Using Learned Models for the Root Cause Analysis of Cyber-Physical Production Systems

Oliver Niggemann[1,2], **Stefan Windmann**[1], **Sören Volgmann**[1], **Andreas Bunte**[2], **Benno Stein**[3]

[1]Fraunhofer Application Center Industrial Automation, Lemgo, Germany
e-mail: {oliver.niggemann, stefan.windmann, soeren.volgmann}@iosb-ina.fraunhofer.de

[2]Institute Industrial IT, Lemgo, Germany
e-mail: {andreas.bunte}@hs-owl.de

[3]Bauhaus University Weimar, Germany
e-mail: {benno.stein}@uni-weimar.de

Abstract

The diagnosis of Cyber-Physical Production Systems (CPPS) comprises two main steps: (1) The identification of anomalous system behavior and (2) the deduction of the underlying root cause. While step 1 requires only models of the OK-behavior of the system, step 2 requires models that can predict the system behavior in OK and especially in fault situations. Over the last years, the question where such models originate has become a major research topic—due to the highly adaptable nature of CPPS which renders a manual modeling infeasible.

Because of the infeasibility of manual modeling, algorithms have been developed for step 1 which learn an OK-model based on system observations. Theoretically, also fault models for step 2 could be learned, but practically we incur a dilemma since fault events occur too seldom to learn a fault model from them.

This paper introduces the new algorithm *MoSDA* which shows a way out of this dilemma. *MoSDA* does not use fault models but extracts more information from learned OK-models than previous algorithms: The main idea is to go from easy-computable anomalies on the system level to hard-computable anomalies on the component level. In practice, efficient heuristics for the deduction of root causes can be given if anomalies are known on a component level while a root cause analysis is hard if anomalies are only known on a system level.

1 Motivation

The diagnosis of distributed production systems has gained new attention due to research agendas such as Cyber-physical Production Systems (CPPS) [Lee, 2008; Rajkumar *et al.*, 2010] or its German pendant "Industrie 4.0". In these agendas, a major focus is on the self-diagnosis capabilities for complex and distributed CPPS. Typical goals of such self-diagnosis approaches are the detection of anomalies, suboptimal energy consumptions, or wear [Christiansen *et al.*, 2011; Isermann, 2004; Struss and Ertl, 2009; Windmann *et al.*, 2013]. As a technical solution approach, model-based diagnosis approaches became the preferred method [Niggemann *et al.*, 2012; Struss and Ertl, 2009; Christiansen *et al.*, 2011], mainly due to the complex causalities and the distributed architecture of such systems.

In the context of CPPS, model-based diagnosis must deal with adaptable, changeable plants, leading to the problem of both many and time-consuming model modifications and model verification steps. So the challenge for the application of model-based diagnosis approaches to CPPS is the synchronization between system and model for fast changing systems.

One possible solution is to learn models automatically from system observations. However, such learned models capture mainly the normal behavior, i.e., the system behavior if no fault has occurred. The learning of faulty behavior would require a large number of fault cases and a manual annotation of these faults—both are unrealistic requirements for most CPPS. Hence, learned models of normal behavior are therefore traditionally used for anomaly detection only [Niggemann *et al.*, 2012], i.e., to detect non-normal situations.

If such learned models are used, the diagnosis task itself, i.e., the identification of the root cause, is in such scenarios normally left to the human expert. The reason for this lies in the way model-based diagnosis identifies the root cause: the use of models which can also predict or simulate faulty behavior. For this, usually faults must be integrated into the model to verify whether a hypothetical fault would cause the observed symptoms. So the field of diagnosis faces a challenge: On the one hand, only learned model can handle modern production plants which are modified frequently. On the other hand, such learned models can for very fundamental reasons not be used for a root cause analysis.

The major contribution of this paper is to present a new algorithm *MoSDA* (Model Separation Diagnosis Algorithm) which tries to use learned models, i.e., models of the normal behavior, for the task of diagnosis and root cause analysis in CPPS. For this, *MoSDA* exploits a situation typical for CPPS: Anomalies often refer to an anomalous behavior of the overall system, e.g., an anomalous energy consumption of the production system. *MoSDA* leverages on the fact that the system measurements create an over-determined system and *MoSDA* can therefore break down—heuristically—the system-related anomalies to component-related anomalies. Starting from such component-related anomalies, heuristics can be easily developed that compute a set of possible root causes. So *MoSDA* does not implement a true root cause analysis but it implements a compromise between the usage of learned models and root cause analysis. *MoSDA* can limit heuristically the set of possible root causes to a small number of anomalous system components which helps the human expert significantly to repair the system and so to reduce plant downtimes. Furthermore, unlike other approaches (see section 2), *MoSDA* uses a generic modeling formalism that requires a minimum of system knowledge and which can be learned automatically for a wide range of production plants.

2 State of the Art

The diagnosis task can be structured into three steps, named anomaly detection (symptom generation), hypothesis generation, and hypothesis discrimination (see e.g. [Benjamins and Jansweijer, 1994]). The first step of anomaly detection computes anomalies and symptoms based on system observations. For this step, several model-based approach exist: Most of them are statical, i.e., they use time-invariant system features (see e.g. [Ferracuti *et al.*, 2011; Goernitz *et al.*, 2013; Chen *et al.*, 2014]). For continuous systems, state space equations are often employed for modeling the temporal transition of hidden process variables, e.g., Kalman filter-based observers [Narasimhan and Biswas, 2007; Williams and Henry, 2002; Zhao *et al.*, 2005]) or particle filters (see e.g. [Wang and Dearden, 2009]) are used. Kalman filters are also used for the anomaly detection of energy consumption in hybrid systems (see e.g. [Windmann *et al.*, 2013]).

