Appeared in: Proceedings of the 14th International Workshop on Principles of Diagnosis (DX03) USA, June 2003.

A TOOLBOX INTEGRATING MODEL-BASED DIAGNOSABILITY ANALYSIS AND AUTOMATED GENERATION OF DIAGNOSTICS

Oskar Dressler[1], Peter Struss[1,2]

[1]OCC'M Software, Gleissentalstr. 22, D-82041 Deisenhofen, Germany
[2]Technical University of Munich, Boltzmannstr. 3, D-85748 Garching, Germany
{dressler,struss}@occm.de, struss@in.tum.de

Abstract**:** Computer tools that allow one to improve design for diagnosability and to generate diagnostics as early as possible in the design process gain importance along with the growing complexity of devices. Model-based systems provide a basis for addressing these challenges. This paper presents theoretical foundations, implementation, and first application results of a toolbox that supports the automated analysis of detectability and discriminability of faults, the selection of sensors based on this analysis and the generation of diagnostics.

Keywords: Model-based Systems, Qualitative Modeling, On-board Diagnosis, Diagnosability Analysis

1 INTRODUCTION

Traditionally, design is understood as the task of creating a conceptual description of a device (e.g. in terms of interacting components) such that its physical realization will perform a specified behavior. For devices with embedded systems, current practice usually focuses on designing a control regime (including hardware and software) that establishes a certain function – under the assumption that the physical system **operates in a normal way**. The analysis of the **effect of potential faults** of its components (called failure-modes-and-effects analysis, FMEA) is often only a subsequent step, carried out separate from control design, and tends to cause additional design loops and, hence, higher costs and longer time to market. The same holds for considerations of how faults can be detected and identified by an embedded system (on-board diagnosis) or in the workshop (off-board diagnosis). The negligence of design for diagnosis can lead to suboptimal fault detection and identification on-board and higher maintenance costs in the workshop.

While this is more or less true for many areas, the car industry provides a particular example for the challenges. They face legal demands (e.g. w.r.t. emission reduction) and a strong competition for customer satisfaction; vehicles are becoming extremely complex (containing dozens of interacting control units), and it is impossible to guarantee an up-to-date education of maintenance staff throughout the world. This is why car manufacturers and suppliers exhibit a strong interest in new technologies that promise effective computer support to address the above issues. The objective of the IDD project[1] was to develop a model of an integrated work process for control design, fault analysis, and diagnosis and model-based computer tools that support this integration and enable the consideration of diagnostics in early design phases. In particular, a collection of tools has been developed integrated, and applied that use a system model to generate information about the diagnosability of a design (and, especially, of a chosen set of sensors) and also diagnostics in special formats.

This paper describes some elements of the IDD toolbox and illustrates their use.

The following section provides a short overview of the toolbox and its implementation basis and introduces an illustrative example that will be used throughout the paper. Section 3 describes model compression, i.e. a preprocessing of a compositional system model aiming at a parsimonious representation of the model for on-board diagnosis. As a focus of the paper, the use of this model for discriminability and detectability analysis is presented. Diagnostics can be generated as decision trees and diagnostic C-code (sections 6 and 7). Finally, we list the most important of the many open issues that need to be addressed in the future.

[1] IDD: Integrated Design Process for Onboard Diagnosis, funded by the Commission of the European Union (Project no. G3RD-CT199-00058), 2000-2003, Brignolo et al. (2001)

2 THE TOOLBOX - OVERVIEW

The toolbox is intended for model designers that need to

- evaluate their model prototypes w.r.t. diagnosability during development and
- finally produce diagnostics.

