USE CASE-BASED FAULT TREE ANALYSIS OF SAFETY-RELATED EMBEDDED SYSTEMS

Dipl.-Ing. (FH) Ephraim Balz, MSc
STZ Softwaretechnik
Im Gaugenmaier 20
73730 Esslingen
Germany
Ephraim.Balz@stz-softwaretechnik.de

Prof. Dr. Joachim Goll
Fachhochschule Esslingen
Kanalstr. 33
73728 Esslingen
Germany
Joachim.Goll@fht-esslingen.de

ABSTRACT

In this paper a use case-based approach for performing the safety analyses of safety related systems will be presented, which makes it possible to analyse complex systems in a straightforward way and helps avoiding mistakes during safety analyses. The safety analyses "Use Case-Based Fault Tree Analysis" and "Use Case-Based Failure Mode and Effects Analysis" are integrated into the context of use case-based system analysis and system design which allows developing systems with system parts of different safety integrity levels in a consistent way. An example will show how a qualitative Use Case-based Fault Tree Analysis is integrated into the use case-based development of a safety related automotive system.

KEYWORDS

Safety critical systems, Fault Tree Analysis

1. Introduction

In the last few years the number of embedded systems in vehicles strongly increased due to new advanced applications. Driver assistant systems e.g., which actively intervene the driving course [1] get more and more important. Thus the requirements on the reliability of systems and components are rising, too. In this paper a development process is introduced, which comprises the development of hardware and software. It is compatible to the German V-Model [2] as well as to the IEC 61508 [3]. This approach is use case-based which is new in the field of safety analysis. This use case-based approach allows to perform the system analysis, the risk analysis and the system design of mixed safety-related and non safety-related systems in a straightforward and consistent way.

For performing the safety analysis of systems and components there exist several well-known methods such as the Failure Mode and Effects Analysis (FMEA) [4, 5] and the Fault Tree Analysis (FTA) [6, 7]. In the Failure Mode and Effects Analysis it will be analysed, which failures might arise in a system component and how this failures would affect the total system. Starting from an undesired event (a dangerous failure of the system, called top level event) in a qualitative Fault Tree Analysis it will be analysed in a top-down approach (tree structure of faults), what reasons might lead to this undesired event. On the other hand, after estimating failure probabilities for the leafs of the fault tree in the quantitative bottom-up evaluation of the fault tree the probability of this dangerous system failure can be computed.

State of the art is that in the qualitative Fault Tree Analysis the whole technical system will be considered. Thus the analyst must have the complete system in mind. In complex systems the qualitative Fault Tree Analysis will result in many pages and will become very complicated, which might lead to mistakes during the safety analysis. It might occur that failures would be forgotten and therefore gaps in the safety concept could arise. This might lead to fatal consequences for the system as well as for its surroundings, in which the system is in operation, and of course for the company which has produced it and has brought it to market.

Here the use case-based approach, in which the safety analysis is carried out in a structured way, can help to avoid errors in the safety analysis. The system will be analysed step by step on the basis of use cases. This means, that not the total system will be analysed all at once, but just the sub systems relevant for the corresponding use case. At the end the results of all qualitative Use Case-Based Fault Tree Analyses will be put together.

The result of the combination is the qualitative Fault Tree Analysis of the total system. By considering the individual use cases separately, one after the other, the complexity of the Fault Tree Analysis will be decreased significantly. This contributes decisively to the correctness of the safety analysis.

2. Use case-based system development

A significant characteristic of embedded systems in contrast to information systems is that not data-carrying objects and their static associations play a central role, but control-objects and dynamic interactions. Embedded real-time applications are often event-driven and state-dependent. The dynamic behaviour of a system can be modelled by state diagrams [8] and interaction diagrams in the form of collaboration diagrams – in UML 2.0 called communication diagrams – and of sequence diagrams. An interaction diagram describes how the use case is realized by the collaboration of system functions (see e.g. figure 3-1) or objects, respectively. On the way to state and interaction diagrams the use case diagram, textual descriptions of the use cases and optionally activity diagrams are helpful.

Within the scope of system analysis the system is composed of system functions – called processes in Structured Analysis – and of data in the framework of classical software engineering or of control and entity objects in the case of object-oriented modelling. The collaboration of the system functions is classically described by data flow diagrams. Using the object-oriented paradigm the collaboration of objects is visualized by collaboration diagrams or sequence diagrams, respectively. In both paradigms – the classical function-oriented as well as the object-oriented paradigm, respectively – these diagrams describe in system analysis the processing of the use cases in an ideal world in a logical way.

In contrast to UML 1.x now UML 2.0 allows that different diagram types can be mixed in order to model mutually dependent system aspects commonly in just one view. This common view may contribute considerably to the consistency of diagrams. The advantage of a common modelling of mutually dependent aspects has been shown by Heinisch and Goll [9] by the introduction of state-based collaboration diagrams and of collaboration diagrams in a layer model.