The next steps are the hypothesis generation and discrimination. Some typical papers are summarized in table 1. All solutions have in common that domain knowledge is required to infer from anomalies and symptoms to possible root causes. The phenomenological approach directly classifies hypotheses based on the symptoms, where a hypothesis is one possible root cause for the symptom. Such classificators use machine learning algorithms such as neural network or support vector machine [Berjaga *et al.*, 2009; Wang *et al.*, 2000; Zhao *et al.*, 2007], but some use hard-coded rules [Blesa *et al.*, 2013].

The model-based approach uses a model which mainly describes the causalities from root causes to symptoms. The knowledge for this approach, i.e., the causalities, are often modeled manually [Abidin *et al.*, 2002; Klar *et al.*, 2011]. Sometimes, the model consists of a set of fault models, where each model represents the behavior for one hypothesis. All models are compared with the observations for the hypothesis generation, and the hypothesis of a matched model is selected [Pomeranz and Reddy, 2009; Minhas *et al.*, 2014]). Other approaches use a single model instead of several fault models. Such a model represents the whole behavior of the system and is influenced by parameters controlling component failures. In these algorithms, those parameters have to be identified which match the observed behavior [de Kleer *et al.*, 2013; Vemuri *et al.*, 2001]. The plant could be modeled based on components from a library, where every component already includes the faulty behavior modes. E.g., this has been done in Modelica, a popular modeling and simulation language where in [de Kleer *et al.*, 2013]) the library is augmented with fault models. Furthermore, often fault probabilities are used to reduce the number of hypothesis (see. e.g. [Stern *et al.*, 2013]).

A special situation exists if the considered subsystems are small. In that case, anomalies and root causes are closely related—of course only in a heuristic manner. An example is given in [Alippi *et al.*, 2013], it deals with a distributed sensor network in which faulty sensors are detected.

As can be seen, only model-based approaches are applied successfully to complex and distributed systems such as CPPS. And most of these solutions require high manual modeling efforts. However, this strategy is not acceptable for CPPS because of their adaptable and variant nature. On the other hand, machine learning and model learning is so-far mainly applied to simple phenomenological diagnosis cases. The paper in hand shows how to combine model learning and model-based diagnosis.

3 Solution Idea

The main idea of the *MoSDA* algorithm can be seen in figure 1: As mentioned before, for the diagnosis of CPPS we must rely on automatically learned models. I.e. based on observations (step 0 in figure 1), system behavior models are learned automatically. And since observations hardly comprise fault situations, these models predict only OK-situations.

To learn such system behavior models, the HyBUTLA algorithm [Niggemann *et al.*, 2012; 2013] is used in step 1 of figure 3. This algorithm splits the overall system behavior into single states— the result is a Timed Probabilistic Hybrid Automaton (see section 4.1). Each state of this automaton correspond to one specific production scenario, e.g. the filling of a tank or one operation mode of a robot. The transitions between states are often triggered by a control signal, e.g. a signal to open a valve or to stop a drive. The advantage of separating the overall model into these separate states is straightforward: Within such a state, the relation between the continuous signals can be modeled and learned much easier than for the overall system (see section 4.3). And because the relation between continuous signals can be expressed easier, generic model formalisms such as polynomial functions can be used, i.e. the behavior can be learned automatically and less manual modeling efforts, e.g. in form of differential equations, are needed. Details for step 1 are found in section 4.2. Based on such models, we can now detect anomalies (symptoms) in the overall system behavior: For this, system observations are compared to the model predictions. Examples are given in [Faltinski *et al.*, 2012; Gilani *et al.*, 2013; Maier *et al.*, 2011].

But because only the overall system behavior is modeled (and analyzed), such an approach does not allow for an identification of the root causes. For this, the behavior of individual components must be known. And this is not possible using the aforementioned learning approach, since in general continuous signals such as power consumptions are only measured on the system and not on the component level—a situation typical for production plants where a lot of measurements such as power consumption, water consumption, output, fluidic pressure are gathered by sensors on a system-level.

If models for the individual component behavior would exist, we could apply the principle of anomaly detection on the component level. This would of course not allow for a precise root cause detection, but would allow for good heuristic solutions. Good examples for this can be seen in [Isermann, 2007; 2004; de Kleer *et al.*, 2013; Cermignani and Tornielli, 1994] which use manually modeled differential equation systems—in these works, parameters are learned automatically while the equations are defined manually. And because these models work on a com-

Table 1: Current state of the art regarding hypothesis generation.

	Hypothesis generation		
Reference	Approach	Techniques	Knowledge base
[Alippi *et al.*, 2013]	Model-based	Markov Model + Statistical Classifier	Parametrized
[Cardenas *et al.*, 2003]	Model-based	State Machine	Manual
[de Kleer *et al.*, 2013]	Model-based	System Discretion Language	Parametrized
[Klar *et al.*, 2011]	Causal	System Discretion Language	Manual
[Struss and Ertl, 2009]	Causal	-	Manuel
[Fries, 2013]	Hybrid	Fuzzy + Fault Propagation Tree	Manual
[Berjaga *et al.*, 2009]	Phenomenological	Case-based Reasoning KNN	Learned
[Zhao *et al.*, 2007]	Phenomenological	SVM	Learned
[Azarian *et al.*, 2011]	Phenomenological	Fault Tree (Network)	Manual
[Pan *et al.*, 2012]	Phenomenological	SE-Tree	Manual
[Wang *et al.*, 2000]	Phenomenological	Neural Network	Learned

