An FDI Method for Manufacturing Systems Based on an Identified Model

Matthias Roth, Jean-Jacques Lesage, Lothar Litz

An FDI Method for Manufacturing Systems Based on an Identified Model

Matthias Roth*,. Jean-Jacques Lesage**,**
Lothar Litz*

**Institute of Automatic Control, University of Kaiserslautern,*
P.O. Box 3049, 67653 Kaiserslautern, Germany, {mroth, litz}@eit.uni-kl.de

***LURPA – Ecole Normal Superieur de Cachan*
61, Avenue du Président Wilson, 94235 Cachan Cedex, France, {roth, lesage}@lurpa.ens-cachan.fr

Abstract: In this paper a generic method for fault detection and isolation (FDI) in manufacturing systems considered as discrete event systems (DES) is presented. The method uses an identified model of the controlled process to be monitored which is built on the basis of observed fault free behavior. A special term of accuracy is motivated that helps to identify an efficient model. This paper gives an overview of the method that consists of the identification and the use of the identified model for fault detection and isolation. Furthermore, the theoretical framework of the method will be explained. Experiences of an industrial application are described to show the relevance of the method for large scale manufacturing systems in operation.

Keywords: Discrete Event Systems, Fault Detection, Fault Isolation, Identification

1. INTRODUCTION

Manufacturing companies have to face a growing competition that affects the quality and price of manufactured goods as well as the manufacturing process itself. To stay competitive it is a key interest of a company to increase the availability of its production lines. Besides a reasonable maintenance management to avoid downtimes due to e.g. mechanical wear it is necessary to detect and to localize unforeseen events like faults. Efficient fault detection and localization facilitates fast and effective repair actions.

Automated manufacturing systems are usually controlled by a Programmable Logic Controller (PLC) and are often highly individual installations. Many systems have a large number of digital inputs and outputs (I/Os) which leads to a complex system behavior. Sometimes FDI methods are directly integrated into the control algorithm running on the PLC. Integrating FDI in existing control programs needs a lot of expertise for the considered system and leads to modifications in the PLC which is usually to be avoided.

The fact that manufacturing systems are often controlled on the basis of binary input information allows considering these systems as discrete event systems. In the last 10 to 15 years a lot of research on fault diagnosis of discrete event systems has been done. Most of the developed approaches are based on a diagnoser which has been introduced by (Sampath, et al., 1996). The diagnoser approach starts with building the models of the system components and of the controller program by including the fault-free, normal system behavior as well as the behavior in case of given faults. The models of system components and of the control algorithm are composed to obtain the monolithic model of the process. This model contains unobservable and observable events. Faults are usually modeled as unobservable events. From this model a special observer called diagnoser is derived that allows detecting and diagnosing the occurrence of an anticipated fault by analyzing only observable events. The diagnoser estimates the current state of the observed system and delivers information of possible faults in the system.

Another class of diagnostic approaches that works on the basis of event order and the timed behavior of a system is presented in (Pandalai and Holloway, 2000). This method does not work with a state estimation but with so called condition templates. It analyzes if the considered system creates events in the right order or in given time delays. A fault is detected if there are missing or wrong reactions in the process.

(Philippot, et al., 2007) presented a combination of these two kinds of approaches that is adapted for fault diagnosis of manufacturing systems. Instead of one central diagnoser, a set of decentralized diagnosers is developed that delivers diagnosis information according to special rules. The timed behavior of the considered system is monitored with condition templates in each state of the diagnoser. The method is very appropriate if a diagnosis system is to be developed during the conception phase of a manufacturing system. It needs detailed models of the system components as well as the determination of the timed system behavior. If a formal description of the control algorithm such as GRAFCET is available the model for the controller can be derived automatically.

As a common drawback of these approaches a high degree of system knowledge is needed. The manual model building or

determination of event orders is a laborious task that can make it impossible from an economical point of view to install an FDI system. Especially for already existing and operating manufacturing systems with many I/Os it can be hard to obtain the necessary information such as models for non standard components or a formal description of the controller program.

