Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A concept for fault diagnosis combining Case-Based Reasoning with topological system models

Jonas Zinn * **Birgit Vogel-Heuser** * **Felix Ocker** *

* *Institute of Automation and Information Systems, Technical University of Munich, Munich, Germany (e-mail: jonas.zinn@tum.de, vogel-heuser@tum.de, felix.ocker@tum.de).*

Abstract: Automated failure recovery plays an important role in improving Overall Equipment Effectiveness and is a building block of industry 4.0. However, in an increasingly dynamic market, failure recovery mechanisms need to be able to adapt to system changes. Starting with fault diagnosis in automated Production Systems for assembly and logistics, this paper proposes a novel approach to combining Model-based Reasoning on topological system models with Case-based Reasoning. The topological models are leveraged for case adaption, which significantly reduces the engineering effort of adding new fault types to the system, compared to signal-based methods. Furthermore, the approach does not rely on complete fault models existing in advance; thus, the case database can be continuously built up during operation.

Keywords: Fault diagnosis, Expert Systems, Case-based Reasoning, Model-driven Reasoning, Model-driven Engineering, AutomationML

1. INTRODUCTION

In order to stay competitive in an increasingly dynamic market, companies need to constantly adapt their production systems (Westkämper, 2003). As a result, traditional approaches to hard-code system capabilities within automation software lead to high engineering effort (Vogel-Heuser et al., 2016). Automation software needs to be able to adapt to new situations and requirements. This new paradigm also applies to failure recovery mechanisms, which are an important building block of industry 4.0. These mechanisms shorten stoppages and thereby increase the Overall Equipment Effectiveness (OEE). New approaches need to be developed that do not rely on hard coded recovery routines or the previous existence of complete behavioral models or fault trees.

Focusing on automated Production Systems (aPSs) for assembly and logistics, one approach to increasing the flexibility of fault recovery relies on leveraging the high number of similar components or component types used in these systems (e.g., pneumatic cylinders, linear actuators). As a process for solving new problems based on the solutions of similar past problems, Case-based Reasoning (CBR) can leverage these similarities. Similarly to experts who draw their knowledge from past experiences, CBR leverages a database of previously occurred fault cases. This method therefore allows the easy integration of existing expert knowledge. While CBR has been extensively researched in the past, generalizing from stored cases to new faults, i.e., case adaptation, remains a challenge.

Model-Driven Engineering (MDE) is another approach aiming to simplify the development process of automation software by leveraging abstract knowledge representations such as topological system models. Due to past efforts to standardize the exchange of these models across companies, e.g., through AutomationML (AML), topological data on aPS is becoming more widely available.

As the main contribution, this paper therefore proposes the combination of Model-based Reasoning (MBR) on topological system models and CBR for fault diagnosis in aPS for assembly and logistics. It is shown that the combination of both approaches overcomes the challenge of case adaptation and reduces the effort of adding new fault types to failure recovery systems. Furthermore, the case database can be built up during operation and only topological system models are required in advance.

The remainder of this paper is structured as follows: in section 2, existing literature on failure recovery and CBR is reviewed to identify the research gap to be addressed. Section 3 introduces a failure recovery use case that is utilized for the description (section 4) and evaluation (section 5) of the proposed approach. Finally, section 6 concludes this paper and suggests directions for future research.

2. STATE OF RESEARCH AND CONTRIBUTION OF THIS PAPER

This section provides an overview of the most important approaches in the field of failure recovery using the terminology defined by Isermann and Balle (1997). Furthermore, the concept of CBR is introduced, and various studies on CBR for fault diagnosis are reviewed. Based on this review, a research gap is identified which is addressed within this paper.

2.1 Existing approaches for fault diagnosis

The main methods used in the field of fault diagnosis have been reviewed by several authors (Venkatasubramanian

Table 1. Overview of existing work on CBR for fault diagnosis in aPS

Reference	Method	Application area	Adaption	Model type
Bregón et al. (2005)	CBR	General process industries	No	-
Berenji and Wang (2006)	CBR with NN[1] and PCA[2]	General process industries	No	-
Tsai (2009)	CBR	Injection molding	No	-
Nasiri and Khosravani (2019)	CBR with Fuzzy Logic	Injection molding	No	-
Feret and Glasgow (1997)	MBR with CBR	Robotic systems	Yes	Fault model
Portinale et al. (2004)	MBR with CBR	General industries	Yes	Behavioral mode
O'Farrill et al. (2005)	MBR with CBR	Electronics manufacturing	No	Test model

