FAULT DETECTION OF DISCRETE EVENT SYSTEMS USING AN IDENTIFICATION APPROACH

Stéphane Klein[*, **], Lothar Litz[*] and Jean-Jacques Lesage[**]

[*] *Institute of Automatic Control – University of Kaiserslautern*
P.O. Box 3049 – 67653 Kaiserslautern (Germany)
{sklein, litz}@eit.uni-kl.de

[**] *LURPA – Ecole Normale Supérieure de Cachan*
61, av. du Président Wilson – 94235 Cachan Cedex (France)
{klein, lesage}@lurpa.ens-cachan.fr

Abstract: In this paper, we focus on the identification of large-scale discrete-event systems for the purpose of fault detection. The properties of a model to be useful for fault detection are discussed. As appropriate model basis the nondeterministic autonomous automaton is chosen and metrics to evaluate the accuracy of the identified model are defined. An identification algorithm which allows setting the accuracy of the identified model is presented. Results are given for two case studies, one of a laboratory and another one of an industrial plant. *Copyright © 2005 IFAC*

Keywords: Discrete-event systems, Fault detection, Identification, Large-scale systems, Behaviour, Accuracy.

1. INTRODUCTION

Increasing the availability of machines is an important economic issue. The challenge is to reduce the downtime of the machines. This happens if failures can be avoided by detecting the faults prior to a stop. The goal is to notice that the machine is running in an abnormal mode and to perform preventive actions. If the failure is inevitable, the challenge is to reduce the time needed to find the origin of the failure.

Several methods like expert systems (Lucas and van der Gaag, 1991) or fault trees (Schneeweiß, 1984) are available to find the origin of a failure when this has occurred but the fault detection is mostly done manually, i.e. by the operators in charge of the production. This is especially the case with large scale and complex systems.

The model based approach offers a way to perform automatically fault detection on large scale system. It relies on the simple principle enounced by (Davis and Hamscher, 1988): “To determine why something has stopped working, it’s useful to know how it was supposed to work in the first place.” The idea is to have a model of the functioning of the system and to compare the evolution of the model (prediction) with the evolution of the real system (observation). If the model is supposed to be accurate and complete, a discrepancy between prediction and observation originates necessarily from a fault in the system

Our goal is to perform fault detection for large-scale industrial systems. Hence we will concentrate on the way a model of the normal evolution can be won. Two ways can be distinguished: The knowledge-based approach and the identification approach.

The knowledge based approach is a white-box method which needs a structural description of the system, i.e. the way the components are connected, and a description of the behaviour of each component. This approach is very cumbersome for large-scale complex systems but works very well for small and medium-sized systems, see (Lunze *et al.*, 2001; Sztipaovits and Misra, 1996; Sampath *et al.*, 1996).

The chosen identification based approach follows another way. Basing on previous observations of the behaviour of the system, a state-transition model of the behaviour is built. Since this approach does not require any prior knowledge on the system, it is suitable for large-scale complex systems integrating control and controlled parts as well as their interactions.

This paper is organized as follows: In section 2, the objectives of the behavioural identification as well as the chosen model are presented. In section 3, quality metrics to evaluate the identified model are defined. In section 4, the identification algorithm is developed using a simple example. In section 5, the results of the identification algorithm applied on two representative case studies are given. Finally section 6 concludes the paper.

2. BEHAVIOURAL IDENTIFICATION OF DISCRETE EVENT DYNAMIC SYSTEMS

The goal of the identification is to win an internal behavioural model of a system basing on the external observed behaviour. For DES, this behaviour is the ordered sequence of input/output signals recorded on the system. The internal model is a mathematical representation of the behaviour of the system, typically for DES automata or Petri nets.

