Downloaded from orbit.dtu.dk on: Mar 20, 2019

DTU Library

SaTool - a Software Tool for Structural Analysis of Complex Automation Systems

Blanke, Mogens; Lorentzen, Torsten

Published in:
Proceedings of the 6. IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes

Link to article, DOI:
10.3182/20060829-4-CN-2909.00104

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Blanke, M., & Lorentzen, T. (2006). SaTool - a Software Tool for Structural Analysis of Complex Automation Systems. In Proceedings of the 6. IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (pp. 673-678). Elsevier. DOI: 10.3182/20060829-4-CN-2909.00104

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

SATOOL - A SOFTWARE TOOL FOR STRUCTURAL ANALYSIS OF COMPLEX AUTOMATION SYSTEMS [1,2]

Mogens Blanke * **Torsten Lorentzen** **

* *Automation at Ørsted•DTU, Technical University of Denmark, Kgs. Lyngby, Denmark and CeSOS at NTNU, Trondheim, Norway. e-mail: mb@oersted.dtu.dk*
** *Automation at Ørsted•DTU, Technical University of Denmark, Kgs. Lyngby, Denmark*

Abstract The paper introduces SaTool, a tool for structural analysis, the use of the Matlab®-based [3] implementation is presented and special features are introduced, which were motivated by industrial users. Salient features of tool are presented, including the ability to specify the behavior of a complex system at a high level of functional abstraction, analyze single and multiple fault scenarios and automatically generate parity relations for diagnosis for the system in normal and impaired conditions. User interface and algorithmic details are presented. © Copyright IFAC 2006.

Keywords: Structural analysis, Fault diagnosis, Fault-tolerant control, Computer aided control systems design.

1. INTRODUCTION

Automated production and utility plants consist of numerous interconnected subsystems. Together, they obtain an overall common objective to provide the plants' normal operation. Each of the subsystems is created to optimize the service by which they contribute to the overall objective but when local faults occur, local self protection functions typically cause local shut down. While this is rarely the ideal reaction to local faults, analysis of the needs and possibilities in large systems have been so complex and time-consuming that there has not been realistic alternatives, except in cases of safety critical applications. As the reactions of local components to faults have implications on overall availability and safety, it would be highly desirable to have a tool that could cope with analysis of complex systems without large penalties on development time and cost. Diagnosis of not-normal operation and the remedial possibilities are of particular interest.

Structural analysis is a theoretic method that aims at offering such possibility. Structural concepts were studied early in the applied mathematics community (Dulmage and Mendelsohn, 1959) and various theoretical algorithms were developed, (Dulmage and Mendelsohn, 1963) and (Hopcroft and Karp, 1973). Structural analysis has been used intensively in Chemical Engineering

[1] 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Beijing, PR China, August 30 – September 1, 2006

[2] Support is gratefully acknowledged from American Power Conversion, Denmark A/S and from the Danish Research Council, under grant number 431-294, the Agrobotics project.

[3] Matlab is a trademark of the Mathworks Inc. USA.

to deal with solving large sets of equations and issues on solvability was pursued (Unger *et al.*, 1995), (Leitold and Hangos, 2001). The structural approach and the features it offers for analyzing monitoring and diagnosis problems was first introduced in (Staroswiecki and Declerck, 1989) and further developed in (Declerck, 1991) and (Staroswiecki *et al.*, 1993). Extensions to analysis of reconfigurability and fault-tolerance emerged (Staroswiecki *et al.*, 1999), (Izadi-Zamanabadi, 1999), (Staroswiecki and Gehin, 2000). Central algorithms for matching were studied by (Dustegor, 2005) and (Krysander *et al.*, 2005). A comprehensive treatment of the structural analysis approach is presented in (Blanke *et al.*, 2006). Structural analysis has hence evolved during several decades. However, the salient features of the theory and the possibilities it offers have only become apparent to a larger community in the field of automation and automatic control over the last few years (Izadi-Zamanabadi and Staroswiecki, 2000), (Åström *et al.*, 2001) and applications reported in e.g. (Izadi-Zamanabadi *et al.*, 2003), and (Blanke, 2005). Reasons for the slow penetration into applications origin mainly in the lack of widely available tools to support the structural analysis method for automated industrial systems.

SaTool (Lorentzen and Blanke, 2004), (Lorentzen and Blanke, 2005*a*), (Lorentzen and Blanke, 2005*b*) is a tool for analysis of system structure at a high level of specification. It is a rapid prototype tool albeit with a user friendly graphical user interface and is based on Matlab®. This paper presents SaTool. The programme structure is discussed based on capture of the requirements industrial developers expressed to this type of design program.

