Model-Based Failure Analysis with RODON

Karin Lunde[1] and **Rüdiger Lunde**[1] and **Burkhard Münker**[2]

Abstract. The model-based reasoning tool RODON supports engineers in their quest for reliable, well-designed technical products, by providing means to analyze system behavior, especially in case of failure, systematically. Based on a quantitative product model, it offers a wide range of analyses, including reliability analyses such as FMEA and FTA or the generation of diagnostic knowledge such as diagnostic decision trees. An object-oriented modeling language enhances the reusability of models and, together with an integrated design environment and extensive model libraries, considerably reduces the modeling effort. In this paper, we describe the modeling framework and the model analyses supported by RODON (considering as example the model of a comfort seat), and characterize the technology behind them. Finally, we discuss the experience gained during the development of RODON.

1 Introduction

During the life cycle of a technical product, various analysis tasks are necessary to ensure system safety and soundness of design, such as requirements analysis, design, reliability analysis, optimization and maintenance. Different teams specialized in certain phases of the development process have to work together in a highly distributed environment. The more people are involved, the more the exchange of knowledge becomes a bottleneck.

Model-based engineering aims to incorporate as much knowledge as possible in formalized, computer-interpretable models and to share those models between the different phases of the engineering process. The scope of those models is not fixed and can include very different aspects of structural and functional knowledge. To support knowledge reuse, domain specific, product specific, and task specific knowledge has to be separated. The goal of model-based engineering is to improve process efficiency as well as product quality by automatization and to reduce errors caused by misinterpretations.

Since the introduction of CAD systems in the early 80s, the exchange of formalized structural product knowledge has continuously increased. Today, CAD models originating from the design phase are reused in almost all later phases, for example to export part specifications for subcontractors like wiring harness suppliers, or to support product management systems. There are also promising examples for the reuse of behavioral models, like model-based code generation for embedded control units (MATLAB tool suite) or failure and reliability analysis (MDS [9], AutoSteve[11], or AUTAS [10], to name only a few tools).

In this paper, we present the model-based reasoning tool RODON. It provides a wide range of analyses, with a focus on failure analysis

[1] University of Applied Sciences Ulm, Germany, emails: k.lunde@hs-ulm.de, r.lunde@hs-ulm.de
[2] Sörman Information & Media AB, Heidenheim, Germany, email: burkhard.muenker@sorman.com

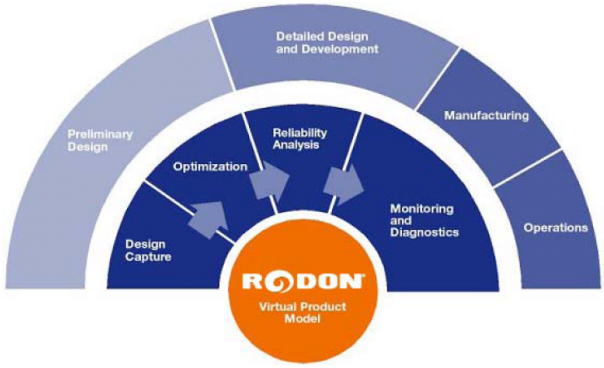

Figure 1. RODON's contribution to engineering tasks within the product life cycle

(see Figure 1). Based on a component-oriented, hierarchical model, which describes component behavior in terms of mathematical relations, RODON allows to

- simulate the behavior of the system in any state (which is part of the model);
- perform reliability analyses such as FMEA and FTA;
- compute optimal architectures for safety-critical applications;
- generate reliability block diagrams and failure trees;
- generate diagnostic knowledge such as diagnostic rules and diagnostic decision trees;
- perform interactive diagnoses with measurement proposal.

Since building and converting models between tools was identified as a major source of engineering effort, we attach great importance to the reusability of models. In the following, we describe the modeling framework and characterize shortly the algorithms which are at work behind the scenes to facilitate all those analyses. In Section 4, typical analyses are demonstrated considering as example a comfort seat model. A short discussion of lessons learned completes the paper.