Before going further on external and internal models, let us isolate the system to be identified. Our objective is to detect faults that occur in a running system composed of a control part – typically a programmable logic controller (PLC) – and an equipment under control running in closed loop, see figure 1. Hence the behaviour we have to identify is the behaviour of both the controller and the equipment interacting in closed loop. The observed signals are the controller input and output signals, represented by an I/O vector at a time *t* noted $u_i(t)$. Each observation corresponds to a stable state of the system. *i* is the index representing the observation sequence of a given production cycle of the system. The ordered series of $u_i(t)$ is named a data observation $\sigma_i = (u_i(1), u_i(2), \dots, u_i(|\sigma_i|))$, where $|\sigma_i|$ is the cardinality of σ_i. Note that the considered systems are cyclic, then the production cycles start and end in the same state, i.e. $\forall (i, j)$, $u_i(1) = u_j(1) = u_i(|\sigma_i|) = u_j(|\sigma_j|)$. The set of all the observations, $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_{|\Sigma|}\}$, contains the observed behaviour we have to translate into a state-transition model.

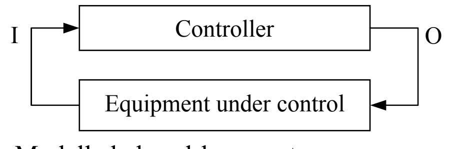

Fig. 1. Modelled closed-loop system.

In the following sections of this paper, we will distinguish three different behaviours: The original behaviour, the observed one and the identified one. The original behaviour (Beh_{Orig}) is the never known behaviour of the real system. That's why no a priori knowledge of the behaviour can be taken into account during the identification and the system to be identified is considered as a "black box". The observed behaviour (Beh_{Obs}) – represented by Σ – is an incomplete part, limited to the observed sequences, of the original behaviour. The identified behaviour (Beh_{Id}) is the behaviour of the identification model used for the fault detection.

For DES, each of these behaviours is a finite set of trajectories between a finite set of states. Figure 2 gives a graphical representation of the different behaviours.

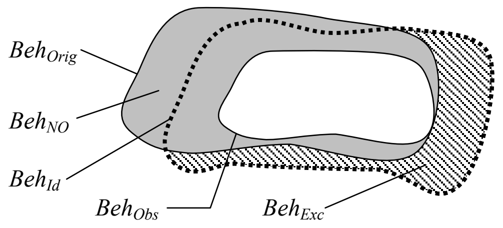

Fig. 2. Graphical representation of the different considered behaviours.

Some remarks can be formulated:

First, $Beh_{Obs} \subset Beh_{Orig}$ and $Beh_{NO} \neq \varnothing$ with $Beh_{NO} = Beh_{Orig} \setminus Beh_{Obs}$ the non-observed behaviour. Indeed, an infinite observation time, i.e. $t \to \infty$, would be required to have $Beh_{NO} \to 0$. In practice non-observed behaviours lead to declare correct trajectories as faulty.

Second, $Beh_{Obs} \subset Beh_{Id}$ and $Beh_{Exc} \neq \varnothing$ with $Beh_{Exc} = Beh_{Orig} \setminus Beh_{Id}$ the exceeding behaviour. The first inclusion traduces the simulation relation enounced by (Lee and Varaya, 2003). It indicates that the model is complete in the sense defined by (Blanke *et al.*, 2003) and so that it is suitable for fault detection. Nevertheless, even if the state spaces on which the different trajectories are built are identical, there are more trajectories in the identified behaviour than those really observed. The general difficulty of the identification problem is to reduce the number of exceeding trajectories. The number of exceeding trajectories represents the accuracy of the model. In terms of fault detection, these exceeding trajectories result in behaviours that will be considered fault free even if they are erroneous. This leads to non detectable faults.

To evaluate the accuracy of the model, a formal definition of the different behaviours is given later in the paper.

