Model-Based Diagnosis for Cyber-Physical Production Systems Based on Machine Learning and Residual-Based Diagnosis Models

Andreas Bunte
OWL University of Applied Sciences
Institute Industrial IT
Langenbruch 6, 32657 Lemgo, Germany
andreas.bunte@hs-owl.de

Benno Stein
Bauhaus-Universität Weimar
Faculty of Media, Webis Group
99421 Weimar, Germany
benno.stein@uni-weimar.de

Oliver Niggemann
OWL University of Applied Sciences
Institute Industrial IT
Langenbruch 6, 32657 Lemgo, Germany
oliver.niggemann@hs-owl.de

Abstract

This paper introduces a novel approach to Model-Based Diagnosis (MBD) for hybrid technical systems. Unlike existing approaches which normally rely on qualitative diagnosis models expressed in logic, our approach applies a learned quantitative model that is used to derive residuals. Based on these residuals a diagnosis model is generated and used for a root cause identification. The new solution has several advantages such as the easy integration of new machine learning algorithms into MBD, a seamless integration of qualitative models, and a significant speed-up of the diagnosis runtime. The paper at hand formally defines the new approach, outlines its advantages and drawbacks, and presents an evaluation with real-world use cases.

Introduction

Current trends in production lead toward intelligent modular production systems, called Cyber Physical Production System (CPPSs). Modular CPPSs are highly adaptive and can be configured in many different ways to produce a broad range of products. If such a CPPS fails, operators may not have gathered sufficient experience with the current configuration, and efficient manual diagnoses becomes harder if not impossible. An adaptive diagnosis system is considered the silver bullet to tackle a quick and reliable CPPS repair.

However, diagnosis concepts from today's production systems are not applicable in this regard: Heuristic methods, such as alarm generation, are hard to create and maintain for large, quickly changing CPPSs and require too much manual engineering effort. Model-based approaches can be used for discrete CPPSs, but the models have to be both created and tested manually after every system adaption. Additionally, the diagnosis runtime become an important issue in large systems (Stern et al. 2015). Nevertheless, the MBD approach is promising for the future needs.

Model-Based Diagnosis (MBD) was introduced by Reiter (1987) as Diagnosis from First Principles and by de Kleer and Williams (1987) as General Diagnostic Engine (GDE). It is a generic diagnosis technique which uses a system model in logic as a basis. Observations from the system are added and they are expected to stay consistent with the system model if all components function correctly. If an inconsistency between the system model and an observation is detected, assumptions about the correct behavior of components are withdrawn until all inconsistencies disappear; the withdrawn assumptions are hence called diagnoses. This approach also has limitations, especially when MBD is used for large, dynamic, and especially, hybrid CPPSs.

Challenge 1: Integration of Continuous Models.
MBD requires a system description in a suitable logic (Reiter 1987) which makes MBD predestined for discrete systems. But CPPSs are typically hybrid and combine both value discrete and value continuous variables. Therefore, models of multiple continuous variables have to be integrated into the diagnosis system: often, a change in the values of discrete variables leads to a different system behavior, called a mode change (Niggemann and Lohweg 2015). Due to a mode change, the system's models are changed and continuous signals can shift abruptly. For example, turning on a heater triggers a mode change and will increase the power consumption by entering a new mode.

Challenge 2: Creation of Diagnosis Models.
The creation of a holistic qualitative model, which is required for existing approaches, has to cover the whole system description. Especially in large CPPSs it is challenging to create such models since they are often designed manually. It requires a lot of knowledge about the system and its causalities, and only experts can create them. However, even for experts it is time consuming and costly. Since CPPSs are adaptive, the models may need to be adapted often, which is not feasible. Hence, for CPPSs the models should be learned automatically (Niggemann and Lohweg 2015).

Challenge 3: Diagnosis Runtime.
The diagnosis runtime is challenging for large-scale systems (Stern et al. 2015). The computation of the often used minimal cardinality diagnosis, a diagnosis with the smallest number of components, is *NP-hard*. I.e., algorithms have an exponential runtime with respect to the number of components (Metodi et al. 2014). To enable an efficient diagnosis for large-scale CPPSs, the search space has to be reduced.

Main idea of this paper is a Residual-Based Diagnosis Algorithm (RDA), which is sketched in Fig. 1. First, in the training phase, a model of the normal behavior of the CPPS is learned from data (top, right-hand side of the figure).

This model is comprised of two submodels: *(i)* an automaton which models the mode changes triggered by discrete variables d_i and *(ii)* a model describing the behavior of continuous variables $e_c(t)$ for each mode M_i. Note that several approaches exist to learn such models, see "Related Work".

Second, in the operating phase, observations of continuous variables $c(t)$ and discrete variables $d(t)$ (top, left-hand side) are made in the running system. For each point in time t, a static diagnosis model (bottom of the figure) is generated, which comprises a system description, the health state of the system's components and observations. The observations contain the current residuals, the current mode, and the truth values of the discrete variables.

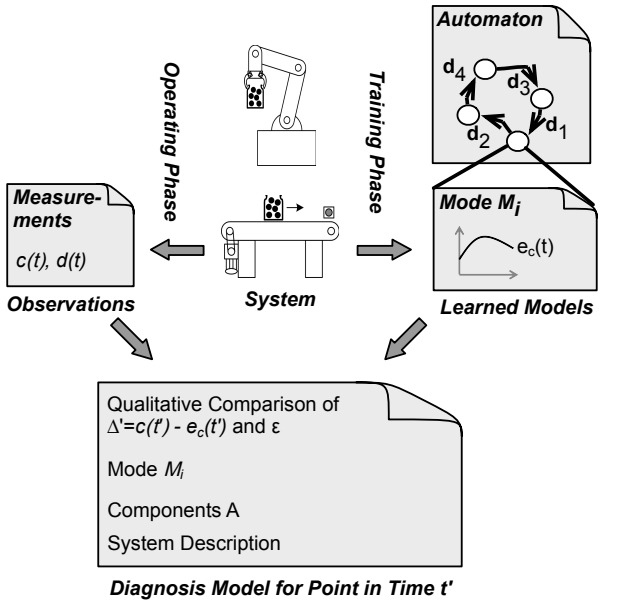

Figure 1: The main idea of the new RDA.