Since model building is the main obstacle for the use of model-based diagnosis techniques an approach based on an identified model is presented in this paper. Only very few physical knowledge of the system is necessary. The presented method can handle large scale manufacturing systems with a very high number of I/Os (>100). In the following parts of the paper it will be explained how an appropriate model of an automated system can be identified. In contrast to the classical diagnoser approach the identified model does not contain the system behavior in case of faults but only the fault free behavior. Furthermore it will be explained how this fault-free model can be used for fault detection and fault isolation even if the faulty behavior is not part of the model. With this technique it is possible to localize and to detect faults even if they have not been considered before. Each behavior that is not part of the identified model is considered as faulty and leads to a fault detection. Although the modeling effort in comparison to the diagnoser approach is much smaller there are some drawbacks. The proposed method usually leads to false alerts and actually has to be improved in the field of fault localization.

2. PROPOSED METHODOLOGY

The proposed method considers a manufacturing system as a closed loop discrete event system. To use an identification approach without the interference of test signals it is necessary that the considered system performs a cyclic (and thus repetitive) production. The closed loop consists of the PLC and the physical plant that is to be observed. Fig 1 shows the principle architecture of the method. The first step is the identification of an appropriate model of the closed loop system. The identification is carried out on the basis of I/O vectors of the PLC. In this work the control algorithm running on the PLC is considered as fault-free.

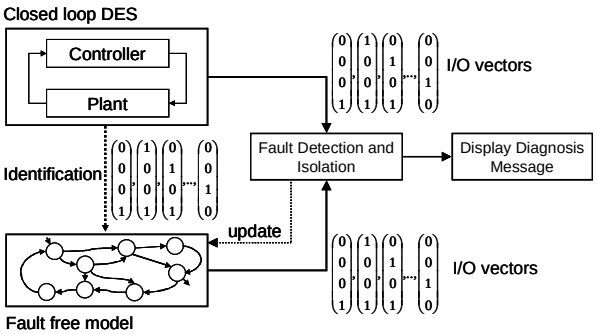

Fig 1: Principle of the proposed method

Fault detection is performed in real time by comparing the output of the model and the output of the closed loop DES in the block “Fault Detection and Isolation”. If discordance between observed and modeled sequences appears a fault is detected. Furthermore, by comparing observed and expected sequences fault localization is possible. The diagnosis information can then be displayed to the system operators. Unlike to approaches that use a reference model with faults like (Sampath, et al., 1996), it is not possible to give guarantees concerning the diagnosabilty of certain faults.

During the fault detection and isolation procedure it may also be possible that a deviant I/O vector sequence is not considered as faulty but as a false alert. This decision can be achieved by the experience of the system operator or by some heuristic rules. In this case the model can be updated with the new observed sequence.

In the following chapters a more detailed description of the elements of the method is given.

3. DATA COLLECTION AND ANALYSIS

In order to get the necessary information for identification and FDI to the diagnosis PC, the I/O vectors of the PLC must be collected. In Fig 2 the principle of this data collection is depicted. After the PLC has read the values of the input data it copies them to a local data sector. Then the control program is executed and the values of the outputs are set according to the control algorithm. At the end of the program execution the input and the newly determined output values are sent to the PC that contains the FDI methods. With an appropriate data link between PLC and PC the method can be considered as noninvasive such that there is only a very small negative effect of the FDI on the time performance and the correctness of the controller. More details on an appropriate data link will be given in chapter 7.

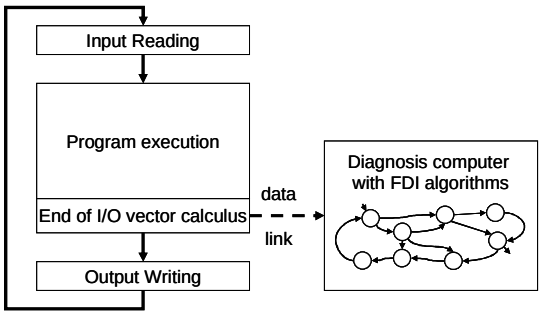

Fig 2: Data collection within the PLC-cycle

Fig 3 shows how the I/O vectors are sampled from the real values of the I/Os. If the PLC performs a cyclic (and not a periodic) execution, the cycle time is not constant. That explains why the I/O vectors in Fig 3 are not sampled in equidistant time intervals. At the end of each PLC cycle the actual vector is sent. If there is at least one I/O with a change in value a new I/O vector is created. For presentation purposes the I/O vectors are denoted with letters.

