INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

MODELING AND ANALYZING FAULT-TOLERANT MECHATRONIC SYSTEMS

Prof. Dr.-Ing. J. Gausemeier[1], Dr.-Ing. M. Pöschl[2], Dipl.-Wirt.-Ing. Sebastian Deyter[1], M. Sc. L. Kaiser[1]
(1) Heinz Nixdorf Institute, University of Paderborn (2) PROSTEP IMP GmbH

ABSTRACT
The increasing functional integration of mechanical and electronic components in mechatronic systems leads to numerous interdependencies. Not recognized in time, they often cause problems regarding the product reliability. Assuming the appearance of a failure can be traced back to a sequence of events, failure-continuation models are useful for the evaluation of the reliability of mechatronic systems. The most established methods are the Failure Modes and Effect Analysis (FMEA) and the Fault Tree Analysis (FTA). Generating fault trees by hand is time-consuming. There is a need to support the generation of fault trees. By extending the specification technique, developed by the University of Paderborn, the causes of unintended system outputs can be traced back in an early development phase. The needed extensions are presented and applied at the principle solution of a fault-tolerant active steering system.

Keywords: specification technique, fault-tolerant, fault tree, mechatronic system

1 INTRODUCTION
The products of mechanical engineering and related industrial sectors, such as the automotive industry, are often based on the close interaction of mechanics, electronics and software engineering, which is aptly expressed by the term mechatronics. They allow the realization of new functions or the substitution of present solutions with an improved cost benefit relation [1], [2]. Three essential trends mark the application of mechatronic systems in the mentioned sectors:

- shorter innovation cycles due to the dynamic technological evolution and the high competitive pressure,
- rising functionality of the products enabled by the increasing efficiency of the information processing and
- increasing product varieties caused by the global competition and customer individual product configurations.

The increasing functional integration of mechanical and electronic components in mechatronic systems leads to numerous interdependencies. Not recognized in time, they often cause problems regarding the product reliability. The established approaches for reliability analysis such as simulations and tests occur mainly in the later development process phases. This results in expensive and time-consuming iteration loops. The application of adequate methods and models in the early design phases supports the developing of reliable systems [3]. Today there is a lack of procedures, methods and tools for the reliability oriented product development in the early development phases. The few existing approaches are not linked to each other sufficiently.

Summing up, there is a need for preventive approaches that identify potential weak spots in the system to be developed. This especially concerns the conceptual design phase [4]. To fill this gap, the established methods for analyzing the reliability have to be applied to the available information in these development phases. As a starting point for analyzing the reliability we use a cross-discipline product concept the so-called principle solution. The principle solution determines the basic structure and the operation mode of the system and, subsequently, it is the basis for further concretization [5].

2 METHODS TO EVALUATE THE RELIABILITY OF MECHATRONIC SYSTEMS

Assuming the appearance of a failure of one unit can be traced back to a sequence of events, failure-continuation models are useful for the evaluation of the reliability of mechatronic systems. The most established procedures are the Failure Modes and Effect Analysis (FMEA) and the Fault Tree Analysis (FTA). A FMEA is a procedure for analysis of potential failure modes within a system for classification by severity or determination of the effect of failures on the system [6]. A FTA is a failure analysis in which an undesired state of a system is analyzed using Boolean logic to combine a series of lower-level events. The FTA is mainly used in the field of safety engineering to quantitatively determine the probability of a safety hazard [7]. To motivate the usage of these procedures its appliance on mechanical engineering, software engineering and mechatronic systems is considered in the following.

The use of 3D geometry models is common in the development of mechanical components. These models can be analyzed in terms of static collisions, of mechanical tensions up to analyzing the systems deforming in case of dynamic collisions. The FTA is also used on mechanical components, especially at security-relevant components. Nevertheless, the FMEA is more established in this domain because of the mainly local restricted fault continuation. A systematic conclusion on possible failure causes by means of a FTA is rarely necessary. Furthermore an extensive knowledge of experience grown for years exists for mechanical components regarding the predominantly abrasion caused faults.

In the automotive sector the paradigm of model-driven software development became common. The software is developed by using graphical notation techniques to create abstract software models. Therefore a set of diagrams is used. Every diagram represents a view of the software under construction. Complex interrelations can be described more transparent than at source code level. The usage of FTA is uncommon in the software development. This is due to the fact that software does not abrade and it is present in a formal or semiformal way. Hence, failures caused by software can be traced back almost at all times on systematic mistakes. To find these failures other methods than FTA or FMEA are used, e.g. testing and formal verification often combined with strict modeling or encoding guidelines.