The paper focus on the users' needs during an initial high level design phase and demonstrates a GUI that supports this phase. Ways to present the user with structural information without getting lost in details is discussed and program interface examples show the use of SaTool. An appendix shows essential data structures.

2. SYSTEM STRUCTURE AND BEHAVIORS

Structural analysis is a graph-based technique where behaviors between variables express the system's properties. Measured and controlled quantities in the system are related to variables through functional relations, referred to as constraints. Normal behavior means the set of constraints describe the relation between input and output. A structure graph of the system has variables and constraints as nodes of the graph. Edges of the graph shows which variables are used by a particular constraint, the edges connect a constraint-nod to variable-nodes. Matching of unknown variables to constraints provides a graph-theory solution that shows how the set of constraints could be solved for unknown variables. Solving for an unknown variable means to trace the matching back to known variables through matched constraints, leaving only one variable unknown in a constraint. Unmatched constraints are used to test the validity of a solution in the unknown variables. A fault in the system means a deviation from normal behavior of one or more constraints, hence a test made using an unmatched constraint will fail if any of the constraints involved in the particular redundancy relation are violated. Unmatched constraints are hence useful as parity relations that can directly be used for diagnosis.

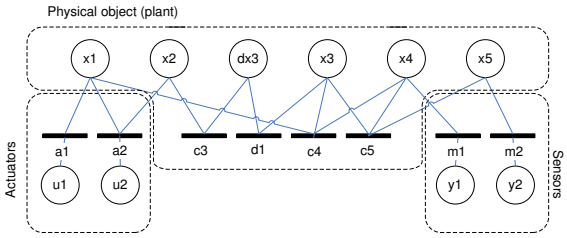

Figure 1. Bipartite graph representing structure.

As a basis for the calculations provided by SaTool, the following is a brief summary of essential definitions used in structural analysis are based on the results of original work by Staroswiecki and co-workers which was extended and brought into a unified form in (Blanke *et al.*, 2006).

Given input u and outputs y of a set of components, normal behaviors of a system is described by a set of constraints and the relations they impose on variables. The following notation is used,

$\mathcal{Z}:$	$z \in \mathcal{Z}$	all variables
$\mathcal{U}:$	$u \in \mathcal{U}, \mathcal{U} \subseteq \mathcal{Z}$	input variables
$\mathcal{Y}:$	$y \in Y, Y \subseteq \mathcal{Z}$	output variables
$\mathcal{K}:$	$k \in \mathcal{K}, \mathcal{U} \cup \mathcal{Y} \subseteq \mathcal{K} \subseteq \mathcal{Z}$	known variables
$\mathcal{X}:$	$x \in \mathcal{X} = \mathcal{Z} \setminus \mathcal{K}$	unknown variables
$\mathcal{C}:$	$c_i \in \mathcal{C}$	set of constraints
$c_i:$	$c_i(z) = 0$	behaviors

Figure 1 shows a systems structure as a structure graph. For convenience in later interpretation of results, symbols a and m have been used instead of c for constraints associated with actuators and sensors, respectively.

Definition 1. Structure graph: A structure graph of the system $(\mathcal{C}, \mathcal{Z}$ is a bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{C} \times \mathcal{Z}$ is the set of edges defined by: $z_j \in c_i \;\Rightarrow\; (c_i, z_j) \in \mathcal{E}$

Definition 2. Matching: A matching $\mathcal{M}$ is a subset of edges $\mathcal{E}$ such that with $e_1 = (c_{i1}, z_{j1}), e_2 = (c_{i2}, c_{j2})$, $\forall e_1, e_2 \in \mathcal{M} : e_1 \neq e_2 \Rightarrow c_{i1} \neq c_{i2} \wedge z_{j1} \neq z_{j2}$

Definition 3. Complete matching on X: A matching is complete on X if $|\mathcal{M}| = |\mathcal{X}|$

Definition 4. Fault: A fault is a deviation from normal behavior, $\exists i : c_i \neq 0$.