If the observed I/O vector sequence in Fig 3 is analyzed it can be seen that the I/O vector *C* appears twice. Suppose that the I/Os in Fig 3 represent the position sensors (index 1-3) and the actuator (index *n*) of a conveyor that is part of a larger system like in Fig 4. The vector *C* can represent both cases when the work piece is between position 1 and 2 and when it is between position 2 and 3. To distinguish the two physical

states by analyzing the I/O vectors it is necessary to take longer sequences into account. The two physical states can be distinguished if the preceding vectors of *C* are also considered. The sequences *BC* and *DC* differ and thus the physical states can be distinguished.

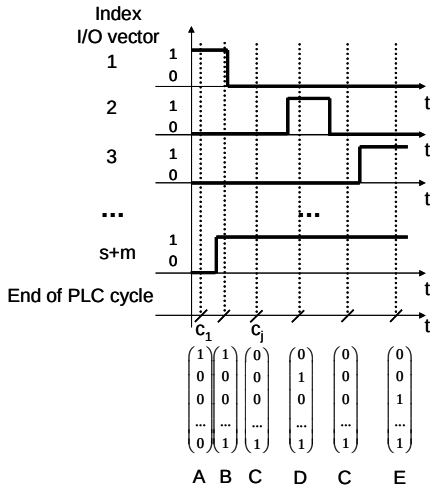

Fig 3: Collection of I/O vectors

For FDI it is crucial to have a very accurate estimation of the physical system state since an event that is correct if the system is in state 1 may be a fault in another system state 2 even if it produces the same I/O vectors for both states. In the example consider a change in value of the sensor P3 in case 1 and in case 2.

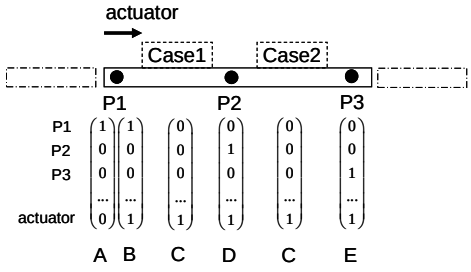

Fig 4: Conveyor with two scenarios

Before translating the observed sequences of I/O vectors into a DES model it is necessary to define the observed language that is used for identification. The identification is carried out on the basis of *p* different fault-free production cycles.

Definition 1: The *j*-th I/O vector in the *h*-th production cycle is defined as $u_h(j) = (I_1(j), ..., I_s(j), O_1(j), ..., O_m(j))_h$ with I_1, .., I_s and O_1, .., O_m denoting the considered inputs and outputs of the closed loop system.

Definition 2: If during the *h*-th production cycle, l_h I/O vectors have been observed, the sequence is denoted as $\sigma(h) = (u_h(1), u_h(2), ..., u_h(l_h))$.

With these definitions it is possible to define the set of observed words with length *q* observed during *p* different production cycles. This set represents the set of observed I/O vector sequences

Definition 3: The observed words of length *q* are denoted as

$$W_{\mathrm{Obs}}^{q} = \bigcup_{i=1}^{p}\left(\bigcup_{j=1}^{l_i-q+1} \left(u_i(j), u_i(j+1), ..., u_i(j+q-1)\right)\right).$$

As explained by the example of Fig 4 longer words describe the physical state of a system more accurately than shorter ones by including the history of the system. An objective of the identification algorithm is to increase the length of correctly identified words. Thus, it works on the basis of words of a given length rather than on single I/O vectors.

The union of the observed words with length from 1 to *n* can be considered as the observed language.

Definition 4: The observed language of length *n* is

$$L_{Obs}^{n} = \bigcup_{i=1}^{n} W_{Obs}^{i}$$

The aim of the identification algorithm is to build a model that produces exactly the same language L_{Ident}^{n} as the observed language:

$$L_{Obs}^{n} = L_{Ident}^{n}$$

Before the identification of the model can be performed, it is necessary to state if the language and thus the words of a given length of the system have been completely observed. If the language is not completely observed it is possible that new words will be observed that are part of the fault free system behavior. Since the identified model will not be able to produce the new words ($L_{Obs}^{n} = L_{Ident}^{n}$) a false alert will be raised. To state if a language or the words of a certain length *n* are completely observed the cardinality of the set W_{Obs}^{n} can be analyzed. If the set converges to a stable level over time it is very likely that each word of a certain length has been observed.