It takes as input qualitative models in a form defined by an xml schema that contains modeling concepts such as components, their behavior, and their interactions. Mathematically, qualitative models consist of relations over variables with finite domains. If V_i is the set of variables local to a component C_i with a domain $DOM(V_i)$, each possible behavior mode $mode_{ij}$ of C_i has a relation

$$R_{ij} \subseteq DOM(V_i)$$

as a **behavior model** associated with it. By adding a variable that represents the mode of C_i and joining these extended behavior model relations, we construct a relation $R_i \subseteq \{mode_{ij}\} \times DOM(V_i)$ that represents the behavior of C_i under all possible modes:

$$R_i = \{mode_{i1}\} \times R_{ij} \bowtie \{mode_{i2}\} \times R_{i2} \bowtie \dots \bowtie \{mode_{in}\} \times R_{in},$$

where $\bowtie$ is the join over shared variables.

The source of these models is irrelevant as long as the models obey the xml schema and further semantic constraints checked by the software. During the project handcrafted qualitative models exported from the Raz'r development environment (see OCC'M (2003)), as well as automatically derived models obtained through abstraction from numerical models developed in MatLab/Simulink were used.

Users of the toolbox can thus evaluate qualitative models from various sources w.r.t to discriminability and detectability in an interactive fashion. In order to facilitate model development, diagnosis sessions can be run with partially completed models on supplied sample data.

Finally, diagnostics can be produced in various forms.

Figure 1 shows the essential modules. The information flows consist of xml documents passed around between the modules. The model compiler produces a system description from the input model fragments. By running diagnosis sessions on these system descriptions partially completed models can be checked for the intended behavior. Prediction with the diagnosis engine is purposely limited to Boolean constraint propagation (BCP). Complete inferences can be achieved by offline model compressing (see section 3). Syntactically and semantically, the output of the model compressor is again a system description, but instead of many qualitative relations describing component behavior under certain modes it consists of just one qualitative relation. The compressed model can be used as an input to the diagnosis engine. Also, the diagnosability analysis engine and the diagnostics generator directly proceed from such a compressed model.

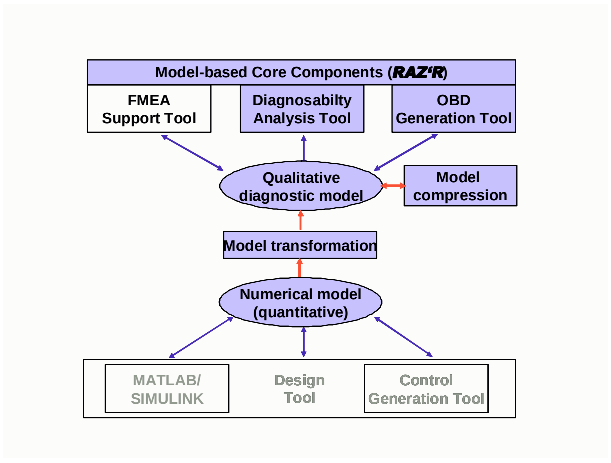


Figure 1 Components of the IDD Toolbox

The latter modules are based on computation with full qualitative relations. Thus, rather than looking at a specific situation, all situations are represented and manipulated simultaneously. Of course, this is only feasible by representing the occurring relations symbolically. It is done by employing a variant of OBDDs (ordered binary decision diagrams) (Bryant (1992)). OBDDs are directed, acyclic graphs, where at each step downward from a top level node a decision about a variable's value is represented. We use a generalized version with varying, but finite domains (OMDDs). Model compressor, diagnosability analysis engine, and diagnostics generator are grounded on operators directly manipulating these dedicated data structures. During the intermediate stages of compressing of some of the more advanced examples we successfully handled relations with trillions of tuples and beyond.

2.1 An Illustrative Example

In the following, we will use an easy to understand example (Rehfus (2002) for illustration (Figure 2).

It contains a battery (*Source*) and a general switch (*Switch*) that controls the supply of several units, e.g. actuator circuits (two in our example). Each of these parallel branches contains a consumer (*Cons1,2*) and a high-side switch (*HSS1,2*) for controlling its power supply. They share the connection to ground (*Kl1*). The command for the high-side switches is called *hssinput* and is considered Boolean (T meaning "closed" and F "open"). Quite commonly, they provide a self-diagnosis bit, which we call *EC* and which is set to T if and only if an open circuit is detected on the consumer's side. These diagnosis bits can be interpreted by the electronic control unit (ECU) and together with *hssinput*, the actuator signals issued by the ECU, form the observables of the system. As the only faults, we consider open circuits inside the consumers and at the connection to ground, and the task is to analyze whether

and how these faults can be detected and identified given the four observable values.
A model of this system is obtained by aggregating the behavior models of the component models and, hence, contains a number of variables that cannot provide relevant information for on-board diagnosis because they are not directly observable by the ECU, such as all voltage and current variables.