2 Modeling in RODON

An object-oriented modeling language together with an integrated design environment and extensible hierarchical model libraries are the building blocks of a modeling environment which allows the engineer to create highly reusable and maintainable models with a moderate effort. In RODON, there are basically three modeling modes which differ widely in their degree of automation:

- The model topology can be imported automatically from CAD data, where component models are chosen from existing libraries.
- A model can be assembled graphically using existing model libraries (drag and drop).

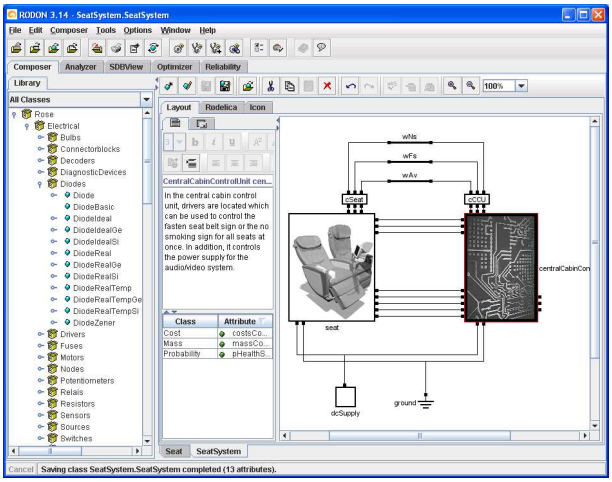

Figure 2. The model composer, at the left side the hierarchical library.

- A model can be built from scratch using the object-oriented modeling language Rodelica [3].

In practice, these modeling approaches are frequently combined: The main structure of a model may be imported from a CAD tool, while some parts can be built re-using parameterizable library components, and company-specific parts have to be modeled by the engineer himself. The degree of possible automatization depends on how accurately the model structure matches the structure of the artifact. All models are organized within hierarchical libraries which can be extended continually by adding new models.

Object-oriented modeling Modeling in RODON is component-oriented, which means that the structure of the model reflects the structure of the artifact itself rather than the signal flow. In particular, this is essential for model-based diagnosis, where the engineer is interested in identifying the component whose failure caused the observed faulty system behavior. More important, a failure of one component can change the signal flow significantly, which restricts the use of a signal-flow-oriented model to certain behavior modes.

It is only a small step from a component-oriented to an object-oriented modeling paradigm. All essential properties of a component type are subsumed in a model class. It describes the interface of the component to other components, the substructure or the quantities which characterize its behavior, and their interactions in terms of constraints.

The concept of a model class is very general. Model classes represent components as well as connectors or physical quantities. They consist of attributes, behavior sections containing the constraints, and `extends` statements which declare base classes whose attributes and behavior are inherited. An attribute is defined by a name, which must be unique within the class, and a type, which is a model class (or an array of those). As primitives, Rodelica provides a set of built-in classes, e.g. `Real` or `Interval`. Besides, attributes and `extends` statements can contain parametrizations. Connections between the connectors of components are expressed by `connect` statements within the behavior section and translated into constraints at instantiation time.

The concept of inheritance greatly enhances the reusability and maintainability of model libraries: General knowledge can be encapsulated in more abstract base classes which are likely to be reused frequently. All changes which are made to a base class propagate automatically to all classes inheriting from it.

Uncertainty and constraints To solve diagnostic problems, explicit representation of uncertainty is needed. It may sometimes be hard even to predict a system's nominal behavior exactly. To scope all possible kinds of faulty behavior is in general infeasible. For some components, the number of all possible fault behaviors is infinite (consider e.g. a leak with arbitrary diameter in a pipe). And even if the number is finite, the number may be too high to be managed efficiently (e.g. shorted pins in a connector block).