The contribution of this paper is an online diagnosis approach for dynamic hybrid and non-linear CPPSs that significantly extends the current state of MBD: *(i)* RDA provides a generic diagnosis approach for CPPSs that uses quantitative models which can be represented by multiple formalisms (challenge 1); *(ii)* learned models can be used for the diagnosis which reduce the modeling effort significantly (challenge 2); *(iii)* the CPPS is separated into modes which can be automatically learned by a finite automaton (challenge 1 + 2); *(iv)* RDA applies a residual-based diagnosis which reduces the diagnosis runtime, but still enables the usage of traditional MBD algorithms, so that existing high-performance algorithms can be used (challenge 3).

The paper is structured as follows: First, we will present the approach and provide an example. We then give a brief overview of related work. This is followed by an empirical evaluation which proves whether the challenges are properly addressed or not. Finally, we conclude the work and give a short overview about future topics.

Proposed Approach

In the following, RDA is presented. At first, definitions are introduced which are needed for the proposed hybrid diagnosis (challenge 1). Then a small example is used to clarify how the approach works, therein we focus on the continuous part. Finally, the algorithm is introduced which consists of a training phase (challenge 2) and an operating phase, where the latter does the optimization (challenge 3).

Integrating Continuous Models

We use the following definitions in this work, to enable an integration of continuous models into MBD.

Definition 1 *A **hybrid CPPS** is defined as a set of devices $A = \{a_i \mid i \leq n; i, n \in \mathbb{N}\}$. Each $a_i \in A$ has properties which are represented by a set of variables $V = \{v_j \mid j \leq m; j, m \in \mathbb{N}\}$, with $V = C \cup D, C \cap D = \emptyset$. Whereas C is a set of value continuous variables and D is a set of value discrete variables.*

Please note that the defined CPPS consists of a set of devices, which means that it is an automated system which always acts similar in a non-faulty case. That means there should be no human in the loop, because of the thereby injected randomness.

Definition 2 *A variable $v \in V$ is **observable** if its value can be determined at each point in time during the systems' runtime.*

There are mainly two reasons why a variable might not be observable: *(i)* it is technically not possible to determine the value, e.g. the distribution of chemicals in a reactor or *(ii)* because no proper sensor is installed, e.g. to reduce costs.

Definition 3 *The **continuous model** of a hybrid CPPS A is defined as a set of functions $R = \{e_c \mid c \in C, e_c : \mathbb{R}^+ \to \mathbb{R}, \}$. Each function e_c returns the expected value for an observable continuous variable $c \in C$ at a point in time $t \in \mathbb{R}^+$.*

There are many formalisms that can be used for the representation of continuous variables. Some of them enable the continuous models to be learned, as we will show later on.

The continuous model enables a quantitative residual Δ for every continuous variable to be determined, which is the difference between the measured value of a variable and its expected value. $\Delta_c = c(t) - e_c(t)$ at a specific time $t \in \mathbb{R}^+$ for a continuous variable $c \in C$, where $e_c(t)$ is the expected value of c at time t.

Definition 4 *A **qualitative residual** for each continuous variable $c \in C$ at each point in time $t \in \mathbb{R}^+$ can be defined by the following predicates:*

- $s^0(c)$ *iff* $|\Delta_c| \leq \varepsilon$
- $s^-(c)$ *iff* $|\Delta_c| > \varepsilon$ *and* $\Delta_c < 0$
- $s^+(c)$ *iff* $|\Delta_c| > \varepsilon$ *and* $\Delta_c > 0$

$\varepsilon \in \mathbb{R}^+$ is a threshold for the residual that is acceptable to assign the non-faulty qualitative residual s^0 to a variable c.

Definition 5 *An **ok-assumption** $ok(a)$ is a property (unary predicate) of a device $a \in A$, which indicates that the device is working correctly.*

The ok-assumption is also referred to as the *health state* of a component.

Definition 6 *A **hybrid system description** SD_H describes the systems' causalities of hybrid systems and consists of a set of first order logic (FOL) sentences, including the predicates ok-assumption and qualitative residuals, but does not include quantifiers.*

The qualitative residuals s^- and s^+ are typically modeled with a negation in the hybrid system description, since their occurrence is not expected during normal operation.

Definition 7 *The **components** $COMPS$ is a set of all devices $COMPS = A$.*

Definition 8 *The **observations** OBS is a finite set of FOL sentences that represent observations about the system.*

That means, the observations represent discrete variables and the qualitative residuals of observable continuous variables. Since we also allow non-equidistant discrete time steps, the OBS can be updated at every point in time $t \in \mathbb{R}^+$.

The presented approach can be used for some systems, but it is challenging to model complex shapes of the continuous variables, because shapes mainly dependent on the systems' mode. Therefore, we divide the system into different modes. Thus, the continuous variables have to be modeled for every mode, but the modeling per mode is much easier.

