

Comparison of Diagnostics Granularity for Lighting Control Systems

M. Behrens, G. Provan & M. Boubekeur

Cork Complex Systems Lab (CCSL),

Computer Science Department,

University College Cork (UCC),

Cork, Ireland.

ABSTRACT: Intelligent buildings are complex systems consisting of embedded computation, technical devices, (inter)acting users and physical quantities. Integrating such different aspects brings together subsystems that must be represented differently. Automated problem-solving on the other side is based on models that capture the essential aspects of the behavior in a uniform way. Here we focus on diagnostics for intelligent buildings, and in particular the automated generation of diagnostics from design models for buildings. In particular, we extend previous work on diagnostics generation in two ways: (1) Defining a meta-model for a specific domain, as illustrated through the lighting control domain, allows us to fully automate the diagnostics generation process. (2) We propose a methodology that allows the system engineer to adjust and compare levels of granularity of the diagnosis according to a specific diagnosis task.

1 INTRODUCTION

1.1 Motivation

Given the importance of minimizing energy use in building applications, researchers are studying the role of many tools for such purposes. One tool, diagnostics, plays a key role in energy optimization and user comfort in buildings by identifying sub-optimal energy use caused by component faults.

Our work has addressed how to use detailed building systems models (such as energy simulation models) as the basis for automatically generating embeddable diagnostics code. In [Behrens, 2010] we showed how we can semi-automatically generate discrete propositional logic models Φ_D for diagnostics from a hybrid systems model, HS . We showed how to use a model-transformation approach to first abstract the HS model into a simpler discrete model, from which we generate propositional logic sentences.

In our work, we examine the *diagnosis task* of finding a representation of the system that is capable of isolating severe system faults while reducing false alarms. Defining such a diagnosis task includes analysis of the severity of faults. The *complexity* of a HS model is associated with the number of discrete states, the number of continuous state variables, and the type of equations defining continuous state-evolution [Mazzi, 2008]; the corresponding diagnosis model Φ_D will have lower complexity than HS , since there are only discrete states in Φ_D .

Fault-isolation using detailed models in on-board applications is resource-consuming, and models must be abstracted in order to compute diagnostics in real time. The challenge is to reduce the complexity of Φ_D such that we can still accomplish the diagnosis task.

Granularity is the extent to which abstraction breaks down a system into smaller parts. The complexity of an abstracted model is a result of the granularity of the abstraction function ξ . In general, fine granularity leads to high complexity whereas coarse granularity results in a model of lower complexity.

Finding the right granularity of abstraction for a system in order to fulfill the diagnosis task at lowest possible complexity is challenging [Darwiche, 2000]. In this paper we compare diagnosis models of different granularity, to show the types of faults that can be captured by the different granularity levels. We use a set of lighting system models as the application domain.

In summary, our contributions are:

1. We propose a meta-model for lighting systems in buildings from which diagnostics at different granularity can be automatically generated.
2. We outline how four different abstraction functions bring forth diagnostics of different granularity and complexity.
3. We compare the different reduced-order models in terms of complexity and how they fulfill the diagnosis task given.

The rest of this paper is organized as follows: We introduce our framework and the main technologies in section 2. Preliminaries that deal with model abstraction are briefly given in section 3, and the methodology of our diagnostics comparison is described in section 4. In section 5 we illustrate our achievements through a detailed example. Section 6 concludes our work and gives an outlook onto our future contributions.

1.2 Related Work

We summarize related work in two areas, model abstraction and auto-generated diagnostics.

Concepts of model abstraction have been widely discussed in the literature [Krantz, 1995], [Cousot, 2000], whereas ideas of adjusting abstraction to a specific “task” are relatively new [Sachenbacher, 2004]. The application to building automation behavioral models is novel.