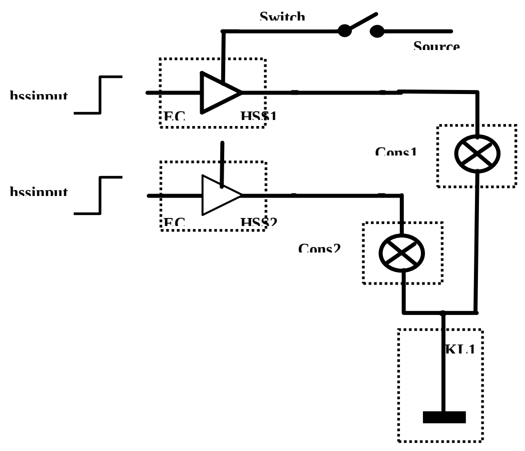


Figure 2 Switch circuit example

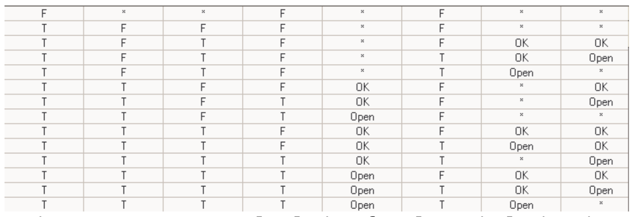
3 MODEL COMPRESSION

The compositional models typically contain several hundred variables and even more qualitative relations. Almost all of the variables, however, are not available for inspection at diagnosis time. Especially, in the on-board situation observable variables are very scarce. Situations with one or two observables and hundreds of non-observables are very common. In applying a compilation step (see e.g Darwiche and Marquis (2002)), we can drop the non-observable variables in a step called model compression.
By pair wise JOINing relations and projecting out non-observable and non-mode variables that are not used by any other relation, the whole network of relations can be compressed into one relation over the set of **relevant variables** V_{rel}, which comprise all observables and mode variables:

$$\text{MODEL}_{\text{compressed}} = \text{Proj}_{\text{rel}} (R_1 \rhd\lhd R_2 \rhd\lhd \dots \rhd\lhd R_n)$$

where the R_i are the relations describing component behavior.
The most important property of the compressed model is that all irrelevant variables have been eliminated. For instance, for on-board diagnosis, we can eliminate all unobservable variables. Running BCP on this model with the remaining observable variables supplied with values is equivalent to using a **complete inference** procedure. In this way, the cost of inference at diagnosis **run time** becomes **linear**..
Figure 3 shows the relation that represents the compressed model for the above example with

V_{rel} == modes. $\cup\{$*Switch.Pos, HSS1.hssinput, HSS1.EC, HSS2.hssinput, HSS2.EC*$\}$



Switch.Pos	*HSS1.hssinput*	*HSS2.hssinput*	*HSS1.EC*	*Cons1*	*HSS2.EC*	*Cons2*	*KL1*
F	*	*	F	*	F	*	*
T	F	F	F	*	F	*	*
T	F	T	F	*	F	OK	OK
T	F	T	F	*	T	OK	Open
T	F	T	F	*	T	Open	*
T	T	F	F	OK	F	*	OK
T	T	F	T	OK	F	*	Open
T	T	F	T	Open	F	*	*
T	T	T	F	OK	F	OK	OK
T	T	T	F	OK	T	Open	OK
T	T	T	T	OK	T	*	Open
T	T	T	T	Open	F	OK	OK
T	T	T	T	Open	T	OK	Open
T	T	T	T	Open	T	Open	*

Figure 3 Compressed relation for the switch circuit example, where '*' means "no restriction on the value. Values T and F for *Switch.Pos* denote closed and open, resp.