[1] Neural Networks [2] Principal Component Analysis

et al., 2003c,a,b; Gao et al., 2015b,a; Bayar et al., 2015) and can be categorized into model-based, signal-based, and knowledge- or process history-based approaches.
Model-based fault diagnosis approaches rely on the availability of models describing the system behavior. For discrete event systems which describe many automated assembly and logistics systems, Automata and Petri-nets are frequently used model types. However, in practice these models are often unavailable. Besides, model-based methods require the explicit design of fault diagnosis algorithms for each fault to be detected. While signal-based methods do not rely on the existence of system models, they also require the definition of signal patterns for each fault. As shown by Vogel-Heuser et al. (2016), implementing diagnosis schemes or signal patterns requires high software engineering effort.
In contrast to model-based and signal-based methods, knowledge-based approaches detect and diagnose faults from historical data instead of using process models or signal patterns. The application of machine learning methods has led to promising results in the area of quantitative knowledge-based fault diagnosis (Lei et al., 2020). However, these types of methods require large amounts of training data, which are often not available, and do not leverage existing qualitative information such as topological models or expert knowledge. In contrast, qualitative knowledge-based fault diagnosis methods, e.g., expert systems, completely rely on these qualitative data. Expert systems can be further categorized depending on the reasoning concepts used. Rule-based Reasoning and Case-based Reasoning are often used (Prentzas and Hatzilygeroudis, 2007).
Since every method has certain advantages and disadvantages, research has lately focused on hybrid methods which combine various approaches.

2.2 Case-based Reasoning for fault diagnosis

The idea behind CBR can be described in a process model consisting of the steps Retrieve, Reuse, Revise, and Retain (Aamodt and Plaza, 1994). Starting with a given problem, the aim of the Retrieve phase is to extract one or several similar cases from a case database. During the Reuse phase, the retrieved cases are adapted in order to fit the problem. The solution of the adapted case is verified in the Revise phase by applying it to the real world. Corrections are then made by domain experts. In the final Retain phase, the new solution is added to the case database.
One of the advantages of using CBR for fault diagnosis is its ability to reason based on existing knowledge, thereby leveraging the similarities between present and known past faults. Especially in systems with a large number of similar components and modules, fault patterns show commonalities that can be leveraged with this approach. Furthermore, the similarity between CBR and the human reasoning process (experts also rely on past experiences when solving complex problems) simplifies the integration of expert knowledge into CBR-based systems (Mitra and Basak, 2005). This similar reasoning process additionally ensures that the diagnoses generated can be easily comprehended by domain experts. In terms of real-world implementation, CBR has the advantage that it can reason with incomplete and imprecise data. One of the major disadvantages of CBR is the high complexity of the case adaptation in the Reuse phase. While some authors address this challenge by combining CBR with other fault diagnosis approaches, many proposed systems completely omit case adaptation (Mitra and Basak, 2005).
Table 1 provides an overview of recent work on CBR for fault diagnosis, categorized by methods used, application areas, use of adaptation, and model-type (in case of combination with MBR). The combination of CBR with MBR is most often discussed in here. Feret and Glasgow (1997) use CBR to increase or lower the confidence in MBR results as well as to suggest diagnoses that might have been overlooked when using the model-based method. Their approach relies on detailed fault models, and is able to generalize cases to all components with a similar component type. CBR has also been used to mitigate the effects of model errors in MBR systems. In their approach, Portinale et al. (2004) rely on CBR to accelerate the MBR process by leveraging existing knowledge. The authors use CBR for diagnosis tasks that are hard to solve with MBR, thereby reducing computation time. For the adaptation of cases retrieved from the knowledge base, the same behavioral model that is also used for MBR is leveraged. Another work by O'Farrill et al. (2005) extends an MBR system for the diagnosis of circuit board faults with CBR in order to enable the learning of new fault types without requiring any adaptation of the underlying model. The authors use detailed test models and do not adapt retrieved cases.

2.3 Research gap and contribution of this work

The review of the existing literature on CBR systems for fault diagnosis shows several limitations:

- Most existing work does not explicitly focus on aPS for assembly and logistics.