In system analysis of non-safety related systems a use case-based approach is state of the art. In the use case approach one starts with the application functions – called use cases – of a system, which can be invoked by the users of the system. These use cases are captured in a use case diagram, which shows the use cases of the system and the actors participating in each use case. Then normally a textual description of use cases will be given, supported by activity diagrams if necessary in the case of complex use cases. Finally the modelling in form of interaction and state diagrams will be performed.

For safety related systems it is useful to keep the use case-based view and to perform a risk analysis for each use case. On the basis of the risk analysis each use case will be assigned a safety integrity level (SIL) [10].

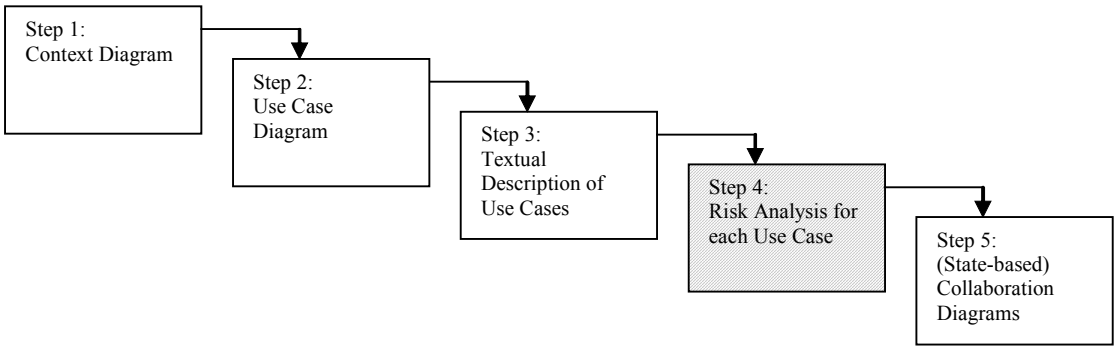

Figure 2-1 Steps in system analysis including safety analysis. The hatched activities are performed only for safety-related systems

One advantage of a use case-based view is, that it allows to treat systems with safety related and non-safety related system parts with the same method. When non-safety related system parts are considered to be of "SIL 0", the suggested method holds for systems, which are composed of system parts with any safety integrity level. Use cases with different safety integrity levels should be decoupled in system design. If use cases with different safety integrity levels are coupled, the highest safety integrity level of the coupled use cases must be assigned to all of them.

In the first step of system design the flow diagrams of system analysis should be mapped to the system hardware, i.e. to the different nodes of a distributed system and to the internal layers of the nodes. The hardware architecture is designed by defining, which components the total system should consist of, what functionality which component should have and how the components should interact to yield the output of the system.

System Functions/ Classes in System Analysis	System Functions/Classes in System Design
Processing Functions	Processing Functions
	Input / Output
	Persistency
	Communication
	Start-Up/Shut-Down
	Concurrency
	Security
	Fault diagnosis / Error processing

Technical Functions/ Technical Classes

Table 2-1 System functions in system analysis and in system design

In a second step the flow diagrams in the layer models should be complemented by further types of system functions, which we call technical functions (see Table 2-1), which are needed to build a real existing, i.e. "physical" system. These technical functions are system functions for input/output, persistency, communication, start-up/shut-down, concurrency, and for access control (security).

The suggested use case-based approach allows recognising, whether use cases with different safety integrity levels are running independently or whether they affect each other. A coupling of use cases may result from the fact that a distinct system function or data can be used by different use cases. This can easily be detected by considering the collaboration diagrams.