4 DISCRIMINABILITY ANALYSIS

We expect discriminability analysis to answer the question *"For a particular design and a chosen set of sensors, determine whether and under which circumstances the considered (classes of) faults considered can be distinguished from each other (based on the sensor readings)"*.
Stating the question like this, rather than as a yes-or-no question, reflects the fact that rarely two faults produce totally different behavior, i.e. disjoint behavior relations. In the switch circuit example, neither an open circuit in one consumer nor one in the connection to source will not cause the respective diagnosis flag to be set if the general switch or the high-side switch is open and, hence, produce the same observations in this case. On the one hand, there are certain situations that provide a natural way of stratifying the behavior specification and are used in design any way. Examples for such **operating conditions** are the idle mode of the engine vs. driving conditions, ranges of engine temperature, and, in the switch circuit example, the position of the main switch (ignition on or off). We represent them as relations on the set of system model variables V:

$$\text{OPC}_i \subset \text{DOM}(V) .$$

But there can be other variables that can influence whether or not two behavior modes can be distinguished from each other which are less evident. This could be the command to both high-side switches in the above example. In Struss et al. (2002)) we assumed that these variables are observable causal variables. The concept of "causal variables" implies that their values are determined independently of the model and of each other which means that the potential values of the causal variables are restricted only by the specification of the operating conditions. In practice, we found this to be too restrictive. Two different variables might be useful for characterizing discriminability properties without being independent, but related by the model. If we would choose both *hssinput* and the diagnosis flag and assume the correctness of the switch, *EC*=T would not occur if *hssinput*=F. Also, certain variables may not be observable on-board, but influence discriminability. For instance, certain faults may only have a significant effect under heavy load of the vehicle which is nor measurable directly.

For these reasons, we generalize the definitions given in Struss et al. (2002) in dropping both preconditions: we allow to select an arbitrary set $V_{char} \subset V$ of "characterizing" variables (that are selected to specify the situations for discriminability) with the respective projection, $PROJ_{char}$.

The task of discriminability analysis is then described as *"Determine the set of situations (i.e. tuples of values of the variables in V_{char}) that allow discriminating between two behavior modes necessarily, possibly or not at all."*

We provide the intuition behind the solution by constructing the solution in steps. First, we determine the observable tuples that are shared by both behavior modes:

$$COMMON\_OBS := PROJ_{obs}(MODEL_{fault1} \cap OPC_i) \cap PROJ_{obs}(MODEL_{fault2} \cap OPC_i) .$$

(Figure 4).

The situations that may generate an observation from this set are

$$PROJ_{char}((MODEL_{fault1} \cup MODEL_{fault2}) \rhd\lhd COMMON\_OBS)$$

(Figure 4). (Note that if the characteristic variables are observable, i.e. $V_{char} \subseteq V_{obs}$, then we need not join COMMON_OBS with $MODEL_{faultj}$, and we obtain the same expression as in Struss et al. (2002), replacing "o-cause" by "char").

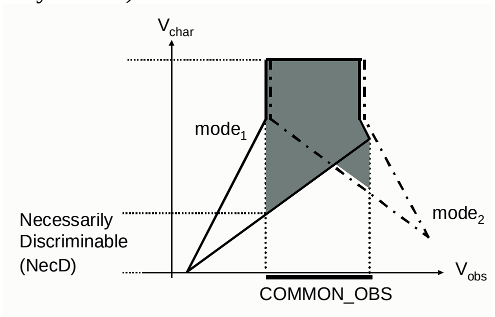


Figure 4 The set of situations that are guaranteed to produce different observations for mode$_1$ and mode$_2$

The set of situations that may produce the same observations for both modes and have to be removed from the set of all possible situations is now restricted by the operating conditions and the model which is

$$PROJ_{char}(OPC_i \cap (MODEL_{fault1} \cup MODEL_{fault2})).$$

In a similar way, we find the set of situations that do not allow any discrimination because they necessarily produce the same observations for both behavior modes, as illustrated in Figure 5. This leads to the following definition.