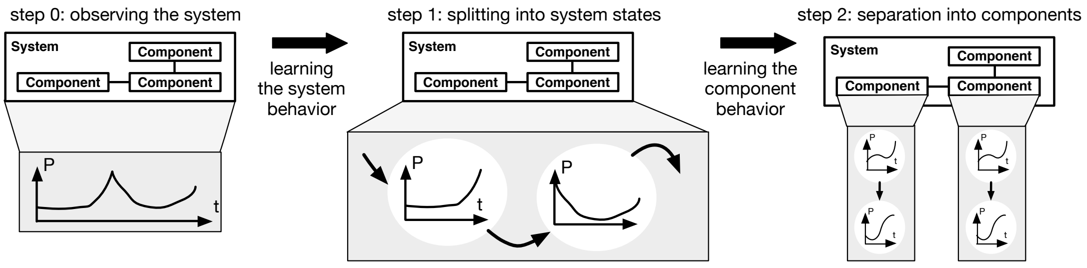
step 0: observing the system
System
Component
Component
Component
P
t
learning the system behavior
step 1: splitting into system states
System
Component
Component
Component
P
t
P
learning the component behavior
step 2: separation into components
System
Component
Component
Component
P
t

Figure 1: The main idea of the *MoSDA* algorithm.

ponent level, parameter changes on the component level can be detected and interpreted with regards to anomalies. In most cases, heuristics could be given which relate behavior changes in components to potential root causes. The disadvantage of these approaches are the high efforts needed for the manual creation of the differential equation models.

The *MoSDA* algorithm presented in this paper combines both solution approach: Generic model formalisms (here: Timed Probabilistic Hybrid Automata and function approximation) are employed, i.e. the models are learned automatically and no expensive manual efforts are needed. And it also computes models for the individual component behaviors, i.e. heuristics for the root causes can be given. For this, the learned system behavior models from step 1 are separated automatically into individual component models (step 2 in figure 1), details are given in section 5.

The reader may notice that this automatic separation into component models (step 2) is only possible because the overall system behavior has been split in step 1 into system states. As will be outlined in section 5, the separation approach exploits the observations that normally only some components are active in one state. I.e. this knowledge can be used to compute the components' influence on a state's behavior and to deduce the components' (correct or incorrect) behavior.

4 Step 1: Learning the System States

The *MoSDA* algorithm presented in this paper in section 5 relies on a special kind of hybrid timed probabilistic automaton as a generic and learnable model formalism to describe a plant's behavior. Both the formalism and the learning algorithm HyBUTLA have been introduced in [Niggemann *et al.*, 2012; 2013] and are shortly summarized in this section. The application of these models and algorithms to the task of root cause analysis is described in section 5.

4.1 Hybrid Timed Probabilistic Automata

An example of a hybrid timed automaton is given in Figure 2. It shows a simple container which is first emptied and then filled. The automaton comprises 2 states. The first state "State 0" is triggered by an event "empty". This event must occur $0-5$ seconds after the system had entered the previous state. In 20% of all cases the system takes this transition. Within this state, the container is emptied, the height of the bulk good in the container is modeled by a continuous variable $h(k)$ within the state. Unlike other hybrid automata formalisms, the continuous variables are here modeled using a state space representation. When the container is empty, a new event "fill" occurs with a probability of 100% and "State 1" is entered. Again, this event must occur $5-10$ seconds after "State 0" has been entered, e.g. timing is modeled relatively to each state entering.

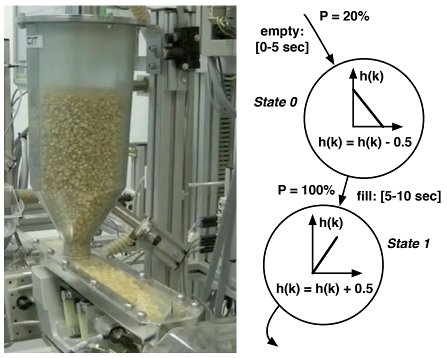

Figure 2: Example of a hybrid automaton.

Formally, hybrid timed automata are here defined as follows:

Definition 1. *Hybrid Timed Automaton: A hybrid timed automaton is a tuple $A = (S, s_0, F, \Sigma, T, \Delta, p, c, Y)$, where*

- *S is a finite set of states, $s_0 \in S$ is the initial state. States correspond to specific phases of the production system such as "Valve A is open" or "Chemical reaction B is running".*
- *Σ is the alphabet comprising all relevant events. Often events correspond to control signals which turn on/off an actor such as a valve or drive.*
- *$T \subseteq S \times \Sigma \times S$ gives the set of transitions. E.g. for a transition $\langle s, a, s' \rangle$, $s, s' \in S$ are the source and destination states and $a \in \Sigma$ is the trigger event.*
- *A set of relative transition timing constraints Δ with $\forall \delta \in \Delta : \delta : T \to I$, where $I = \mathbf{R} \times \mathbf{R}$ is the set of time intervals. A transition $(s_0, *, *)$ with an associated timing interval $[t_0, t_1]$ can only be taken if the automation has been between t_0 and t_1 time units in state s_0.*
- *Transition probabilities $\forall e \in T : p : T \to [0, 1]$, all outgoing-transition probabilities of a state s must summarize to a value ≤ 1, the difference to 1 defines the probability that the automaton remains in s.*
- *A set of functions $Y = \{\mathbf{y_0}, \dots, \mathbf{y_{n_s}}\}\}, n_s \in \mathbf{N}$ where y_s describes the continuous signals within states $s \in S$:*

$$\mathbf{y_s}(t) = f_s^{\mathbf{p}_s}(\mathbf{y_s}(t), \mathbf{y_s}(t-1), \dots, \mathbf{y_s}(t-i)), \quad (1)$$
$$i \in \mathbf{N}, \mathbf{p}_i \in \mathbf{R}^m, m \in \mathbf{N}$$

I.e. f_s computes the vector of continuous signals $\mathbf{y_s}$ based on the current and on older values for $\mathbf{y_s}$.
$\mathbf{p}_s$ is a vector of parameters, these parameters are later on learned automatically using system observations.