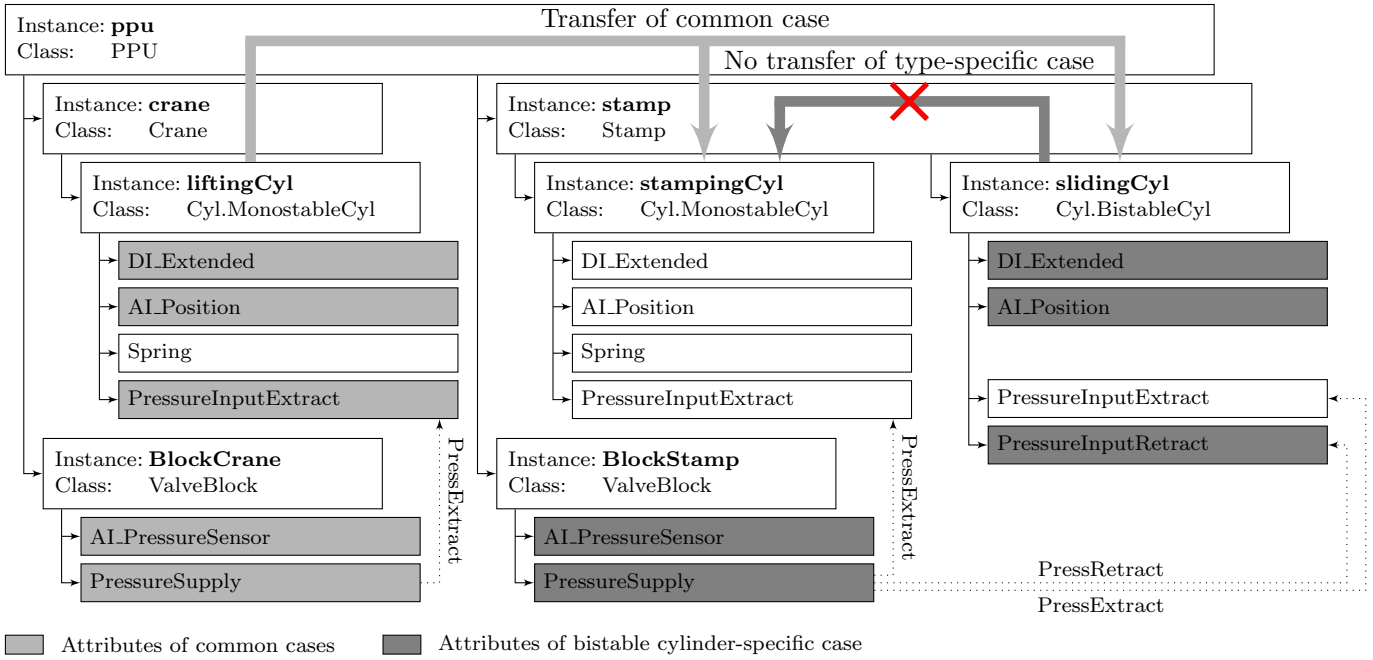

Fig. 1. Topological model of the Pick and Place Unit (PPU), and visualization of the case transfer across component types. Only fault cases including attributes common to all cylinder types are transferred.

- Only a few suggested systems are able to automatically adapt cases retrieved.
- In systems combining CBR with MBR, the former is mostly used as an extension to compensate for the limitations of the MBR approach.
- MBR-based approaches rely on detailed test models, fault models, or behavioral system models that are often unavailable in practice.

Therefore, the following part of this paper proposes a novel approach to integrating CBR and MBR for fault diagnosis in aPS for assembly and logistics. In contrast to existing research, this approach uses CBR as the main method, extended by MBR for the automated adaptation of retrieved cases. Furthermore, it relies purely on topological system models. In contrast to behavioral data or other models such as testing models or fault trees, topological data is more widely available in industry. Without loss of generality, the topological model used for the evaluation of the approach suggested in this paper is provided via AML, which is a standard engineering data exchange format (International Electrotechnical Commission, 2015).

3. USE CASE FOR EVALUATION

As a use case for the evaluation of the suggested approach, the PPU, a research demonstrator, is chosen (Vogel-Heuser et al., 2014). The system resembles an aPS with a focus on assembly and logistics tasks. The PPU consists of various modules, i.e., a stack, a crane, a stamp, and a conveyor belt, that has been built with industrial components. The sensors and actuators (e.g., inductive sensors, pneumatic cylinders) used are part of many industrial aPS. Furthermore, similar sensors and actuators are used within the modules; for example, the PPU contains a total of 6 monostable and 1 bistable cylinders. The system therefore provides a good use case for evaluating CBR approaches for fault diagnosis. Additionally, the PPU has been used as an evaluation example in various other works (Legat et al., 2013; Bareiss et al., 2016).
In order to use the PPU within this work, the existing SysML models have been extended and converted to AML. Fig. 1 includes an extract of the topological model for the pneumatic cylinders, used within the crane and stamp module, as well as for their pressure supply.
A common failure of these pneumatic cylinders is the failure to extend or retract. The underlying faults causing a failed extension can be leakage in the pneumatic system, a defective compressor, or a mechanical blockage of the cylinder. The causes of a failed retraction vary between monostable and bistable cylinders. For the bistable cylinder, they are similar to those for the failed extension while for the monostable cylinder, only the mechanical blockage and a breakage of the retraction spring applies. Further failures occur if an extension or retraction of both cylinder types is not detected due to defective end position sensors.

4. AN APPROACH TO COMBINING CBR WITH MBR

This section describes the suggested approach for fault diagnosis, which combines MBR on topological system models with CBR. After introducing the general system architecture, an algorithm for retrieving and adapting similar cases from the case database is developed.