Several theoretical works in the field of the identification have been done in the sixties and seventies in computer science, see (Kella, 1971; Biermann and Feldman, 1972; Veelenturf, 1978;

Booth, 1967). Most of these works aim at identifying languages in the form of Moore or Mealy machines. This implies especially that the modelled system establishes a causal relationship between its inputs and outputs. In the case of interest, such models are not suitable since we want to model trajectories in a finite state space without outlining the relationship between the input and the output signals. To do that, we define a specific class of automata named Non Deterministic Autonomous Automaton with Output (NDAAO). This model, based on a non deterministic autonomous automaton (Litz, 2004), is formally defined as:

NDAAO = $(\mathbf{X}, \mathbf{\Omega}, f_{\text{nd}}, \lambda, \mathrm{x}_0, \mathrm{x}_\mathrm{f})$ with:
$\mathbf{X}$: finite set of states,
$\mathbf{\Omega}$: finite set of output symbols,
$f_{\text{nd}}: \mathbf{X} \to 2^{\mathbf{X}}$: transition function ($2^{\mathbf{X}}$ represents the power set of $\mathbf{X}$),
$\lambda: \mathbf{X} \to \mathbf{\Omega}$: output function,
x_0: initial state,
x_f: final state.

x_0 is the state representing the first observed I/O vector of each observation whereas x_f represents the last observed I/O vector. Even the output associated to these states are identical, they are distinguished for the identification process. For the purpose of fault-monitoring, x_0 and x_f cannot be defined. The initial state of the fault detection will be identified during the initialization of the fault monitoring process.

The dynamics of this automaton is: Given a current state x_i, the automaton can evolve in any state x_j such as $\mathrm{x}_j \in f_{\text{nd}}(\mathrm{x}_i)$. When several reachable states are possible, the choice is not determined.

The output function λ associates each state of the automaton with an element of the set of output symbols $\mathbf{\Omega}$. This set is composed of the different observed I/O vectors $\mathrm{u}_i(t)$. As for Moore machine, the output holds as long as the automaton is in the corresponding state.

As for a deterministic automaton (Cassandras and Lafortune, 1999), the language accepted by a NDAAO can be defined. We note $L^n_{\mathrm{x}_i}$ the set of words of length n that can be accepted starting from state x_i and L^n the set of words of length n accepted by the NDAAO. Formally, these are defined as:

$$L^n_{\mathrm{x}_i} = \left\{ \begin{array}{l} s \in \mathbf{\Omega}^n : s = \left(\lambda\left(x(1)\right), \lambda\left(x(2)\right), ..., \lambda\left(x(n)\right)\right) : \\ \left[\exists\left(x(1), x(2), ..., x(n)\right) : x(1) = \mathrm{x}_i \in \mathbf{X}, \right. \\ \left. \forall\, 1 \le t \le n-1, x(t+1) \in f_{\text{nd}}\left(x(t)\right)\right] \end{array} \right\}$$

and

$$L^n = \bigcup_{\mathrm{x}_i \in \mathbf{X}} L^n_{\mathrm{x}_i} .$$

3. QUALITY OF THE IDENTIFIED MODEL

As explained in section 2, the identified model must be complete and accurate. In this section, quality metrics are defined to evaluate these two qualities of the identified automata.

The goal of the fault detection is to compare the current trajectory with the trajectories that are possible in the model. A fault is detected when there is a discrepancy between the observed trajectory and the ones possible in the model. These trajectories are theoretically infinite. Comparing infinite trajectories is not possible; hence we will compare trajectories of a given length n. A trajectory of length n in the identified model corresponds to a word of the accepted language L^n. The observed sequences of length n can be compared with a word of the language L^n. To compare both behaviours, we first need a formal definition of these behaviours.