In contrast to the other definitions [Alur *et al.*, 1995; Henzinger, 1996], we have here opted for a single clock and relative timing constraints Δ because this ensures the efficient learnability. In [Verwer, 2010] it is shown that, in contrast to n-clock automata, 1-clock automata can be learned efficiently. Furthermore, we model continuous signals by means of explicit state space models and not by means of differential equations.

4.2 Learning the Discrete States

Learning the timed automaton comprises *(i)* the identification of the states and transitions (described in this section) and *(ii)* the identification of the continuous signals (described in section 4.3). Input to the algorithm are system observations where each observation is a sequence of timed events.

The algorithm HyBUTLA is defined formally in [Niggemann *et al.*, 2012; 2013], its major steps are also sketched in figure 3: Based on system observation (step 0 in figure 3), it computes in step *(1)* all relevant events Σ and their timings. For this, each event appearing in the measurements is analyzed whether it stems from a single-mode probability density function (PDF). If not, new events are generated for each separate mode of the PDF. Events usually correspond to control signals, i.e. we can be sure that within a state no control signals occur—a fact which will be essential for the algorithm *MoSDA* .

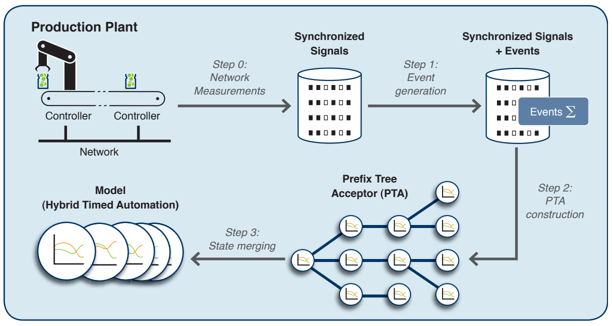

Figure 3: The main steps of the algorithm HyBUTLA.

In step *(2)* in the algorithm a prefix tree acceptor is created, which stores each equal prefix of the incoming observations in a dense form. The prefix tree acceptor represents therefore all used observations, but it does not generalize. In order to generalize, i.e. to learn, similar nodes must be merged into one node.

In step *(3)* in the algorithm, in bottom-up order each pair of states v, w is checked for compatibility. If they are found to be compatible, the states are merged. A very important issue is the compatibility criterion: The used compatibility check is following ALERGIA [Carrasco and Oncina, 1999]. The main idea is to check whether the probabilities for taking a specific transition or for stopping in the state are rather similar for both states. If the two nodes are found to be compatible, the compatibility of the respective subtrees must be checked too. This is done by applying the compatibility check recursively to all nodes in the subtrees. Since after the merging a non-deterministic automaton could have emerged, the resulting automation is determinized by merging states in the subtree.

4.3 Learning the continuous behavior in the states of the hybrid automaton

The continuous signals, e.g. energy consumptions, in each state $S = \{s_0, \dots, s_{n-1}\}$ of a hybrid automaton (see definition 1) are modeled by the functions $\mathbf{y_s}(t) = f_s^{\mathbf{p}_i}(\mathbf{y_s}(t), \mathbf{y_s}(t-1), \dots, \mathbf{y_s}(t-i))$: Usually state-based approaches or polynomial function approximation approaches are used.

The learning of the parameters $\mathbf{p}_i$ is described in section 5 and is an essential part of the solution idea. Here it is worth noting that learning these functions get much easier by splitting the overall system behavior into single states S—a state may model the filling phase of a container, a chemical reaction, the time intervall when a valve is open or a maintenance phase. And only because the learning is eased, we can use generic modeling formalisms such as Hybrid Timed Automata.

5 Step 2: Automatic Model Separation

The *MoSDA* algorithm and the model separation step will be explained first using figure 4, algorithm 1 defines it formally: As outlined in section 3, first of all a timed hybrid automaton is learned which models the overall system behavior. Algorithm 1 gets this automaton as an input in steps (a).

Next we have to notice that most continuous variables θ_s within the learned states s (see definition 1) model summarized observations. E.g. at the top of figure 4 a production

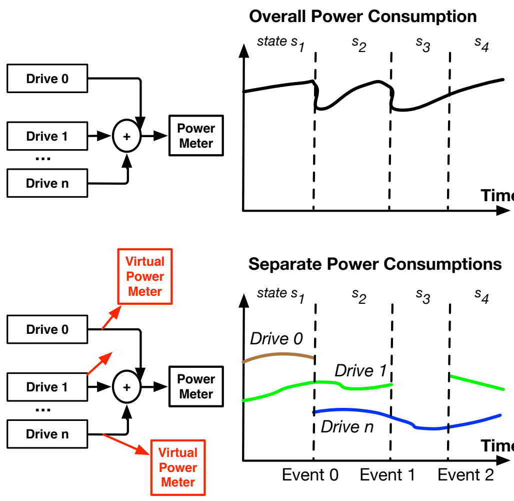

Figure 4: Model separation as a core idea of *MoSDA* .

plant's power consumptions can be seen. This power consumption is the sum of the power consumptions of n drives. Other examples for such summarized variables are water consumptions, buffers which are filled by the product flows from several transport mechanisms or filling levels which are caused by incoming and outcoming tubes.