Our research work concerns the auto-generation of diagnostics. The approach we propose is different from most existing approaches, which generate models from diagnosis components rather than abstract a diagnosis model from more complex behavior models. Among this class of approaches, we examine one particular paper. [Dressler, 2002] describes a methodology of diagnostics generation for car subsystems, based on the assumption that models of car systems can be automatically composed from model libraries.

2 ARCHITECTURE

2.1 Global Framework

Within the development of auto-generation of diagnostics for building automation systems, one of the main difficulties is to tap resources from which diagnostics models can be transformed. Building Information Modeling (BIM) involves integration of CAD drawings, geospatial data and other graphical and non-graphical data. It serves as a shared source of information on a building. Yet, behavioral aspects are rarely modeled and the Industry Foundation Classes (IFC), the standard interchange format for building models, support for smart objects is limited [Halfawy, 2002].

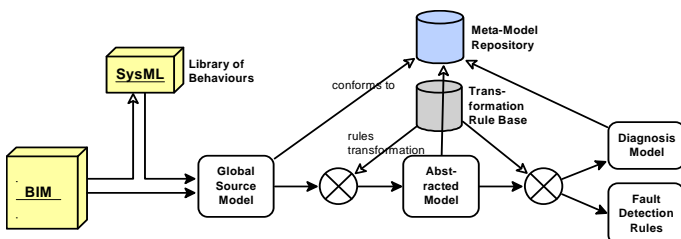


Figure 1. Global framework.

We assume that a given BIM application provides a model that describes detailed information about components of control systems, i.e., types of sensors, actuators and control strategies and component interconnectivity. Further, we assume a library that provides us with behavioural models associated with the components. Based on these assumptions, we have built our framework, which uses a global source model to combine the spatial data and interconnectivity extracted from the BIM application with behaviours extracted from the library of behaviour models. Fig. 1 illustrates the global architecture of the framework: This global source model is transformed, using appropriate meta-models and transformation rules, into a model for model-based diagnostics.

2.2 Model-based Diagnostics

Fault diagnosis is the process of analyzing abnormal system behavior in order to identify and localize the components that are the root cause of a failure. Compared to rule-based diagnostics, which associates sets of if-then rules to certain malfunctions, model-based diagnostics (MBD) uses forward models of systems and then applies diagnostics algorithms on these to decide whether a certain behavior is normal or abnormal [Darwiche].

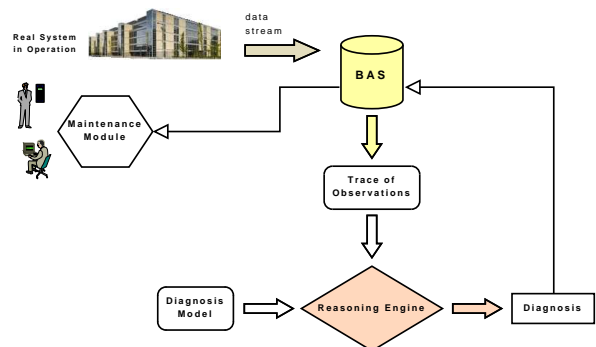


Figure 2. Model-based diagnostics in building automation.

Fig. 2 illustrates how diagnostics is embedded in our framework for fault detection and diagnosis of building automation. The reasoning engine compares the observed values with the predicted state of the system. In case of a conflict the reasoning engine provides the most possible affected component(s) as diagnosis output to be forwarded to a maintenance module.

Although MBD has the capability to isolate more diagnoses than rule-based approaches, it faces the challenge of defining the appropriate MBD models. Typically, diagnostics are generated independent of the design process. In our work, we assume that we can use design models to automate the process of diagnostics, and we apply a model-transformation approach to the diagnostics, using design models to auto-generate the MBD models.

A second drawback of MBD is the complexity of its models and inference, as compared to rules. Hence, it is important to use MBD models (in embedded-systems applications) that provide a good tradeoff of fault-isolation capabilities for memory and CPU requirements.