Definition 9 *A **finite automaton** for a hybrid CPPS is a 3-tuple $E = (M, \Sigma, T)$, where:*

- *M is a finite set of modes. Each mode $m \in M$ is a tuple $m = (id, \boldsymbol{u})$, where id is a unique identifier for the mode and $\boldsymbol{u}$ is a vector that represents the truth values of all discrete variables $\boldsymbol{u} = (d_0, ..., d_n)^T, d_0, ...d_n \in D$.*
- *Σ is the alphabet, the set of events, where an event is a change of the truth value of at least one discrete variable.*
- *$T \subseteq M \times \Sigma \times M$ gives the set of transitions, e.g. for a transition $\langle m, b, m' \rangle$, $m, m' \in M$ are the source and destination mode, and $b \in \Sigma$ is the trigger event. (Maier 2014)*

We use the unique identifiers of the modes as variable names in FOL to describe whether the CPPS is currently located in a certain mode or not. For example, if its variable $M1$ holds true, the CPPS is in mode $m_1 = (M1, \mathbf{u})$. The mode identifier $M1$ that represents the mode $m1$ is integrated as FOL variable $M1$ to a sentence in the hybrid system description SD_H if the FOL sentence holds true in mode $m1$. The observations OBS contain the mode identifier of the mode that holds true, at the current point in time. However, the system must not change its mode during a diagnosis session. Additionally, the whole system should be synchronous to determine modes properly. An asynchronous part of the system would cause much more modes and thus much more training data to create a suitable model. Anyway, the modes provide a context and this enables the detection of not only point anomalies but also contextual anomalies, because a specific behavior is only suitable in a certain context.

Definition 10 *The **clock** is a continuous variable $t \in \mathbb{R}^+$ that increases linear strictly monotonically and is set to zero by changing the mode $m \in M$ of the finite automaton E. In modeless systems, clock and time are the same.*

The structure of the RDA in the operational phase is shown in Fig. 2, where white boxes are components that do not differ from traditional MBD, whereas grey boxes are adapted for RDA. Arrows represent the flow of data. RDA processes the observations from the CPPS and the generic diagnosis model, which consists of an automaton, a continuous diagnosis model, the components, and a hybrid system description, to get a static diagnosis model. The static diagnosis model, which is used as input for the traditional MBD, consists of SD_m, $COMPS_m$, and OBS. The subscripted m in $COMPS_m$ and SD_m indicates that it holds true just for a single mode, so $SD_H = \bigcup_{m \in M} SD_m$ and $COMPS = \bigcup_{m \in M} COMPS_m$. We introduce the same notation for the continuous model $R = \bigcup_{m \in M} R_m$.

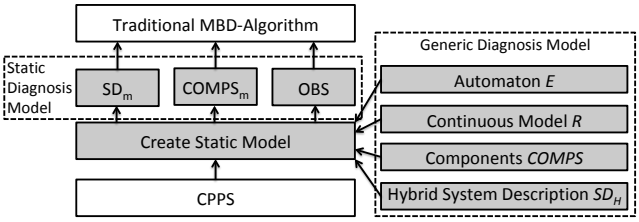

Figure 2: Structure of the RDA during operational phase.

Example

The approach is presented on a simple CPPS, shown in Fig. 3. It is a tank, where a defined volume of liquid is filled in (mode M0), a blender stirs the liquid during the heating process (mode M1) and then it is drained through a valve (mode M2). All shown components are part of the hybrid CPPS, so $l, \vartheta, b, h_1, h_2, d, k \in A$. To simplify the example, we suggest the height of the liquid x and the viscosity of the product p as variables (properties) of tank l. The system contains the following variables $v_k, v_b, v_{h_1}, v_{h_2} \in D$ and $v_x, v_p, v_\vartheta, v_d \in C$, where only v_x and v_p are not observable.

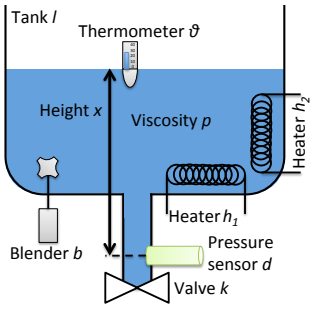

Figure 3: Example of simple CPPS.

Fig. 4 shows a finite automaton that represents the relation between different modes and one continuous model in each

mode. Only one continuous variable is represented, because it is suggested the other variables constant to simplify the example. We define for mode m_1, where $e^{m_1}_{c_\vartheta}(t)$ was identified empirically:

$$\left.\begin{array}{c} ok(b) \wedge ok(h_1) \wedge ok(h_2) \wedge M1 \rightarrow s^0(v_\vartheta) \\ ok(b) \wedge M1 \rightarrow \neg s^+(v_\vartheta) \\ ok(h_1) \wedge ok(h_2) \wedge M1 \rightarrow \neg s^-(v_\vartheta) \end{array}\right\} SD_{m_1}$$

$$\left.e^{m_1}_{c_\vartheta}(t) = 60 - 30 \cdot 2^{2.56e^{-4} \cdot t}\right\} R_{m_1}.$$

The mode includes the allocation of discrete variables to be true, e.g. for *M1:* $v_b \wedge v_{h_1} \wedge v_{h_2} \wedge \neg v_k$ is true. So, modes are explicitly modeled by the automaton and thus represents the discrete values in the system description SD_H. Since we can learn automaton, as we show later on, SD_H only contain relations between health state of components, modes and qualitative residuals, which simplify its creation significantly. However, the whole system description just represents the normal behavior, including situations that do not occur during normal operation. The qualitative residuals s^- and s^+ have a negation to indicate that the behavior is not expected. Please note that these are not strong fault models. Strong fault models also represent faulty behavior and enable a consistency, even if a component fails.

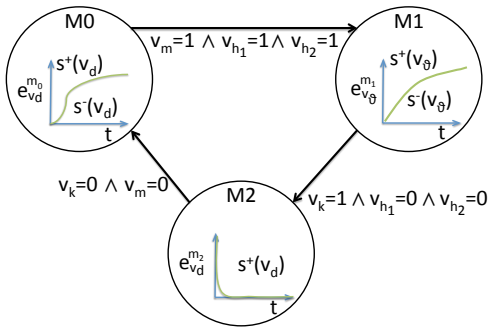

Figure 4: Automaton for the simple CPPS from Fig. 3.