4. IDENTIFICATION OF A DES MODEL

To decrease the modeling effort in comparison with the classical approaches mentioned in the introduction, the observed language of the closed loop DES is to be translated into a finite state machine by an appropriate identification algorithm. Since a system with interaction of controller (deterministic behavior) and physical process (non-deterministic behavior) has to be identified, an appropriate model is a non-deterministic autonomous automaton with output (NDAAO):

Definition 5: NDAAO = ($\mathbf{X}$, $\mathbf{\Omega}$, r, λ, x_0)

$\mathbf{X}$	finite set of states,
$\mathbf{\Omega}$	output alphabet,
$r : \mathbf{X} \rightarrow 2^{\mathbf{X}}$	non-deterministic transition relation,
$\lambda : \mathbf{X} \rightarrow \mathbf{\Omega}$	output function,
x_0	initial state.

The dynamics of this automaton is: Given a current state x(*i*), the automaton can evolve in any state x(*j*) such that

$x(j) \in r(x(i))$. When several reachable states are possible, the choice is not determined.

The output alphabet Ω consists of the observed I/O vectors such that in the case of an identified NDAAO $\boldsymbol{\Omega} = W^1_{Obs}$. λ assigns a word and thus an I/O vector from the output alphabet to each state.

The NDAAO generates a set of words of length *n* from each state x(*i*) that is defined as:

Definition 6:

$$W^1_{x(i)} = w \in \boldsymbol{\Omega}^1 : w = (\lambda(x(i))) \quad \text{and}$$
$$W^{n>1}_{x(i)} = \{w \in \boldsymbol{\Omega}^n : w = (\lambda(x(i)), \lambda(x(i+1)), ...,$$
$$\lambda(x(i+n-1))) : x(j+1) \in r(x(j)) \forall\, i \le j \le i+n-2\}$$

The definition of the language of length *n* of the NDAAO follows definition 4:

Definition 7: The language of length *n* of an NDAAO is

$$L^n_{Ident} = \bigcup_{i=1}^{n} \bigcup_{x(i) \in X} W^i_{x(i)}$$

Thus, the language consists of all the words up to length *n* that can be produced starting from each state $x(i) \in X$.

In (Klein, et al., 2005) an original identification algorithm that delivers an appropriate model for FDI is described in detail. It works on the basis of words of the parametric length *k*. Basically, states that represent observed words of length *k* are connected in the order the words have been observed. The algorithm delivers a model that is *k*+1-complete (Moor, et al., 1998). This means that the automaton generates *exactly* the observed words of length *k*+1: $L^{k+1}_{Obs} = L^{k+1}_{Ident}$. The parameter *k* can be chosen such that each physical system state can unambiguously be identified by an I/O vector sequence of length *k*. Each sequence of length *k* is assigned to exactly one automaton state even if it appears several times during the observation. The choice of *k* in the example of Fig 4 would thus be *k*=2.

As a small example suppose that the following three sequences have been observed:

$\sigma_1 = (\mathrm{A,B,C,D,E,C,A})$ $\quad \sigma_2 = (\mathrm{A,D,B,C,D,A,C,A})$ and $\sigma_3 = (\mathrm{A,D,B,C,F,D,E,C,A})$.

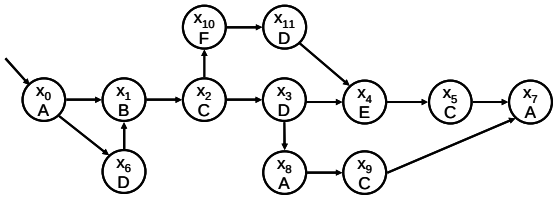

Fig 5: Identified NDAAO with *k*=2

Fig 5 shows the identified automaton for parameter *k*=2. Even if it contains unobserved sequences of length 4 ($\sigma_4 = (\mathrm{A, B, C, F})$), it does only contain words of length 3 that have been observed since $L^{k+1}_{Obs} = L^{k+1}_{Ident}$. In the case that each two physical system states can be distinguished by a sequence of length *k*, each model state represents exactly one system state which is advantageous for FDI as explained for the example in Fig 4.