2.3 Model Transformation

We have two applications, a source and a target application, with a corresponding meta-model for each application. We use the theory of model transformation [Mens, 2006] to formalize our transformation process in terms of a rewrite procedure. Model transformations that translate a source model into an output model can be expressed in the form of rewriting rules. According to Mens et al. [Mens, 2006], the transformation we adopt is an exogenous transformation, in that the source and target model are expressed in different languages, i.e., behaviour models represented as FSM or hybrid automata and propositional logic languages.

3 QUALITATIVE ABSTRACTION

3.1 Preliminaries

Abstraction is the process of separating essential information from details, focusing on the former while ignoring the latter. An *abstraction function* maps a model M into a model M' of lower complexity. Any abstraction entails a loss of information. Therefore, not all questions can be answered through the abstracted model; hence, in our application not all faults can be diagnosed precisely through the resulting diagnostics model. The diagnosability of the abstract model depends on the type, granularity and coordination of the abstraction method.

In this section we analyze the methods of domain abstraction and function abstraction, and evaluate the granularity of different aspects using the diagnostics of a lighting control system.

3.2 Domain Abstraction

A domain abstraction maps values of a domain $\text{dom}_1(v)$ to a domain $\text{dom}_2(v)$ that is an ordered set of intervals taken from dom_1 . Sachenbacher et al. [Sachenbacher, 2005] define an observable distinction that is the granularity of the observable variables and a target distinction that is the granularity of the solution of the diagnosis problem. They further describe the “goal of task-dependant qualitative domain abstraction is to determine maximal partitions for the variables’ domains [...] that retain all the necessary distinctions”.

Definition. (Domain Abstraction Function) Given a continuous domain $\text{dom}_c(v_i)$ for a variable v_i and an ordered set of landmarks $L(v_i) = \{l_1, l_2, \dots, l_n\}$ as-

sociated with v_i the domain abstraction function ξ_{dom} provides the abstracted discrete domain $\text{dom}_d = \xi_{dom}(\text{dom}_c, L, Y(L))$ where $Y(L)$ assigns one of the values $\{b, o, a\}$ to each landmark l_i indicating whether l_i is included in the interval below, forms an own interval $[l_i, l_i]$ or is included in the above interval.

Definition. (Granularity of Domain Abstraction) The granularity of the domain abstraction $g(\xi_{dom})$ is the number of intervals in which a domain $\text{dom}_c(v_i)$ is split by a given abstraction function ξ_{dom} .

Remark. It is obvious that $n - 1 \leq g(\xi_{dom}) \leq 2n + 1$ for $n = |L|$.

3.3 Function Abstraction

Abstraction of a function f includes rewriting the continuous input and/or output to a representation that maps discrete arguments to discrete values. We define two different types of function abstraction:

1. We represent f through discrete directions in which the value changes: increase, decrease, hold and jump-to. The output can then be modeled as a separate automaton.
2. We represent the f by tracking its value. The number of discrete states of the system must therefore be multiplied by the granularity of the abstracted output domain

We call (1) the *local function abstraction* because functions are replaced through additional states and transitions within the same component; and we call (2) the *global function abstraction* in which the functions are replaced through states and transitions in an additional component changing the global structure of the model.

Definition. (Granularity of Function Abstraction) The granularity of the function abstraction $g(\xi_{fun})$ is the number of transitions between states that were created by the abstraction function ξ_{fun} in order to represent f .

Remark. Given n possible output values for a function f , the granularity of the abstracted representation $g(\xi_{fun})$ is $n \leq g(\xi_{fun}) \leq n^2$ for both abstraction types.

3.4 Combined Abstraction

Assuming a variable v that is the state/output variable q_i of a component C_i and that is forwarded as input y_j to a component C_j , in the ideal case the landmarks $L(v) = \{l_1, l_2, \dots, l_n\}$ are the same in C_i and C_j . However, if this is not the case, abstraction functions must be composed.