As mentioned above mechatronic systems functions are based on the cooperation of a huge number of distributed and mostly extremely linked up system elements, consisting of mechanical components, sensors, actuators, energy supply systems, wiring as well as control units consisting of hardware and software. Requirements according to the safety and reliability refer inevitable to the totality of these systems. As a result they have to be examined or validated at this level. Equal decisions on redundancy have to be made and to be validated at this level. The FTA is appropriate for analyzing the fault continuation in these linked up systems. One of its strengths is the applicability with non-formal input data. Nevertheless, the appliance with non-formal input data without technical support is time-consuming. These expenses put the expected benefit into perspective. There is a need to support the generation of fault trees. For generating fault trees in an automatically way formal input data are needed. The input data have to be provided by the architecture model which describes the principle solution of the system to be developed.

In the following three specification techniques for mechatronic systems are introduced. Subsequent an extension of one of these techniques is presented which enables to apply the common FTA on the principle solution. At the end of this paper the methodology is applied to the example of a fault-tolerant active steering system.

3 DOMAIN-SPANNING SPECIFICATION TECHNIQUES FOR MECHATRONIC SYSTEMS

A method for modeling and analyzing the fault-tolerance of mechatronic systems should enable to

- model the system elements,
- model the dependences of system elements in- and output,
- model the fault states of system elements and
- model the causes of the fault states

for examine the fault continuation in the overall system automatically. Several techniques for a domain-spanning specification of mechatronic systems exist. In the following three approaches are briefly shown and compared in terms of their suitability for modeling and analyzing fault-tolerant mechatronic systems.

Object-Process Methodology: Object-Process Methodology (OPM) is a system development and specification approach that combines the major systems aspects – function, structure and behavior – in a single graphic and textual model. OPM is designed to be very intuitive. It uses Object-Process Diagrams (OPDs) for the graphic specification and Object Process Language (OPL) for the textual specification. The two models are redundant and complement each other for are easier understanding. If the text is not well understood at some point along the OPL script, the corresponding OPD sentence – a construct of one or more OPD graphic symbols – can be examined to obtain clarification [12].
The OPD is build of three types of entities: objects, processes and states with objects and states being higher-level building blocks. According to [11] Objects, symbolized by rectangles, are things that exist. Processes, symbolized by ellipses, are things that transform objects by changing their states or by generating or consuming them. States, symbolized by rountangles, are situations objects can be in.
For connecting the entities the OPD uses two types of links: structural links and procedural links. Structural links express persistent, long-term relations among objects or among processes in the system. Aggregation and specialization are two examples of structural links. Procedural links express the systems behavior. Result links, input and output links are examples for procedural links [11].
Figure 1 shows the most important concepts that OPM is build upon.

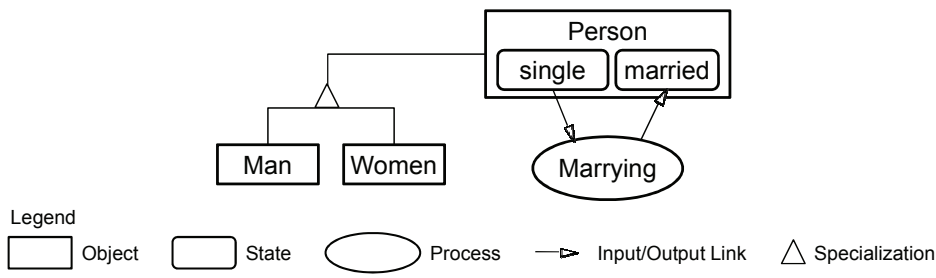

Figure 1. Example of an Object-Process Diagram [11]

The following sentences in OPL describing the meaning of
Figure 1 additionally: Person can be single or married. Man and women are persons. Marrying transform Person from single to married.
The Object-Process Methodology is suitable for specifying technical systems in an intuitive way. However, for modeling and analyzing fault tolerant mechatronic systems it is necessary to describe the interaction of the systems elements by their in- and outputs as well as the dependencies between the in- and outputs. The Input/Output link in the OPM is a procedural link. It specifies an intended order of states and processes. The physical or logical outputs of the systems elements in terms of a flow of material, energy or information cannot be specified.

System Modeling Language (SysML): SysML is a semiformal, graphic language for the modeling, analysis and verification of systems. Furthermore, it implies an amount of diagrams that are used to describe the views requirements, parameters, structure and behavior. SysML is a further development of the UML 2.0 for the purpose of system engineering. In contrast to UML, SysML is able to portray physical flows and continuous functions as well as input and output parameters [14]. By using SysML, technical systems can be described continuously from their requirements up to their conceptual design. For modeling and analyzing fault-tolerant mechatronic systems it is necessary to describe the interaction of the systems elements in an intuitive way. For this purpose the following specification technique is more suitable.