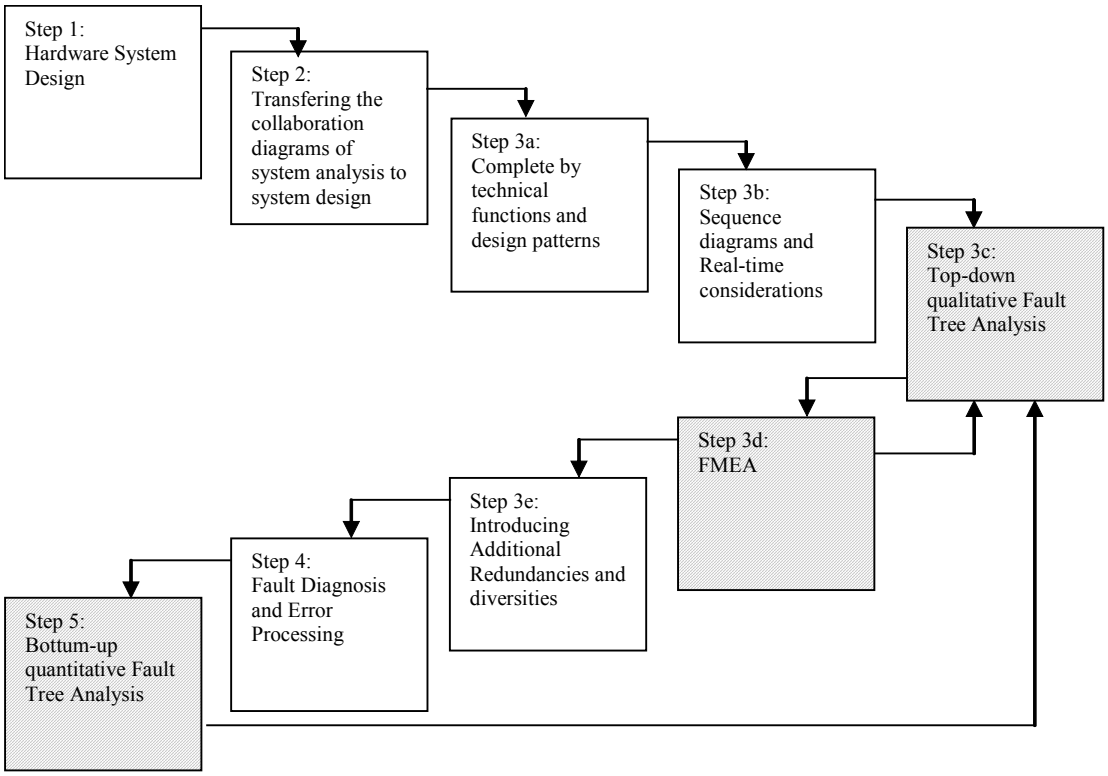

Figure 2-2 Steps in system design including safety analyses. The hatched activities are performed only for safety-related system

In the third step system functions for fault diagnosis and error processing are added to the system. Hardware components like sensors and actuators are implemented redundantly if required by the safety integrity level. In this case further system functions for fault diagnosis and error processing are needed which also influence the concept of error processing. It makes sense that in the second step the system is designed without fault diagnosis and error processing to get the adequate system architecture and to find all required system functions/objects. In principle each hardware unit and each system function can fail. Therefore these components should first be defined. They are then the basis for a safety analysis and the safety analysis in turn is the basis for designing the fault diagnosis and error processing.

Now, the architecture in hardware and software is designed. A qualitative top-down Fault Tree Analysis and

a bottom-up Fault Tree Analysis will be performed to check whether the system design fulfils the safety requirements. A safety evaluation should be performed before the system is implemented. In this stage changes in the design and safety concept, which follow from the results of the safety evaluation, can be performed cost-effective. The qualitative Fault Tree Analysis should be performed before the FMEA to yield inputs about possible faults.

3. Use case-based safety analysis in the scope of system analysis

By means of an example it will be shown, how a safety integrity level is assigned to processing functions of system analysis. The example here is performed according to classical software engineering with Structured Analysis.

The use case "Braking" of an automotive braking system consists of several scenarios, namely the main scenario "Braking in normal case" and several alternative scenarios. In Table 3-2 the scenarios of the use case "Braking" are listed as well as the processing functions, which are used by the respective scenario.

In this example we shall consider just two scenarios, namely the alternative scenario 1, "Braking at a low adhesion factor", and the alternative scenario 2, "Braking at the driving dynamic limit". In Figure 3-1 and Figure 3-2 the data flow diagrams of the use case "Braking" [10] with alternative scenario 1 and alternative scenario 2 will be illustrated. In both diagrams the parts of the alternative scenarios, which differ from the main scenario, are hatched differently.

The data flow in Figure 3-2 is very simplified. In a real system the driving dynamic sensors consist of several sensors which determine the yaw rate, the lateral acceleration, the steering angle and the wheel speed. The date "Driving Dynamics Data", which is an input for the processing function "Determining Driving Dynamics", contains data, which are required to compute the brake forces for stabilizing the vehicle in the driving dynamic limit. The simplification of this data flow was made for the better understanding of the data flow diagram. However, it does not have any affect on the illustration of the use case-based approach.

Next, each scenario of the use case "Braking" is assigned a safety integrity level. The risk analysis with the aid of a conventional risk graph [10] will not be covered by this paper.

The use cases, which are described in system analysis, are provided with a Safety Integrity Level and the Safety Integrity Levels of the processing functions can be easily derived from the use cases taking into account possible couplings between different use cases.

