**In:** Hong-Yue Zhang (ed.). Fault Detection, Supervision and Safety of Technical Processes (Safeprocess 2006), Elsevier, ISBN**:** 978-0-08-044485-7, 2006

# A MODEL-BASED METHODOLOGY FOR THE INTEGRATION OF DIAGNOSIS AND FAULT ANALYSIS DURING THE ENTIRE LIFE CYCLE

**Peter Struss**

*Technische Universität Munich and OCC'M Software GmbH, Deisenhofen*

The work presented here uses a library of behavior models of system components as a core for supportive tools for several work processes during the life cycle. Models of complex systems can be automatically composed from such library elements and provide the basis for the automated generation, instead of programming, of diagnostics and fault analysis. The feasibility of this model-based methodology is illustrated by case studies in the automotive and aeronautics industry. This includes a demonstrator for the automated generation of on-board diagnostics for a vehicle system. The same model basis is used to perform other tasks, such as failure-modes-and-effects analysis, diagnosability analysis, and the generation of tests. 

Keywords: Diagnosis, Test generation, Modeling, Diagnosability, Failure-modes-and-effects analysis, Life cycle.

## 1. INTRODUCTION

Tasks of diagnosis and fault analysis occur during the entire life cycle of a product. Due to increasing complexity of systems, their solution becomes more challenging and crucial. In many areas, the existence of many variants of technical devices makes them highly repetitive. Also, the same knowledge is exploited in many work processes during the life cycle. In current practice, these two kinds of repetition are related to human activities and, hence, consume time and precious labor. For instance, failure-mode-and effects analysis (FMEA), the production of on-board diagnostics, and the generation of diagnosis manuals have to be carried out whenever the structure of the designed or manufactured system changes. And the knowledge about possible faults and their impact that was used to produce an FMEA report has also to be applied (and retrieved or reproduced) in writing on-board diagnosis code or in workshop diagnosis.

A significant reduction of the time and efforts spent on these tasks can only be expected based on an approach that makes the shared knowledge explicit, represents it in a knowledge-base for further processing by human experts or computer-based problem solvers in the various life cycle tasks. What is needed in order to reduce the efforts spent on the various tasks is a comprehensive, general (system- and task-independent) and systematic approach to fault analysis and the generation of diagnostics and their automation. The methodologies, techniques, models, and tools currently available and used in engineering processes are usually domain and task dependent and fail to address these issues.

The aim of this paper is to show that model-based systems technology, which has been developed in Artificial Intelligence, is a significant contribution to addressing this challenge, and, rather than discussing the theoretical and technical details, its focuses on the methodological aspects.

In this approach, a library of generic, reusable component models represents the sharable knowledge. Automated model composition based on this library and a structural description of a specific plant together with domain-independent problems solving algorithms allow for the automated generation of the required plant-specific results, thus reducing or even eliminating the human efforts involved in the analysis and programming.

In the following section, we outline the foundations of this approach. Then, we demonstrate its maturity by describing its application to the automated generation of code for vehicle on-board diagnosis and the generation of FMEA reports. Finally, we discuss the reuse of the models and the algorithmic basis for other tasks during the lifecycle such as diagnosability analysis and test generation.

## 2. MODEL-BASED SYSTEMS

In order to understand why and how the model-based approach to problem solving in general, and diagnosis in particular, bears the potential of a significant reduction of time and costs, we ask what enables a human expert to perform diagnosis and develop diagnostic code for all kinds of variants of known systems and even for new types of systems. Obviously, it is her or his understanding of how these systems behave and how observed or measured aspects of this behavior relate to possible faults. That she or he is able to cope with novel devices, say an electrical circuit with a new structure shows that one part of this knowledge is generic. This part is mainly the knowledge about the **behavior** of the individual **components** that establish the device. It is applied in a repetitive way to each single device based on information about its **structure**, i.e. the (device-specific) interaction among the components.

As long as this foundational knowledge is available only in the heads of the diagnostics developer, the production of the diagnosis software will remain a repetitive one, and this means: costly. Any computer-based solution that addresses this issue has to move this knowledge into an explicit representation in a computer program and have the program apply this knowledge in a step of automated diagnostics generation, thus liberating the human from this repetitive task. This is what model-based diagnosis does.