5. FAULT DETECTION AND ISOLATION

5.1 Fault detection and model initialization

To use the identified model for FDI it must be run in parallel with the considered system. In Fig 1 the principle is shown: The I/O vector sequence of the closed loop system is compared with an I/O vector sequence of the model. If the model is not able to produce the observed I/O vector sequence, a fault is detected. The comparison of sequences is realized as follows: It is checked if the output of the actual state $\lambda(x(i))$ or the output of one of its post states $\lambda(x(j)) \mid x(j) \in r(x(i))$ corresponds to the newly observed I/O vector. If this is the case, the determined state becomes the new actual state. If not, a fault is detected.

When a fault has been detected the model must be reinitialized. Hence, a new actual model state that corresponds to the system state must be determined. Since the model is identified on the basis of *k*-long I/O vector sequences, it is necessary to collect up to the next *k* new I/O vectors as the observed word of length *k* W^k_{Obs}. If an automaton state exists that can be reached by a state trajectory starting in a state $x(i) \in X$ with $W^k_{\mathrm{x(i)}} = W^k_{\mathrm{Obs}}$ state x(*i*+*n*-1) according to definition 6 becomes the actual state of the model.

If the model is reinitialized very quickly after a fault detection (e.g. within *k* new vectors) it is possible that the detected fault was a false alert. In this case it is possible to perform an online update of the model (see section 6) depending on the decision of a system operator.

5.2 Fault Localization Techniques

Since the identified model does not contain any information of the system in case of a fault like in the classical diagnoser approach, specific fault localization techniques are necessary. For the considered systems with many I/Os, fault localization consists of giving possible inputs or outputs that may lead to the faulty component. Thus, sensor faults are related to inputs and actuator faults are related to outputs.

The localization technique presented here is inspired by the residual technique known from continuous time systems (Isermann 2005). The main idea is to compare the observed and the expected behavior of the system (Roth et al., 2009). Since the localization aims at giving inputs and outputs as fault candidates it is necessary to give some definitions that introduce rising and falling edges of I/Os in the model.

Definition 8:
$E = (I_1\_1, I_1\_0, ..., I_s\_1, I_s\_0, O_1\_1, O_1\_0, ..., O_m\_1, O_m\_0)$ denotes the set of rising and falling edges of the system I/Os. I_1_1 denotes the rising edge and I_1_0 denotes the falling edge of input I_1.

Definition 9: $ES(u(1), u(2))$ denotes the set of observed rising and falling edges between two consecutive I/O vectors $u(1)$ and $u(2)$.

Using the function given in definition 9 it is possible to get all the rising and falling edges that appear when a state is left by iterating over its transition relation.

The fault localization consists of operations on sets defined in definition 9. As an example of such an operation consider the automaton given in Fig 6. From the actual state x_1 an I/O vector with the rising edge c_1 and the falling edge b_0 is observed on the system. The observed I/O vector can not be produced by the model as explained in chapter 5.1. Thus, a fault is detected. One possible question that leads to the localization of the fault is: "what happened unexpectedly?" An answer can be found in the calculus of a residual using the previously defined sets. u_{actual} denotes the I/O vector that led to fault detection and x denotes the actual state before the fault was detected:

$$\text{Res} = ES(\lambda(x), u_{actual}) \setminus \bigcup_{\forall x' \in r(x)} ES(\lambda(x), \lambda(x'))$$

First, all the observed rising and falling edges between the last actual state and the newly observed I/O vector are taken ($ES(\lambda(x), u_{actual})$). The observed set is: {c_1, b_0}. Then, the union of all edges is built that appear when the actual state x is left to another state $x' \in r(x)$ resulting in {b_0, d_0}. For the example in Fig 6, the residual delivers the rising edge c_1 that is not part of the modeled following behavior but has been observed ({c_1, b_0}\{b_0, d_0}={c_1}). If a fault happens that leads to a directly observable change in value of an I/O, this I/O is usually part of the presented residual.

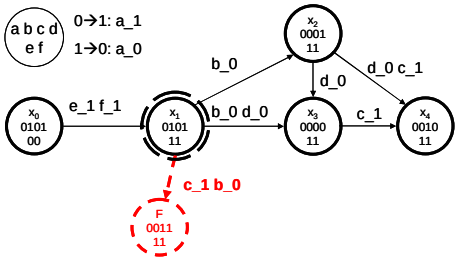

Fig 6: Automaton with rising and falling edges of I/Os

If large scale systems with numerous I/Os are considered, the presented residual approach can help to deliver a small set of fault candidates to the system operator which represents a good estimate where the fault could have happened.