Constraint techniques provide a convenient framework to represent this kind of behavioral knowledge. Arbitrary dependencies between values of different variables can be expressed by constraints which represent relations corresponding to the underlying physical laws. The representation is declarative. It abstracts from details how to perform simulations. The modeler does not need to direct relations by transforming equations into assignments or arrange sets of equations in a special order. It just suffices to formalize the relationships between the relevant variables. This abstraction makes it possible to arrange knowledge in a context-free component-oriented way. Especially the fact that constraint networks support reasoning about sets of possible values rather than exact values makes them well suited for diagnostic reasoning.

Constraint and data types The built-in classes can be understood as the data types provided by the modeling language. In RODON, the most important data types represent sets of integer numbers (built-in class `Discrete`), sets of boolean values (`Boolean`) or interval sets (`Interval`). Besides, the usual data types `Integer`, `Real` and `String` are available, as well as arrays and records. Several kinds of constraints are supported, providing the means to combine the quantitative and qualitative modeling approaches as appropriate:

- algebraic (in-)equalities,
- boolean relations (as formulas or truth tables),
- qualitative relations (tables),
- alternative behavior: conditional constraints and disjunctions,
- spline interpolation (for measured data or characteristics).

In some applications, it is adequate to use a directed modeling approach. For this reason, RODON supports assignments and algorithmic sections in models where values can be computed using elements known from programming languages, e.g. loops. Thus, the modeler can choose the modeling style and the level of abstraction which are particularly suitable for the task at hand with great flexibility.

An integrated design environment RODON's COMPOSER module supports the user in assembling, editing and debugging models. To the left, the library of the current project is shown. The user can choose between the compact tree view or a palette view, where the icons of selected library packages are listed. To the right, one or several class editors may be active (see Figure 2).

Each class editor features three main panels: the layout pane, the Rodelica pane, and the icon pane. In the layout pane, a model can be assembled graphically, by dragging classes from the library and

[3] Rodelica is a dialect of Modelica, which is a standardized object-oriented language to describe physical systems in a component-oriented and declarative way (see `www.modelica.org`). It differs mainly in the behavior representation, by using constraints rather than differential equations. This deviation from Modelica is due to the requirements of model-based diagnosis, where model behavior is often underdetermined (see [4]).

dropping them in the desired location on the canvas (thereby creating an attribute of the corresponding type), renaming and connecting them. A dialog assists the user in the parametrization, by showing the available parameters of the selected attribute and their admissible values. Besides, documentation texts can be assigned to the class as a whole and to each attribute. From the layout, a Rodelica text is generated, which can be viewed and edited after switching to the Rodelica pane. The Rodelica text and the layout view are synchronized — changes made in the Rodelica pane are visible after switching back to the layout pane. Finally, in the icon pane the user edits the external view at attributes which use the current class as their type, by assigning an icon and arranging connectors and labels around it. At any time, it can be checked whether the recent changes are syntactically correct and compatible with the project library.

3 Behind the scenes: The reasoning framework

The approach to model-based engineering followed by RODON is strongly inspired by the general diagnostic engine (GDE) [1] and its extensions for handling different behavioral modes [2] [13] and state transitions [14]. It is based on a reasoning framework which incorporates concepts from the classical approach to model-based diagnosis, such as component-oriented modeling, dependency tracking and conflict directed search for candidates. Reasoning is divided into two layers.

The first layer is the prediction layer. In this layer inferences are drawn to compute logical consequences based on a behavioral model and information about relevant system states (e.g. assumed fault mode assignments, symptoms, measurement results or top events). We describe technical systems as discrete-time systems. Constraints represent relations between different variables at the same point in time, and difference equations the evolution of state variable values from the current to the next point in time. During behavior prediction, two main analysis steps are performed in a loop.

Given value assignments of at least the dynamic state variables, the intra-state analysis tries to determine the values of all variables at a certain point in time. This step is an iterative domain reduction step. Based on the specified relations between the variables, the constraint solver successively prunes out those regions of the variable domains which do not occur in a solution.