Specification Technique for the Description of Self-Optimizing Mechatronic Systems: Within the Collaborative Research Centre (CRC) 614 "Self-Optimizing Systems and Structures in Mechanical Engineering" of the University of Paderborn, a set of specification techniques in order to describe the principle solution of self-optimizing systems has been worked out. By using this specification technique, the system that is to be developed will be described in a holistic domain-spanning way. The description of the principle solution is divided into aspects. According to Figure 2, those aspects are requirements, environment, system of objectives, application scenarios, functions, active structure, shape and behavior. The mentioned aspects are mapped on computer by partial models. The principle solution consists of a coherent system of partial models because the aspects are in relationship with each other and ought to form a coherent system.

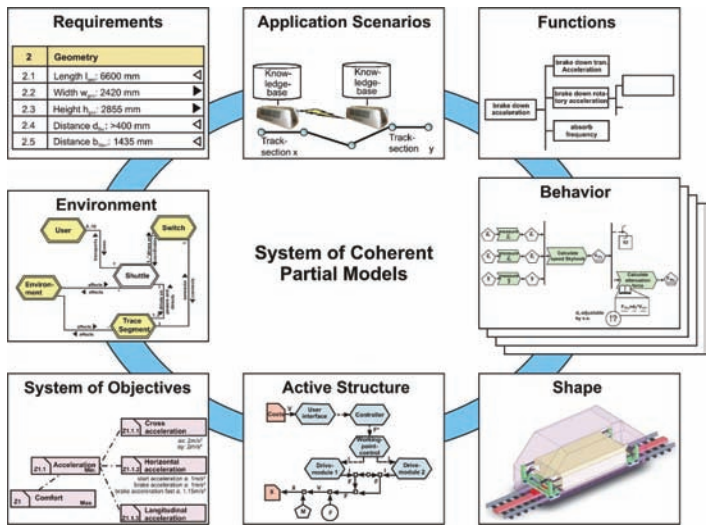

Figure 2. Partial models for the domain-spanning description of the principle solution of self-optimizing systems [8]

In the following the partial models are briefly described.

Environment: This model describes the environment of the system that has to be developed and its embedding into the environment. Relevant spheres of influence (such as weather, mechanical load, superior systems) and influences (such as thermal radiation, wind energy, information) will be identified. Disturbing influences on the system's purpose will be marked as disturbance variables. Furthermore, the interplays between the influences will be examined. We consider a situation to be one consistent amount of collectively occurring influences, in which the system has to work properly. We mark influences that cause a state transition of the system as events. Catalogues, that imply spheres of influences and influences, support the creation of environment models.

Application scenario: Application scenarios form first concretizations of the system. They describe the system's behavior in a special state and a special situation and furthermore, what kinds of events initiate a certain state transitions. Application scenarios characterize a problem, which needs to be solved in special cases, and also roughly describe the possible solution.

Requirements: This aspect considers the computer-intern representation of the requirements. The list of requirements sets up its basis. It presents an organized collection of requirements that need to be fulfilled during the product development (such as overall size, performance data).
System of objectives: This aspect includes the representation of external, inherent and internal objectives and its connections. The external and inherent objectives are represented as a hierarchical tree. The hierarchy relations are specified by logical relations with declarations of the hierarchy criterion "is part-objective of". The potential internal objectives derive from the external and inherent objectives. The system of objectives is used to identify the need for self-optimization.
Functions: This aspect concerns the hierarchical subdivision of the functionality. A function is the general and required coherence between input and output parameters, aiming at fulfilling a task. Functions are realized by solution patterns and its concretizations. A subdivision into sub functions is taking place until useful solution patterns can be found for the functions.
Active structure: The active structure describes the system elements, its attributes as well as the relation of the system elements. It is the target to define the basic structure of a self-optimizing system, including all system configurations which can be thought ahead. The active structure consists of system elements, such as actuators, sensors and information processing.
Shape: This aspect needs to be modeled because first definitions of the system's shape have to be carried out already in the phase of the conceptual design. This especially concerns working surfaces, working places, surfaces and frames. The computer-aided modeling takes place by using 3D CAD systems.
Behavior: This group of partial models comprises several kinds of behavior. Basically, what is needed to be modeled are the system's states with the connected operation processes and the state transitions. The partial model **behavior – states** defines the states and state transitions of a system. All of the system's states and state transitions, which can be thought ahead and thus, need to be considered, as well as the events initiating a state transition need to be described. Events can be characteristic influences on the system or already finished activities. The partial model **Behavior – Activities** describes the mentioned operation processes, that take place in a system's state, and the adaptation processes, which have the typical features of self-optimization. All in all, the processes are modeled by activities.
Of high importance are the interrelations between the partial models which describe the coherence of the partial models. Those interrelations are built up between the constructs of the relating partial models [8].

None of the introduced specification techniques fulfills all the requirements mentioned above. However, the Specification Technique for the Description of Self-Optimizing Mechatronic Systems is the most suitable technique for modeling and analyzing fault-tolerant mechatronic systems. In the following a method for applying the FTA on the principle solution specified by the specification technique is introduced.