6. ONLINE ENHANCEMENT OF THE MODEL

In large scale systems with high concurrency it is usually not possible to observe the whole fault-free language within a short time. Thus, it can be necessary to apply an incomplete model for the FDI process and to perform updates when an unknown fault-free I/O vector sequence has been observed after the learning phase. A newly observed sequence that is not yet part of the model can be declared as fault free by the system operator or by the analysis of the re-initialization process as sketched in chapter 5.1. An update algorithm must ensure that the model is still k+1-complete after processing a new sequence.

Suppose that the new sequence that is to be included in the model is σ_{new}. During fault detection, the model tries to produce the sequence with a state trajectory that ends in x_{actual} from where a certain I/O vector $u_{new}(i) \in \sigma_{new}$ cannot be produced by the state itself or by one of its successors.

Update algorithm:

Check if $\exists x \in X \mid W_x^k = (u_{new}(i), .., u_{new}(i+k))$.
If yes: $r(x_{actual}) := r(x_{actual}) \cup x$ and
$x_{actual} := x(i+n-1)$ according to definition 6.
If no: create $x_{new} \mid (r(x_{new}) = \{\}, \lambda(x_{new}) = u_{new}(i))$,
$r(x_{actual}) := r(x_{actual}) \cup x_{new}$ and $x_{actual} := x_{new}$

Proceed with the analysis of $u_{new}(i+1) \in \sigma_{new}$. If $u_{new}(i+1)$ cannot be produced by x_{actual} itself or by one of its successors start the update algorithm again starting at $u_{new}(i+1)$.

Remarks: The check at the beginning of the algorithm also delivers a "no" if there are less than k new I/O vectors in the sequence. It also delivers a "no" if a certain state can produce the given sequence $(u_{new}(i), .., u_{new}(i+k))$ *and other additional* sequences.

As an example suppose the following sequence has been observed with an NDAAO$^{k=2}$ starting in state x_0 (see Fig 7): $\sigma_{new} = (\text{A,B,G,C,F,D,E})$. From state x_1 the automaton will not be able to produce the sequence and must thus be updated. Since k=2, sequences of length k+1=3 must be analyzed. There does not exist a state x with $W_x^3 = (G, C, F)$, hence a new state with $\lambda(x_{new}) = G$ is created. The analysis proceeds with the vector *C*. Although there exist states with the output *C*, there does not exist a state that can *only* produce the next observed subsequence *CFD*. State x_2 can produce *CFD* but also other sequences like *CDE* or *CDA*. Connecting the new State $x_{\text{new}} = x_{12}$ to x_2 would lead to a sequence of length 3 that is in the model but has not been observed (*GCD*). Thus, a new state with $\lambda(x_{new}) = C$ is created and the next vector of σ_{new}, *F*, is analyzed. Since state x_{10} is only able to produce the new following sequence *FDE*, the before created state is connected to x_{10} and the update is finished. The identification algorithm in (Klein, et al., 2005) ensures that there is only one state from where a certain sequence of length k+1 can be produced.

It is possible to implement the update algorithm in a very efficient way, such that even automata with up to several hundred states can be treated online with a standard PC.

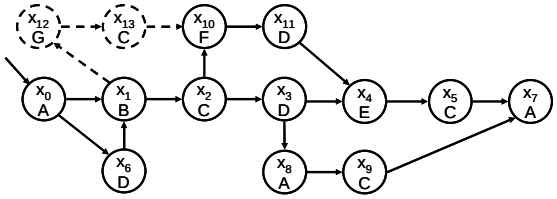

Fig 7: Updating of an NDAAO with k=2

7. APPLICATION

The described method has been applied to an industrial coil winding process with about 80 I/Os that is controlled by a PLC. The process automatically changes a coil when a predefined batch of fabric is coiled up. The inclusion of FDI algorithms in the PLC was not possible since only minimal change of the control program was desired. To implement a noninvasive data link between PLC and diagnosis PC a UDP (User Datagram Protocol) connection was established. The PLC sends its I/O vectors at the end of each controller cycle to the PC using a direct Ethernet link without passing the message through a network. The efficiency of the PLC program is not affected if the UDP data connection fails. It just keeps on sending I/O vectors without waiting for a confirmation message. Since the machine has already been running for over 10 years the necessary knowledge for manual model-building was not available and could not be obtained within reasonable time and cost. Hence, the identification approach was chosen.