The results of the intra-state analysis are fed into the inter-state analysis, which checks by evaluating the difference equations whether another state should be added to the current sequence, which value sets to assign to which state variable, and which time increment to use. By allowing both continuous and discrete time variables as state variables, events as well as continuous dynamic behavior can be simulated.

The second reasoning layer is the explanation layer. The methods applied here aim to find input parameter settings for the model such that it behaves consistently to some given external observations. In diagnosis, this includes at least assignment of a value to each fault mode variable. Additionally, a search in the possible space of other unknown inputs like switch states or hidden memory states can be included. Potential explanations are called *candidates*. The search for the most plausible candidates is determined by

- the definition of the candidate space, which is given by a set of variables and their possible values,
- the conflict information, which is obtained in the prediction layer by tracking the dependencies between the input parameters and their consequences during intra-state analysis, and additionally by back-mapping [14] if a sequence of more than one state is analyzed, and
- some candidate orders defining minimality and plausibility.

Over the past years, this reasoning framework has been refined and extended in order to increase the reasoning efficiency without restricting the flexibility and expressiveness of the modeling paradigm. Most important are the following two modifications.

Inference completeness was improved by combining local consistency techniques with interval arithmetics, domain splitting and network decomposition (see [6]). The constraint solver extends the branch&prune algorithm (known from continuous constraint satisfaction problem solving, see [3]) by a network decomposition step and an interface to a reason maintenance system.

The reasoning efficiency was greatly improved by replacing the ATMS by a light-weight dependency tracking tool which we call the value manager (see [7]). In contrast to classical reason maintenance systems, the value manager actively controls resource consumption by applying data reduction techniques. Additionally, it incorporates strong focusing and data buffering.

A detailed account about the technology behind RODON can be found in [8].

4 Applications

4.1 Failure analysis of a comfort seat

In the following, we demonstrate several analyses provided by RODON by means of a simplified model of a comfort seat, like those in the first class of an aircraft. The seat model comprises an overhead unit featuring a reading light and visual signals, an audio/video unit, and three actuators to move the seat back horizontally and the foot rest both vertically and horizontally. Those elements are controlled by the operation unit which is situated in the arm rest. The seat control unit (SCU) contains drivers which process the incoming electric signals and drive the corresponding actuators or transmit messages via a bus to the central cabin control unit (CCU), as needed. The diagnostic trouble codes set by those drivers can be used by the service to locate faults.

In practice, the CCU controls all seats in the aircraft. From here, signals can be transmitted to all seats at once, e. g. to switch off the audio/video systems in case of a power drop. Each seat is connected to the CCU via a bus (modeled by electrical wires and connector blocks). The scope of the current model is restricted to only one seat and the CCU. Overall, the model consists of 183 components, 2540 variables and 2361 constraints.

Model-based diagnosis Consider a test of the comfort seat which involves checks of the three actuators, in the following order: Move the seat back forwards, then move the foot rest upwards, then move the foot rest inwards, and move the seat back backwards again.

When performing the test, it is observed that the seat back moves correctly, but the foot rest does not move at all. These observations, together with the switch positions in the four states of the test, are used as additional information for the diagnosis. Figure 3 shows the diagnosis: among all single faults there is only one candidate that can explain the system behavior in all four states, namely `seat.operationUnit.FootRestControl disconnected`; the next candidate is already a double fault. The dots at the end of the diagnosis denote the possibility of further double or higher order explanations. The diagnosis can be continued