The reader may note that this situation is typical: In production plants, the number of sensors is limited due to energy and cabeling reasons. Therefore, most continuous signals are only measured in the form of summarized signals, e.g. one energy sensors measures the energy consumptions of a whole subsystem, time measurements capture the time duration for several production steps or the measured water consumption refers to a whole production plant.

In order to better locate the anomalous subsystems or components, e.g. drive, we must separate the summarized variable, e.g. power consumption, into the individual behavior of the components, e.g. into the individual power consumptions of the drives. The easiest solution to this is the installation of new power meters but for cost and effort reasons this solution is pure theoretical and can not be implemented in real-world plants. In step (b) of the algorithm 1 the components C and their individual, i.e. separated, continuous signals (e.g. power consumptions) $\mathbf{y_{s,c}}(\mathbf{t})$ are defined.

So instead we separate the summarized variables using mathematical methods: For this, we first have to notice that at one point in time only a subset of drives are active, this can be seen as interrupted power consumption functions at the lower part of figure 4. Since drives are turned on and off using control signals (e.g. event 0-2 in figure 4), the intervals between the events correspond to the states of the learned hybrid automaton in section 4.2. I.e. by analyzing the learned discrete states (see section 4.2), we can derive automatically for each state of the hybrid automaton whether a drive is turned on or off: For this, only the events on the path from the initial state to the current state must be analyzed. As external knowledge, the algorithm only needs to know which events, i.e. signals, refer to which actuator (e.g. drive); this knowledge can easily be obtained from the engineering tool and is available for all automation systems. Algorithm 1 gets the computed information "component c is on/off in state s" as an input $f(s, c)$ in steps (c).

```
Algorithm MoSDA :
Given:
(a) Learned Automaton $A = (S, s_0, F, \Sigma, T, \Delta, p, c, \Theta)$
 // Learned by HyBUTLA [Niggemann et al., 2012]
(b) subsystems $C = \{c_0, \dots, c_{k-1}\}$ where for all states $s \in S$
   $c_i$ adds $\mathbf{y_{s,c}}(\mathbf{t})$ to $\mathbf{y_s}(\mathbf{t})$ and
   $\mathbf{y_{s,c}}(t) = f_{s,c}^{\mathbf{p}_{s,c}}(\mathbf{y_{s,c}}(t), \mathbf{y_{s,c}}(t-1), \dots, \mathbf{y_{s,c}}(t-i))$
   is similar to equation 1
// I.e. $\mathbf{y_{s,c}}(\mathbf{t})$ is that part of a cont. signals caused by component $c$
(c) function $f : S \times C \to \{0, 1\}$ with $\forall s \in S \forall c \in C : f(s, c) =$
$1 \Leftrightarrow c$ is turn on in state $s$
    function $f$ is computed automatically based on input (a)
Result: possible root causes
(1)  during learning phase:
(1)      observe summarized continuous variables $\mathbf{y_s}\tilde{(\mathbf{t})}$
(1)  $p_{s,c} = separate(A, \mathbf{y_s}\tilde{(\mathbf{t})}, C, f)$
(2   during operation phase:
(2)      observe summarized continuous variables $\mathbf{y_s}\tilde{(\mathbf{t})}'$
(2)  $p'_{s,c} = separate(A, \mathbf{y_s}\tilde{(\mathbf{t})}', C, f)$
(3)  compute probable root causes based on $p_{s,c} - p'_{s,c}$

Subroutine separate:
Given:
(a) Learned Automaton $A = (S, s_0, F, \Sigma, T, \Delta, p, c, \Theta)$
(b) $\mathbf{y_s}\tilde{(\mathbf{t})}, s \in S$: observed summarized continuous variables
(c) subsystems $C = \{c_0, \dots, c_{t-1}\}$
(d) function $f : S \times C \to \{0, 1\}$
Result: the learned parameters $p_{s,c}$ for the functions $\mathbf{y_{s,c}}$
(i)  compute $argmin_{p_{s,c}} \sum_{s \in S} \sum_t \left| \mathbf{y_s}\tilde{(\mathbf{t})} - \sum_{c \in C} f(s, c) \mathbf{y_s^c}(t) \right|$
     // where || denotes a distance measure
```

Algorithm 1: Model separation algorithm *MoSDA*

Using methods from the field of mathematical optimization, the *MoSDA* algorithm learns to which extent a drive (i.e. components $C = \{c_0, \dots, c_{k-1}\}$ in algorithm 1) is responsible for the overall power consumption in a specific state. E.g. given enough observations, we can derive mathematically the individual power consumptions of the drives. This corresponds to virtual power meters seen in figure 4.

Using this approach, summarized variables can be mathematically separated into their single components. These separated variables characterize in detail the individual component's behavior and allow therefore to detect a component's misfunction. In algorithm 1 the separation is done in the subroutine **separate** by computing in step (i) the functions $f_{s,c}^{\mathbf{p}_{s,c}}$. The structure of these functions $f_{s,c}^{\mathbf{p}_{s,c}}$ is predefined, so learning these functions means finding parameter values $p_{s,c}$ which minimize the difference between observations and function values.