Using the device model to generate diagnostics requires a diagnosis algorithm that is **generic**, i.e. independent of the specific device. This is the second contribution of model-based diagnosis. As a result, the **task-specific** software element (diagnosis) is **separated** from the description of the **subject** of the task (the model of the device to be diagnosed) which allows

- the re-use of tasks-specific algorithms for different devices and, in particular, the automated diagnostics generation and
- the re-use of device-specific models in different tasks (horizontal integration).

It should be obvious that this separation has a strong impact on the utility and cost-effectiveness of the solution in industrial applications and clearly distinguishes it from other approaches that also exploit system models, but interwoven with procedural diagnostic elements. In the following, we illustrate the idea of domain-independent problem solvers by explaining the basis of a general diagnosis algorithm and then outline some of the modeling principles.

### *2.1 The Foundation of Model-based Diagnosis*

We consider a trivial example comprising a pump which is mechanically driven and pumps air into a container (Figure 1). There are sensors measuring the inflow and the pressure in the container, and also the command that controls the mechanical drive is known. If we assume a scenario where there is a command which requires speeding-up the pump, while the sensors show an increased inflow to the pump and a pressure drop in the container, we are certain that a fault is present, and, moreover, we will reckon that the pump or the container has a leakage or the pressure sensor is failing.

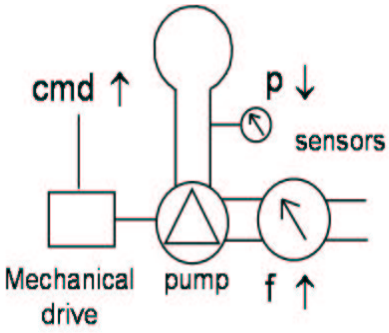


**Figure 1** A small diagnosis problem

How could an algorithm that is fed with models of the physical behaviors of the components plus information about their interconnections achieve this result? The core of the algorithm checks the consistency of parts of the device model with the given observations (which is why the approach is called consistency-based diagnosis). For instance, assuming the correct behavior of the pump, the container, and the two sensors is contradicting the sensor readings, because the models of normal behavior of these components imply that both values show the same tendency (sign of the first derivative). A logical implication of this check is

NOT OK(pump)
OR NOT OK(container)
OR NOT OK(pressure-sensor)
OR NOT OK(flow-sensor)

which indicates the **detection** of a fault. In the same way, the models of the mechanical drive, the pump, the container, and the pressure sensor together are in conflict with the command and the measured pressure which yields

NOT OK(mech-drive)
OR NOT OK(pump)
OR NOT OK(container)
OR NOT OK(pressure-sensor) .

Together these two “conflicts” logically imply

NOT OK(pump)
OR NOT OK(container)
OR NOT OK(pressure-sensor)
OR (NOT OK(flow-sensor)
AND NOT OK(mech-drive))

which is a (incomplete) **localization** (isolation) of the fault. We realize that this result not only reproduces the diagnostic hypothesis stated above, but also illustrates that the algorithm can easily generate multiple fault hypotheses (by suspecting the flow sensor and the mechanical drive of the pump). Furthermore, we emphasize that fault localization can be based on models that capture the correct behavior only and does not require models of component faults. They are needed if the fault has to be further **identified**, and they can be checked for consistency with the observations in the same way. For instance, the model of a stuck pump will be

refuted, whereas a leaking pump remains a hypothesis. (For a more detailed presentation of the theoretical foundations, see (Dressler-Struss 1996)
This admittedly trivial example should suffice to illustrate the possibility to generate diagnostic hypotheses based on a behavior model, and, more importantly, to suggest that this

- can be formalized in a rigorous **logical theory** and
- turned into an algorithm which is not reflecting the specific content of the models and, hence, **device- and even domain-independent**.

It is actually a simplification of a turbo charger application that was the subject of a real-world demonstrator on a test vehicle in the Brite-EuRam VMBD (Vehicle Model Based Diagnosis) project. In this project, Volvo Car Corp., Robert-Bosch GmbH, OCC'M Software GmbH, and the Technische Universität München developed a demonstrator for on-board detection and localization of black-smoke-related faults in a turbo charger Diesel engine (Volvo 850 TDI). The demonstrator vehicle had several built-in faults that could be activated through a switchboard, such as a leakage in the air hose between the turbine outlet and the intake manifold, a mechanical failure in the exhaust gas re-circulation valve, and sensor faults. Sensor and actuator signals from the engine ECU were transmitted via a serial line to a notebook running the model-based diagnosis runtime system. The signals exploited were atmospheric pressure, boost pressure, mass airflow, engine speed, duty cycle of the turbo control valve, and injected fuel quantity, i.e. the ordinary ECU signals with no additional sensors for diagnostic purposes.
The diagnosis runtime system was based on models of the (correct) physical behavior of the turbo charger system components and achieved fault localization with a performance meeting real-time requirements (Sachenbacher-Struss-Weber 2000).
The same principles have been applied to a variety of technical systems ranging from digital circuits and high-voltage power-transportation networks to ballast water tank systems.
The example also indicates that a model (and its execution) suited for this diagnostic task must have certain properties which we discuss in the following.