Scenarios		Processing Function
Main scenario:	Braking in normal case	Determining must deceleration
		Computing brake forces
Alternative scenario 1:	Braking at a low adhesion factor	Determining must deceleration
		Computing brake forces
		Determining slip
Alternative scenario 2:	Braking at the driving dynamic limit	Determining driving dynamics
		Determining must deceleration
		Computing brake forces
Alternative scenario 3:	Braking in emergency case	Recognizing emergency case
		Strengthening must deceleration
		Determining must deceleration
		Computing brake forces
Alternative scenario 4:	Braking at uphill driveaway	Determining vehicle speed
		Determining must deceleration
		Computing brake forces

Table 3-1 Use case "Braking"

In Table 3-2 a matrix is shown, which shows the processing functions participating in the realization of a use case. To a processing function the highest safety integrity level of a scenario which uses that processing function, will be assigned.

Here to each of the scenarios of the use case "Braking" a safety integrity level of 3 is assigned. The reason for this common level is that in each scenario a system fault could cause, that the system performs a braking operation without a driver's request and could lead to death of several people.

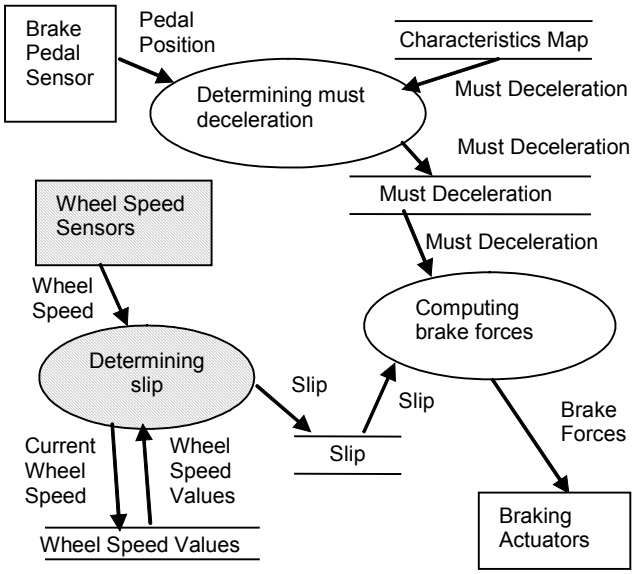

Figure 3-1 Data flow diagram of alternative scenario 1, "Braking at a low adhesion factor"

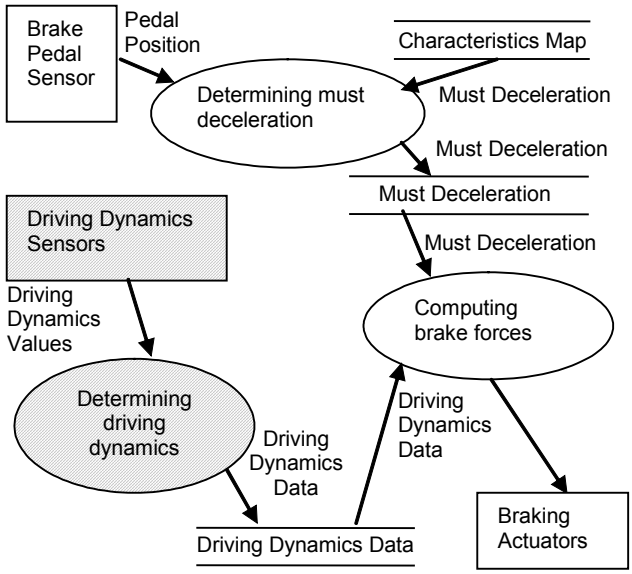

Figure 3-2 Data flow diagram of alternative scenario 2, "Braking at the driving dynamic limit"

Processing functions of the systems analysis	Use case "Braking"					SIL of the processing function (Couplings are not considered)
	Scenario "Braking in normal case" SIL3	Scenario "Braking at a low adhesion factor" SIL3	Scenario "Braking at the driving dynamic limit" SIL3	Scenario "Braking in emer-gency case" SIL3	Scenario "Braking at uphill driveaway" SIL3	
Determining driving dynamics	-	-	X (SIL3)	-	-	SIL3
Determining slip	-	X (SIL3)	-	-	-	SIL3
Determining vehicle speed	-	-	-	-	X (SIL3)	SIL3
Recognize emergency case	-	-	-	X (SIL3)	-	SIL3
Determining must deceleration	X (SIL3)	X (SIL3)	X (SIL3)	X (SIL3)	-	SIL3
Computing brake forces	X (SIL3)	X (SIL3)	X (SIL3)	X (SIL3)	X (SIL3)	SIL3

Table 3-2 Determining the safety integrity level of the processing functions

4. Use case-based safety analysis in system design

In system design the flow diagrams of system analysis are mapped onto the system hardware, on which the software is going to run. Figure 4-1 contains in addition to the system functions already present in Figure 3-1 and Figure 3-2, further system functions which act as interfaces between the processing functions and the sensors or the actors, respectively. The system function "Determining wheel speed" determines the speed of the wheels and transmits them to the system function "Determining Slip". The brake pedal position is determined by the system function "Determining brake pedal position", which reads out the brake pedal sensor. The brake forces, which are computed by the system function "Computing brake forces", are transmitted to the individual brakes by the system function "Controlling Brake Actuators". The driving dynamics sensors are read in by the system function "Polling Driving Dynamics Sensors" and the driving dynamics values are transmitted to the system function "Determining Driving Dynamics". After the system functions are designed, a qualitative Fault Tree Analysis is performed.