In this example, we observe $OBS = \{s^+(v_\vartheta), M1, v_b \wedge v_{h_1} \wedge v_{h_2} \wedge \neg v_k\}$, since $\Delta_{c_\vartheta} > \varepsilon$. This is not consistent with the system description, it contradicts $ok(b) \wedge M1 \rightarrow \neg s^+(v_\vartheta)$ if we assume all components are working correctly. The consistency is achieved by canceling the ok-assumption for b, which is the only, as well as the correct, diagnosis.

Creating Diagnosis Models

The general (non-static) diagnosis model contains a hybrid system description SD_H, the components $COMPS$, an automaton E and a continuous model R. The SD_H only contains relations between health state of components, modes, and qualitative residuals. So, the creation of SD_H is much easier, since neither the modes nor the continuous variables have to be modeled. Before we introduce the algorithm for the training phase, we introduce some more notations. V^t refers to the measured values of variables V at specific time $t \in \mathbb{R}^+$. V^* is a set of measured values of the variables V at different points in time $V^* = \bigcup_{t \in [t_1, t_2]} V^t$ with $t_1, t_2 \in \mathbb{R}^+$, called historical data. As $V = C \cup D$ holds true, both notations can be used equivalently: $V^t = C^t \cup D^t$ and $V^* = C^* \cup D^*$. Additionally, all symbols from definitions 1 – 10 are the same as in the algorithms.

The training phase is presented in Algorithm 1. In line 1, a finite automaton is learned by using discrete variable names and historical data. The historical data have to represent the normal behavior of the CPPS without any faults. Not all variables have to be used to learn the model. There might be some variables which should not be used, such as variables controlled by pulse duration modulation, since they toggle for a long time without changing the mode in the sense of the process. Mostly there is a flag that can be used instead. All modes of the learned automaton are determined (line 2) and used to iterate over each modes (line 3) to learn the continuous model for each mode. Therefore, the continuous model R_m for mode m is initialized by an empty set (line 4). Then, the expected values e^m_c for all continuous variables c in the mode m are learned, based on historical data, and added to R_m (line 6 + 7). Finally, the continuous model of the mode R_m is added to the overall continuous model R (line 8) and the automaton and the overall continuous model are returned (line 9).

Algorithm 1: Learning Modes and Continuous Model

Input: variables $V = C \cup D$, historical data $V^* = C^* \cup D^*$
Output: finite automata E, continuous model R

```
1 E ← learnAutomaton(D, D*)
2 Modes M ← getAllModes(E)
3 forall modes m ∈ M do
4     R_m ← ∅
5     forall continuous variables c ∈ C do
6         e^m_c ← learnExpectedValues(m, c, C*)
7         R_m ← R_m ∪ {e^m_c}
8     R ← R ∪ R_m
9 return E, R
```

Learning Modes: Some learning algorithms for finite automata can be found in the literature, such as MDI (Carrasco and Oncina 1994), ALERGIA (Thollard, Dupont, and de la Higuera 2000), RTI+ (Verwer 2010), BUTLA (Maier 2015), or OTALA (Maier 2014). All these algorithms learn automata without expert knowledge.

By reviewing the possible algorithms, we identified the Online Timed Automaton Learning Algorithm (OTALA) as a good choice. OTALA is an online learning algorithm that uses the current values of discrete variables to define modes, so there are $k^{|D|}$ possible modes, where $|D|$ is the number of discrete variables and k is the possible number of discrete values, so $k = 2$ for binary values. It is possible to define criteria to decide whether the learning is completed or not, which helps to automatize the learning process and save resources. So OTALA exactly fulfills the needs of the mode separation.

Learning Continuous Models: RDA is a generic approach, which does not limit the learning of continuous variables to a specific technique. Learning methods have to be compatible to definition 3, so that they provide an expected value at every point in time and thus enable RDA to determine qualitative residuals regarding definition 4. Many methods known in machine learning fulfill these requirements.

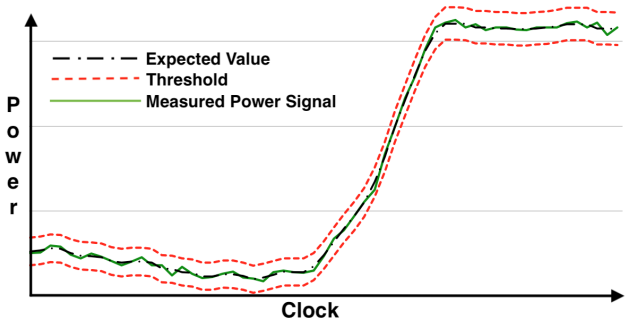

Figure 5: Measurement of the variable power with the expected value, learned with regression, and boundaries.

As an example, we use non-linear regression to represent the power consumption during the operation mode of turning on a heater, which is performed after a short reaction time. In Fig. 5, the regression model (which determines $e_c(t)$) and the threshold (ε) are shown as well as a measured signal. The model is learned with a two layer neural net with 48 neurons on the hidden layer, using the gradient descent algorithm with a learning rate of 0.001. As one can see, with this model it is possible to determine a residual Δ for every point in time. So regression is one suitable method for the representation of continuous variables in RDA.