Definition 5. Parity relation: A parity relation $p_j(k_i, c_i)$ is a behavior between known variables $k_i \in K^{(p_j)} \subseteq \mathcal{K}$, $c_i \in C^{(p_j)} \subseteq \mathcal{C}$ with the property $p_j(k_i, c_i) = 0 \Leftrightarrow \forall c_i \in C^{(p_j)} : c_i = 0$

Corollary 6. Parity relation from unmatched constraint: A parity relation $p_j(k_i, c_i) = 0$ is obtained from an unmatched constraint $c_j(x_i, k_i) \rightarrow zero$ $x_i \in \mathcal{X}$ and $k_i \in \mathcal{K}$ by backtracking unknown variables x_i through constraints to which they were matched and hence arrive at a behavior between known variables in the system $p_j(k_i, c_i) = 0$.

Hence, each unmatched constraint will will give one parity relation that could be used for diagnosis since a violation of a constraint that is used in constructing parity relation would give a violation of the behavior $p_j(k_i, c_i) = 0$. If more than one complete matching was obtained, the set of constraints $c_i \in C^{(p_j)} \subseteq \mathcal{C}$ used in a particular parity relation $p_j(k_i, c_i) = 0$ clearly depend on the matching used.

A system with m constraints and n parity relations will give a relation showing which residuals depend on which constraints. One view on these relations is the boolean mapping,

$$\mathcal{F} : r \leftarrow M \otimes (c_i \neq 0) \tag{1}$$

from which structural detectability and isolability can be assessed.

Lemma 7. A violation of a constraint is structurally detectable if and only if it has a nonzero boolean signature in the residual, $c_i \in C_{\text{det}\,ectable}$ iff $\exists j : c_i \neq 0 \Rightarrow r_j \neq 0$

Lemma 8. A violation of a constraint is structurally isolable if and only if it has a unique signature in the residual vector, i.e. column m_i of M is independent of all other columns in M, $c_i \in C_{isolable} \Leftrightarrow \forall j \neq i : m_i \neq m_j$

2.1 Analysis of impaired system

When a system is impaired, one or more constraints of normal operation are violated $c_{if} \neq 0$

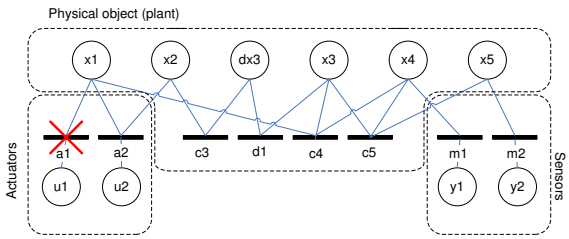

Figure 2. Impaired system with actuator failure. A constraint need be disregarded in re-analysis.

and a modified set of constraints is needed to describe the behavior of the impaired system. Let the set of constraints that describe the impaired system be $\mathcal{C}^f$ and without loss of generality, assume $\mathcal{C}^f \subset \mathcal{C}$. Analysis of the impaired system, where $c_{if} \in \mathcal{C} \backslash \mathcal{C}^f$ requires analysis of all structural properties of the system $(\mathcal{C}^f, \mathcal{Z})$. A case of and impaired system is illustrated in Figure 2.

3. ANALYSIS BASED ON STRUCTURE

The user enters a list of these relations that together describe the entire system. The list of such variables and functional relations constitute the system's structure graph. Normal operation means all functional relations are intact. Should faults occur, one or more functional relations cease to be valid. In a structure graph, this is seen as the disappearance of one or more nodes of the graph. SaTool makes analysis of the structure graph to provide knowledge about fundamental properties of the system in normal and faulty conditions.

From matching all unknown variables to constraints we have identified the over-determined subgraphs that can be used as analytical redundancy relations in the system.

When a matching has been found, the set $C^{(u)} \subset C$ of unmatched constraints $C^{(u)} = \{c_j | c_j(x_i, k_i) \rightarrow zero\}$ $x_i \in X$ and $k_i \in K$ is determined. To obtain parity relations for diagnosis, we need to substitute the unknown variables in c_j by known ones entering through matched constraints. Backtracking along paths in the matching will enable such an elimination of the unknown variables. This procedure can be described formally.