3.1 Definition of behaviours.

Identified Behaviour. The identified behaviour of length n is defined as the set of trajectories of length lower or equal to n that can be followed in the state space of the identified automaton. This is equivalent to the number of words of length lower or equal to n accepted by the automaton. Formally,

$$Beh^n_{Id} = \bigcup_{i=1}^{n} L^i .$$

Observed Behaviour. The observed behaviour of length n is the finite set of trajectories of length lower or equal to n that have been observed. If we define the observed language L^k_{Obs} as the set of words of length k that have been observed, i.e.

$$L^k_{\text{Obs}} = \bigcup_{\sigma_i \in \Sigma} \left(\bigcup_{j=1}^{|\sigma_i|-k+1} \left(\mathrm{u}_i(j), \mathrm{u}_i(j+1), ..., \mathrm{u}_i(j+k-1)\right) \right),$$

we can define the observed behaviour of length n as

$$Beh^n_{Obs} = \bigcup_{k=1}^{n} L^k_{\text{Obs}} .$$

3.2 Quality Metrics.

Before the identification algorithm is presented, let us define some criteria to evaluate the quality of the produced solution. (Geffroy *et al*., 1995) define two criteria: The structural and the behavioral complexities.

Structural Complexity. The structure of the identified model is characterized by the number of states and the number of transitions of the model. Looking at the number of states allows seeing if the produced model can be handled. A large model is a handicap for the online fault detection. Hence the model should be kept as small as possible. The number of transitions itself does not bring lots of information about the structure but the number of transitions in regard of the number of states gives information on the degree of non determinism of the machine. Basing on these considerations, two metrics for the structural complexity are defined:

$C_{S1} = |\mathbf{X}|$, representing the number of states of the identified model, and

$C_{S2} = \frac{\sum_{x \in \mathbf{X}} |f_{nd}(x)|}{|\mathbf{X}|}$, representing the mean number of transitions per state.

These metrics are quite similar to those defined by (Gilb, 1977) to characterize the structural complexity of software programs. Instead of considering states, Gilb considers the number of modules and the number of module connections.

Behavioural Complexity. The main identification problem is to obtain an accurate model, i.e. a model that does not allow too many non-observed behaviours. A way to qualify the accuracy of the model is to quantify the exceeding number of words of length *n* accepted by the model. Hence we define the behavioural complexity at rank *n* as the ratio between the number of words of length *n* accepted by the model and the number of observed words of length *n*, i.e. $C_B^n = \frac{|L_{Id}^n|}{|L_{Obs}^n|}$.

This metric can be interpreted in terms of non detectable faults. Assume we have $C_B^3 = 1.25$. That means that the identified model can accept 25% more words of length 3 than the original system does. During the fault detection, once the current state is clearly defined, after two evolutions, the probability that a non detected fault has occurred is $1 - 1/1.25 = 0.2 = 20\%$. That means although the followed trajectory is in the identified model, there are 20% risks that this trajectory is erroneous.

In terms of behaviours, reducing the amount of non detectable faults aims at having $Beh_{Obs}^n = Beh_{Id}^n$ for a given value of *n*. That means, $\forall i \le n, L_{Obs}^i = L_{Id}^i$ and $\forall i \le n, C_B^i = 1$.

4. IDENTIFICATION ALGORITHM

In this section, our algorithm that identifies a machine generating exactly L_{Obs}^{k+1} is detailed. This machine respects the condition $Beh_{Obs}^{k+1} = Beh_{Id}^{k+1}$. In order to do so, we use a parameter *k* that allows adjusting the accuracy of the model. This approach was inspired from (Biermann and Feldmann, 1972) who used such a parameter for the identification of Mealy machines.

To identify the NDAAO, we start with a set Σ of observed sequences. Building an automaton that represents the trajectories of length $k+1$ between the states aims at building an automaton that accepts the languages composed of words of length $k+1$. Each letter of these words is an observed I/O vector.

Our algorithm proceeds in six steps:

1. For each observed sequence σ_i, construction of sequences of *k* vectors $u_i(t)$ where *k* is the a priori fixed parameter.
2. Construction of the NDAAO.
3. Renaming of the output function.
4. Reduction of the last state.
5. Merging of equivalent states.
6. Closure of the automaton.