4 APPLIENCE OF THE FTA ON THE PRINCIPLE SOLUTION

Figure 3 shows an example of the **active structure** of a simplified active steering system. The active steering system allows changing the steering transmission ratio according to the driving situation. At low speed the steering transmission ratio is low for easy parking and turning. At high speed the steering transmission ratio is high to handle lane changes easier. The active structure of the steering system consists of a steering wheel, a steering wheel angular sensor, a battery, a µC, a DC motor and a planetary drive. The steering wheel transforms a hand force into a wheels steering torque. The steering wheels torque is the input for the planetary drive. The steering wheel angular sensor measures the current steering wheel angle and transfers it as information to the µC. The µC calculates a needed angle that has to be added to the driver's steering angle according to the current car's speed. The DC motor transfers the needed angle into a torque. The DC motor's power supply is not in the scope of this investigation. The planetary drive adds the driver's steering torque to the motor's torque and transfers it to the front wheels.

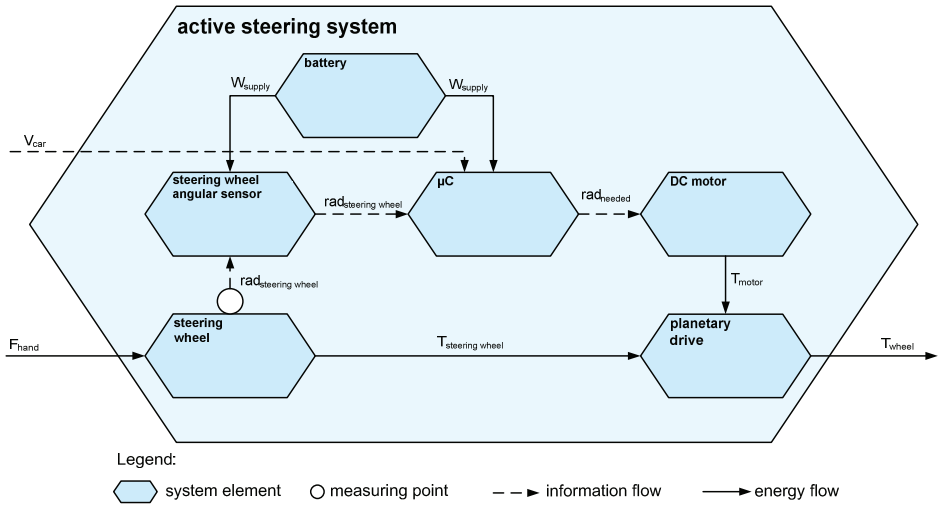

Figure 3. Exemplary partial model active structure of an active steering system

According to [8] the states of a system to be developed are described with the partial model **behavior – states**. Figure 4 shows exemplary the partial model behavior – states for the active steering system's µC. It specifies the states of the µC and the events initiating state transitions. It is also possible to describe the states of the whole steering system. The exemplary partial model consists of the three states "off", "start-up phase" and "on". The process starts with the state "on". If the event "W_{supply} on" occurs, the µCs state change to "start-up phase" and so on.

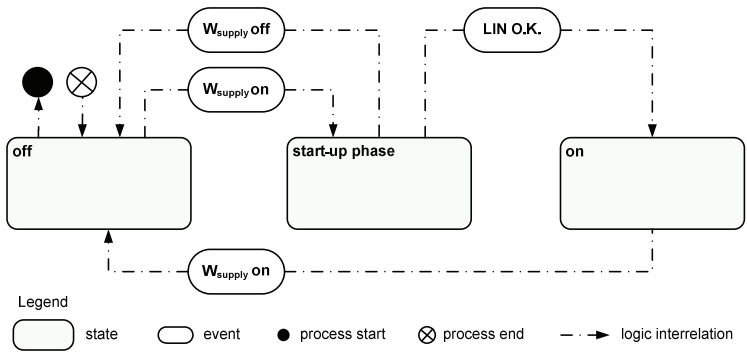

Figure 4. Exemplary partial model behavior-states of the µC

The partial model behavior – states describes the intended states and state transitions of a system. This is adequate for specifying a systems behavior. For analyzing the reliability of a system the possible fault states of a system are needed. For that purpose the partial models have to be extended.

To model the fault states of a system state tuples are used. A state tuple consists of several states. For state transitions the following rules are used:

1. Always one state of a state tuple is active.
2. A state is active if all its needs are fulfilled

The preconditions of a state to be active are modeled as needs. For a better overview the state tuples and needs are added within the active structure. If more than one need has to be fulfilled for a state to be active, the needs are connected by Boolean operators to express the kind of interrelations. Figure 5 shows an example of an active structure added with state tuples.