4.1 Architecture

The proposed approach for fault diagnosis builds on a fault detection scheme which utilizes operational states with pre- and postconditions (Schütz et al., 2012; Priego et al., 2015). One implementation of this operational states concept for failure recovery has been suggested by Bareiss et al. (2016). In their architecture, pre- and postconditions

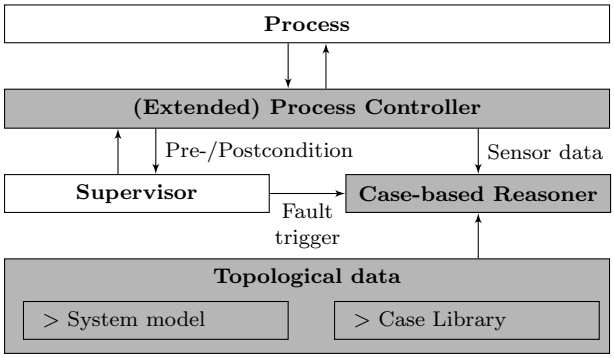

Fig. 2. Extension of architecture from Bareiss et al. (2016) (Extended parts are shaded grey)

are passed from the process controller to a supervisor, which checks whether the conditions have been fulfilled and triggers a diagnosis and recovery module in case a fault is detected. Faults can be detected by comparing the current execution time of an operation to the average execution time. If the current execution time exceeds the average by a defined threshold, meaning an error has been detected, a fault must have occurred. Pre- and postconditions are usually directly linked to sensor signals; e.g., the post-condition for the extension of a pneumatic cylinder could be the activation of the respective end-position sensor *DI_Extended*.

In order to utilize CBR for fault diagnosis, the architecture suggested by Bareiss et al. (2016) is extended. A Case-based Reasoner is used for the Diagnosis Module and an interface is added to access topological system models and historical fault cases. In order to use high-level programming languages for the implementation of the Case-based Reasoner, it is, without loss of generality, implemented on the same hardware as the Supervisor instead of including it in the (Extended) Process Controller. Fig. 2 provides an overview of the extended architecture.

The main premise for the design of the case library is to minimize the effort needed for the creation of new cases. Therefore, cases are purely constructed of attributes that are necessary to unambiguously describe the respective fault case, as well as the conditions for the sensor and control values. Attributes used encompass sensor values, the control values of actuators, and the physical properties of structural components. The shaded attributes in Fig. 1 are examples. The hierarchy of the attributes within the case must be similar to the attribute's hierarchy in the topological system model. For example, the sensor value of the crane module would be added to *case1* in the case library as *case1.ppu.crane.liftingCyl.DI_Extended*. While this seems to be tedious, all information is available in the system's topological model. Thus, the operation can be easily implemented as a copy-and-paste procedure of the respective attributes from the system model into the case. Then, the conditions for the case-describing sensor and control values need to be added manually to the attributes. While cases can be stored in any format that is able to represent topological data, AML is used for the evaluation of the approach suggested in this paper. Each case is represented as an InternalElement at the top level within a special InstanceHierarchy functioning as a case library. The conditions for the sensor and actuator values are added as constraints to the attribute definitions.

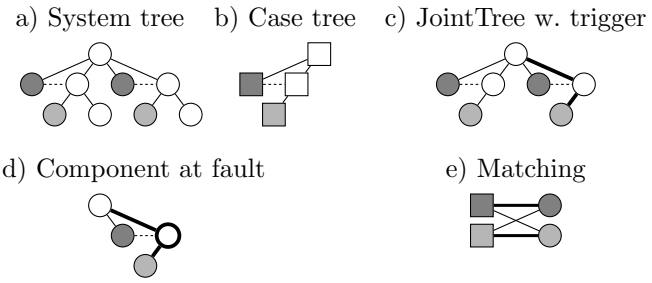

Fig. 3. Illustration of graph operations used by the proposed algorithm

4.2 Algorithm for retrieving and adapting similar cases

In contrast to most CBR systems discussed in section 2, the approach suggested not only adapts the best matching case but all retrieved ones. The similarity of the adapted cases is then compared, and the one that matches best is selected. The extend to which the cases are similar is calculated based on attribute names, attribute values, and context. The instance names of the attributes that describe a fault need to match the ones from the similar case and the attribute values need to be in the specified range. A similar context is defined by matching instance and class names of parent components as well as internal links. While similarities between parent components help to identify similar modules (e.g., cylinder types), the internal links allow the identification of similar module groups (e.g., cylinders with connected valve blocks). This similarity definition allows to identify components and modules independent of their position in the system topology, and can be seen as an extension of the one used by Wolfenstetter et al. (2018).

The algorithm for retrieval and adaptation of cases relies on the fact that the topological system model and the fault cases can be represented as trees. Fig. 1 provides an extract of the system model of the research demonstrator, described in section 3, in the tree representation. While the root of the tree is the complete system (e.g., the PPU), the leaves represent its attributes (i.e., its sensor values, control values, or properties of structural components). When considering internal links between interfaces of components, e.g., the pressure supply for the cylinders as illustrated by dotted lines, nodes of different branches become connected, and the representation becomes a directed graph. While the link is created between the interfaces of components, the connection within the graph is directly created between the components, e.g., the *liftingCyl* connects to the *BlockCrane*. This complies with the representation of internal links in AML.