### *2.2 Automated Compositional Modeling*

The solution sketched in section 2.1 is based on the recognition of inconsistent **parts** of the overall system model. A black box model of the entire device would suffice for fault detection, but not yield any contribution to fault localization. For this, we need a **compositional** model, i.e. a model that allows the algorithm to identify the origin of a detected inconsistency in terms of the diagnosis-relevant entities of the device.
This property meets an important requirement that arises from the perspective of software development: given a generic diagnosis algorithm, the main step for generating device-specific diagnostics lies in the provision of a suitable device model. Since we try to address the variant problem and want to avoid unnecessary repetition of tasks, producing the model for each individual device and its variants from scratch is prohibitive. The solution is to collect models of (types of) components in a library and have the device model generated automatically by composing it from the library elements based on the structural description of the respective device.
Also these features,

- the automated composition of the device model from a library, and, hence,
- the re-usability of the library elements for different devices,

are essential for high gains in the development process, and again mark a distinction from other model-based solutions that depend on the development of device-specific models.
The rate of re-use of the library elements is further influenced by the fact that, for diagnostic purposes, often a qualitative description of the component behavior suffices.

### *2.3 Qualitative Modeling*

The example in section 2.1 illustrates that very coarse information, like the sign of the derivatives of the measured quantities, can suffice to derive at least a first set of diagnostic hypotheses. Using numerical values would not improve the diagnostic result in this case. As a consequence, the behavior models need to represent only distinctions that are relevant to discriminate correct from faulty behavior.
To give an example from the demonstrator described in the next section: the information about the measured position of the window in a car door provided by Hall sensors can be abstracted to the qualitative positions "top", "bottom", and "between" which completely suffices to check their consistency, say, with information about the direction of change. This turns the model into a very generic and re-usable one which requires only a mapping between the numerical position and the three qualitative positions.
Besides increased re-usability, qualitative models are essential to the effectiveness and efficiency of the model-based on-board diagnostics: they provide the basis for

- the representation of the model as a finite one which allows for the application of efficient algorithms for manipulation of relations and consistency checking,
- a compact representation of the model, and
- a significant reduction of computational efforts at runtime, because the input signals can be abstracted to qualitative values, and only changes in this qualitative information needs to be fed to the diagnostic algorithm.

## 3. AUTOMATED GENERATION OF ON-BOARD DIAGNOSIS CODE

The described model-based on-board diagnosis demonstrator of VMBD proved the principled feasibility of the approach, but, having been developed in a research project and running on a

Windows/Pentium PC, prompted two important questions:

- How can the production of such solutions be introduced in the industrial process of on-board software development?

These challenges have been addressed during the last years and the results were evaluated in a case study using a physical demonstrator of a comfort system of a Volkswagen Polo that had been created in the STEP-X project (Harms et al. 2003). This demonstrator comprises a rack with four window lifters and two mirrors with mirror positioning and the respective switches to operate these systems (Figure 2).

The functions are controlled by four ECUs (one for each door) that are connected via CAN bus. To evaluate diagnostics, roughly 100 faults could be switched on, including short and open circuits affecting the electrical motors and Hall sensors of the windows, detuning of switch resistances, and faults of the CAN bus and the connections to the various ECUs. For the benchmark, a fifth ECU, based on an Infineon C167 microprocessor, running the diagnosis software was connected to the CAN bus. CAN messages containing in total about 100 signals were transferred to a diagnosis ECU every 50 ms.

The software that performs diagnosis based on these messages is C code that is **automatically produced by a code generator** based on models of the four subsystems, their communication, and their power supply (Fig. 3). This means, the software does not only detect and identify faults at the subsystem level (window lifters and the respective control panels), but also at the level of interacting and interdependent subsystems. It should be noted that the decision to implement the diagnosis on a separate ECU was not due to principled, but only pragmatic reasons, namely the locally distributed development of the control and the diagnostic software. Usually, the subsystem-related diagnostics could be run on the respective ECU and the centralized part on a separate one or one or more of the existing ECUs.