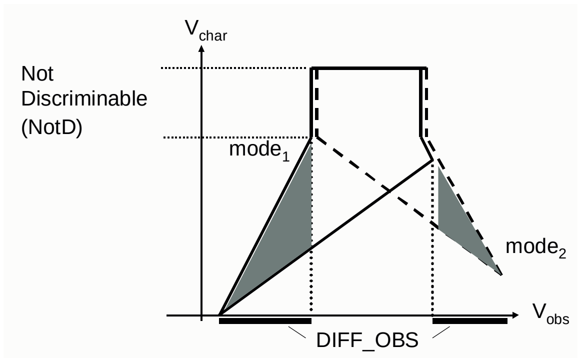


Figure 5 The set of situations that do not allow discrimination between the two behavior modes

Definition 1 (Discriminability of behavior modes)

Let $MODEL_{fault1}$, $MODEL_{fault2}$ be the behavior relations of two modes, OPC_i an operating condition, and $SIT \subset DOM(V_{char})$ a non-empty relation on the characteristic variables.

For OPC_i and SIT, two faults are called

- not discriminable, written

$NotD(fault_1, fault_2, OPC_i, SIT)$,

iff

(i) $SIT \subseteq SIT_{NotD} = PROJ_{char}(OPC_i \cap (MODEL_{fault1} \cup MODEL_{fault2})) \setminus (PROJ_{char}((MODEL_{fault1} \cup MODEL_{fault2}) \rhd\lhd DIFF\_OBS)$

where

$$DIFF\_OBS := PROJ_{obs}(MODEL_{fault1} \cap OPC_i) \setminus PROJ_{obs}(MODEL_{fault2} \cap OPC_i) \cup PROJ_{obs}(MODEL_{fault2} \cap OPC_i) \setminus PROJ_{obs}(MODEL_{fault1} \cap OPC_i)$$

- necessarily discriminable , written

$NecD(fault_1, fault_2, OPC_i, SIT)$,

iff

(ii) $SIT \subseteq SIT_{NecD} = PROJ_{char}(OPC_i \cap (MODEL_{fault1} \cup MODEL_{fault2})) \setminus (PROJ_{char}((MODEL_{fault1} \cup MODEL_{fault2}) \rhd\lhd COMMON\_OBS)$,

where

$$COMMON\_OBS : = PROJ_{obs}(MODEL_{fault1} \cap OPC_i) \cap PROJ_{obs}(MODEL_{fault2} \cap OPC_i) .$$

- possibly discriminable , written

$PotD(fault_1, fault_2, OPC_i , SIT)$,

iff

$$SIT \subseteq PROJ_{char}((MODEL_{fault1} \cup MODEL_{fault2}) \cap OPC_i) \setminus (SIT_{NotD} \cup SIT_{NecD}),$$

where SIT_{NotD} and SIT_{NecD} are the maximal relations that satisfy (i) and (ii), respectively.

For the switch circuit example, we make the following choices:

- The only interesting **operation condition** is given by ***Switch.Pos*=T**, because for an open switch, the task is meaningless.
- The **observable variables** are V_{obs} = {*HSS1.hssinput, HSS1.EC, HSS2.hssinput, HSS2.EC*}.
- The **characterizing variables** are V_{char} ={*HSS1.hssinput, HSS2.hssinput*}.

The computation is based on a compressed model that preserves, besides the mode variables, the set
{*Switch.Pos, HSS1.hssinput, HSS1.EC, HSS2.hssinput, HSS2.EC*}
as relevant variables. The result of the analysis for discriminabilty of *Open(Cons1)* vs. *Open(Kl1)* (with all other components being OK) under the specified operating condition is the following. The two faults are

- not discriminable for
  NotD = {(HSS1.hssinput=T, HSS2.hssinput=F), (HSS1.hssinput=F, HSS2.hssinput=F)},
- necessarily discriminable for
  NecD = {(HSS1.hssinput=T, HSS2.hssinput=T), (HSS1.hssinput=F, HSS2.hssinput=T)},
- never only possibly discriminable: PosD= $\varnothing$.

