A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems

Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter

Automation and Control Institute, Vienna University of Technology
Gußhausstraße 27-29, A-1040 Vienna, Austria
Email: {steinegger,melik-merkumians,zajc,schitter}@acin.tuwien.ac.at

***Abstract*—In this paper, a framework for automatic generation of a flexible and modular system for fault detection and diagnosis (FDD) is proposed. The method is based on an ontology-based integration framework, which gathers the information from various engineering artifacts. Based on the ontologies, FDD functions are generated based on structural and procedural generation rules. The rules are encoded as SPARQL queries which automatically build logical segments of the entire manufacturing system in the ontology, assign sub-processes to these segments, and finally generate the appropriate FDD system for the sub-process. These generated modular FDD functions are additionally combined in a modular way to enable the fault detection and diagnosis of the entire system. The effectiveness of the approach is demonstrated by a first application to a conveyor system.**

I. Introduction

Undetected or unhandled component failures of manufacturing systems not only pose a threat to the operation personnel, the environment, and the plant itself but are also one of the major cost factors during the production time [1]. To counteract and significantly reduce the probability of the mentioned threats, system faults have to be detected, localized and also handled at an early stage. Therefore, *fault detection and diagnosis* (FDD) systems are included into the control architecture of the manufacturing system to monitor the plant.

Venkatasubramanian [2] summarized generally applied FDD methods which are classified into four categories: (1) qualitative and (2) quantitative model-based methods [3], and (3) qualitative and (4) quantitative process-history based methods [4]. However, none of these methods encompasses all desired characteristics like fast detection and analysis, robustness, and adaptability. Furthermore, the lack of a consistent plant data management hinders the efficient and re-usable modeling of inherent coherence and dependencies of components required for modular and easily reconfigurable FDD systems.

A promising approach to overcome the above-mentioned issues is to combine model-based approaches with process history methods [5]. For instance, the proposed fault diagnosis approach described in [6] is based on the structural plant knowledge mapped to the CAEX (*computer-aided engineering exchange*) model combined with process knowledge given by formalized process descriptions. However, for flexible and reconfigurable Industry 4.0 systems, modular and scalable fault diagnosis approaches are required for such systems [7].

In this paper, the ontology-based fault diagnosis approaches proposed in [8] and [9] are both extended and combined to achieve a modular, scalable, and easily configurable FDD system for manufacturing systems. The approach is based on the formalization of structural plant knowledge in terms of mapping the data of various engineering documents to the defined ontologies. The formalized plant model is then analyzed and logical segments of the underlying manufacturing systems are build by user-definable rules encoded as SPARQL queries. Afterward, predefined sub-processes are automatically assigned to the built logical segments encompassing also the diagnosis information of the individual segment.

The remainder of the paper is organized as follows. In Sec. II, the integration and formalization of the system knowledge with an ontology-based integration framework is described. The approach for generating FDD systems automatically based on the formalized system knowledge is summarized in Sec. III. Section IV describes the evaluation of the proposed approach by application to a laboratory transportation system, and Sec. V concludes the paper.

978-1-5090-6505-9/17/$31.00 © 2017 IEEE

II. Formalization of System Knowledge

The proposed approach for automatically generating FDD systems is based on the a-priori knowledge of the underlying manufacturing system captured within a modular structured ontology-based integration framework, as illustrated in Fig. 1 and also Fig. 2.

A. System Knowledge

The system knowledge encompasses the plant model representing, for example, the mechanical structure and the electrical wiring of the underlying manufacturing system. Here, the system and its components can be described based on AutomationML [10]. Furthermore, the description of processes executed on the plant (e.g., given as process recipes) is required for fault diagnosis. However, since those sources of system

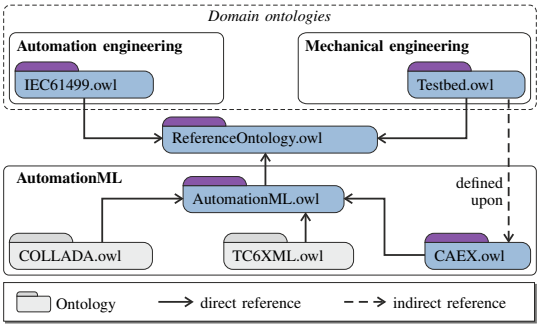

Fig. 1. Overview of the ontology-based integration framework, where several domain ontologies are bridged by a modular-structured reference ontology. The ontology modules, which are relevant for the proposed approach, are highlighted with color.