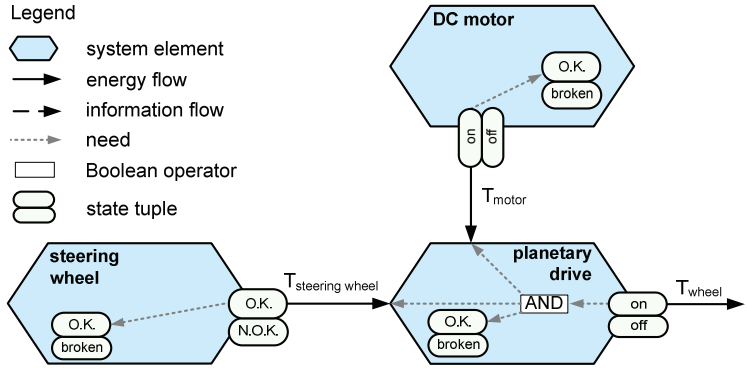

Figure 5. Exemplary active structure with added state tuples

Figure 5 is to read as follows. The planetary drive's output is a torque transferred to the front wheels. The output state tuple consists of the states "on" and "off". The state "off" implies simplified a not intended torque. "Off" is the fault state of the planetary drive's output. The need for the state "on" is the state "O.K." of the planetary drive, the state "on" of the steering wheels output "$T_{steering\ wheel}$" and the state "on" of the DC motors torque output. The need for the steering wheel's torque output is the steering wheel to be in state "O.K.". The need for the DC motors torque output is the DC motor to be in state "O.K.".

To complete the state tuples and its consequences the application of FMEA is useful. To trace back the causes of unintended fault states of outputs or system elements the application of FTA is recommended. If the system is modeled as shown in Figure 5, a fault tree can be generated automatically for every state of interest. For generating a fault tree for the modeled system a fault of interest have to be indentified, the so called top event. Figure 6 shows the fault tree for the planetary drive's output-state "off". For tracing back faults which cause this state, the needs of the other state of the tuple has to be negated. To get into the off-state a need for the on-state has not to be fulfilled (see rule no. 2). The modeled needs for the T_{wheel} "on"-state are the planetary drive's state "O.K." **and** the O.K.-state of the input $T_{steering\ wheel}$ **and** the T_{motor} "on"-state. Consequently the T_{wheel} is in state "off" if the planetary drive's state "broken" is **active** or the $T_{steering\ wheel}$ state "N.O.K." is active **or** the input T_{motor} state "off" is active. For generating the full fault tree the fault causes have to be traced back until no more upstream cause can be found.

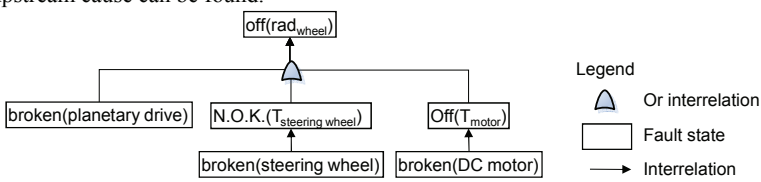

Figure 6. Example of a fault tree

In the following the introduced method is applied at a more comprehensive example to show its strengths accurately.

5 APPLICATION EXAMPLE

Subsequent the complete active steering system serves as an example. Figure 7 shows the active structure with the state tuples of the system elements of the steering system.