At this point, it must first be discussed how these learned and separated subsystem behaviors $f_{s,c}^{\mathbf{p}_{s,c}}$ can be used to identify the root cause: The main idea is the comparison of the learned behavior so-far (i.e. learned parameters $p_{s,c}$ in step (1) of algorithm 1) with the currently observed behavior (i.e. current parameters $p'_{s,c}$ in step (2) of algorithm 1). If the learned and the observed component models differ (i.e. the parameters $p_{s,c}$ and $p'_{s,c}$), the component is classified as anomalous in step (3) of algorithm 1.

By using *MoSDA* , we do not classify only the overall plant behavior $f_s^{\mathbf{p}_s}$ as anomalous, instead we now classify the behavior $f_{s,c}^{\mathbf{p}_{s,c}}$ of single components (e.g. drives) as anomalous. This a significant advantage since those subsystems form the root causes and must be therefore analyzed separately. E.g. for the energy consumption example, while we can classify normally only whole production lines as anomalous, we can now classify single drives as anomalous. Hence *MoSDA* does a major step towards a root cause analysis with learned models: The learned models now refer to individual components and not to overall systems.

But a major problem remains: Due to the chaining effect in CPPS, a root cause (or several root causes) can still cause anomalous behavior in error-free but related subsystems. But in CPPS, in most cases the root causes, i.e. the broken components, can be identified by several heuristics: (1) A-priori error probabilities, (2) often, the behavior of the root causes degrade first, (3) manual expert knowledge can be used.

6 Applications of the Algorithm

6.1 Diagnosis of a High Storage System

First, the *MoSDA* has been applied to a high storage system. The system comprises real world components such as drives, conveyer belts and automation devices. 5 conveyor belts and 7 drives are used to move objects from and to 4 storage positions on two levels. 14 inductive sensors are used to get the objects' positions and an energy sensor captures continuously the overall power consumption. Typically, wear and faults occur mainly in the drives, causing too high energy consumptions, too long transport durations and finally plant downtimes. To prevent this, any wear in a drive must be detected as early as possible. This has been implemented using the *MoSDA* algorithm from section 5.

For this, in each state of the hybrid automaton, 3 continuous signals are modeled: power consumption $p(k)$, velocity $v(k)$ and acceleration $a(k)$—corresponding to functions $\mathbf{y_s}(k)$ in definition 1. The functions $p(k)$ are modeled by a function template $p(k) = c + c_a a(k) + c_v v(k) + c_{a2} a^2(k) + c_{v2} v^2(k) + c_{av} a(k) v(k), c, c_a, c_v, c_{a2}, c_{v2}, c_{av} \in \mathbf{R}$—i.e. $c, c_a, c_v, c_{a2}, c_{v2}, c_{av}$ correspond to the parameters $\mathbf{p}_s$ in definition 1.

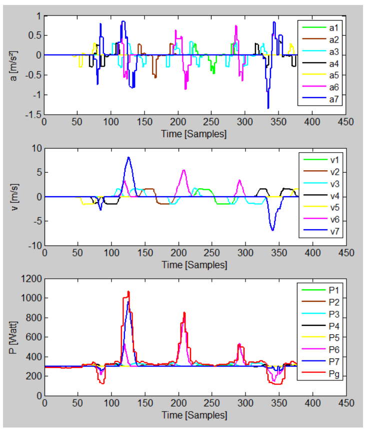

Figure 5: The results of the *MoSDA* algorithm.

The overall power consumption $p(k)$ of the high storage system is now split into the individual power consumptions of the 7 drives—the drives are the components C in line (c) of algorithm 5 and for each drive the power consumption corresponds to $\mathbf{y_{s,c}}(k)$ of algorithm 5 . The results of the *MoSDA* algorithm are shown in figure 5: The first row of figure 5 shows the 7 accelerations, the second row the 7 velocities and the last row shows the measured overall power consumption Pg and the 7 separated power consumptions as computed by the *MoSDA* algorithm.

In the research high storage system (unlike in real plants), the separated power consumptions can be compared to the real power consumptions: 10 cycles have been measured, the average prediction error was between 1.7% and 7.4%. I.e. the separate power consumptions can be learned effectively. Therefore, in this example, deviations of more than 7.4% from the normal behavior of the single drives can be used to detect wear and faults. Furthermore, in this case the root cause was always the first drive to show deviations.

6.2 Diagnosis of Drives

Next, the algorithm *MoSDA* has been evaluated using a set of simulated drives, the same set of functions is used as in section 6.1. Here, a state-based functional template is used: $\mathbf{p}(k) = A\mathbf{p}(k-1) + B\mathbf{v}(k), A, B \in \mathbf{R}^5 \times \mathbf{R}^5$, where the total power consumption is summed up through $y(k) = Cp(k)$.

Figure 6 depicts a simulated scenario of 5 similar drives with a predefined velocity for each drive as input parameter. The resulting total energy consumption is shown in the second row of Figure 6. Furthermore, the *MoSDA* algorithm separates the energy into the individual power consumptions of each drive by estimating the model parameters of the template function.

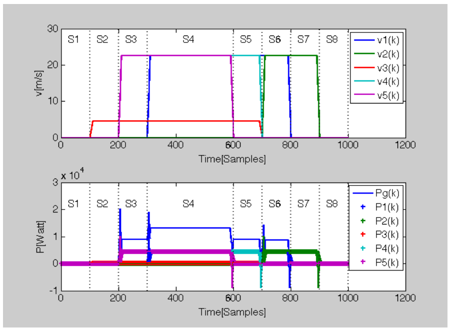

Figure 6: Simulated scenario of 5 drives and result of the *MoSDA* algorithm.