The algorithm can be divided into four main steps. While steps 1-3 focus on the retrieval and adaptation of cases, as well as the calculation of their similarity, and are therefore executed for each case in the case library, step 4 covers the selection of the best matching case. The following part of this section describes each step in detail using the PPU as an example. Additionally, graph operations are explained based on the illustrations provided by Fig. 3. Fig. 4 presents the complete algorithm in pseudo code.

```
1  For Case in CaseLibrary:
2     Find instance names of case tree leaves within system tree leaves, and copy respective
        branches into new JointTree
3     If JointTree contains FaultTrigger:
4        For TriggerNode in FaultTrigger (from leaf):
5           Get all Leaves of TriggerNode in JointTree considering InternalLinks
6              If leaves from case are part of Leaves:
7                 Set FailedNode = TriggerNode and exit loop
8     Calculate similarity between branches from case tree and branches from system tree
        containing the FailedNode, and build pairs using maximum weight matching
9     Compare sensor and control values with conditions from case for pairs, and calculate
        share of matching conditions
10 Suggest case with highest matching and FailedNode that is closest to leaf
```

Fig. 4. Pseudo-code of the algorithm suggested for retrieving and adapting similar cases

1) Retrieval of a case (line 2-3) In the first step, the algorithm selects a single case from the case library and extracts the relevant elements for this case from the topological model. To start with, it identifies all attributes of the case within the system tree using the instance name as matching criteria. The branches containing the matching attributes are added to a new tree named *JointTree* (line 2 in Fig. 1). Fig. 3a-c illustrates this process. The *JointTree* in Fig. 3c only includes the branches of the system tree (3a) the leaves of which are part of the case tree (3b), which are light or dark grey.
Next, the sensor signal for which the post-condition has not been fullfilled, i.e., the fault trigger, needs to be checked to determine whether it exists within the *JointTree* (line 3). The trigger of a fault detected by the nonactivated end-position switch *DI_Extended*, which is part of the *crane's liftingCyl*, would look like *ppu.crane .liftingCyl.DI_Extended*. If the fault trigger exists in the *JointTree*, the fault case is considered to be retrieved and the algorithm continues with its adaptation. In Fig. 3c the fault trigger is illustrated by the bold branch.

2) Adaptation of the retrieved case (line 4-7) A case adaptation is generated by identifying the component of the fault trigger, the children of which (including all grandchildren) encompass all attributes of the case. Therefore, the algorithm compares those attributes that are part of the subtree of each fault trigger component, starting with the one closest to the leaves, with the ones from the case (lines 4-6). For the comparison of attributes, internal links between components are considered, i.e., the *JointTree* is transformed into a directed graph. In the research demonstrator, the attributes of the *liftingCyl* therefore also encompass the *AI_PressureSensor*. This approach allows the diagnosis of faults, the features of which include attributes of other components. For example, the fault of a cylinder that does not extend due to leakage in the pneumatic system can only be diagnosed by considering the sensors of the cylinder itself as well as the sensors of the valve block. The first component, i.e., the one closest to the leaves, that encompasses all attributes is then selected (line 7). This component can be seen as the best choice for the failed component when considering the selected case. In Fig. 3d, a bold outline highlights the failed component that has been identified. It is the first node in the fault trigger (bold branch) that has the case attributes (grey nodes) as children, when also considering the links (dashed connections). The failed component's subtree represents the adapted case.

3) Similarity calculation for the adapted case (line 8-9) In the third step, the values of the attributes from the adapted case at failure time are compared with the conditions from the original case taken from the case database. First, the attributes of the adapted case are matched to the attributes of the original case using maximum weight matching (Galil, 1986) (line 8 in Fig. 4). An undirected graph is built by connecting each attribute of the adapted case with each attribute of the original case. A weight on each connection represents the similarity between the instance names, classes, and internal links of the two connected attributes and their branches. For the instance names and classes, the maximum number of connected similar nodes, starting from the attributes (i.e., the leaf of the branch), N_i and N_c is calculated. For the internal links, the total number of nodes in a branch with similar internal links N_l is determined. For example, if the instance names of node one, two, and four match, N_i is set to 2. If the internal links match for node 1 and 3, N_l is also set to two. The weight is then defined as

$$w = N_i + N_c + aN_l, \tag{1}$$

where a is a large number, e.g., $a = 1000$. This weight definition ensures that nonmatching internal links cannot be compensated by matching instance names or classes. Matching internal links, i.e., the connection to the same type of pressure source, provide a stronger indication for similarity than the existence of identical parent modules. Fig. 3e illustrates the undirected graph for the matching algorithm, which consists of the attributes of the illustrations 3b and 3d. The bold lines highlight the pairs that are identified via the maximum weight matching.
For each resulting attribute pair that describes a sensor or control value, the value at the time of failure is compared to the conditions from the case (line 9). The similarity is then calculated by dividing the number of matching values by the total number of sensor and control values.