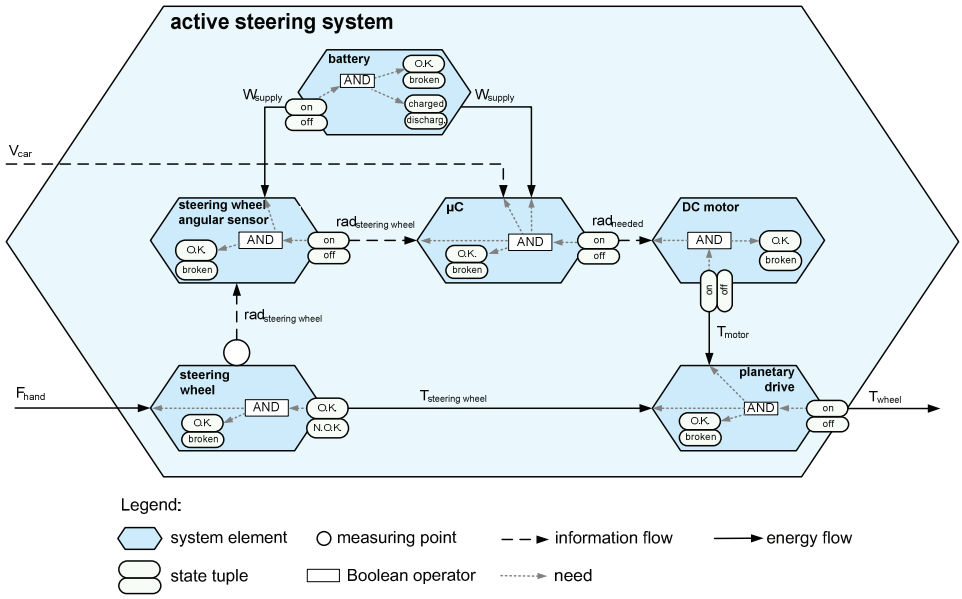

Figure 7. The steering systems active structure with added state tuples

An unintended wheel angle can cause deadly injuries on the cars occupants and third parties. Therefore this fault state cannot be accepted during the system's lifecycle. Figure 8 shows the fault tree of the steering system. Top event is the planetary drives output state "off".

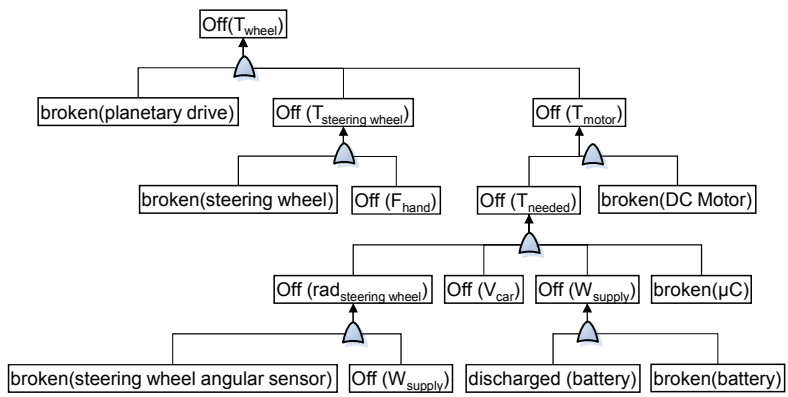

Figure 8. Fault trees for the active steering system

According to Figure 8 a breakdown of the DC motor or the µC or the steering wheel angular sensor or the battery can cause this top event. A breakdown of these system elements can not be excluded during the lifecycle of the steering system. The system's architecture has to be modified to meet the safety requirements of a steering system.

To assure a fault-tolerant functionality of the steering system in case of a breakdown of the mentioned system elements a requirement to a fail tolerant behavior for the steering system has to be claimed. A system is fault-tolerant when it is continuing a claimed operation, while an error or a deviation occurs [9]. A German automobile manufacturer has implemented a steering system which meets this requirement [10]. Figure 9 shows the simplified active structure of this system with added state tuples. The μC and the steering wheel angular sensor are designed redundant. The results of the μCs are compared by a compiler. If the results of the μCs are not equal, e.g. because of a breakdown of one μC, a mechanical break will be activated. The mechanical break will also be activated in case of a compiler breakdown, e.g. caused by a battery breakdown. The break fixes the shaft which connects the brushless DC motor with the planetary drive. The brushless DC motor is equipped with permanent magnets. The motor only turns in case of an alteration of the electric field in the stator. In case of a breakdown the motor itself fixes the connecting shaft.

The active steering system implements a decreased functionality. Within the planetary drives output state “O.K.” the system is full functional. In the state “steerable” the car is steerable but a steering wheel’s displacement is possible. In the state “N.O.K.” an unintended wheel angel occurs.