Diagnosis Runtime

The RDA algorithm is executed during the operational phase at a specific point in time t. At this point in time a static diagnosis model is created by capturing the observations and the generic diagnosis model. Additionally, it is used to reduce the search space and thus decrease the diagnosis runtime (challenge 3). This is achieved by computing SD_m and $COMPS_m$. Therefore, all statements from SD_H are checked if they hold true in the current mode, and only those statements are added to SD_m. If they do not hold true, they are not able to cause an inconsistency during diagnosis, so they need not be considered. All statements that are added to SD_m are checked whether or not they contain a device $a \in COMPS$, which reduces the search space and is mainly responsible for the increased performance. Again, only those devices that appear in the SD_m can cause an inconsistency. All other devices do not need to be considered without loss of generality.

This computation is shown in the first part of algorithm 2. At first, the current mode is determined (line 1). Then the outputs are initialized with empty sets (line 2-4). All sentences of the hybrid system description are browsed through (line 5) and they are checked if they hold true in the current mode (line 6). If the sentence holds true, it is added to the system description for the current mode SD_m (line 7). Furthermore, all components of the sentence are extracted and added to $COMPS_m$ (line 8+9). This reduces the search space for the diagnosis algorithm.

Algorithm 2: RDA Compute Diagnoses

Input: variables V, continuous variables C, measurement V^t, finite automaton E, continuous model R, hybrid system description SD_H, components $COMPS$
Output: Diagnoses Ω

1 Mode $m \leftarrow determineMode(E, V^t)$
2 $SD_m \leftarrow \emptyset$
3 $COMPS_m \leftarrow \emptyset$
4 $OBS \leftarrow \emptyset$
// Compute SD_m and $COMPS_m$
5 **forall** $\alpha \in SD_H$ **do**
6 **if** α *holds true in* m **then**
7 $SD_m \leftarrow SD_m \cup \alpha$
8 **if** α *contains* $a \in COMPS$ **then**
9 $COMPS_m \leftarrow COMPS_m \cup a$
// Compute OBS
10 **forall** $v \in V$ **do**
11 **if** $v \in C$ **then**
12 $\Delta_v \leftarrow v(t) - e_v(t)$
13 $OBS \leftarrow OBS \cup determineResidual(\Delta_v, \varepsilon)$
14 **else**
15 **if** $v(t)$ **then**
16 $OBS \leftarrow OBS \cup v(t)$
17 **else**
18 $OBS \leftarrow OBS \cup \neg v(t)$
19 $OBS \leftarrow OBS \cup getID(m)$
20 $\Omega \leftarrow computeDiagnoses(SD_m, COMPS_m, OBS)$
21 **return** Ω

The preprocessing of observations is necessary to perform the hybrid diagnosis with traditional MBD algorithms. Two steps are needed: determine the residuals for each continuous variable and determine the current mode. Discrete variables are added directly to the OBS, since they do not require any preprocessing. Algorithm 2 shows how to compute OBS in the second part. The algorithm iterates over all variables (line 10). In line 11, it is differentiated between continuous and discrete variables. For continuous variables, the qualitative residual is calculated, the residual is determined and it is added to the observations (line 12+13). If v is a discrete variable, the variable is added to the observations that the observations are always true (line 15-18). Then, the mode identifier of the current mode is added to the observations (line 19). Finally, diagnoses Ω are computed from the static diagnosis model, by using a traditional MBD algorithm, and they are returned (line 20 + 21).

Related Work

Model-Based Diagnosis Although MBD has been well known for over 30 years, it is still an active field of research. One focus of research is to improve the algorithms' performance, since the complexity is a major challenge in MBD (Feldman, Provan, and van Gemund 2010). There are a some fast algorithms available, such as SAFARI (Feldman, Provan, and van Gemund 2010), SDA (Siddiqi and Huang 2011) or SATdB (Metodi et al. 2012). Each of these algorithms optimizes the computation of diagnoses and performs better than the previous ones, often by at least one order-of-magnitude. Jannach, Schmitz, and Shchekotykhin (2015) introduce another algorithm-based approach by parallelize hitting set computation, which utilize a multi-core computer better than the usual algorithms and reduce the computation time by up to 85 %. Since we use traditional MBD algorithms, our approach is compatible with those works and they can be used for an efficient implementation.

But not only algorithms are used to improve the performance, Guo et al. (2011) merge components with same constraints to the output set. This reduces the number of components and thus reduces the runtime of the diagnosis algorithm. The work at hand also reduces the number of components, but we use information about modes and do not search for similar constraints. The approach of Guo et at. is compatible, so it can be applied to every mode of the work in hand to increase the performance further. However, there is potential for a further increase in performance, e.g. Perdomo-Ortiz et al. (2017) applied MBD to a quantum computer.

Qualitative Reasoning The idea of qualitative reasoning (QR) is to reason like humans, since they do not calculate differential equations for inferences, but mostly use qualitative information. Many approaches can be found in the literature, such as (Kuipers 1986), (Wiley, Sammut, and Bratko 2014) or (Izmirlioglu and Erdem 2018). The approach presented in this paper does not use classical concepts of QR. The only concept we use from QR is discretization of residuals as *high/+*, *low/-* or *as expected/0*. But we use a threshold to rate them and not a sign-function as is typical for QR.

Machine Learning Many approaches can be found in literature, which focus on the learning of system models to detect anomalies, such as (von Birgelen and Niggemann 2017; Hranisavljevic, Niggemann, and Maier 2016). However, those approaches are not able to identify precise root causes. If they do, it is based on a similarity measurement between the expected and the observed values. This can be helpful in detecting a wrong sensor value, but those methods are not able to identify more complex root causes. For example, the chaining effect propagates over multiple modules before it is detected and thus they are complex to identify.

In the work at hand, machine learning is used to represent continuous variables over the time. Therefore, use nonlinear regression is used, which is realized by a neural net, regarding (Russell and Norvig 2010). But also other approaches can be used, such as deep neural networks and autoencoder (la Rosa, Yu, and Li 2016) or switched kalman filter (Nguyen and Goulet 2017). More details can be found in (Osborne et al. 2012) and (Gao, Cecati, and Ding 2015).