In this paper the Use Case-Based Fault Tree Analysis will be introduced, the Use Case-Based Failure Mode and Effects Analysis will be published in [11].

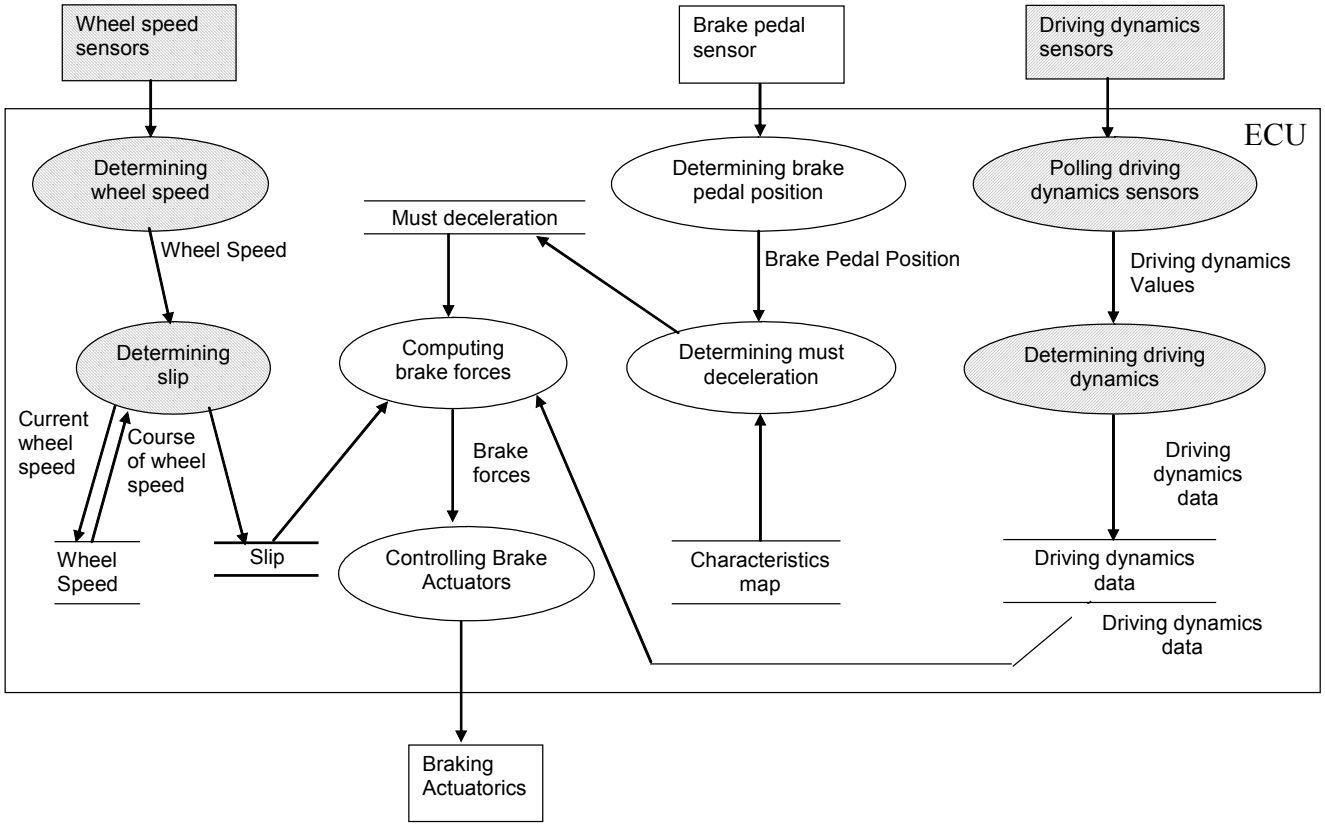

Figure 4-1 System design

4.1. Qualitative Fault Tree Analysis

According to DIN 25424 [5] the Fault Tree Analysis has the following goals:

1. Systematic identification of all possible failure combinations, which lead to the undesired event.
2. Determination of reliability measures.