4) Selection of the most similar case (line 10) After executing the previous steps for each case in the case library, the case with the closest sensor and control value match (1 for an exact match) is suggested. In case of multiple matches, the case with the failed component that is closest to the leaves is chosen. Multiple matching can occur since the algorithm compensates for mismatching attributes of dissimilar cases by suggesting a failed com-

ponent on a higher hierarchy level. In an extreme case, the root is selected as the component at failure since it always encompasses all case attributes. This is also illustrated in Fig. 3d, where not only the highlighted node but also the root has all the grey attributes as (grand-)children.

5. EVALUATION BASED ON THE USE CASE

This section focuses on evaluating the proposed approach using the research demonstrator described in section 3. First, the approach is assessed for its ability to transfer fault types across component instances and component types. Next, the suggested system and signal-based fault diagnosis methods are compared to determine the effort of adding additional fault types for diagnosis. Finally, the approach's transferability and scalability are discussed.
For evaluation purposes, the system model and case library are implemented in AML (CAEX version 2.15) using the AML editor. The Case-based Reasoner executing the algorithm described in Fig. 4 is implemented in Python. The fault trigger is provided as a string while the sensor values are read from an external Java Script Object Notation (JSON) file.

5.1 Transfer across component instances and types

This section evaluates the transfer of faults across component instances and types. Therefore, it validates the algorithm's ability to differentiate between common faults, which are transferable to all cylinder types, and type-specific faults. As described in section 3, the demonstrator contains monostable and bistable cylinders. This evaluation focuses on the four common faults, which need to be diagnosed for both cylinder types, and the two faults that only apply to the bistable cylinder. The attributes required to build the cases for the common and bistable cylinder-specific faults are shown in the topological model of the PPU provided by Fig. 1.
The common faults, i.e, a failure to extend or retract due to a mechanical blockage, as well as a failed extension due to a defective compressor or a leakage in the pneumatic system, are characterized by the sensor signals *DI_Extended*, *AI_Position*, *AI_PressureSensor*, and the interface *PressureInputExtract*, which are part of all monostable and bistable cylinders. Thus, one case per fault is sufficient to diagnose these faults across the PPU.
For the two faults that are specific to the bistable cylinder, i.e., a failed retraction due to a defect compressor or a leakage in the pneumatic system, two additional cases are necessary. These cases include the interface *PressureInputRetract*, which only exists for the bistable cylinder. Thus, the case is not transferred to the monostable cylinders.
The evaluation shows that the algorithm is able to differentiate between common and type-specific faults. Common faults are successfully transferred across component types.

5.2 Software development effort

Next, the proposed approach is assessed for its ability to reduce the software development effort of adding additional fault types for diagnosis. CBR and the traditional approach of diagnosing faults through signal-based methods within the control code as implemented by Vogel-Heuser et al. (2016) are here compared to determine the effort of implementing one additional fault type. For the evaluation, two faults are considered. The first fault is the undetected cylinder extension due to a defective end position switch, which is common to all cylinder types. The second fault, a failure to retract due to a defective spring, only applies to monostable cylinders.

Table 2. Comparison of the number of added cases, adjusted functions, or adjusted function blocks to add new fault types for diagnosis

Approach	**Common fault**	**Monostable cylinder-specific fault**
Signal-based (non-modular)	6	5
Signal-based (modular)	2	1
Case-based Reasoning	1	1

Utilizing the proposed approach, both additional faults can be implemented for the six cylinders and the five monostable cylinders, respectively, by adding one case for each fault to the case library. Assuming that no modular control code design is used, a traditional signal-based diagnosis within the controller requires one function to be adapted for each cylinder. For the PPU, these are six functions for the common faults and five functions for the monostable cylinder-specific fault. In case of a modular control code design, the adaptation effort scales with the number of component types for which the fault needs to be diagnosed. The common fault therefore requires the adaptation of two function blocks, while only one function block needs to be adapted for the second fault. A summary of this comparison is provided by table 2. Due to the strong differences in the implementation of the diagnosis schemes, i.e., the creation of a case in AML versus the adaptation of control code, a more granular comparison (e.g., of added, changed, removed lines of code) is not feasible.
The comparison shows that the approach suggested strongly reduces the implementation effort, compared to traditional approaches which lack a modular control code design, if systems include a large number of similar components. As shown by Fischer et al. (2018), non-modular control code design is still widely applied in practice. In case of a modular design, CBR also provides an advantage if faults occur across different component types. Component specific faults show no advantage compared to the traditional approach with modular code design.

5.3 Transferability and Scalability

This section discusses the transferability and scalability of the proposed approach. It also differentiates between transferability to other components and fault types, as well as transferability from the research demonstrator to industrial use cases. In terms of scalability, the time complexity of the suggested algorithm as well as the manageability of the case library is discussed.

