysis for each FT in the PLFT-MDD is performed. Afterwards the superpositon of each MCSs generates the MCSs-MDD.

The MDD-based MCSs representation allows to efficiently investigate several scenarios: for example the identification of the MCSs w.r.t. CAP—or for given MCSs one might be interested in the configuration, state, or SW variant which are affected. Such queries are based on intersect operations and MDD-based structures encoding the query (for example a specific vehicle configuration).

IV. Conclusion and Future Work

The automotive state-of-the-art method is to generate for each valid architecture a FT and its corresponding MCS. The novel approach in this paper is more efficient compared to the state-of-the-art method due to the reason, that all valid architectures of a specific function are analyzed and compared in one iteration and in one data structure approach. Therefore, all different fault propagations of variant-rich and stateful safety-critical functions are determined and stored in one framework. The automatic generation is given due to the fact, that each input is used as basic event and each component has an internal fault. The underlying data structure is given as MDDs which are not only used to encode the fault propagation of all configurations, states, and SW variants of a function but also to represent the corresponding MCSs in one framework. Moreover, the systematic approach how to transfer the function development into a MDD-based safety analysis was explained.

An ongoing work is the development of a formal model which combines the architecture, the configuration, and the state space to transfer the necessary properties in an MDD structure.

In the near future, we will evaluate the presented approach with piloted driving systems. Moreover, following basic analyze options will be evaluated:

- Searching of MCSs for each valid variant and state.
- Identification of all affected variants and states of a function with a given MCS.
- Comparison of the MCSs of configurations, SW variants, and states of a safety-critical function, and analysis which cardinal number is equal or not.
- Identification of similar or differing safety mechanisms in configurations, SW variants, and states of a function.

Based on that, we want to identify metrics to improve our approach, and to measure the impact of change requests affecting the vehicle configuration and feasible states of a function. However, we plan to integrate the MDD analysis approach into an extended model-based safety and variant management framework (cf. [20], [21]). Finally, in order to keep the approach practicable, we want to investigate the impact of diverse variable ordering methods—cf. [10], [11]—upon the MDD-based representations of both, the PLFT and the MCSs.

References

[1] Ebert, C., Jones, C.: Embedded Software: Facts, Figures, and Future. IEEE Computer **42** (2009) 42–52

[2] Technical Committee 22 (ISO/TC 22): Road vehicles – Functional safety (2011)

[3] Dehlinger, J., Lutz, R.: Software Fault Tree Analysis for Product Lines. In Proceedings of Eighth IEEE International Symposium on High Assurance Systems Engineering. (2004)

[4] Lu, D., Lutz, R.: Fault Contribution Trees for Product Families. In: Proceedings of the 13th International Symposium on Software Reliability Engineering, 2002. ISSRE 2003. (2002) 231–242

[5] Lam, W.: A Case-study of Requirements Reuse Through Product Families. Ann. Softw. Eng. **5** (1998) 253–277

[6] Feng, Q., Lutz, R.R.: Bi-directional Safety Analysis of Product Lines. J. Syst. Softw. **78** (2005) 111–127

[7] Noda, A., Nakanishi, T., Kitasuka, T., Fukuda, A.: Introducing Fault Tree Analysis into Product Line Software Engineering for Exception Handling Feature Exploitation. In: Proceedings of the 25th Conference on IASTED International Multi-Conference: Software Engineering. SE'07, Anaheim, CA, USA, ACTA Press (2007) 229–234

[8] Gómez, C., Liggesmeyer, P., Sutor, A.: Variability Management of Safety and Reliability Models: An Intermediate Model Towards Systematic Reuse of Component Fault Trees. In: Proceedings of the 29th International Conference on Computer Safety, Reliability, and Security. SAFECOMP'10, Berlin, Heidelberg, Springer-Verlag (2010) 28–40

[9] Kaiser, B., Liggesmeyer, P., Mäckel, O.: A New Component Concept for Fault Trees. In: Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33. SCS '03, Darlinghurst, Australia, Australia, Australian Computer Society, Inc. (2003) 37–46

[10] Berndt, R., Bazan, P., Hielscher, K.S.: On the Ordering of Variables of Multi-Valued Decision Diagrams. In: MMBnet, Hamburg. (2011)

[11] Berndt, R., Bazan, P., Hielscher, K.S.J., German, R.: Construction Methods for MDD-based State Space Representations of Unstructured Systems. In 2014, M., ed.: Proceedings of the 17th International GI/ITG Conference on Measurement, Modelling and Evaluation of Computing Systems and Dependability and Fault-Tolerance. Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (2014) 43–56