The IEC 1025 [12] also describes the Fault Tree Analysis and points out another goal for the Fault Tree Analysis:

- Determination of the factor(s) which most seriously affect a particular reliability measure and the changes required to improve that measure.

According to DIN 25424 the Fault Tree Analysis is performed in the following steps:

1. Analysing the system
2. Defining the undesired events (top level events)
3. Defining the reliability measures, which are relevant for the FTA
4. Determining the failure modes of the components
5. Establishing the fault tree
6. Evaluating the fault tree

The DIN 25424 assumes that the Fault Tree Analysis is applied on an already existing system. Therefore the first step "Analysing the system" contains the decomposition of the system into components (see steps 1, 2, 3a and 3b in Figure 2-2) and the analysis of their failure modes. In the case of Use Case-Based Fault Tree Analysis, however, the Fault Tree Analysis is integrated in the system design, so the "Analysing of the system" does not have to be done again.

According to DIN 25424 the undesired events are defined, which represent a dangerous failure of the system (see Step 3c in Figure 2-2). In the qualitative analysis it is analysed which failure combinations (cut-sets) lead to the top level event. In the quantitative evaluation (see Step 5 in Figure 2-2) reliability measures are defined, which are used to calculate the probability of occurrence of failure combinations, the occurrence probability of the top level event or the unavailability of the system on demand.

4.2. Qualitative Use Case-based Fault Tree Analysis

In a Fault Tree Analysis all combinations of failure states of the components, the faulty operations of the system as well as harmfully environmental affects to the system are systematically determined and can potentially be the cause for a distinct top level event. However, the backward reasoning from a certain top level event to its causes can be very difficult, if the complete system has to be considered all at once. It is much easier to think in single use cases and to perform the qualitative Fault Tree Analysis in a use case-based way and to consider only one use case when modelling a fault tree. In this way the safety analysis is done in a use case-based manner like the system analysis, the system design and the system test. This procedure has the huge advantage that one can trace a use case from the beginning of development up to the system test without a gap. This traceability allows to develop the system in a straightforward and homogeneous way, to understand it easily and to check, whether the results of different activities are consistent to each other. The total fault tree will then be created by superimposing the fault trees of all use cases, which have the same top level event.

In the Use Case-Based Fault Tree Analysis, which is integrated in the use case-based system design, the same steps are performed as in the conventional Fault Tree Analysis, but for each use case or its scenarios (main scenario and alternative scenarios) a separated fault tree is modelled.

In a qualitative Use Case-Based Fault Tree Analysis, which is integrated in the use case-based system design, the following analysing steps are performed:

1. Defining the undesired events (top level events)
2. Defining the reliability measures, which are relevant for the FTA
3. Determining the failure modes of the components
4. Identification of the components, which are relevant for each use case
5. Preparing the use case-based fault trees

In a quantitative Use Case-Based Fault Tree Analysis the following analysing steps are performed:

6. Combining the fault trees of the different use cases
7. Evaluating the total fault tree

In the example shown below the qualitative Fault Tree Analysis is performed for the alternative scenario 1, "Braking at a low adhesion factor", and for the alternative scenario 2, "Braking at the driving dynamic limit". In the qualitative Use Case-Based Fault Tree Analysis the analysing steps two and three are performed in the same way as in the conventional Fault Tree Analysis. Therefore these steps will not be considered here.

4.2.1. Defining top level events

In an automotive braking system there are basically two different kinds of failure modes:

1. The braking operation is performed too late, too low or not at all despite the driver's request.
2. The braking operation is performed too powerful or at a wrong time, i.e. without a driver's request.

In this example the following top level events are defined:

1. No braking or too low braking operation at the driver's request.
2. Self-acting braking without a driver's request.

For the illustration of the qualitative Use Case-Based Fault Tree Analysis the fault tree for the top level event "No braking operation or too low braking at the driver's request" is shown in Figure 4-2 and Figure 4-3.

4.2.2. Identification of the relevant components

After the identification of the top level events, the hardware components and system functions, which are used by the individual use cases, are identified. The hardware component's faults and the system function's faults are the elements of the fault tree.

In the following example the hardware components for the alternative scenario 1, "Braking at a low adhesion factor", are the braking actuators, the pedal sensor, the wheel speed sensors and the computer hardware. The involved system functions are "Determining brake pedal position", "Determining must deceleration", "Determining slip", "Computing brake forces" and "Controlling brake actuators".

The hardware components for the alternative scenario 2 are the braking actuators, the pedal sensor, the wheel speed sensors and the computer hardware. The involved system functions are "Determining brake pedal position", "Determining must deceleration", "Determining driving dynamics", "Computing brake forces" and "Controlling brake actuators".