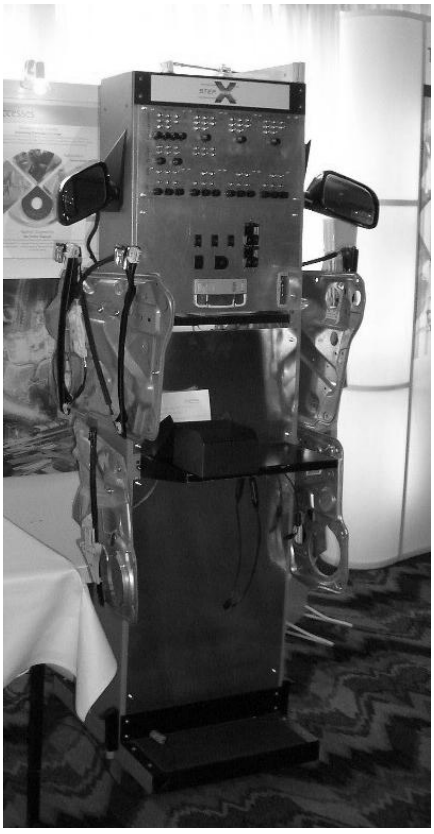

**Figure 2 The comfort system demonstrator**

The production of a device-specific diagnosis system becomes a sequence of two automated generation steps: First, the device model is generated based on the library and the "blueprint" of the device. Second, this model is compiled together with the general diagnosis algorithm into C-code for the ECU. Once the library exists and contains the necessary model elements, editing the structure and parameterization of the device or, ideally, importing it from a design system is the essential step in the development of diagnosis software. What remains to be done, is to specify how the available signals have to be mapped to the variables occurring in the model. The code generation approach has a major impact on the development process: any change or variation in physical systems to be diagnosed, such as exchanging a component or modifying the structure, requires only the respective modification of the model and re-running the automatic generation procedure in order to reflect the physical change in the diagnostics. No interaction on the software level is needed, but of course the code can be inspected and interfaced with e.g. control software.

The generated C-code when compiled is quite compact, requiring about **25 KB** of memory in total, i.e. including the signal preprocessing, the compressed system model, and the diagnosis algorithm. Even for the implemented centralized version of diagnosis, the system performed in **real-time** and with the processor running at 19,5 MHz used strictly less than **20%** of the **maximum performance capacity**. An important feature of the diagnosis system is that it is not restricted to single faults; even for each single subsystem, it handles **multiple faults** of arbitrary size. In the context of this demonstrator, it delivers **complete** diagnostic information in terms of a fault code for each 50 ms time window, where this code represents the complete **set of all possible faults** in this window (**not** a set of symptoms as current trouble codes). This can be the input for further processing, filtering, debouncing, statistics etc.

An evaluation of the diagnosis results based on the complete set of built-in faults was performed in cooperation by the Technical University of Braunschweig and Carmeq GmbH. Correctness of the diagnosis results was attested to be greater than **95 %**, where correctness meant that the generated set of possible faults contained the actual one (provided the fault was detectable at all for principled physical reasons) and did not contain faults inconsistent with the signals. The 5% incorrect diagnoses revealed that they were due to limitations of the behavior models (rather than indicating defects in the diagnosis algorithm).

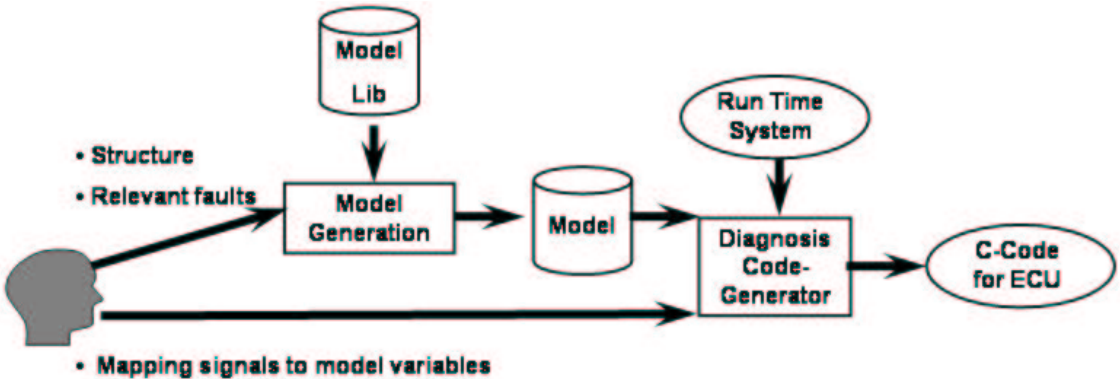


**Figure 3** Automated generation of on-board diagnostic code

This benchmark proves that a major step has been achieved:

- Model-based solutions for on-board diagnosis are small and efficient enough to achieve real-time performance on standard ECUs.
- They provide the systematic basis for automated code generation and, hence, a radical improvement in the software development process for on-board systems.

The essential use of the models is to check consistency of models with observations. If both are represented as relations, as illustrated in an abstract way in Fig. 4a, this amounts to checking whether a behavior relation has an intersection with the observations (such as "Behavior $f_1$") or not ("Behavior $f_2$"). In the following, we show that the same type of model can be re-used by applying similar relational operations and checks.

## 4. AUTOMATED GENERATION OF FMEA REPORTS

FMEA involves the process of determining the impact different component faults may have on the function of the component and of higher level systems under certain scenarios. While diagnosis is seen as the process of inferring the faults that may have caused an particular observed system behaviour, FMEA has to infer how certain faults affect the system behavior. Obviously, the knowledge required to perform both tasks concerns the behavior of components under normal and fault conditions, and we can expect to automate this analysis based on the same type of models that were used for diagnostic reasoning.

This has been realized in AUTAS, a project of some aeronautics companies, software suppliers and academic institutions, which developed system for automated FMEA generation based on qualitative models (see (Picardi et al. 2004)).

FMEA requires that the consequences of each single fault of each component under each specified scenario are determined; this corresponds to "simulating" the system in the scenario where the component under examination is assigned a fault mode and all the others are in the OK mode. This analysis has to determine whether an effect out of a set of pre-specified effects occurs. Effects correspond to certain violations of the intended function of the faulted component itself, the subsystem it belongs to ("next level effects") and the entire system ("end effects"). Technically, this is done by constraint solving which provides values for the variables that specify the effects. From these values, the engine can infer which effects are entailed by the faults and are included in the table.

For the computation, again, models of behavior modes are stated as relations (constraints) over a set of system variables. Scenarios specify certain exterior conditions and a particular state of the system and are expressed as a relation over the respective model variables ("scenario" in Figure 4b). The intersection of the model relation of a fault ("Behavior $f_1$") and the scenario relation describes the behavior of the system under this scenario. Also the relevant effects are represented as relations over a different subset of variables, namely those that can be used to characterize the function and, hence, its violation.

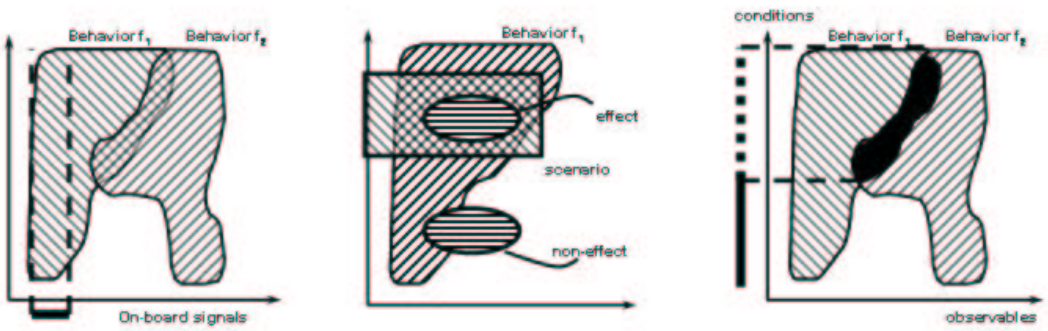


**Figure 4 a) Diagnosis computation b) Computation for FMEA c) Computation for diagnosability**

The FMEA analysis can then be stated as the problem of computing the join of the fault mode relation under a scenario and the effect relations. If this is empty, the effect is not implied by the fault under the scenario ("non-effect"). If the effect is contained the relation describing the behavior under the scenario, it is an effect. In AUTAS, this computation is done an implementation of constraints based on Ordered Multiple Decision Diagrams (OMDD).