4. SATOOL IN THE DEVELOPMENT CYCLE

In an initial concept phase of development, the designer wish to consider whether overall functionality of the plant will be met, and which local failures might affect the fulfilment of the overall objectives for the plant. To work effectively with structural properties of a fault tolerant control systems design, the following features are needed:

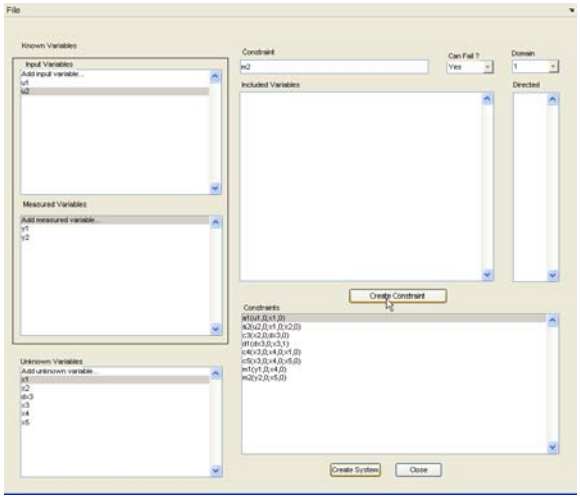

Figure 3. SaTool constraint editor window.

- work with abstract representation of functional relations;
- specify functional relations by name;
- analyze fault detection and isolation possibilities;
- analyze structural controllability;
- output parity relations in symbolic form;
- analyze cases of multiple faults;
- modify structural models interactively;
- store and reload structural models.

This set of features were implemented in SaTool (Lorentzen and Blanke, 2005*b*)

4.1 Specify and manipulate the structural model

SaTool automates the process of creating the structure model and performing the structural analysis.

Constraints represent the functional relations in the system, i.e. originating in a physical model using first principles. The constraints needed for structural analysis are far more simple. Instead of using the explicit system equations, structural analysis need to know whether a certain constraint makes use of a particular variable. Parameters that are known from the physics of the plant or from properties of the automation system, e.g. a control gain, are treated as part of the constraint in which the particular parameter is used.

There are three different kinds of variables in the program: *Input variables* are known, externally defined; *Measured variables* are entities measured in the system; *Unknown variables* are internal physical variables. Input and measured variables both belong to the set K but are separated for calculation of controllability.

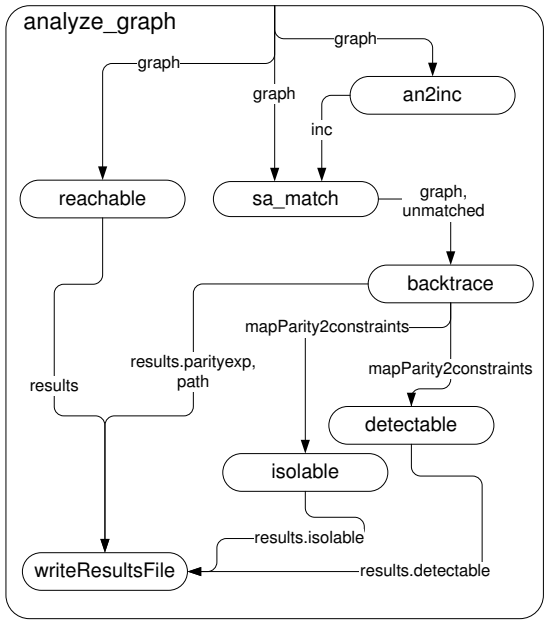

Figure 4. Software architecture for the part that analyze a graph

5. SOFTWARE ARCHITECTURE AND ALGORITHMS

SaTool has two main parts, the *constraint editor* and the *analyze graph* functions.

The *constraint editor* uses the Matlab GUI builder as a basis. Results of interactive work in the editor is stored in the structure *graph* when the user selects *create graph* on the GUI. The system files can be stored in Matlab .mat format or in a dedicated XML format. The *constraint editor* GUI is shown in Figure 3.

The *analyze graph* window in shown in Figure 5.

Figure 4 illustrates dataflow in the *analyze graph* part of the software architecture.

5.1 Matching

The central idea in the structure graph approach is to match all unknown variables using available constraints and known variables, if possible. If successful, the matching will identify overdetermined subgraphs that can be used as analytical redundancy relations in the system. Overdetermined subsystems are complete with respect to Z and incomplete with respect to C_Z. All matchings must be causal i.e. a matched variable must be calculable from the constraint by which it is matched. Non-injective functions and table look-up are causal only in one direction and can hence not be used to match all of the variables involved.

Causality is represented in the structure graph in marking it *directed*. Graphically this is represented by a dotted line in the GUI. Non directed edges are causal in both directions.