Transferability In order to diagnose not only the faults of pneumatic cylinders within the PPU but of all components used in industrial aPS, the proposed approach must be able to uniquely identify any component type within the system model, as well as be able to differentiate fault types for a given component. To ensure this, the proposed approach relies on characteristic attributes which encompass

sensor values, control values, and the physical properties of structural components to identify similarities between occurred faults and historical cases. Any physical component can be uniquely described by its physical properties and therefore identified by the proposed approach, as long as the topological model provides sufficient details. All available sensor and control values of the failed component as well as all other components (through internal links) can be included in the fault cases. Thus, the approach is able to uniquely diagnose any fault that can be diagnosed with the available sensor and actuator values of the system. The approach is therefore not limited to pneumatic cylinders and can be applied to all components within aPS.
Besides, the proposed approach should allow for cost efficient integration into industrial environments. On the one hand, the approach suggested does not require a specific format for the topological data and, therefore, allows the use of existing models, e.g., models stored in Product Lifecycle Management (PLM) systems. On the other hand, the support for AML as a standardized engineering data exchange format limits the effort needed to integrate the approach into industrial tool chains. The case database can be built up during operation, since CBR is able to reason based on incomplete information. In contrast to many MBR approaches, no complete fault model is required in advance. However, if partial or complete fault models are available, they can be easily converted into cases.
Thus, the proposed system should transfer well from the research demonstrator to industrial use cases.

Scalability To evaluate whether the execution time of the algorithm suggested and, thus, the system's ability to diagnose faults in "real time" is a limitation in terms scalability, the algorithm's time complexity is analyzed. The most complex operations within the algorithm include the creation of the *JointTree* as well as maximum weight matching. The creation of the *JointTree* has a complexity of $O(NM)$, where N is number of nodes in the system tree and M the number of nodes in the case. For the worst case, where the case includes the whole system, the complexity becomes $O(N^2)$. The worst case complexity of the maximum weight matching algorithm used in the prototype is $O(N^{1.5})$, where N is the number of nodes (i.e., modules, components, and attributes) in the system tree (Galil, 1986). While the execution time for the retrieval and adaptation of a single case scales with the size of the system model and case, the overall executing time also scales with the number of cases in the case database. However, once created, the *JointTree* for each case can be stored in memory in order to avoid the need to recreate it during each diagnosis activity. Thus, for the worst case, that the *JointTree* of each case in the case database includes the fault trigger, the complexity of the maximum weight matching algorithm $O(N^{1.5})$ is scaled with the number of cases in the database. However, steps 2 and 3 of the algorithm, which include maximum weight matching, are only executed for the retrieved cases. In practice, it is not expected that every case will be retrieved during a single diagnosis activity. Thus, the only operation that is expected to scale with the whole number of cases in the database is the check that determines whether the *JointTree* includes the fault trigger, which has a complexity of $O(n)$. Considering these remarks, and the fact that fault diagnosis is often a manual operation, execution time does not limit the approach's scalability.
Besides, the creation of new cases, as well as the management of existing cases, needs to remain simple even for large system models and case libraries. The use of topological models with hierarchical structure assures, that it is possible to have an overview of large system models. Fault cases follow the same structure and the option of building cases with copy-and-paste operations allows for a simple case creation independent of the system model size.
In conclusion, the proposed approach is expected to scale well to the size of common industrial use cases.

6. CONCLUSION AND OUTLOOK

This paper proposed a novel approach to fault diagnosis in aPS for assembly and logistics combining MBR on topological system models with CBR. The models are provided via the standard engineering data exchange format AML, which is also used for storing cases of past faults. Building on the failure recovery architecture of Bareiss et al. (2016), the approach suggested relies on pre- and postconditions for fault detection. In contrast to many CBR systems from literature, all retrieved cases are adapted and case selection is based on the similarity to these adapted cases. An algorithm for retrieval and adaptation of similar cases that relies on tree and directed graph representations of the topological system model provided is described. The approach is evaluated on a research demonstrator resembling an aPS with a focus on assembly and logistics tasks. It is shown that the suggested CBR system strongly reduces the implementation effort compared to signal-based fault diagnoses approaches without a modular code design. Even with modular code design, the implementation effort is reduced if faults occur across different component types, e.g., across monostable and bistable cylinders. In contrast to many model-based approaches, the system suggested is able to reason from incomplete fault data. The case database can be built up continuously during operation and there is no need for complete fault models to exist in advance. Existing fault data can still be used after its conversion into fault cases. The approach is expected to transfer and scale well to real-world industrial use cases.
There are several directions in which this work could be extended in the future. While this paper focuses on CBR for fault diagnosis, the natural next step would be an extension to fault recovery. The integration of automatic recovery routines into the cases and their automatic adaptation to fit similar cases would be an interesting research area. Since the proposed CBR system has been only implemented as a prototype relying on sensor data provided via JSON files, a production-ready implementation of the system and an integration into the physical PPU would be another research direction. Despite the similarities between the PPU and industrial aPS for assembly and logistics, further evaluation of the approach's accuracy and diagnosis speed in more complex and real-world industrial use case should be conducted in the future.

REFERENCES

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1), 39–59.