To illustrate this identification process, let us consider the example of an elementary plant with a controller having two inputs and one output, see figure 3. Two sequences have been observed. Each I/O vector is coded as $u_i(t) = \begin{pmatrix} i_1(t) \\ i_2(t) \\ o_1(t) \end{pmatrix}$. Using this notation, let us consider the observed sequences are:

$$\sigma_1 = \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right) \text{ and}$$

$$\sigma_2 = \left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right).$$

In order to simplify the notation, each I/O vector is coded as A, B, C, D or E. These letters represent the letters of the observed alphabet. With this coding, the observed sequences are: $\sigma_1 = $ (A, B, C, D, E, A) and $\sigma_2 = $ (A, C, B, C, D, A).

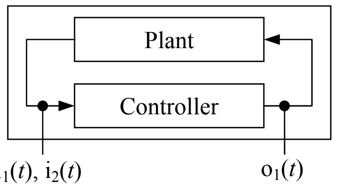

Fig. 3. Sample system used for the identification.

4.1 Step 1: Construction of vector sequences.

In this first step, we transform the observed sequences of letters into sequences of words of length *k*. The defined sequences are built on elements of L_{Obs}^k. In order to respect the hypothesis that each sequence starts and ends with the same word, the first and the last I/O vector of each sequence have to be duplicated $k-1$ times. Setting $k = 2$, we obtain for the example:

$\sigma_0^2 = ((A,A),(A,B),(B,C),(C,D),(D,E),(E,A)$ $(A, A))$ and $\sigma_1^2 = ((A,A),(A,C),(C,B),(B,C)$ $(C, D),(D, A),(A, A))$.

4.2 Step 2: Construction of the NDAAO.

In this step, we build the NDAAO that accepts the observed language L_{Obs}^{k+1}. We proved that a NDAAO that associates each state with an element of the observed language L_{Obs}^k and represents the transitions between these words accepts exactly the observed language L_{Obs}^{k+1}. The first word of each sequence, defined in section 4.1, is associated with x_0 whereas the last word is associated with x_f. The identification principle is to associate each different word with a single state, see figure 4.

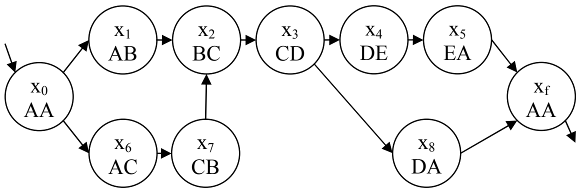

Fig. 4. Identified NDAAO.

4.3 Step 3: Rename the output function.

Each state of the NDAAO of figure 4 corresponds to a unique and stable value of the input and output signals. This value is described by the last letter of each sequence of length k. The corresponding state is renamed with this last letter, see figure 5.

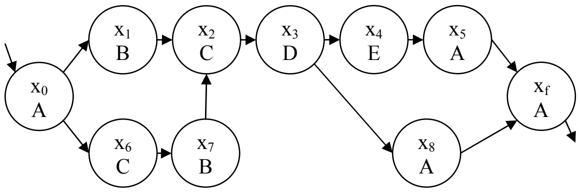

Fig. 5. Identified NDAAO after renaming the output function.

4.4 Step 4: Reduction of the final state.

In the NDAAO of figure 5, the last k states of each branch ending with x_f are associated with the same letter. These states, used as a construction artefact, can now be reduced. The following procedure has to be iterated $k-1$ times. First, merge the pre-states of x_f. Second redefine this new state as the final state x_f and delete the former x_f from the set of states. In the example, states x_5 and x_8 are merged and x_5 is the new final state, see figure 6.

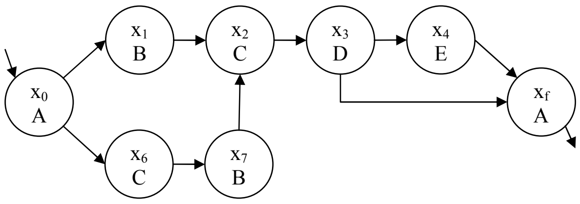

Fig. 6. Identified NDAAO after the reduction of the last state.