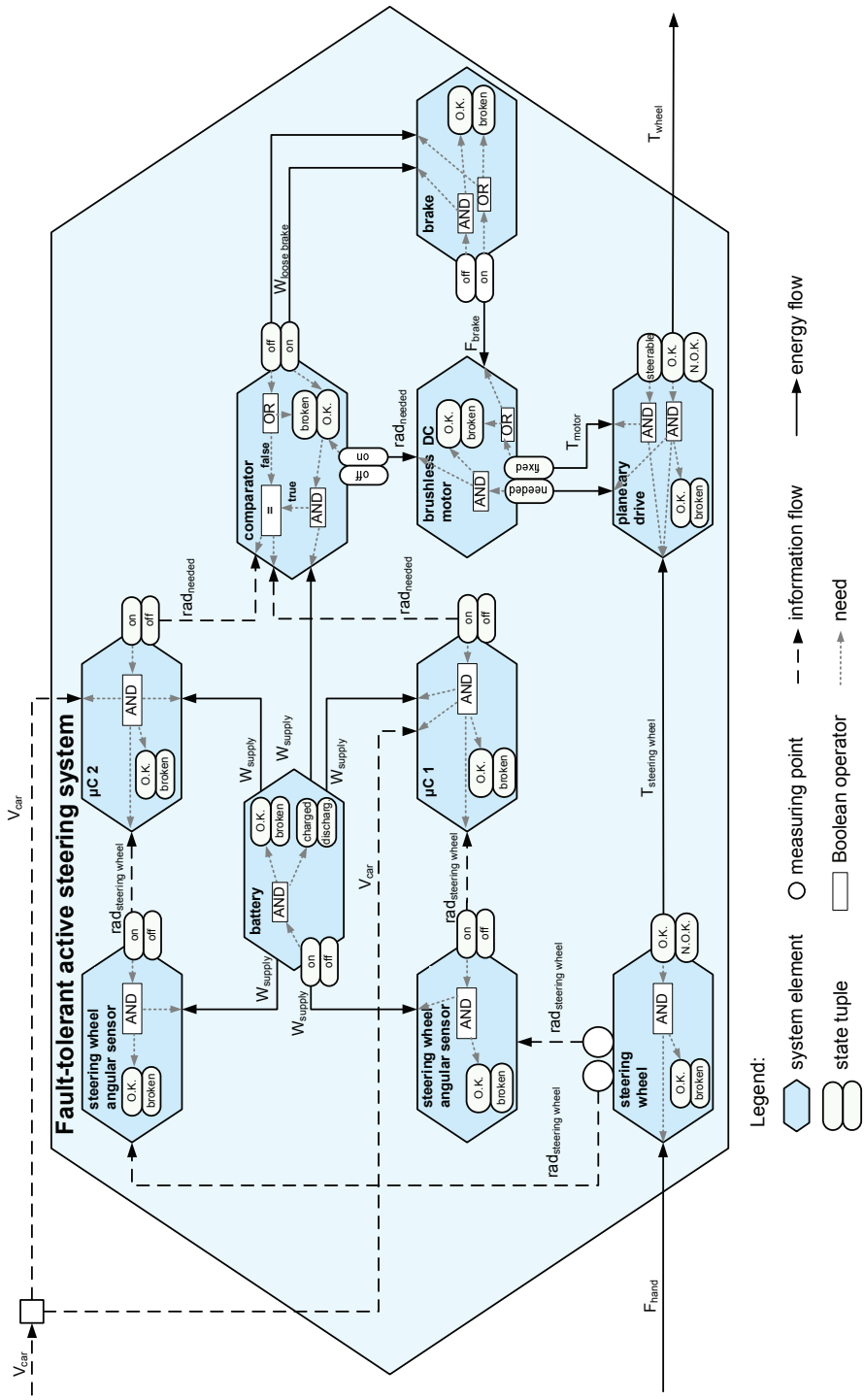

Figure 9. Active structure with added state tuples of the fault-tolerant active steering system

Figure 10 shows the fault trees for the top events “steerable” and “N.O.K.”. According to Figure 10 the active steering system is no not steerable anymore in case of a breakdown of the planetary drive or the steering wheel or one of the µC´s and one of the steering wheel angular sensors breaks down cross over. In case one of the µC´s, the comparator, the break, the battery, one of the steering wheel angular sensors, the DC motor breaks down or the battery is discharged the car is still steerable.

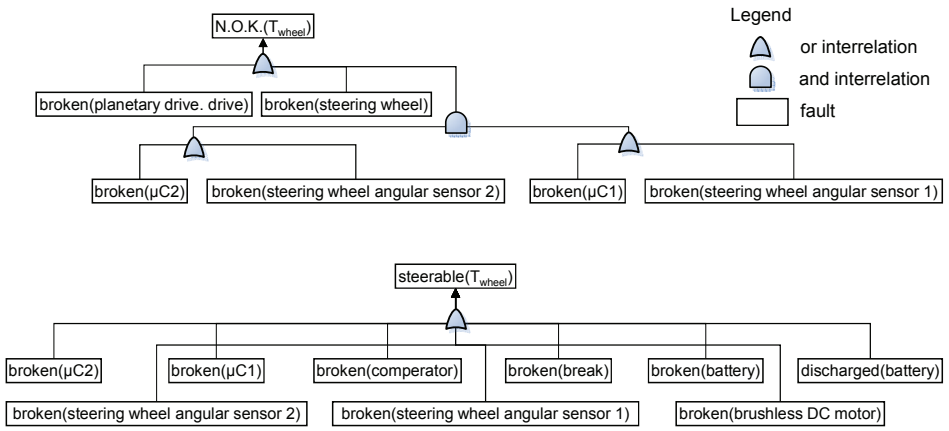

Figure 10. Fault trees for the active steering system

For reasons of acceptance it is necessary to form fault trees compactly and expressively. Software has been implemented for specifying mechatronic systems and generating fault trees in the shown way. The generated fault trees are reduced to causal fault states of the single system elements. Feedback loops in the graph, as they appear with regulated systems, are recognized reliable and taken into account. By using the Boolean minimization an at most shortened representation is reached.