Hybrid Model Based Diagnosis There is a lot of work available that deals with the diagnosis of hybrid systems. Hybrid systems are mostly modeled as a combination of an underlying discrete model, where each state (we call it *mode* in this work) contains the representation of the continuous signals, such as (Biswas et al. 2003; Grastien 2014; Taktak, Triki, and Kamoun 2017; Prakash, Samantaray, and Bhattacharyya 2017). We follow this paradigm in our work.

An often used modeling formalism for those systems are temporal causal graph (TCG) or hybrid bond graphs (HBGs) which can be transformed to a TCG for the further processing. Biswas et al. (2003) use TCG to model the hybrid system. An observer is implemented as a combination of extended Kalman filter and a hybrid automaton which is used to determine residuals. The diagnosis is done in three steps: qualitative roll-back, qualitative roll-forward, and quantitative parameter estimation. The approach is demonstrated on a fuel transfer system of an aircraft. An extension of the work is represented in Narasimhan and Biswas (2007), where HBG are used for the modeling. Since the scalability of centralized diagnosis approaches is bad, Prakash, Samantaray, and Bhattacharyya (2017) extend the HBG approach to a distributed algorithm for sequentially occurring faults in hybrid dynamical systems. The advantage of the decentralized algorithm is less needed computing power. A supervisory diagnoser handles the situation if more than one algorithm detects a fault. But the presented approach requires slow dynamics. The presented approaches only detect parametric faults and they require significant manual effort, because the models have to be created by experts. So, these approaches are suitable for process industries, where discrete faults do not play a major role and where the system does not change regularly. Instead our approach is suitable for the manufacturing industry, where processes change regularly and discrete faults play a major role, e.g. a workpiece holder on a conveyor triggers a light barrier.

Voigt, Flad, and Struss (2015) introduce the application of MBD in a bottling plant by extending a traditional MBD algorithm. The authors represent the system with precise equations and transform them into a qualitative mathematical description. They used a real world application, but they use classical QR concepts for abstraction, not residuals, they are limited to total fault, i.e. system breakdown, and their environment does not change, so they create models manually.

Grastien (2014) introduces an approach for diagnosing hybrid systems with satisfiability modulo theories (SMT). He separates the system into states (*modes*), but he does not differentiate between continuous and discrete variables, they are all used to define states. The major drawback of his approach is the diagnosis runtime.

Provan (2009) introduces an approach, in which he uses modes and discretizes continuous variables, as known from qualitative reasoning. We also use modes and standard MBD algorithms, but Provan is limited to a polynomial representation of continuous variables, which are not learned and he does not optimize the diagnosis runtime.

Evaluation

This section consists of two parts: Evaluation of hybrid diagnosis (challenge 1) and of the diagnosis runtime (challenge 3). The learning of models (challenge 2) is not evaluated on its own, it is covered in the evaluation of hybrid diagnosis, since wrong models will lead to a wrong diagnosis.

Evaluation of Hybrid Diagnosis

Two CPPSs are used to evaluate the diagnosis approach. To evaluate the approach on real-world CPPSs, we made the trade-off to use a small real-world system, instead of a large artificial one. The first is a demonstration plant of a modular CPPS (called versatile production system (VPS)) in the SmartFactoryOWL that processes corn and makes popcorn from it. The CPPS consists of four modules, namely delivery, storage, dosing, and production, which can be easily adjusted regarding the current needs. In total the modules have 23 components, such as conveyor, level sensors, aspirator, heater or power meter. The components have three continuous variables (power meter, weight of corn, and weight of popcorn) and 20 binary variables (conveyor on/off, sensor on/off, etc.). A data set is used with a total of 34,973 observations, where 27,187 present the correct behavior and have been used for learning. The observations were taken with equidistant time steps of 50 ms. We used eight different arrangements with two to four modules. For example, if small batches are to be produced, the dosing module has to be used. Whereas this is not necessary if big batches are produced and the weight does not have to be so exact. In that case, the corn can be portioned by the runtime of the conveyor (VPS 6 in Fig. 6).

The OTALA algorithm was used to learn the automata which identified up to 13 modes for each arrangement. Then, the continuous models are learned within each mode. Therefore, we used non-linear regression, as described in the previous section. We set the threshold, to decide whether a signal is anomalous or not, manually.

As a second CPPS one of the well-known two-tank systems (TTS) in two different arrangements was used, because only two are possible with two tanks. Within the CPPS two tanks, pumps, values, heating, and mixing units perform a batch process, similar to the example in the previous section (see figure 3).

For evaluation, we injected 10 discrete and 10 continuous faults into each arrangement of both systems within different modes. The evaluation was successful if an inconsistency was detected and the diagnosis algorithm provided diagnoses that contain the effected component. Therefore, inconsistencies have to be detected, which require a correct continuous model and a proper threshold to detect continuous faults, and a correct conclusion of the diagnosis algorithm.

Fig. 6 shows a comparison between traditional MBD and RDA. Both approaches are able to identify all discrete faults, which were expandable since they use the same method for discrete faults. The MBD was not able to identify a continuous fault in any system. In its extended version, RDA has identified all continuous faults in the VPS correctly and

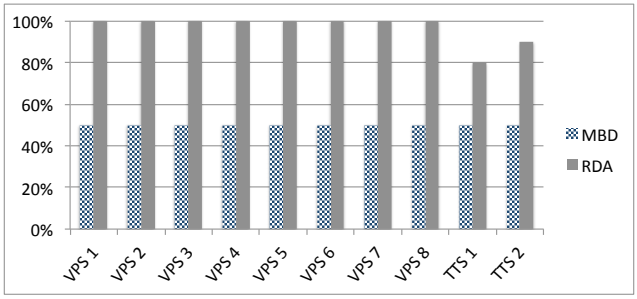

Figure 6: Comparison of MBD with RDA in CPPS.