Bareiss, P., Schütz, D., Priego, R., Marcos, M., and Vogel-Heuser, B. (2016). A model-based failure recovery approach for automated production systems combining SysML and industrial standards. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–7. IEEE.

Bayar, N., Darmoul, S., Hajri-Gabouj, S., and Pierreval, H. (2015). Fault detection, diagnosis and recovery using artificial immune systems: A review. *Engineering Applications of Artificial Intelligence*, 46, 43–57.

Berenji, H.R. and Wang, Y. (2006). Case-based reasoning for fault diagnosis and prognosis. In *2006 IEEE International Conference on Fuzzy Systems*, 1316–1321. IEEE.

Bregón, A., Simón, M.A., Rodríguez, J.J., Alonso, C., Pulido, B., and Moro, I. (2005). Early fault classification in dynamic systems using case-based reasoning. In *Conference of the Spanish Association for Artificial Intelligence*, 211–220. Springer.

Feret, M.P. and Glasgow, J.I. (1997). Combining case-based and model-based reasoning for the diagnosis of complex devices. *Applied Intelligence*, 7(1), 57–78.

Fischer, J., Bougouffa, S., Schlie, A., Schaefer, I., and Vogel-Heuser, B. (2018). A qualitative study of variability management of control software for industrial automation systems. In *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 615–624. IEEE.

Galil, Z. (1986). Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys (CSUR)*, 18(1), 23–38.

Gao, Z., Cecati, C., and Ding, S.X. (2015a). A survey of fault diagnosis and fault-tolerant techniques – Part II: Fault diagnosis with knowledge-based and hybrid/active-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6), 3768–3774.

Gao, Z., Cecati, C., and Ding, S.X. (2015b). 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.

International Electrotechnical Commission (2015). IEC 62714 – Engineering data exchange format for use in industrial automation systems engineering – AutomationML.

Isermann, R. and Balle, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5), 709–719.

Legat, C., Folmer, J., and Vogel-Heuser, B. (2013). Evolution in industrial plant automation: A case study. In *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 4386–4391. IEEE.

Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., and Nandi, A.K. (2020). Applications of machine learning to machine fault diagnosis: A review and roadmap. *Mechanical Systems and Signal Processing*, 138, 106587.

Mitra, R. and Basak, J. (2005). Methods of case adaptation: A survey. *International Journal of Intelligent Systems*, 20(6), 627–645.

Nasiri, S. and Khosravani, M.R. (2019). Faults and failures prediction in injection molding process. *The International Journal of Advanced Manufacturing Technology*, 103(5-8), 2469–2484.

O'Farrill, C., Moakil-Chbany, M., and Eklow, B. (2005). Optimized reasoning-based diagnosis for non-random, board-level, production defects. In *IEEE International Conference on Test, 2005.* IEEE.

Portinale, L., Magro, D., and Torasso, P. (2004). Multimodal diagnosis combining case-based and model-based reasoning: a formal and experimental analysis. *Artificial Intelligence*, 158(2), 109–153.

Prentzas, J. and Hatzilygeroudis, I. (2007). Categorizing approaches combining rule-based and case-based reasoning. *Expert Systems*, 24(2), 97–122.

Priego, R., Schütz, D., Vogel-Heuser, B., and Marcos, M. (2015). Reconfiguration architecture for updates of automation systems during operation. In *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, 1–8. IEEE.

Schütz, D., Legat, C., and Vogel-Heuser, B. (2012). On modelling the state-space of manufacturing systems using uml. *14th IFAC Symposium on Information Control Problems in Manufacturing*, 45(6), 469–474.

Tsai, Y.T. (2009). Applying a case-based reasoning method for fault diagnosis during maintenance. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 223(10), 2431–2441.

Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S.N. (2003a). A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Computers & Chemical Engineering*, 27(3), 313–326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S.N., and Yin, K. (2003b). A review of process fault detection and diagnosis: Part III: Process history based methods. *Computers & Chemical Engineering*, 27(3), 327–346.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S.N. (2003c). A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3), 293–311.

Vogel-Heuser, B., Legat, C., Folmer, J., and Feldmann, S. (2014). Researching evolution in industrial plant automation: Scenarios and documentation of the pick and place unit. Technical report, Institute of Automation and Information Systems, Technische Universität München. URL `https://mediatum.ub.tum.de/node?id=1208973`.

Vogel-Heuser, B., Rösch, S., Fischer, J., Simon, T., Ulewicz, S., and Folmer, J. (2016). Fault handling in PLC-based industry 4.0 automated production systems as a basis for restart and self-configuration and its evaluation. *Journal of software engineering and applications*, 9(01), 1–43.

Westkämper, E. (2003). *Adaptable production structures*, 87–120. Springer.

Wolfenstetter, T., Basirati, M.R., Böhm, M., and Krcmar, H. (2018). Introducing TRAILS: A tool supporting traceability, integration and visualisation of engineering knowledge for product service systems development. *Journal of Systems and Software*, 144, 342–355.