The process of modeling and analyzing mechatronic systems is iterative and will be passed several times. By using the software fault trees do not have to be generated by hand in case of changing the specification new. Fault trees can easily be generated new. Furthermore the developed software indicates changes to the preceding specification versions.
In case of decomposing system elements into subsystem elements, the information described on top level must be likewise modeled for the subsystem elements. It originates one or several parallel requirement descriptions with regard to the fault tolerance – on the one hand for the higher system element, on the other hand for its subsystem elements. The software is able to compare the requirements on consistency and to announce inconsistencies. In this manner the process of architecture refinement is optimally supported.

6 CONCLUSION

To support the procedures for analyzing the reliability of systems to be developed the information provided by the principle solution is useful. The presented specification technique offers the possibility to create a principle solution for advanced mechatronic systems. Extending this specification technique by modeling state tuples and its needs the causes of unintended system outputs can be traced back automatically already in the early stages of the development. The methodology has been exemplified through the principle solution of a fault-tolerant active steering system. In use the methodology weak spots in the system to be developed can be found and expensive iteration loops can be avoided.

Future research will focus on generating test cases automatically based on specified requirements. This will improve checking the consistence between the specified system’s behavior and requirements. For a quantitative analysis of the generated fault trees the concept of scenario-based FMEA will be adopted.

7 ACKNOWLEDGEMENT

This work is implemented within an ongoing project "Instrumentarium für die frühzeitige Zuverlässigkeitsanalyse mechatronischer Systeme (InZuMech)", funded by the Federal Ministry of Education and Research (BMBF) of Germany.

REFERENCES

[1] Verein Deutscher Ingenieure (VDI). Design methodology for mechatronic systems. *VDI-guideline 2206,* 2004 (Beuth-Verlag, Berlin)

[2] Liske, S.; Westerhüs, J. Das Rennen um den SOP – Kernmerkmale eines Produkt-Entstehungs-Prozesses der Spitzenklasse. *Automotive Executive Newsletter der Unternehmensberatung Arthur D. Little*, 2004

[3] Ponn, J and Lindemann, U. *Konzeptentwicklung und Gestaltung technischer Produkte – Optimierte Produkte - systematisch von Anforderungen zu Konzepte,* 2008 (Springer-Verlag, Berlin)

[4] Jäger, P. and Bertsche, B. An Approach for Early Reliability Evaluation of Mechatronics Systems. European Safety and Reliability Conference. In *European Safety and Reliability Conference, ESREL 2005*, Santa Fe, USA, June 2005

[5] Gausemeier, J. and Frank, U. and Steffen, D. Specifying the principle solution of tomorrows mechanical engineering products. In *International Design Conference - DESIGN 2006*, Dubrovnik, 2006

[6] Deutsches Institut für Normung e.V. Analysetechniken für die Funktionsfähigkeit von Systemen – Verfahren für die Fehlerzustandsart- und Auswirkungsanalyse (FMEA), *DIN-Norm EN 60812,* 2006

[7] Deutsches Institut für Normung e.V. Fehlerbaumanalyse – Methode und Bildzeichen, *DIN-Norm 25424 Teil 1*, 1981

[8] Gausemeier, J. and Frank, U. and Donoth, J. and Kahl, S. Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus. *Konstruktion, part 1: July/August, part: 2 September 2008*

[9] Deutsches Institut für Normung e.V. Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbar elektronischer Geräte - Begriffe und Abkürzungen, *DIN EN 61508-4*, 2002

[10] Isermann, R. and Beck, M. *Fehlertolerante Systeme - Prinzipien und Ausführungsbeispiele.* Survey for the Deutsche Gesellschaft für die Anwendung der Mikroelektronik e.V., 2005

[11] Dori, D.: Object-Process Methodology – A Holistic Systems Paradigma. Springer-Verlag 2002. ISBN 3540654712.

[12] Siau, K.: Advanced Topics in Database Research. Idea Group Inc (IGI), 2006. ISBN 1591409357

[13] Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H.: Engineering Design – A Systematic Approach. Springer Verlag, London, 3rd edition, 2007

[14] Weilkiens, T.: Systems Engineering mit SysML/UML - Modellierung, Analyse, Design. dpunkt verlag GmbH, 2006

Contact: Prof. Dr.-Ing. Jürgen Gausemeier
Heinz Nixdorf Insitute, University of Paderborn
Fürstenallee 11
33102 Paderborn
Germany
Tel: Int +49 5251 60-6267
Fax: Int +49 5251 60-6268
Email: Juergen.Gausemeier@hni.uni-paderborn.de

Contact Dr.-Ing. Martin Pöschl
PROSTEP IMP GmbH
Taunusstraße 42,
80807 München
Phone.: +49 (89) 350 20 - 246,
E-Mail: martin.poeschl@prostep.com