85% of the continuous faults in the TTS. The not-detected faults are caused by undetected inconsistencies, because the residuals have been too small to reach the threshold. A lower threshold leads to more false positives, whereas a higher threshold leads to more false negatives. So, one has to make a trade-off between them, which might depend on the use case. In the examples above, we minimized the total faults. A decrease of the threshold led to an increase of the total number of faults, so it is use case specific that we did not get some faults positives. However, the root-cause for all detected faults was determined correctly. So, from a logical point of view, the diagnoses have been correct in all cases (challenge 1) and only the learned continuous models respectively the residuals have to be improved (challenge 2).

Evaluation of Diagnosis Runtime

To evaluate the diagnosis runtime, we used a MBD algorithm that we implemented on our own. There was no implementation of a high-performance algorithm, as mentioned in 'Related Work', available. To evaluate the diagnosis runtime we compared RDA with a traditional MBD approach, where both approaches use the same MBD algorithm. We only used discrete faults, so the hybrid diagnosis capability of RDA was not used. Since continuous faults are represented as predicates, as discrete faults, there is no difference in the diagnosis runtime.

We used a generated SD with a different number of components. For two components the system description contains one mode and four statements. So the proportion was constant and set to a proportion similar to the model of the demonstrator in the previous section. For evaluation an iMac with 2,5 GHz Intel Core i5-2400S, 8 GB DDR3 RAM, and MacOS Version 10.13.1 was used.

Fig. 7 presents the runtime in seconds to compute diagnoses over the number of components for the traditional MBD and for RDA. We computed one diagnosis in each mode and performed each computation 10 times to reduce side effects due to the non-real time operation system. The traditional MBD, marked by blue dots, shows an exponential correlation to $COMPS$, whereas the optimized version with RDA, marked by red squares, shows a linear correlation to $COMPS$, with a low rise. The linear correlation also holds true for a higher number of components (100 $COMPS$), which is not presented in Fig. 7. For a mode-based system, RDA performance is better than the traditional MBD, only

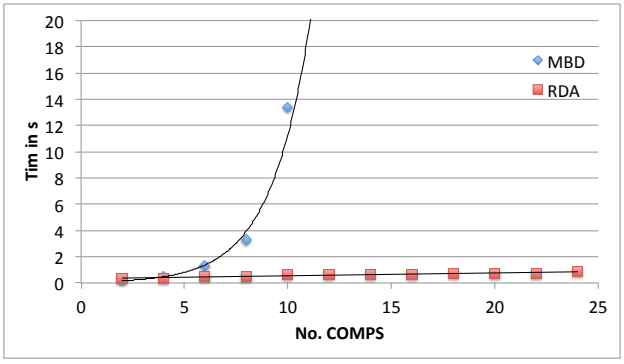

Figure 7: Runtime of RDA and a MBD. Due to representation reasons, we did not print all experiments of the MBD.

in modeless systems (such as in the 2 $COMPS$ experiment, see Fig. 7) the traditional algorithm is slightly faster. This is because, RDA tries to optimize the SD and $COMPS$, which is not possible in this case, but it still needs a short period of time. The optimization takes typically between 0,2% and 1% of the RDA runtime. We showed that RDA has a significant better performance (challenge 3) for large systems.

Conclusion

Three challenges have been identified and tackled in order to apply MBD to large hybrid CPPSs, namely, the integration of continuous models, the creation of models, and the diagnosis runtime. The novelty of this paper is a diagnosis algorithm RDA that learns quantitative models (training phase), performs a diagnosis based on residuals, and, due to an optimized runtime, is able to create diagnoses for large hybrid CPPSs (operational phase). A finite automaton is learned that covers the discrete variables, and within each state a continuous model is learned. As a result, the manual modeling effort boils down to modeling only the relations between modes, qualitative residuals, and the health state of components. Our evaluation shows that the approach is able to identify discrete and continuous faults, and that it finds the correct diagnosis in most cases. Due to its runtime optimization the approach can be used for large CPPSs; the larger the CPPS, the bigger is the advantage of RDA, compared to traditional MBD. The approach is limited with regard to complex continuous behaviors within a single mode, such as CPPS with only continuous or only rarely switching discrete variables, as shown with the two-tanks system.

One major issue for further work is the integration of reaction times and thus the aggregation of modes. Because reaction times lead to multiple modes for the same action, e.g. if a tank is drained by opening a valve, it takes a short time before the maximum level sensor will switch its value, which indicates that the level has dropped. The change of the sensor leads to a mode change, but has no influence on the behavior and thus similar modes have to be modeled multiple times. Handling these situations reduces the modeling effort additionally. Another issue is the improvement of continuous models, since the non-detected faults in the evaluation are based on an undetected deviation of the variables. The improvement can be achieved by an automatic selection of an appropriate machine learning method along with an adaptive threshold that learns the expected deviation. Up until now, the threshold is set in a manual process that relies on expert knowledge and experience, since the threshold is effected by many parameters such as model quality and use case requirements. An adaptive threshold will also enable the diagnosis with less training data, since a poor model can be used with an high threshold which decreases if more data are available for the model learning and thus the model quality increases. Another requirement of CPPSs is that they have to stay in their mode during diagnosis which might be hard to guarantee in real applications. Therefore, the algorithm could be extended to achieve a multimode diagnosis which is able to use the information from different modes. This has the advantage that the diagnoses would be smaller and thus more precise, since more observations are added.

Acknowledgments

The work was supported by the German Federal Ministry of Education and Research (BMBF) under the project "KOARCH" (funding code: 13FH007IA6).