In this simple example, 5 drives are triggered with a predefined velocity at different points in time. The *MoSDA* algorithm estimates the parameters of the functional template by iterating through states $S1$ to $S8$. The resulting average prediction error during the constant velocity phase is $< 0.5\%$.

The simulation setup in figure 7 is used to illustrate anomaly detection and is similar to figure 6. Therefore, the parameters of the estimated separated power consumptions are used to detect anomalies. Figure 7 depicts occurring errors in state $S4$ and $S7$. The simulated total energy differs from the measured consumption. By using the estimated model parameters and the given velocity as input data, the simulated energy reveals to the normal behavior. This can be used to detect faults, i.e. in state $S4$ the drive 5 consumes more energy than normal. Also, small peaks can be detected, as depicted in state $S7$, where the power consumption of drive 2 differs from the expected power consumption.

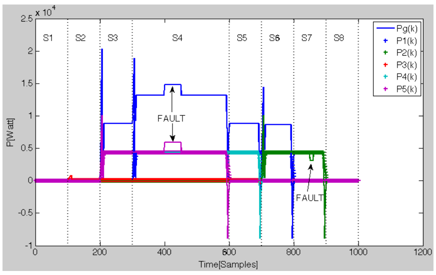

Figure 7: Anomalies detected by the *MoSDA* algorithm.

7 Conclusion and Outlook

Only learned models are a basis for the diagnosis of Cyber-physical Production Systems (CPPS) since no manual modeling approach can keep up with the dynamics of modern products and production systems. And in practice, learned models cover only OK-situations. But such OK-models can not be used for the root cause analysis step of traditional model-based diagnosis approaches–mainly because they classify only the whole system behavior as anomalous and not single components. Hence the design of new diagnosis algorithms which only use OK-models is a key research topic for CPPS. Such a new algorithm, the *MoSDA* algorithm, is presented for the first time in this paper. Furthermore, the *MoSDA* algorithm has been evaluated using two first experimental setups.

In the future, the *MoSDA* algorithm will be evaluated using more production systems. Another focus will be the support for more classes of technical devices such as valves, pumps, and cylinders (so-far, the focus was on drives) and the usage of other types of continuous summarized signals such as water consumptions.

References

[Abidin *et al.*, 2002] M.Shukri Zainal Abidin, Rubiyah Yusof, Marzuki Khalid, and Shamsuddin Mohd. Amin. Application of a model-based fault detection and diagnosis using parameter estimation and fuzzy inference to a dc-servomotor. In *Intelligent Control, 2002. Proceedings of the 2002 IEEE International Symposium on*, 2002.

[Alippi *et al.*, 2013] C. Alippi, S. Ntalampiras, and M. Roveri. A cognitive fault diagnosis system for distributed sensor networks. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(8):1213–1226, Aug 2013.

[Alur *et al.*, 1995] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. h. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[Azarian *et al.*, 2011] A. Azarian, A. Siadat, and P. Martin. Optimization of a knowledge-based system by a meta-heuristic approach for the automotive diagnosis. In *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on*, 2011.

[Benjamins and Jansweijer, 1994] R. Benjamins and W. Jansweijer. Toward a competence theory of diagnosis. *IEEE Expert*, 9(5):43–52, Oct 1994.

[Berjaga *et al.*, 2009] X. Berjaga, A. Pallares, and J. Melendez. A framework for case-based diagnosis of batch processes in the principal components space. In *Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–9, Sept 2009.

[Blesa *et al.*, 2013] Joaquim Blesa, Fatiha Nejjari, Damiano Rotondo, and Vicenc Puig. Robust fault detection and isolation of wind turbines using interval observers. In *Control and Fault-Tolerant Systems (SysTol), 2013 Conference on*, 2013.

[Cardenas *et al.*, 2003] C. Cardenas, J. Olmos, D. Garcia, and E. Baeyens. Modelling, supervision and diagnosis of a manufacturing cell. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, volume 1, pages 69 – 74 vol.1, 2003.

[Carrasco and Oncina, 1999] R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. In *RAIRO (Theoretical Informatics and Applications)*, volume 33, pages 1–20, 1999.

[Cermignani and Tornielli, 1994] Stefano Cermignani and Giorgio Tornielli. Model-based diagnosis of continuous static systems. In *Annals of Mathematics and Artificial Intelligence*, volume 11, pages 367–380, 1994.

[Chen *et al.*, 2014] Huanhuan Chen, P. Tino, A. Rodan, and Xin Yao. Learning in the model space for cognitive fault diagnosis. *Neural Networks and Learning Systems, IEEE Transactions on*, 25(1):124–136, Jan 2014.

[Christiansen *et al.*, 2011] L. Christiansen, A. Fay, B. Opgenoorth, and J. Neidig. Improved diagnosis by combining structural and process knowledge. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, Sept 2011.

[de Kleer *et al.*, 2013] Johan de Kleer, Bill Janssen, Daniel G. Bobrow, Tolga Kurtoglu Bhaskar Saha, Nicholas R. Moore, and Saravan Sutharshana. Fault augmented modelica models. *The 24th International Workshop on Principles of Diagnosis*, pages 71–78, 2013.

[Faltinski *et al.*, 2012] Sebastian Faltinski, Holger Flatt, Florian Pethig, Björn Kroll, Asmir Vodenčarević, Alexander Maier, and Oliver Niggemann. Detecting anomalous energy consumptions in distributed manufacturing systems (in press). In *IEEE 10th International Conference on Industrial Informatics INDIN, Beijing, China*, 2012.