5.1.1. Matching algorithm Several algorithms exist to find complete matchings in a structure graph, see (Blanke *et al.*, 2006), (Dustegor, 2005), (Krysander *et al.*, 2005). In this context, the purpose it to provide a complete matching on X and to provide a list U of unmatched constraints. The *ranking algorithm* has complexity $O(nm)$ where n is number of constraints and m the number of unknown variables. A more computationally efficient algorithm of complexity $O((n+m)\sqrt{n})$ could be used (Hopcroft and Karp, 1973) at the expense of implementation effort. Use of either of the more advanced algorithms is possible. Finding the matching is straightforward, as SaTool generates the incidence matrix *inc*, see Figure A.1. SaTool also requires backtracking of results, using information stored in the *path* structure. Replacement of the matching algorithm is hence straightforward but will require a modification to store the path information. The

6. USING SATOOL

Constraints are generated using the constraint editor. Variables are loaded before constraints can be generated. The constraint editor with the variables loaded in the left hand side lists is seen on Figure 3. When creating a constraint, the user can mark a constraint as *can fail: (Yes, No)*. Mathematical definitions, e.g. a derivative can't fail. Physical fundamentals, e.g. velocity is the derivative of position. Further, a directed entry can be marked, e.g. in a differential constraint where position can't be obtained from integration of velocity if the initial position is not known. A 1 in the field *directed* means this variable can not be calculated from the constraint. Constraints and variables may be given arbitrary names.

Analyze the graph

When all constraints are generated a graphical representation of the structure graph is shown (Figure 5). When analyzing the graph, the results are presented graphically by coloring of the graph and a text document is generated containing the result of the analysis.

A particular important feature is analysis of impaired systems. When a fault occurs, and a fault-tolerant control scheme handles the fault, it is essential for system safety that no critical faults

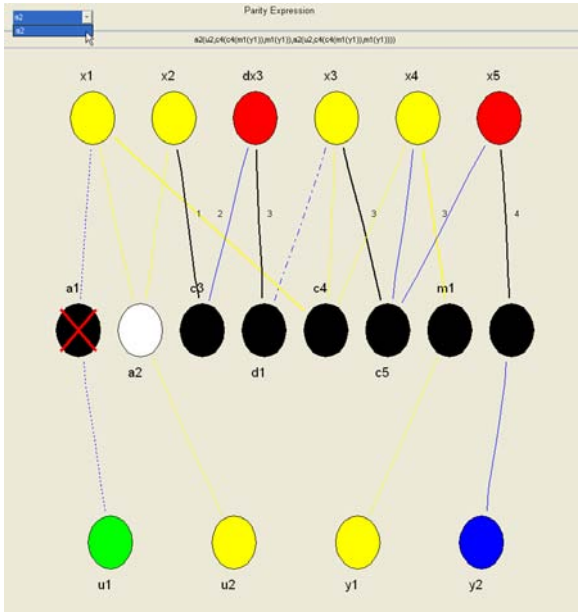

Figure 5. SaTool GUI with structure graph for impaired system - a defect in a1.

remain undetectable in the reconfigured system. Therefore, analysis of the properties of the impaired system is essential.

SaTool has a facility to disable each constraint one at a time and make a report of detectability / isolability for each of the impaired cases. If one or more constraints are already disabled by the user, from the GUI, this condition is taken as the starting point for the systematic analysis. This makes it relatively easy to investigate the cases of all relevant, multiple faults.

Results file

The analysis performed by SaTool are stored in a report file. It includes detectability, isolability, symbolic form of parity relations for the normal and impaired cases selected by the user.

7. SUMMARY

This paper has described SaTool, an implementation of structural analysis methods for computer aided control systems design. Salient features were discussed and information provided on software architecture and methods used to enable the community to contribute to the further development of tools for structural analysis.

Appendix A. ESSENTIAL DATA

System structure graph is represented in a cell array structure carrying all persistent information about the graph, including strings to represent the names of variables and constraints. The cells

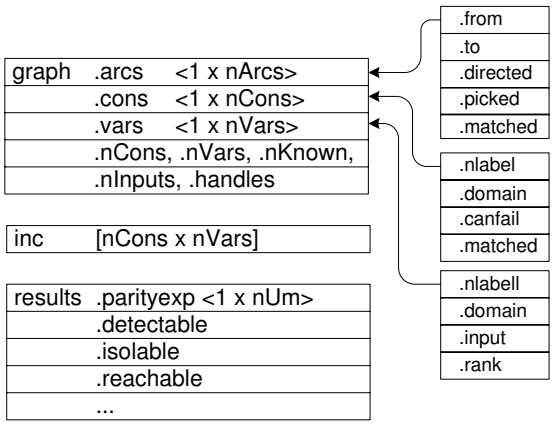

Figure A.1. Structures graph (cellarray), inc (2 dim. array) and results (cell array)

in *graph.cons* refer to constraints; *graph.vars* refer to the variables in the order: input, output, unknowns; *graph.arcs* comprise information about connections between nodes in the graph. These structures also comprise fields that holds information used for the graphical presentation. The array *inc* is a plain representation of the incidence matrix of the structure graph. The *inc* matrix is calculated every time needed since the user has access to change the contents of *graph* (i.e. the system structure) from the graphical user interface. The structure *results* comprise results for presentation, including the symbolic parity relations. Figure A.1 illustrates essentials of these structures.