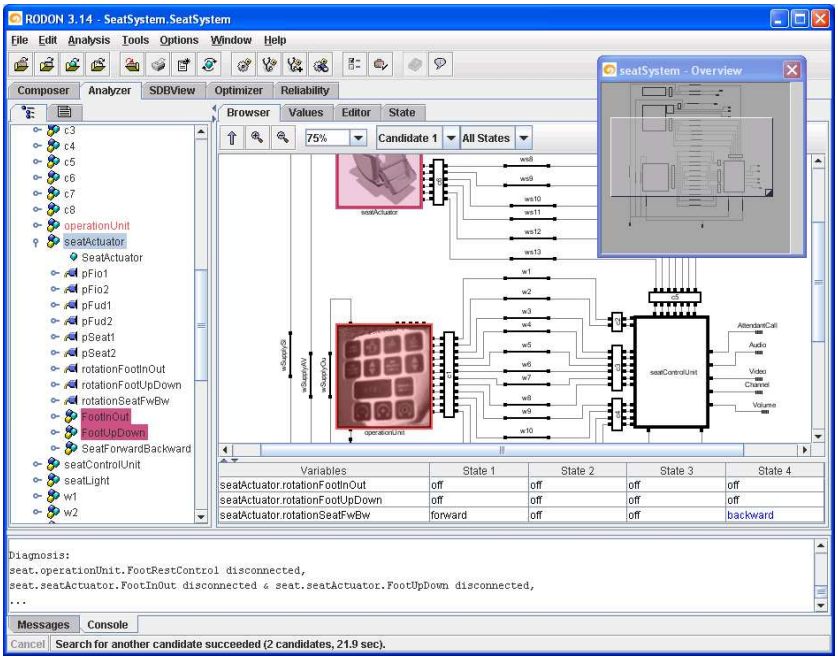

Figure 3. Model-based diagnosis of the seat system

upon request and would examine the candidates in the order of descending probability.

If the available information is insufficient to narrow down the possible explanations as far as in the last example, the tool is able to propose measurements whose results might help to eliminate candidates. Consider only the first two steps of the testing procedure above, with the same observation. Without the information about the outcome of the third test step, the model-based diagnosis finds alone 13 single faults which explain the malfunctioning foot rest. In such a situation, the interactive entropy-based measurement proposal may be used to restrict the set of candidates. The proposed measurements are listed in the order of their information gain, although the user is not bound to perform the measurements in that order.

Reliability analysis Comfort seats are not safety-critical systems, but frequent failures would annoy the passengers and are therefore to be avoided. A thorough reliability analysis in the design phase helps to reduce the probability of failures.

A common kind of reliability analysis is a failure mode and effects analysis (FMEA). RODON is able to generate the first columns of an FMEA table automatically. To this end, the user specifies which single faults and which operational states of the system are relevant for the analysis. The Cartesian product of all those operational states with the set of fault states (plus the state *System ok*) defines a so-called state space. An automatic simulation control module can then be used to perform simulations systematically for each state of the state space and to write the simulation results into a database, which we call state database (SDB). Finally, the desired table can be generated automatically, by evaluating and abstracting the SDB.

In Figure 4, a part of an automatically generated FMEA for the seat system is shown. The SDBVIEW module of RODON supports a great number of different views at the collected simulation results, among them a fault-oriented and an effects-oriented view. Customized table formats can be defined according to the requirements of the user.

4.2 Further applications

In addition to the systematic computation of an FMEA, the state data base can be used to generate other knowledge representations, like diagnostic decision trees or diagnostic rules.

Diagnostic decision trees are used in the service bay to guide the mechanic through a failure analysis with minimal effort and costs. In a recent press release [12], Volkswagen confirmed that the RODON-generated test programs provide increased service quality concerning the maintenance of vehicle electronics.

Diagnostic rules represent a compact, compiled version of the model which can be evaluated efficiently. Therefore, diagnostic rules are well-suited for onboard diagnostics, where resources are usually scarce. For instance, the rules used in the onboard system's diagnostics (SD) of the Mercedes SL- and E-classes were generated by means of RODON.