[Ferracuti *et al.*, 2011] F. Ferracuti, A. Giantomassi, S. Longhi, and N. Bergantino. Multi-scale pca based fault diagnosis on a paper mill plant. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–8, Sept 2011.

[Fries, 2013] T.P. Fries. Automation of rapid fault diagnosis in manufacturing systems using multiple fuzzy agents. In *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pages 65–70, Aug 2013.

[Gilani *et al.*, 2013] S. Gilani, S. Windmann, F. Pethig, B.Kroll, and O. Niggemann. The importance of model-learning for the analysis of the energy consumption of production plant. In *18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep 2013.

[Goernitz *et al.*, 2013] Nico Goernitz, Marius Micha Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. In *Journal Of Artificial Intelligence Research*, volume 46, pages 235–262, 2013.

[Henzinger, 1996] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–292, Washington, DC, USA, 1996. IEEE Computer Society.

[Isermann, 2004] Rolf Isermann. Model-based fault detection and diagnosis - status and applications. In *16th IFAC Symposium on Automatic Control in Aerospace*, St. Petersbug, Russia, 2004.

[Isermann, 2007] Rolf Isermann. Fahrdynamische regelungen und überwachung. *Automatisierungstechnik*, 55(6):279–280, 2007.

[Klar *et al.*, 2011] D. Klar, M. Huhn, and J. Gruhser. Symptom propagation and transformation analysis: A pragmatic model for system-level diagnosis of large automation systems. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–9, Sept 2011.

[Lee, 2008] E.A. Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369, 2008.

[Maier *et al.*, 2011] Alexander Maier, Oliver Niggemann, Asmir Vodenčarević, Roman Just, and Michael Jaeger. Anomaly detection in production plants using timed automata. In *8th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*. Noordwijkerhout, The Netherlands, Jul 2011.

[Minhas *et al.*, 2014] Raj Minhas, Johan de Kleer, Ion Matei, Bhaskar Saha, Bill Janssen, Daniel G. Bobrow, and Tolga Kurtoglu. Using fault augmented modelica models for diagnostics. In *International ModelicaConference*, 2014.

[Narasimhan and Biswas, 2007] S. Narasimhan and G. Biswas. Model-based diagnosis of hybrid systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 37(3):348 –361, May 2007.

[Niggemann *et al.*, 2012] Oliver Niggemann, Benno Stein, Asmir Vodenčarević, Alexander Maier, and Hans Kleine Büning. Learning behavior models for hybrid timed systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.

[Niggemann *et al.*, 2013] Oliver Niggemann, Asmir Vodencarevic, Alexander Maier, Stefan Windmann, and Hans Kleine Büning. A learning anomaly detection algorithm for hybrid manufacturing systems. In *The 24th International Workshop on Principles of Diagnosis (DX-2013)*, Jerusalem, Israel, Oct 2013.

[Pan *et al.*, 2012] Yuxiong Pan, Qingdong Loi, Zhang Ren, Lei Dong, and Xiaojun Zhang. Test point optimization for model-based fault diagnosis expert system. In *Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on*, 2012.

[Pomeranz and Reddy, 2009] Irith Pomeranz and Sudhakar M. Reddy. Selection of a fault model for fault diagnosis based on unique responses. In *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE '09*, 2009.

[Rajkumar *et al.*, 2010] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.

[Stern *et al.*, 2013] Roni Stern, Meir Kalech, Alexander Feldman, Shelly Rogov, and Tom Zamir. Finding all diagnoses is redundant. *The 24th International Workshop on Principles of Diagnosis*, pages 216–221, 2013.

[Struss and Ertl, 2009] Peter Struss and Benjamin Ertl. Diagnosis of bottling plants - first success and challenges. In *20th International Workshop on Principles of Diagnosis, Stockholm*, Stockholm, Sweden, 2009.

[Vemuri *et al.*, 2001] A.T. Vemuri, M.M. Polycarpou, and A.R. Ciric. Fault diagnosis of differential-algebraic systems. In *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 2001.

[Verwer, 2010] Sicco Verwer. *Efficient Identification of Timed Automata: Theory and Practice*. PhD thesis, Delft University of Technology, 2010.

[Wang and Dearden, 2009] M. Wang and R. Dearden. Detecting and Learning Unknown Fault States in Hybrid Diagnosis. In *Proceedings of the 20th International Workshop on Principles of Diagnosis, DX09*, pages 19–26, Stockholm, Sweden, 2009.

[Wang *et al.*, 2000] Zhenyuan Wang, Yilu Liu, and Paul J. Griffi. A combined ann and expert system tool for transformer fault diagnosis. In *Power Engineering Society Winter Meeting, 2000. IEEE*, 2000.

[Williams and Henry, 2002] Brian C. Williams and Melvin Michael Henry. Model-based estimation of probabilistic hybrid automata. Technical report, 2002.

[Windmann *et al.*, 2013] Stefan Windmann, Shuo Jiao, Oliver Niggemann, and Holger Borcherding. A stochastic method for the detection of anomalous energy consumption in hybrid industrial systems. In *INDIN*, 2013.

[Zhao *et al.*, 2005] Feng Zhao, Xenofon D. Koutsoukos, Horst W. Haussecker, James Reich, and Patrick Cheung. Monitoring and fault diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(6):1225–1240, 2005.

[Zhao *et al.*, 2007] Wen-Qing Zhao, Yong-Li Zhu, De-Wen Wang, and Xue-Ming Zhai. A fault diagnosis model for power transformer based on statistical theory. In *Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07. International Conference on*, 2007.