4.2.3. Modelling the use case-based fault trees

The preparation of the fault trees is performed in the same way as in the conventional qualitative Fault Tree Analysis. In Figure 4-2 and Figure 4-3 the fault trees for the discussed alternative scenarios of the use case "Braking" are shown.

The use case-based fault trees are created in the same way for the top level event "Self-acting braking without a driver's request".

4.2.4. Combining the fault trees

After the fault trees of all use cases have been created for a special top level event, the combination of all fault trees follows for this top level event. The result is a total fault tree for this top level event for the total system.

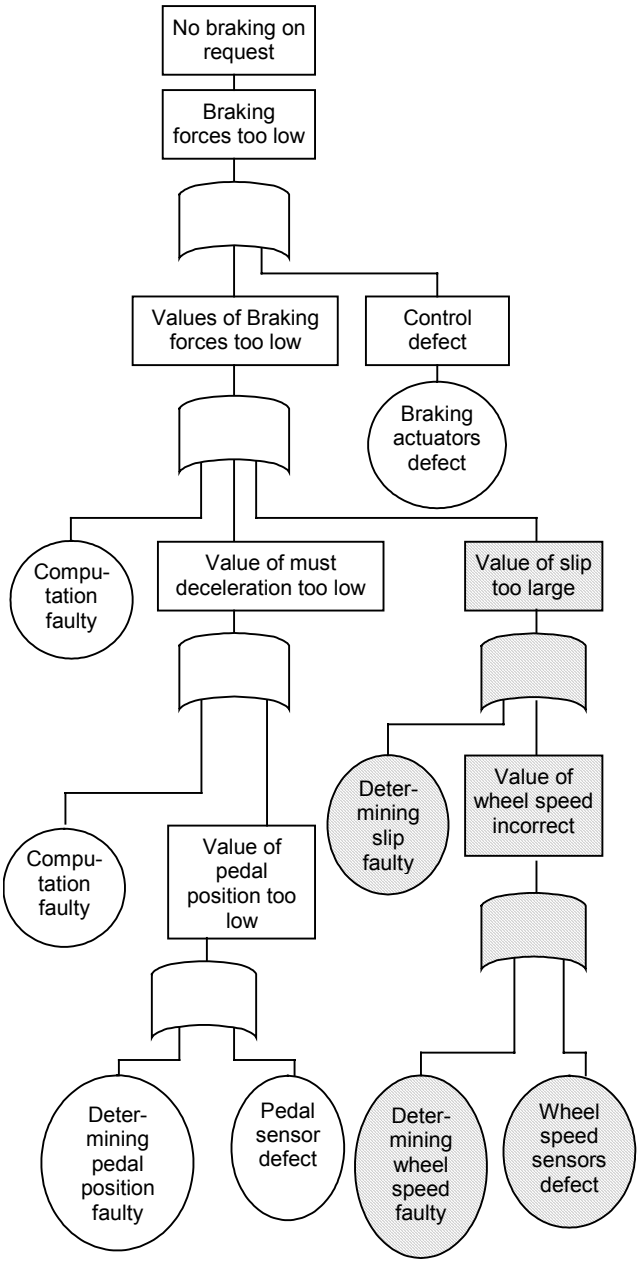

Figure 4-2 Fault tree for "Braking at a low adhesion factor"

The fault tree of the total system is developed step by step by adding all use case-based fault trees to the total fault tree. The differences between the use case-based fault trees in Figure 4-2 and in Figure 4-3 are hatched.

When comparing both fault trees it can easily be recognised, that the main scenario "Braking in normal case" of the use case "Braking" is modelled in both fault trees. The main scenario of the use case "Braking" is represented un-hatched. Such an overlapping of fault trees represents couplings, i.e. dependencies within the system.