For the reliability analysis of safety-critical systems, it is usually not enough to calculate the failure probabilities with respect to single faults (compare Figure 4). In such a case, RODON's special reliability module may be of use. For models which observe certain modeling conventions, it is able to compute minimal cut sets and failure probabilities for safety-critical functions algebraically. In addition, it generates the corresponding customary representations, namely reliability block diagrams and failure trees.

5 Conclusion

About seven years ago, RODON 3 started as a pure model-based diagnosis software but evolved gradually into a life-cycle support tool. The change in scope had mainly the following reasons, derived from our experience with various industrial customer projects:

- Besides the necessity to improve the diagnostic process with respect to flexibility and reliability, customer projects showed clearly that there is a growing need for automated diagnostic knowledge generation. Due to increasing system complexity, exploding variant diversity, and shortening product development

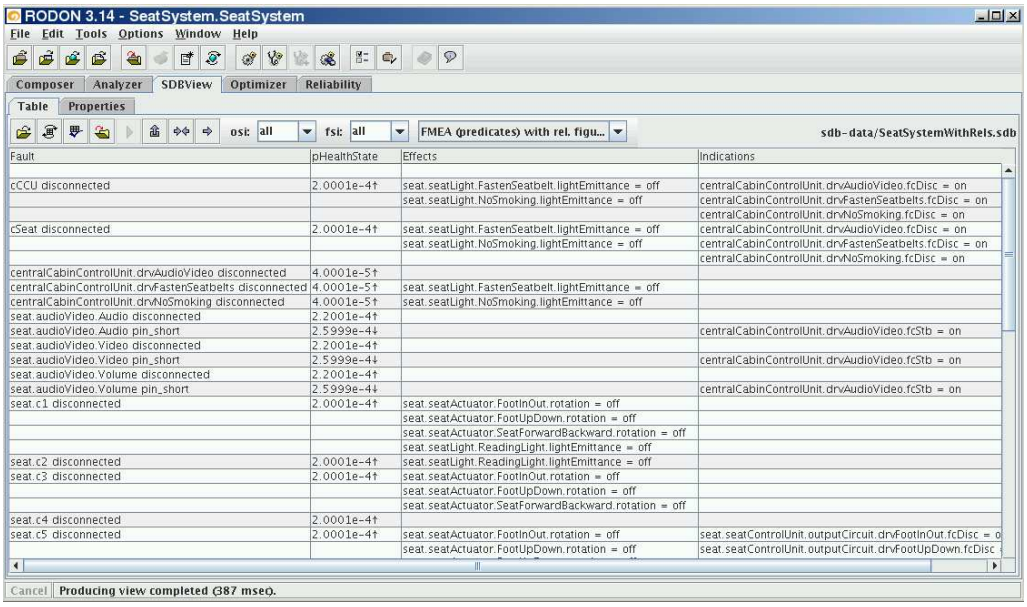

Figure 4. Part of an FMEA of the comfort seat system.
To extend the FMEA table, the model has been enriched with probabilities, for all component fault modes in the model. During SDB generation, in every state a probability `pHealthState` of the overall system is computed and written to the SDB, based upon the individual probabilities and with respect to the current fault state. As a consequence, this FMEA table provides, as an extra information (second column), a lower bound for the probability that a certain effect occurs.

cycles, diagnostic knowledge generation moves steadily from a practical experience-driven level towards a more theoretical level based on reasoning about construction data. For data processing and exchange, formal product description formats are needed, which cover structure as well as specified behavior. Component-oriented models like those used in model-based diagnosis are well suited for this purpose. To be able to serve as a vehicle for knowledge transfer, models must be generated in early stages of the product life cycle. But engineers which are involved in those early stages will spend time on modeling only if they benefit from this effort. This requires the modeling tool to provide additional functionality which helps them performing their own tasks.