The result matches our intuition that the two faults can only be discriminated if the diagnosis flag of *HSS2* provides information, namely whether it indicates an open circuit, which must then be due to a failing connection to ground or not, which leaves *Open(Cons1)*. This information requires that *HSS2* receives a close command. But, at least at a second glance, this result may also be surprising, because the input to *HSS1* is not relevant for discriminabilty of the two faults, although its consumer may be faulted. If this is the case and *HSS1* is not closed, there would not be any indication of a fault! However, one has to keep in mind which question we posed: *"**If one of Open(Cons1) or Open(Kl1) is present**, under what conditions can we unambiguously confirm one of them?"*. Indeed, this possible by looking only at *HSS2.EC*, regardless of the value of *HSS1.hssinput*. This does not imply that evidence for each one of the faults is generated in the **same situation**. This information is provided by detectability analysis which is discussed in the next section.

5 DETECTABILITY ANALYSIS

Detectability analysis has to answer the question whether and under which condition the presence of a particular can be inferred, i.e. the fault can be discriminated from the mode of correct behavior. Hence, it can be regarded as a specialization of the above analysis for the case that one of two modes is the OK mode, with a certain asymmetry, in the sense that only the discrimination of the fault from the OK mode is relevant, not the converse. This is reflected by introducing the set FDIFF_OBS, as illustrated in Figure 6 and also by the fact that only the situations consistent with the fault need to be considered.

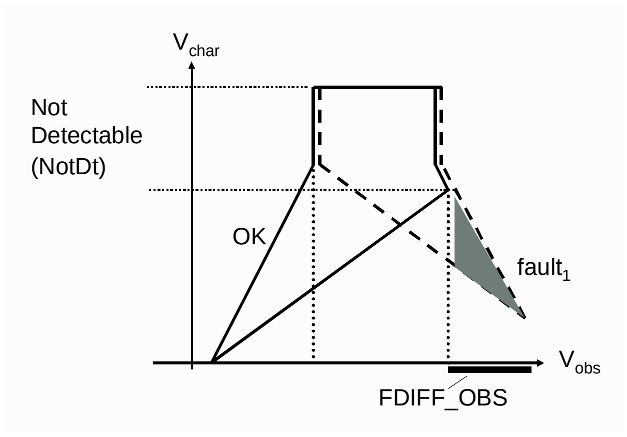


Figure 6 Computation of the situations that do not allow the detection of fault$_1$

Figure 7 illustrates the case where the fault is possibly or necessarily detectable.
Like discriminability, this can be cast into the manipulation of the behavior relation which we omit due to the lack of space.

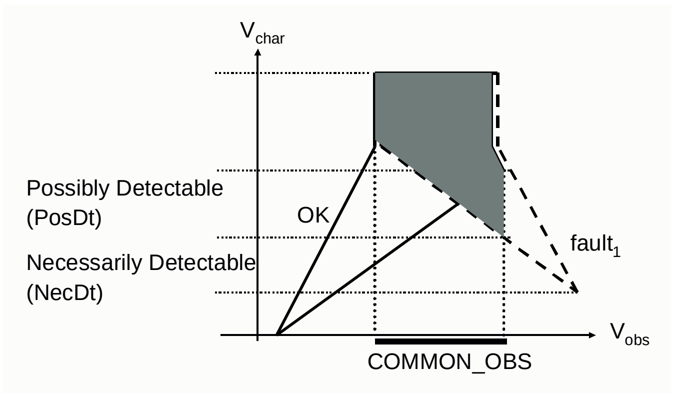


Figure 7 Situations that allow the detection of the fault

When applied to Open(Cons1), this reproduces our intuitions, namely that this fault is detectable if and only if HSS2 is closed:

- NotDetectable for situations
  {(*HSS1.hssinput*=F, *HSS2.hssinput*=T),
  (*HSS1.hssinput*=F, *HSS2.hssinput*=F)},
- NecDetectable =for situations
  {(*HSS1.hssinput*=T, *HSS2.hssinput*=T),
  (*HSS1.hssinput*=T, *HSS2.hssinput*=F)},
- PosssiblyDetectable = $\varnothing$.