We outline two methods of composition:

1. Abstracting domains $\text{dom}_c(q_i)$ and $\text{dom}_c(y_j)$ with different abstraction functions and then provide a mapping function from q_i to y_j , that is from the discrete domain $\text{dom}_d(q_i)$ to the discrete domain $\text{dom}_d(y_j)$.

- Combining the abstraction functions $\xi(q_i)$ and $\xi(y_j)$ by either adding their granularities in case of domain abstraction or multiplying their granularities in case of function abstraction.

Definition. (Granularity of Abstraction) Given an Abstraction A of a model M , the granularity of the abstraction is the pair of the granularity of the domain abstraction and the granularity of the function abstraction $g(A) = (g(\xi_{dom}), g(\xi_{fun}))$.

4 METHODOLOGY OF DIAGNOSTICS COMPARISON

A diagnosis problem arises when some symptoms are observed, that is, when the system's actual behaviour is in contradiction with the expected behaviour. We adopt definitions from [Provan, 2009] for propositional diagnosis mode, observation and diagnosis:

Definition. (Propositional Diagnosis Model) A discrete diagnosis model is specified by a tuple $\Phi_D = \{I, V, E, \Pi\}$ where $I \subset \mathfrak{N}$ is a temporal index; V is a set of discrete-values indexes by I , such that $V_f \subset V$ is the set of failure mode variables, and $V_o \subset V$ is the set of observable variables; $E \subseteq V_f \times \mathcal{L}_n$ consists of propositional equations (where \mathcal{L}_n is a propositional well-formed formula over $(V \setminus V_f)$); and Π is a discrete probability distribution over the equations and/or variables.

Definition. (Observation) An observation α is an instantiation of the set of observable variables V_o .

Definition. (Diagnosis) Given a diagnostic model $\Phi_D = \{I, V, E, \Pi\}$, diagnosis δ is an assignment to all failure mode variables V_f of the diagnostic system Φ_D such that $\Phi_D \wedge \alpha \wedge \delta \neq \perp$ for an observation α .

We further define the abstraction of a diagnosis is the pair of the abstraction of the model and the abstraction of the observation.

Definition. (System Abstraction) Assuming a System S that is represented through a model Φ_H and instantiated with an observation α , an abstraction of S is the pair $A_i = (M_i, O_i)$ where $M_i = \xi_i(\Phi_H)$ is the abstraction of the propositional diagnosis model and $O_i = \chi_i(\alpha)$ is the abstraction of an observation.

Remark. Φ_D in this definition is the abstracted model $M_k = \xi_k(\Phi_H)$ in case of highest possible granularity of the abstraction function ξ_k .

As shown in Fig. 3, different abstracted systems are constructed using different granularities. Each system is composed by the abstracted model and the corresponding observation. The diagnosability of the resulting systems is analyzed based on the granularity and the complexity of the abstraction function.

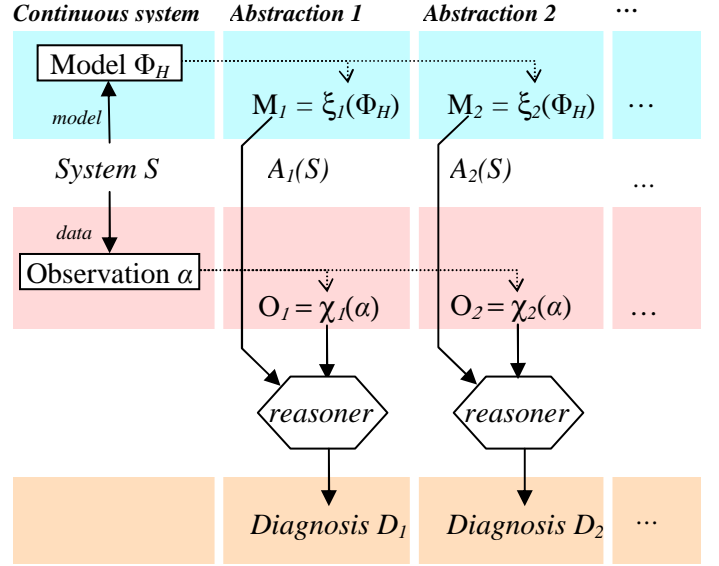


Figure 3. Methodology for Abstraction Granularity

In the following section we will compare four abstractions of a typical lighting system in order to reason about their granularity, complexity and whether the diagnosis task is fulfilled.