## 5. OTHER TASKS

Model-based technology is a fundamental contribution, in particular, to the consideration of diagnostic aspects early in the design process. The IDD project has joined European OEMs and suppliers in producing a toolbox which allows one to perform model-based diagnosability and detectability analysis and to generate diagnostics (also in the form of decision trees) from the same model (Dressler-Struss 2003).

**Diagnosability analysis** solves the task of determining whether, and how, two different behavior modes (the normal behavior and a faulty behavior in detectability analysis, or two faults in discriminability analysis) can be distinguished given a certain set of observable variables and, perhaps, under which conditions (external influences or internal states) this is the case. Again, the qualitative, relational models used for diagnosis or FMEA provide the basis for automating this analysis: the above question requires to determine the conditions under which the two observable behavior relations overlap (projection to the vertical axis in Fig. 4c), which means they are, potentially, indistinguishable, while the complement of this set of conditions guarantees the discriminability of the two relations.

The **design of tests** takes a similar perspective: in end-of-line testing, one is looking for stimuli that make the distinction between the correct behavior and the faulty one observable, while in diagnostic testing, the aim is to enforce observations that discriminate between two different faults that may be present. This requires the same kind of calculations involved in diagnosability analysis, that the "conditions" must be controllable by the test equipment or a human. We are currently applying this approach, which we originally developed for testing physical systems to test generation for Electronic Control Unit software on vehicles (see (Esser-Struss 2006)).

## 6. SUMMARY

The work presented here provides evidence that this technology has grown mature and is ready for making a difference to current practice. The techniques effectively automate the entire production process of on-board diagnostics once a model library is provided. This becomes the focus of the next steps required for the actual industrial exploitation of this technology.

Since one cannot underestimate this non-trivial activity, the question arises whether the expected pay-off for on-board diagnosis justifies such an effort. As we emphasized above, re-use is not only a guiding principle concerning the diagnosis algorithm; it also applies to the models. A model library can be understood as an important repository of corporate knowledge about the technologies applied. In this form, the knowledge becomes explicitly and formally represented and available to both human experts and software programs for different work processes independently of persons, time, and location. Models turn into an important basis for a horizontal integration of these work processes easing the exchange of knowledge and making it coherent.

## REFERENCES


Bidian, P., Tatar, M., Cascio, F., Theseider-Dupre D., Sachenbacher, M., Weber, R., Carlen, C. (1999):. Powertrain Diagnostics: A Model-Based Approach. In: *Proceedings of ERA Technology Vehicle Electronic Systems Conference '99*, Coventry, UK.

R. Brignolo, F. Cascio, L. Console, P. Dague, P. Dubois, O. Dressler, D. Millet, B. Rehfus, and P. Struss (2001). Integration of Design and Diagnosis Into a Common Process. In: *Electronic Systems for vehicles* pages 53-73. VDI Verlag, Duesseldorf.

Dressler, O. , Struss, P (1996) The Consistency-based Approach to Automated Diagnosis of Devices. In*: Brewka, G. (ed.), Principles of Knowledge Representation*, CSLI Publications, Stanford, , p. 267-311.

Dressler, O., Struss P (2003). A Toolbox Integrating Model-based Diagnosability Analysis and Automated Generation of Diagnostics. In: *International Workshop on Principles of Diagnosis DX'03*, Washington, USA.

M. Esser, P. Struss (2006) Model-based Test Generation for Embedded Software. In: *17th International Workshop on Principles of Diagnosis*, Burgos, Spain..

M. Harms, T. Ehlers, J.-U. Varchmin, A. Breuer. (2003). Diagnose im strukturierten Entwicklungsprozess: STEP-X In: *Elektronik im Kraftfahrzeug*, Haus der Technik.

C. Picardi, L. Console, F. Berger, J. Breeman, T. Kanakis, J. Moelands, S. Collas, E. Arbaretier, N. De Domenico, E. Girardelli, O. Dressler, P. Struss, B. Zilbermann (2004). AUTAS: a tool for supporting FMECA generation in aeronautic systems. In: *Proceeding of the 16th European Conference on Artificial Intelligence*, Valencia, Spain, pp. 750-754

Sachenbacher, M.; Struss, P; Weber, R. (2000). Advances in Design and Implementation of OBD Functions for Diesel Injection Systems based on a Qualitative Approach to Diagnosis, In: *SAE 2000 World Congress*, Detroit, USA.

Struss, P., Price, C (2004). Model-based Systems in the Automotive Industry. In: *Artificial Intelligence Magazine*, Winter 2004