References

Biswas, G.; Simon, G.; Mahadevan, N.; Narasimhan, S.; Ramirez, J.; and Karsai, G. 2003. A robust method for hybrid diagnosis of complex systems. *IFAC Proceedings Volumes* 36(5):1023 – 1028. 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes 2003, Washington DC, 9-11 June 1997.

Carrasco, R. C., and Oncina, J. 1994. Learning stochastic regular grammars by means of a state merging method. In *GRAMMATICAL INFERENCE AND APPLICATIONS*, 139–152. Springer-Verlag.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

Feldman, A.; Provan, G. M.; and van Gemund, A. J. C. 2010. Approximate model-based diagnosis using greedy stochastic search. *J. Artif. Intell. Res.* 38:371–413.

Gao, Z.; Cecati, C.; and Ding, S. X. 2015. A survey of fault diagnosis and fault-tolerant techniques – part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics* 62(6):3757–3767.

Grastien, A. 2014. Diagnosis of hybrid systems with SMT: opportunities and challenges. In *ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, 405–410. IOS Press.

Guo, T.; Li, Z.; Guo, R.; and Zhu, X. 2011. Large scale diagnosis using associations between system outputs and components. In *AAAI Conference on Artificial Intelligence*.

Hranisavljevic, N.; Niggemann, O.; and Maier, A. 2016. A novel anomaly detection algorithm for hybrid production systems based on deep learning and timed automata. In *International Workshop on the Principles of Diagnosis (DX)*.

Izmirlioglu, Y., and Erdem, E. 2018. Qualitative reasoning about cardinal directions using answer set programming. In *AAAI Conference on Artificial Intelligence*.

Jannach, D.; Schmitz, T.; and Shchekotykhin, K. 2015. Parallelized hitting set computation for model-based diagnosis. In *AAAI Conference on Artificial Intelligence*.

Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence* 29(3):289 – 338.

la Rosa, E. D.; Yu, W.; and Li, X. 2016. Nonlinear system modeling with deep neural networks and autoencoders algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2157–2162.

Maier, A. 2014. Online passive learning of timed automata for cyber-physical production systems. In *IEEE International Conference on Industrial Informatics (INDIN)*. Porto Alegre, Brazil.

Maier, A. 2015. *Identification of Timed Behavior Models for Diagnosis in Production Systems*. Ph.D. Dissertation, University of Paderborn.

Metodi, A.; Stern, R.; Kalech, M.; and Codish, M. 2012. Compiling model-based diagnosis to boolean satisfaction. In *AAAI Conference on Artificial Intelligence*.

Metodi, A.; Stern, R.; Kalech, M.; and Codish, M. 2014. A novel sat-based approach to model based diagnosis. *J. Artif. Int. Res.* 51(1):377–411.

Narasimhan, S., and Biswas, G. 2007. Model-based diagnosis of hybrid systems. *IEEE Transactions on systems, manufacturing and Cybernetics* 37:348–361.

Nguyen, L. H., and Goulet, J.-A. 2017. Anomaly detection with the switching kalman filter for structural health monitoring. *Structural Control and Health Monitoring* 25(4).

Niggemann, O., and Lohweg, V. 2015. On the Diagnosis of Cyber-Physical Production Systems - State-of-the-Art and Research Agenda. In *AAAI Conference on Artificial Intelligence*.

Osborne, M.; Garnett, R.; Swersky, K.; and de Freitas, N. 2012. Prediction and fault detection of environmental signals with uncharacterised faults. In *AAAI Conference on Artificial Intelligence*.

Perdomo-Ortiz, A.; Feldman, A.; Ozaeta, A.; Isakov, S. V.; Zhu, Z.; O'Gorman, B.; Katzgraber, H. G.; Diedrich, A.; Neven, H.; de Kleer, J.; Lackey, B.; and Biswas, R. 2017. On the readiness of quantum optimization machines for industrial applications. *ArXiv e-prints*.

Prakash, O.; Samantaray, A. K.; and Bhattacharyya, R. 2017. Model-based diagnosis of multiple faults in hybrid dynamical systems with dynamically updated parameters. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Provan, G. 2009. Model abstraction for diagnosing hybrid systems. In *International Workshop on Principles of Diagnosis (DX)*.

Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artificial Intelligence* 32(1):97–130.

Russell, S., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach*. Series in Artificial Intelligence. Upper Saddle River, NJ: Prentice Hall, third edition.

Siddiqi, S., and Huang, J. 2011. Sequential diagnosis by abstraction. *J. Artif. Int. Res.* 41(2):329–365.

Stern, R.; Kalech, M.; Rogov, S.; and Feldman, A. 2015. How many diagnoses do we need? In *AAAI Conference on Artificial Intelligence*.

Taktak, M.; Triki, S.; and Kamoun, A. 2017. Real time algorithm based on time series data abstraction and hybrid bond graph model for diagnosis of switched system. *Engineering Applications of Artificial Intelligence* 59:51 – 72.

Thollard, F.; Dupont, P.; and de la Higuera, C. 2000. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *International Conf. on Machine Learning*, 975–982. Morgan Kaufmann.

Verwer, S. 2010. *Efficient Identification of Timed Automata: Theory and Practice*. Ph.D. Dissertation, Delft University of Technology.

Voigt, T.; Flad, S.; and Struss, P. 2015. Model-based fault localization in bottling plants. *Advanced Engineering Informatics* 29(1):101–114.

von Birgelen, A., and Niggemann, O. 2017. Using self-organizing maps to learn hybrid timed automata in absence of discrete events. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

Wiley, T.; Sammut, C.; and Bratko, I. 2014. Qualitative planning with quantitative constraints for online learning of robotic behaviours. In *AAAI Conference on Artificial Intelligence*.