Similar to the case of discriminability , the analysis of the above formulas yields a data flow diagram with re-usable parts representing common subexpressions. Some subexpressions can even be shared with the discriminability computation. The implementation is straightforward as before.

6 DECISION TREE GENERATION

The cofactor of a qualitative relation restricts the relation by fixing a variable to a specified value. When given observations for observable variables, recursively computing the cofactors with the given values yields a relation that enumerates all consistent values for the remaining variables. If we use a compressed model with all non-observable variables eliminated, the remaining variables are the mode variables. Hence, the consistent values for the remaining variables are exactly the diagnoses under the given observations. A depth-first walk over the OMDD and caching the diagnoses resulting from complete observation values suffices.

7 C-CODE GENERATION

We again use the compressed model, i.e. the model is represented as an OMDD. The intuition behind decision diagrams already makes use of the notion of a path from a top level node to two possible terminal nodes. Provided that the variable order first lists the observable variables and then the mode variables, a series of observation values describes a path to a uniquely determined node. This represents all the consistent valuations for the remaining (mode) variables, i.e. diagnoses. From the decision diagram we directly generate C-Code suited for 16-Bit Microcontrollers including the Infineon C166/7 series that for each node contains a sequence of comparisons between possible qualitative values and observed ones, and branching to the appropriate successor nodes for matching values.

8 OUTLOOK

The results obtained so far are important steps towards our long term goal of providing commercially viable model-based reasoning solutions for industrial usage. The integration of diagnosability considerations into early design phases promises to be especially fruitful since it can avoid high costs occurring further ahead in the product lifecycle. Commercially available tools are needed. The toolbox as presented is a further step in this direction. Though first promising results are now available we will need to improve several elements along the process chain.

Most importantly, the automated transformation of available, mostly numerical models from the engineering world into qualitative models suited for the described tasks must be achieved on a wide variety of models. Furthermore, a complementary qualitative modeling effort is needed for devices where no numerical models are available. This means training and consulting for engineers building qualitative models.

The IDD project has identified a new design process for car makers. But some of the steps are new to the involved people and it will be crucial to see how well our tools can support them.

The compressed model can also serve as ground for accomplishing support to FMEA provided that the variables where negative effects become visible are kept in the model. The model compression module needs improvement as stated before. It is a key element in the process chain on which all of the discussed tasks and possibly many other, future tasks depend. There is a large space of unexplored possibilities for improvement. It will be vital to exploit the particular form of qualitative models we are using, especially the component models with their interaction among a limited number of neighbors.

REFERENCES

R.Brignolo, F. Cascio, L.Console, P.Dague, P.Dubois, O.Dressler, D. Millet, B.Rehfus, P. Struss (2002). Integration of Design and Diagnosis into a Common Process. In: *Electronic Systems for Vehicles*, pp. 53-73, VDI Verlag, Duesseldorf

R. Bryant (1992). Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams ACM Computing Surveys, Vol. 24, September 1992

A. Darwiche and P. Marquis (2002). A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17:229-264

OCC'M (2003). www.occm.de

B. Rehfus (2002) Personal communication.

P.Struss, B.Rehfus, R.Brignolo, F.Cascio, L.Console, P.Dague, P.Dubois, O.Dressler, D.Millet (2002). Model-based Tools for the Integration of Design and Diagnosis into a Common Process – A Project Report, *DX-02*, Semmering, Austria, 2002

P. Struss (2002). Automated Abstraction of Numerical Simulation Models- Theory and Practical Experience, Proc. of the 16th International Workshop on Qualitative Reasoning, Sitges, Spain