5 EXAMPLE

5.1 Lighting System Model

We illustrate our approach through a simple example of a system that controls a lamp with four actuation levels in a zone that is also illuminated by natural (ambient) light. A light sensor and a presence sensor provide a numeric value for the combined light intensity and a truth-value for the occupancy as input parameters to the control unit. A model of such a system is depicted in Fig. 4.

The controller has four states $Q_{Ctrl} = \{\text{off, hold, decr, incr}\}$ to set the actuation level v_{ActL} within its domain $\text{dom}(v_{ActL}) = \{0, 1, 2, 3\}$. The input variables v_{CombL} and $v_{Occupancy}$ guard the transitions between the states in such way that off is activated to set $v_{ActL}' = 0$ under the condition that $v_{Occupancy} = \text{false}$. If $v_{Occupancy} = \text{true}$ and the light-level is within an optimal range $[\alpha, \beta]$, the controller enters or remains in state *hold* and does not change v_{ActL} . If $v_{Occupancy} = \text{true}$ and the light-level is not in an optimal range, the controller can either decrease the actuation level step-wise or increase the actuation level “jump-wise”. I.e., if $v_{LightL} > \beta$ the controller enters state *decr* with the function f_{decr} : $v_{ActL}' = v_{ActL} - 1$. If $v_{CombL} < \alpha$, then v_{ActL} is increased through the function f_{incr} (1) of the state *incr*:

$$v_{ActL}' = v_{ActL} + \left\lceil \frac{\alpha - v_{LightL}}{200} \right\rceil \quad (1)$$

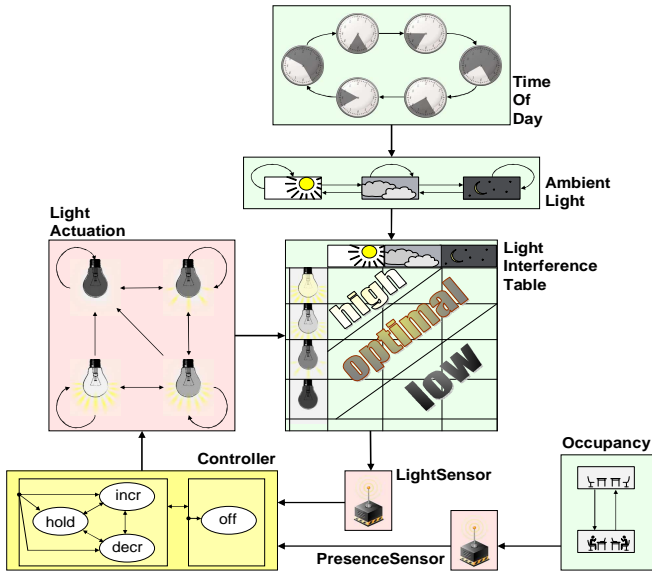


Figure 4. Typical lighting system.

Table 1 lists the combined light of the ambient and artificial light at the position where the light sensor is located. The artificial light directly depends on the actuation level that is discrete by its nature. There is no sensor to measure the intensity of the ambient light in the system; therefore it is estimated based on the time of day (and eventually astrological time).