- Modeling is always an investment since it consumes the expensive time of engineers. To maximize the return of invest, general reusable models should replace special purpose models in future, and be exploited wherever possible. Reliability analysis and diagnosis are related in many aspects. The needed product knowledge largely overlaps, and the level of abstraction in the knowledge representation is comparable. Additionally, common analysis algorithms can be used in both tasks. By extending the functionality of a model-based diagnostic tool to support reliability analysis, obvious synergy effects can be obtained.
- Compared to pure simulation approaches, our constraint-based approach with its ability to represent fuzziness and uncertainty explicitly gives the modeler more flexibility in his choice of abstraction level. This includes very high level abstractions, possibly qualitative ones, which are especially well-suited for very early stages of the product life cycle.

Our approach is well suited for realistic engineering applications (see also the examples in [5] and [8]). The main functional advantages in comparison to pure simulation approaches are the support of conflict-directed search strategies by dependency tracking, and the coverage of ranges of possible behaviors by set-valued reasoning about system behavior.

This additional functionality is not provided free of charge. For behavior prediction, pure simulation engines are significantly faster than the algorithms used within RODON. Although reason maintenance can compensate part of this difference, performance still remains a major issue when faced with the analysis of large systems. Further improvements seem to be possible by combining numerical with algebraic methods. Especially in component-oriented models of electric circuits, a high amount of constraints consists of simple linear equations, which can easily be simplified by algebraic transformations. Therefore, the integration of such algebraic transformations into the reasoning framework seems to be a promising direction of further investigation.

REFERENCES

[1] J. de Kleer and B. C. Williams, 'Diagnosing multiple faults', *Artificial Intelligence*, **32**, 97–130, (1987).
[2] J. de Kleer and B. C. Williams, 'Diagnosis with behavioral modes', in *Proceedings of the IJCAI'89*, pp. 1324–1330, (1989).
[3] P. Van Hentenryck, D. McAllester, and D. Kapur, 'Solving polynomial systems using a branch and prune approach', *SIAM Journal on Numerical Analysis*, **34**(2), 797–827, (April 1997).
[4] K. Lunde, 'Object-oriented modeling in model-based diagnosis', in *Proceedings of the First Modelica Workshop, Lund*, (2000).
[5] K. Lunde, 'Ensuring system safety is more efficient', *Aircraft Engineering and Aerospace Technology*, **75**(5), 477–484, (2003).
[6] R. Lunde, 'Combining domain splitting with network decomposition for application in model-based engineering.', in *19th Workshop on (Constraint) Logic Programming W(C)LP 2005*, ed., A. Wolf et al., Ulmer Informatik-Berichte, (2005).
[7] R. Lunde, 'Introducing data reduction techniques into reason maintenance', in *Proceedings of DX06*, Burgos, Spain, (2006).
[8] R. Lunde, *Towards Model-Based Engineering – A Constraint-Based Approach*, Ph.D. dissertation, Universität Hamburg (to appear), 2006.
[9] J. Mauss, V. May, and M. Tatar, 'Towards model-based engineering: Failure analysis with MDS', in *Proceedings of ECAI 2000, Berlin, Germany*, (2000).
[10] C. Picardi, L. Console, and F. Berger et al., 'Autas: A tool for supporting fmeca generation in aeronautic systems.', in *Proceedings of ECAI 2004, Valencia, Spain*, pp. 750–754, (2004).
[11] Chris Price, 'AutoSteve: Automated electrical design analysis', in *Proceedings of ECAI 2000, Berlin, Germany*, pp. 721–725, (2000).
[12] Volkswagen Media Services. Reliable Volkswagen-electronics by new diagnostics - Improved service quality in VW workshops. press release, February 2006. http://www.volkswagen-media-services.com.
[13] P. Struss and O. Dressler, 'Physical negation: Integrating fault models into the general diagnostic engine.', in *Proceedings of IJCAI 1989*, pp. 1318–1323, (1989).
[14] M. Tatar, 'Diagnosis with cascading defects', in *Proceedings of ECAI 1996, Budapest, Hungary*, pp. 511–518, (1996).