4.5 Step 5: Merging of equivalent states.

In step 3, several states can be associated with the same output. It must be shown if these states are equivalent before they can be merged. Two states x_i and x_j are equivalent if and only if:

1. They are associated with the same output, i.e. $\lambda(x_i) = \lambda(x_j)$.
2. They have the same set of post states, i.e. $f_{nd}(x_i) = f_{nd}(x_j)$.

We proved that, taking into account this definition, the merging of equivalent states does not affect the languages accepted by the NDAAO. The interest of this merging is to reduce the structural complexity of the automaton by limiting the number of states and transitions. This is particularly important for large-scale systems. In our example, states x_1 and x_7 are equivalent. Hence they are merged and only x_1 remains, see figure 7.

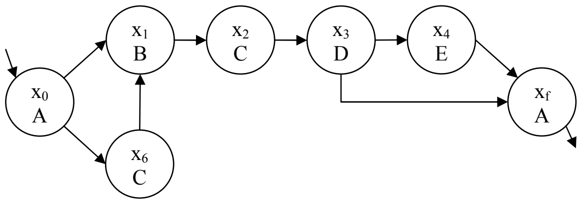

Fig. 7. Identified NDAAO after the merging of x_1 and x_7.

4.6 Step 6: Closure of the automaton.

With the hypothesis that each observed sequence corresponds to a single production cycle, the states x_0 and x_f of the NDAAO figure 7 are identical. Thus the NDAAO can be closed resulting in a strongly connected NDAAO, see figure 8.

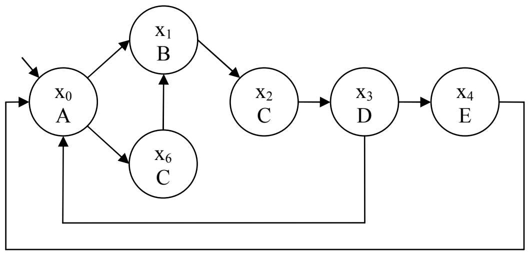

Fig. 8. Strongly connected identified NDAAO.

5. SOME CONCRETE RESULTS

In this section, two systems – an academic one and an industrial one – of different type and size are presented to illustrate the proposed approach.

5.1 Academic example.

The first case study is an assembly station of a Bosch Rexroth manufacturing line implemented at the LURPA, ENS Cachan. This station handles 35 input and output signals controlled by a Schneider PLC. We observed these signals during 18 production cycles.

Table 1 gives some quality indicators of the automata identified with these sequences. We find:

- k, the identification parameter related to the length of the words accepted by the automaton.
- C_{S1}, C_{S2} and C_B^n, the metrics defined in section 3.2

It can be noticed that when k increases, the number of states of the NDAAO does not grow considerably. The mean number of transitions decreases slowly but the model is far more accurate. On this example, a very accurate and relatively small model can be built. Hence a great k should be used in this case.

Table 1. Quality metrics for the identified automaton of the academic example

k	C_{S1}	C_{S2}	C^3_B	C^4_B	C^5_B	C^6_B
1	103	1.23	1.05	1.14	1.28	1.46
2	108	1.20	1	1.05	1.12	1.21
3	115	1.18	1	1	1.03	1.09
4	124	1.16	1	1	1	1.03

5.2 Industrial Example.

The second example is the rewinding stand of a machine producing fleece material. This machine, located in Kaiserslautern in the company Freudenberg, handles 336 input/output signals. A production cycle generates about 1000 events, representing 350 different I/O vectors. 51 machining cycles have been recorded.

As for the previous example, the NDAAO have been computed and the results are given in table 2 with the same notations as for table 1. "*n.c.*" stands for "*not computed*". That means that the metrics could not be calculated in a reasonable time.