Actuation level	Ambient light		
	dark (< 100 lx)	dawn (100 – 1000 lx)	bright (> 1000 lx)
0 (0 lx)	0 – 90 lx	50 – 900 lx	> 500 lx
1 (200 lx)	176 – 290 lx	226 – 1100 lx	> 676 lx
2 (400 lx)	352 – 490 lx	402 – 1300 lx	> 852 lx
3 (600 lx)	528 – 690 lx	578 – 1500 lx	> 1028 lx

Table 1. Interference table.

For the purpose of the diagnosability analysis we construct trace tables that contain the system observations and the corresponding unobservable system parameters. The following tables describe the traces for the nominal behavior (table 2) and behavior in case of a sensor fault (table 3) and in case of an actuation fault (table 4).

t	7:10	7:15	7:16	7:25	...
v_{AmbL}	100 lx	120 lx	120 lx	250 lx	...
v_{CombL}	83 lx	97 lx	598 lx	705 lx	...
v_{CombL_S}	85 lx	95 lx	600 lx	705 lx	...
$v_{Occupancy}$	false	true	true	true	...
$v_{Occupancy_S}$	false	true	true	true	...
q_{Ctrl}	off	incr	hold	decr	...
v_{actL}	0	3	3	2	...
v_{ArtL}	0 lx	600 lx	600 lx	400 lx	...

Table 2. Trace of the continuous model, nominal behavior.

t	7:10	7:15	7:16	7:17	7:18	...
v_{AmbL}	100 lx	120 lx	120 lx	122 lx	122 lx	...
v_{CombL}	83 lx	97 lx	598 lx	98 lx	455 lx	...
v_{CombL_S}	85 lx	95 lx	2000 lx	100 lx	460 lx	...
$v_{Occupancy}$	false	true	true	true	true	...
$v_{Occupancy_S}$	false	true	true	true	true	...
q_{Ctrl}	off	incr	decr	incr	incr	...
v_{actL}	0	3	0	2	3	...
v_{ArtL}	0 lx	600 lx	0 lx	400 lx	600 lx	...

Table 3. Trace of the continuous model, sensor fault.

t	7:10	7:15	7:16	7:17	7:18	...
v_{AmbL}	100 lx	120 lx	120 lx	122 lx	122 lx	...
v_{CombL}	83 lx	97 lx	97 lx	98 lx	98 lx	...
v_{CombL_S}	85 lx	95 lx	95 lx	100 lx	100 lx	...
$v_{Occupancy}$	false	true	true	true	true	...
$v_{Occupancy_S}$	false	true	true	true	true	...
q_{Ctrl}	off	incr	incr	incr	incr	...
v_{actL}	0	3	3	3	3	...
v_{ArtL}	0 lx	0 lx	0 lx	0 lx	0 lx	...

Table 4. Trace of the continuous model, actuator/light fault.

5.2 Abstractions

As mentioned earlier, four abstractions have been calculated mainly driven by the domain of the combined light.

- The coarse domain abstraction (A_1) uses three intervals to determine the light level based on the optimal range that drives the controller.
- The fine domain abstraction has a finer granularity where the light level is represented by 15 intervals corresponding to the interference table. The composed domain abstraction (A_2), based on merging the landmarks of both the coarse and the fine domain abstraction, represents the domain of the light level by 16 intervals.
- The function abstraction of f_{incr} (A_3) combines a local function abstraction with corresponding domain abstraction of five intervals.
- The finest possible abstraction ($A_4 = \Phi_D$) combines all the domain abstractions used above with the finer local function abstraction.

5.3 Discussions

Table 5 shows evaluations of the four abstractions of the lighting system model. Since all of the abstractions either match or extend the range of nominal behavior of the original model, no false alarms are raised, if undisturbed operation of the diagnosis framework is assured. The fault of a sensor reading being out of range (table 3) is covered by those abstractions with the finer granularity of the domain abstraction for v_{CombL} , A_2 and A_4 . A faulty actuator, i.e., a burnt-out bulb, (table 4) is detected by all ab-

stractions. However, only if we use the local abstraction function for f_{incr} , as in A_3 and A_4 , can we detect the malfunction immediately. Using global abstraction, the actuation level v_{ActL} cannot be tracked, and the detection of faulty behaviors of the actuator takes as many observations as there are actuation levels above the current one.