For the identification of the fault free model over 380 production cycles have been observed. Fig 8 shows how the cardinality for the word sets containing the words of length n from n equals 1 to 5. It can be seen that for small values of n the plots almost converge to a stable level. As described in chapter 3 the identification parameter k must be determined such that each system state can unambiguously be determined by a sequence of k I/O vectors. For the industrial application, situations comparable to the example of Fig 4 have been determined as the most critical ones. Hence, for the identification of an appropriate FDI model k should be chosen to k=2. The resulting model has 562 states.

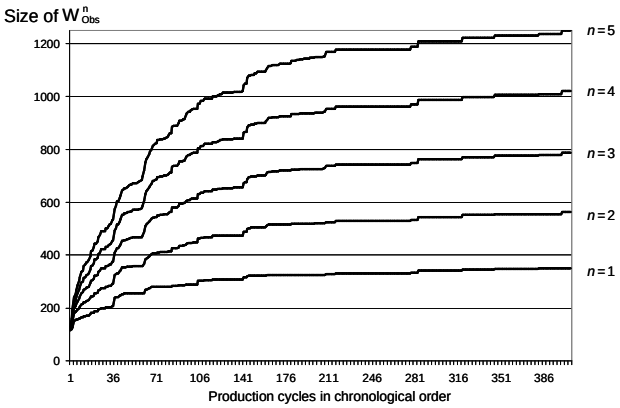

Fig 8: Cardinality of the observed word sets

As explained in chapter 3 the evolution of $W_{obs}^{k+1} = W_{obs}^{3}$ should converge to a stable level. Fig 8 shows that the set does not perfectly converge but it seems to be relatively near to that point. It can be seen that for larger values of n achieving a stable level takes much more time than for smaller values. As the non-convergence of the graph for n=3 implies, new fault-free I/O vector sequences have been observed using the identified model as an observer. Each new I/O vector sequence of length k+1 leads to an alert and to a re-initialization of the model. If the alert is declared as a false alert, the new sequence can be included into the model by using the online update algorithm of chapter 6. An FDI-tool using the described method is currently running at a production site of the industrial partner.

8. OUTLOOK

Current research aims at a decentralized identification approach that should cope better with the problem of incomplete observation. The formulation of appropriate residuals is also a field of current research. To systematically develop further residuals, symptoms of possible faults in manufacturing system that are observed with the presented approach are analyzed. Furthermore the timed behavior of manufacturing systems is soon to be included in the model.

REFERENCES

Isermann, R. (2005). *Model-based fault-detection and diagnosis – status and applications*, Annual Reviews in Control 29 (2005), pp. 71-85.

Klein, S., Lesage, J.-J., Litz, L. (2005). Fault Detection of Discrete Event Systems Using an Identification Approach. *16th IFAC World Congress*, Prague (Czech Republic) CDRom paper n°02643

Moor, T., Raisch, J. and O'Young, S. (1998). *Supervisory Control of Hybrid Systems via l-complete approximations*, Proceedings of the 4th IEEE Workshop on Discrete Event Systems WODES'98, Cagliari (Italy), pp. 426-431.

Pandalai, D.N. and L.E. Holloway (2000). Template Languages for Fault Monitoring of Timed Discrete Event Processes. *IEEE Transactions on Automatic Control*, vol 4, No. 5, pp. 868-882.

Philippot, A., Sayed-Mouchaweh M., Carré-Ménétrier V. (2007). Unconditional Decentralized Structure for the Fault Diagnosis of Discrete Event Systems. *Proceedings of the 1st IFAC Workshop on Dependable Control of Discrete Systems*, Cachan (France), pp. 255-260

Roth, M., Lesage, J.-J., Litz, L. (2009). A Residual Inspired Approach for Fault Localization in DES. *Proceedings of the 2nd IFAC Workshop on Dependable Control of Discrete Systems*, Bari (Italy), (accepted for publication)

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.