REFERENCES

Åström, K. J., P. Albertos, M. Blanke, A. Isidori, W. Schaufelberger and R. Sanz (eds) (2001). *Control of Complex Systems*. Springer.

Blanke, M. (2005). Fault-tolerant sensor fusion with an application to ship navigation.. In: *Proc. IEEE MED*. pp. 1385–1390.

Blanke, M., M. Kinnaert, J. Lunze and M. Staroswiecki (2006). *Diagnosis and Fault-tolerant Control*. 2nd ed.. Springer.

Declerck, Ph. (1991). Analyse structurale et fonctionnelle des grands systèmes. Application à une centrale PWR 900 MW.. PhD thesis. Université des Science et Technologies de Lille,. Villeneuve D'Ascq, France.

Dulmage, A. L. and N. S. Mendelsohn (1959). A structure theory of bi-partite graphs.. *Trans. Royal Society of Canada. Sec. 3.* **53**, 1–13.

Dulmage, A. L. and N. S. Mendelsohn (1963). Two algorithms for bipartite graphs.. *Journal of the Society for Industrial and Applied Mathematics*. (1), 183–194.

Dustegor, D. (2005). PhD thesis. Université des Science et Technologies de Lille,.

Hopcroft, J. E. and R. M. Karp (1973). An n/sup 5/2 algorithm for maximal matchings in bipartite graphs.. *SIAM Journal on Computing* pp. 225–231.

Izadi-Zamanabadi, R. (1999). Fault-tolerant Supervisory Control - System Analysis and Logic Design. PhD thesis.

Izadi-Zamanabadi, R. and M. Staroswiecki (2000). A structural analysis method formulation for fault-tolerant control system design. In: *39th IEEE CDC*. pp. 4901–4902.

Izadi-Zamanabadi, R., M. Blanke and S. Katebi (2003). Cheap diagnosis using structural modeling and fuzzy-logic based detection. *Control Engineering Practice* **11**(4), 415–422.

Krysander, M., J. Åslund and M. Nyberg (2005). An efficient algorithm for finding over-constrained sub-systems for construction of diagnostic tests.. In: *Proc. 16th Int. Workshop on Principles of Diagnosis (DX'05)*.

Leitold, A. and K. M. Hangos (2001). Structural solvability analysis of dynamic process models. *Computers and Chemical Engineering* **25**, 1633–1646.

Lorentzen, T. and M. Blanke (2004). Industrial use of structural analysis - a rapid prototyping tool in the public domain. In: *Proc. Workshop on Advanced Control and Diagnosis*. pp. 166–171.

Lorentzen, T. and M. Blanke (2005*a*). Satool - software reference (version 1.31). Technical report. Technical University of Denmark. Ørsted-DTU, Lyngby, Denmark.

Lorentzen, T. and M. Blanke (2005*b*). Satool - users manual (version 1.31). Technical report. Techniocal University of Denmark. Ørsted-DTU, Lyngby, Denmark.

Staroswiecki, M. and A. L. Gehin (2000). Control, fault tolerant control and supervision problems. In: *IFAC Safeprocess' 2000*. Budapest, Hungary.

Staroswiecki, M. and P. Declerck (1989). Analytical redundancy in nonlinear interconnected systems by means of structural analysis. In: *Proc. IFAC AIPAC'89 Symposium.*. Vol. 2. Elsevier - IFAC. pp. 23–27.

Staroswiecki, M., J. P. Cassar and V. Cocquempot (1993). Generation of optimal structured residuals in the parity space. In: *12th IFAC World Congress*. Vol. 8. pp. 299–305.

Staroswiecki, M., S. Attouche and M. L. Assas (1999). A graphic approach for reconfigurability analysis. In: *Proc. DX'99*.

Unger, J., A. Kröner and W. Marquardt (1995). Structural analysis of differential-algebraic equation systems - theory and applications.. *Computers and Chemical Engineering* (8), 867–882.