[12] Schmiedle, F., Gunther, W., Drechsler, R.: Selection of Efficient Reordering Heuristics for MDD Construction. In: Proceedings. 31st International Symposium on Multiple-Valued Logic, IEEE (2001) 299–304

[13] Xing, L., Dugan, J.B.: Dependability Analysis using Multiple-valued Decision Diagrams. In: Proc. of 6th International Conference on Probabilistic Safety Assessment and Management. (2002)

[14] Xing, L., Dai, Y.: A New Decision-Diagram-Based Method for Efficient Analysis on Multistate Systems. IEEE Transactions on Dependable and Secure Computing **6** (2009) 161–174

[15] Mo, Y., Xing, L., Dugan, J.: MDD-Based Method for Efficient Analysis on Phased-Mission Systems With Multimode Failures. IEEE Transactions on Systems, Man, and Cybernetics: Systems **44** (2014) 757–769

[16] Mo, Y., Xing, L., Amari, S.: A Multiple-Valued Decision Diagram Based Method for Efficient Reliability Analysis of Non-Repairable Phased-Mission Systems. Reliability, IEEE Transactions on **63** (2014) 320–330

[17] Käßmeyer, M., Berndt, R., Bazan, P., German, R.: Product Line Fault Tree Analysis by Means of Multi-valued Decision Diagrams. In 2016, M., ed.: Proceedings of the 18th International GI/ITG Conference on "Measurement, Modelling and Evaluation of Computing Systems" and "Dependability and Fault-Tolerance". Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (2016)

[18] Rauzy, A.: Mathematical Foundations of Minimal Cutsets. IEEE Transactions on Reliability **50** (2001) 389–396

[19] Rauzy, A.: Binary Decision Diagrams for Reliability Studies. In Misra, K., ed.: Handbook of Performability Engineering. Springer London (2008) 381–396

[20] Käßmeyer, M., Velasco Moncaday, D., Schurius, M.: Evaluation of a systematic approach in variant management for safety–critical systems development. In 12th International Conference on Embedded and Ubiquitous Computing (2015)

[21] Käßmeyer, M., Soden, M.: A Model based Difference Approach and Change Impact Rules Language to manage Variability and Change Requests in Safety Critical Automotive Functions. SAE International, Technical Paper 2016-01-0125, 2016, doi:10.4271/2016-01-0125. (2016)

ACRONYMS

BDD	Binary Decision Diagram
CAP	Combined Automotive Properties
CFT	Component Fault Tree
ECU	Electronic Control Unit
FCT	Fault Contribution Tree
FMEA	Failure Mode and Effects Analysis
FT	Fault Tree
FTA	Fault Tree Analysis
HW	hardware
MCS	Minimal Cut Set
MDD	Multi-valued Decision Diagram
OEM	Original Equipment Manufacturer
PLE	Product Line Engineering
PLFT	Product Line Fault Tree
SFMEA	Software Failure Modes and Effects Analysis
SFTA	Software Fault Tree Analysis
SW	software
TLE	Top Level Event

APPENDIX

Table I
EXTRACTION OF PROPERTIES TO DEFINE CONFIGURATIONS

Domain	Attribute	Property
D_S(Steering)	DS	Dynamic Steering
	FS	Front Steering
	RS	Rear Steering
	CS	Combined Steering
D_D(Drive)	FWD	Front-wheel drive
	RWD	Rear-wheel drive
	$4WD$	4-wheel drive
D_{ED}(Environment Detection)	US	Ultrasonic Sensor
	MC	Mono Camera
	SC	Stereo Camera
	RS	Radar Sensor
⋮	⋮	⋮

Table II
EXTRACTION OF PROPERTIES TO DEFINE STATE SPACE

Domain	Attribute	Property
D_{Ign}(Ignition)	ION	Ignition on
	IOF	Ignition off
	EON	Engine on
	EOF	Engine off
	SE	Stop Engine
	VTS	Vehicle traction standby
D_{Spe}(Speed)	$VHAC$	Vehicle accelerates
	$VHSD$	Vehicle decelerates
	$COSP$	Constant speed
	$VHST$	Vehicle stopped
D_{IS}(Interstate)	FL	Fast Line
	OM	Overtaking Maneuver
	CI	Cutting In
	⋮	⋮
⋮	⋮	⋮