Table 2. Quality metrics for the identified automaton of the industrial example

k	C_{S1}	C_{S2}	C^3_B	C^4_B	C^5_B	C^6_B
1	2303	2.10	2.09	5.66	n.c.	n.c.
2	3723	1.65	1	1.59	2.73	n.c.
3	5246	1.44	1	1	1.36	2.08
4	6746	1.32	1	1	1	1.25

For $k=1$, the model is quite compact but it does not allow an efficient fault detection. Indeed after two state evolutions, the risk that a non detectable fault has occurred is about 52% ($1-1/2.09=0.52$). After the third evolution, this probability is about 82%. Using a great value of the parameter k ($k\geq 3$) the accuracy of the model is correct but its size leads to a less convenient use for online fault detection.

6. CONCLUDING REMARKS

In this paper, we focused on the identification of discrete event systems for the purpose of model based fault detection. We defined a specific class of automata called non deterministic autonomous automaton with output. Our identification algorithm allows setting a parameter k to adjust the accuracy of the identified automaton. Using this algorithm, it is possible to build a model that reproduces exactly the observed trajectories of length $k+1$. To qualify the accuracy and the structural complexity of the identified model, quality metrics have been defined. The method works very well with large laboratory plants and has proved to be appropriate for large industrial plants. Nevertheless, there are some problems to be solved in the latter case. They concern first the reduction of the automaton size in order to decrease the initialisation time of the fault detection process and second the interpretation and minimization of the discrepancies.

REFERENCES

Biermann, A.W., and J.A. Feldman (1972). On the Sysnthesis of Finite-State Machines from Samples of Their Behavior. *IEEE Transactions on Computers*, **Vol. 21**, pp. 592-597.

Blanke, M., M. Klinnaert, J. Lunze and M. Staroswiecki (2003). *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, Berlin, Heidelberg, New York.

Booth, T.L. (1967). *Sequential Machines and Automata Theory*. John Wiley and Sons, Inc., New York, London, Sidney.

Cassandras, C.G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*, chapter 2. Kluwer Academic Publishers, Boston, Dordrecht, London.

Davis, R. and W. Hamscher (1988). Model-based Reasoning: Troubleshooting. In: *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, (Schrobe H. (dir), Morgan Kaufmann), pp. 297-346.

Geffroy, J.-C., C. Baron and K. El Maadani (1995). Identification des systèmes séquentiels – Une approche unifiée. *Technique et science informatique*, **Vol. 14 – n° 7**, pp. 809-837.

Gilb, T. (1977). *Software Metrics*. Winthrop Publishers, Inc., Cambridge.

Kella, J. (1971). Sequential Machine Identification. *IEEE Transactions on Computers*, **Vol. 20**, pp. 332-338.

Lee, E.A. and P. Varaya (2003). *Structure and Interpretation of Signals and Systems*, chapter 3. Addison Wesley, Boston.

Litz, L. (2004). *Grundlagen der Automatisierungstechnik*, chapter 3. Oldenbourg Verlag, München, Wien.

Lucas, P. and L. van der Gaag (1991). *Principles of Expert Systems*. Addison-Wesley Publishing Company, Wockingham.

Lunze, J., J. Schröder and P. Supavatanakul (2001). Diagnosis of Discrete Event Systems: the Method and an Example. In: *Proceedings of the 12th International Workshop on Principles of Diagnosis (DX'01)*, Via Lattea (Italy).

Sampath, M., R. Sengutpa, S. Lafortune, K. Sinnamohideen and D. Teneketzis (1996). Failure Diagnosis using Discrete-Event Models. *IEEE Transactions on Control Systems Technology*, **vol 4, No. 2**, pp. 105-124.

Schneeweiß, W.G. (1984). Tutorial on Advanced Concepts in Fault Tree Analysis. *Informatik Berichte Fernuniversität Hagen*, **Nr. 52**.

Veelenturf, L.P.J. (1978). Inference of Sequential Machines from Sample Computations. *IEEE Transactions on Computers*, **Vol. 27**, pp. 167-170.