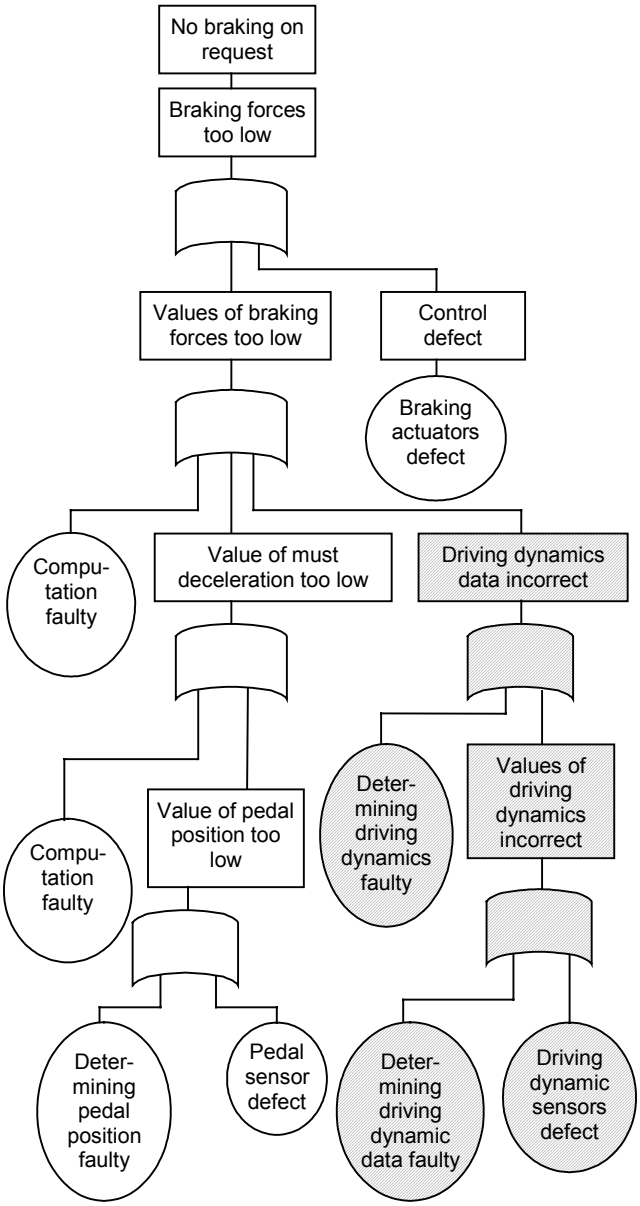

Figure 4-3 Fault tree for "Braking at the driving dynamic limit"

4.2.5. Evaluation of the fault trees

In the evaluation of the use case-based fault trees independently of the total fault tree only the cut-sets within the use case-based fault trees can be identified. If there are cut-sets with only few events, it is possible to think about whether measures in the architecture like introducing redundancies have to been taken or not, before the fault trees are put together. A quantitative evaluation of the use case-based fault trees makes only sense for computing the probability of the undesired event as a basis for the decision, whether measures to improve the reliability have to be taken or not.

Measures, for instance, could be adding hardware redundancies or replacing error-prone components by components with lower failure rates.

After the fault trees are put together and the total fault tree is established, the cut-sets are determined and the quantitative evaluation of the fault tree is performed for the total system in the conventional way.

5. Conclusion

The use case-based approach for the safety analysis of safety-related systems represents the natural continuation of the use case-based system development. Due to the existing system structure, which was developed in system design (see steps 1, 2, 3a and 3b in Figure 2-2), it is not necessary in safety analysis to set-up a separate decomposition of the system. This saves time and contributes to the correctness of safety analysis, too. Furthermore, the integrated use case-based approach allows tracing a use case from the requirements up to the system test, including all steps of development as well as of safety analysis.

In the safety analysis in system design the qualitative Use Case-Based Fault Tree Analysis was introduced. The logically structured way of this procedure reduces the complexity of the analysis which considerably contributes to its correctness.

Acknowledgements

The authors want to thank Dr. Melanie Cossy and Juergen Gaugele for taking part in discussions, which have contributed to the creation of this paper.

References

[1] Ismail Dagli, *Detection of cutting-in vehicles – a feasibility study*, VDI 2003

[2] http://www.v-modell.iabg.de/vm97.htm

[3] IEC 61508; *Functional Safety of electrical/electronic/programmable electronic safety related systems*, International Electrotechnical Commission, 1998

[4] IEC 60812:1985, *Analysis techniques for system reliability – Procedure for failure mode and effects analysis (FMEA)*, International Electrotechnical Commission 1985

[5] DIN 25448, *Ausfalleffektanalyse (Fehler-Möglichkeits- und –Einfluss-Analyse)*, Deutsches Institut für Normung, 1990

[6] IEC 61025:1990, *Fault Tree Analysis (FTA)*, International Electrotechnical Commission, 1990

[7] DIN 25424, *Fehlerbaumanalyse, Methode und Bildzeichen*, Deutsches Institut für Normung 1981

[8] D. Harel and M. Politi, *Modeling Reactive Systems with Statechart,* (McGraw-Hill)

[9] Heinisch C., Goll J., Consistent Object-Based Software Construction for Embedded Applications, Proceedings of *Software Engineering 2004*, Innsbruck

[10] Melanie Cossy, Development process and software architecture for safety-related and distributed applications in vehicles, dissertation University of Tuebingen 2005

[11] E. Balz and J. Goll, Use Case-basierte Ausfalleffektanalyse von sicherheitsbezogenen eingebetteten Systemen, to be published in the procedings of *Zuverlaessigkeit in eingebetteten Systemen*, Aachen, 2005

[12] CEI/IEC 1025, *Reliability and Maintainability: Fault Tree Analysis (FTA)*, International Electrotechnical Commission 1990