	A_1	A_2	A_3	$A_4 = \Phi_D$
Nominal behavior	ok	ok	ok	ok
Faulty behavior Sensor out of range	no	ok	no	ok
Faulty behavior Broken actuator	3 steps	3 steps	1 step	1 step
Granularity $g(A) = (g(\xi_{dom}), g(\xi_{fun}))$	(3,14)	(16,14)	(5,30)	(18,30)
Complexity of diagnosis	low	medium	medium	high

Table 5. Evaluation of four different abstractions.

6 CONCLUSION AND FUTURE WORK

The advantages of model-based diagnostics over state-of-the-art rule-based diagnostics increase with growing system complexity, because the number of possible fault combinations that explain an anomalous observation rises exponentially with the number of components. We showed that a model-transformation approach can generate model-based diagnostics using models defined for the design process. Through an example, we showed that different abstractions can be constructed using different granularities. We provided a methodology to construct and compare the abstracted systems where the diagnosability and the complexity of the granularity are analyzed. Using this approach, we can tailor the abstraction level to the required diagnosis-model granularity. This can help embed diagnostics code that satisfies the goal of isolating faults appropriately, while using the least amount of memory and CPU.

We plan to extend this work in a variety of ways, including: (1) extend the class of diagnosis models to incorporate explicit fault behaviors, rather than simply deviations from nominal behavior; (2) extend the class of source (reference) models to cover a wider range of control systems, sensors, actuators and physical environments found in building automation systems.

7 ACKNOWLEDGEMENT

This work was funded by SFI grant 06-SRC-I1091.

8 REFERENCES

- Behrens, M. & Provan, G. 2010. Temporal Model-Based Diagnostics Generation for HVAC Control Systems. *Proceedings of the 3rd International Conference on Theory and Practice of Model Transformation*, 2010.
- Cousot, P. 2000. Abstract Interpretation Based Formal Methods and Future Challenges. *Lecture Notes in Computer Science*, Volume 2000, 2001, Pages 138-156.
- Darwiche, A. 2000. Model-Based Diagnostics under Real-World Constraints. *AI Magazine*, Vol. 21, No 2, Summer 2000, Pages 57-73.
- Dressler, O., Struss, P.: Generating instead of programming diagnostics. *25 Jahre Elektronik-Systeme im Kraftfahrzeug. Haus der Technik Fachbuch 50*. 2005, Pages 159-169
- Fagin, R. & Kolaitis, P.G. & Popa, L. & Tan, W.-C. 2005. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transactions on Database Systems (TODS)*, Volume 30, Issue 4, December 2005, Pages 994-1055.
- Halfawy, M.R. & Froese, T. 2002. Modeling and Implementation of Smart AEC Objects: An IFC Perspective. *Proceedings of the International Council for Research and Innovation in Building and Construction*, Aarhus School of Architecture.
- Krantz, F.K. 1995. A taxonomy of model abstraction techniques. *Proceedings of the 27th conference on simulation*, 1995, Pages 1413-1420.
- Mazzi, E., Vincentelli, A.S., Balluchi, A. & Bicchi, A. 2008. Hybrid Systems Reduction. *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, Dec. 2008.
- Mens, T. & Van Gorp, P. 2006. A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, Volume 152, March 2006, Pages 125-142.
- Provan, G. 2009. Model Abstraction for Diagnosing Hybrid Systems. *Proc. Intl. Workshop on Principles of Diagnosis, DX-09*, June 2009.
- Sachenbacher, M. & Struss P. 2004. Task-dependant qualitative domain abstraction. *Artificial Intelligence*, Volume 162, Issue 1-2, February 2005, Pages 121-143.