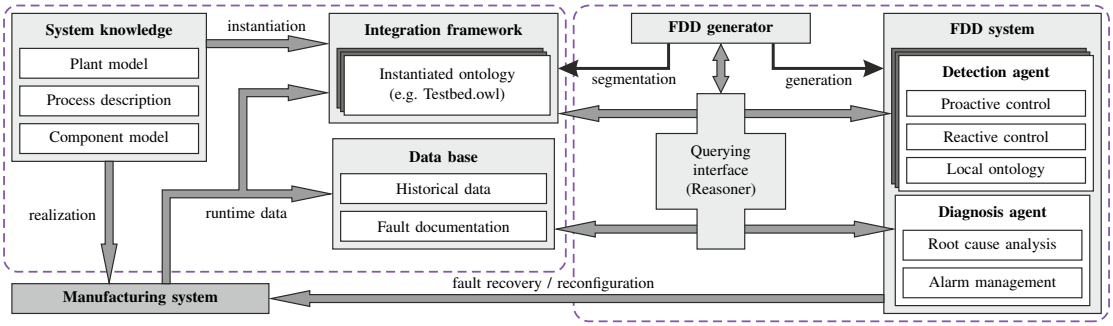

Fig. 2. Concept for the automatic generation of fault detection and diagnosis (FDD) systems. The integration framework represents one of the core component of the approach, which integrates and formalizes the system knowledge given from various sources. Based on the integrated system knowledge, the FDD generator automatically builds logical segments of the underlying manufacturing systems in the ontology and generates the modular FDD system.

knowledge often do not exist in a formalized form, they have to be formalized in the first step for further processing. The formalization is achieved by mapping the information of the engineering documents to previously defined ontologies and is described in Sec. II-C.

B. Integration Framework

The integration framework is composed out of several individual domain ontologies which are bridged by a superior reference ontology. This modular structure is defined according to the concept proposed in [11] and has several advantages like, for instance, the seamless integration of additional ontology models in terms of ontology mapping. For further details, advantages, as well as a formal definition of the integration framework it is referred to [9].

C. Ontology Instantiation

For instantiating the developed ontologies, a combined engineering and supervisory and control tool called Testbed Manager has been developed. This tool enables the graphical planning of the desired transportation system based on standard components as depicted in Fig. 3. After the system design, the corresponding ontologies are automatically instantiated according to the internal tool model. Furthermore, the Testbed Manager acts as supervisory and control system where the desired path, the routing algorithm and options can be selected and detected faults are reported.

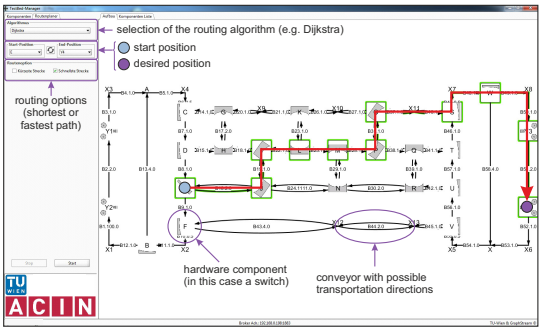

Fig. 3. Screenshot of the Testbed Manager, which represents the engineering tool for the underlying transportation system as well as the supervisory and control system combined in one single tool.

The instantiation of the ontologies is achieved by an implemented tool connector, which maps the internal concepts of the Testbed Manager model to the concepts and properties of the ontologies. Referring to Fig. 4 and Fig. 3, it can be seen that, for instance, the conveyor component in the Testbed Manager is mapped to the class `Conveyor` with all associated properties and references to other components the conveyor is connected with (see *property assertions* of instance `C1` in Fig. 4).

An example of the implemented and automatically instantiated ontologies is depicted in Fig. 4. It can be seen that the ontology for representing the concepts of a transportation system (Testbed.owl) are defined upon the CAEX ontology due to presentation purposes. However, the references between the individual ontologies are usually defined in the reference ontology. The highlighted AutomationML concepts (AutomationML.owl) represent generic connector classes as defined in [10]. In the proposed approach, an adapted implementation of the AutomationML standard interface class `Order` is applied to represent the order of sensors and actuators attached to a conveyor. This simplifies the entire segmentation process since only the outer sensors and actuators are relevant for building the logical segments of the underlying system. The implemented class `Order`, wherein the sensors are referenced, is depicted on the top right side of Fig. 4.

III. Knowledge-Based Fault Detection

After formalization of the engineering knowledge the segmentation and FDD generation process is started.

A. Segmentation and Process Assignment

The initial step of the FDD generation process consists of the segmentation step. Here, the goal of the segmentation process is to (semi-)automatically build logical segments of functional coherent components in the ontologies of the integration framework. The segmentation process is described by application to the laboratory conveyor system as depicted in Fig. 5 and described in Sec. IV-A.

Considering Fig. 5, it can be seen that the logical segments are limited by the outer sensors of each physical conveyor. Since each conveyor has an assigned instance of the class

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter, "A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247705

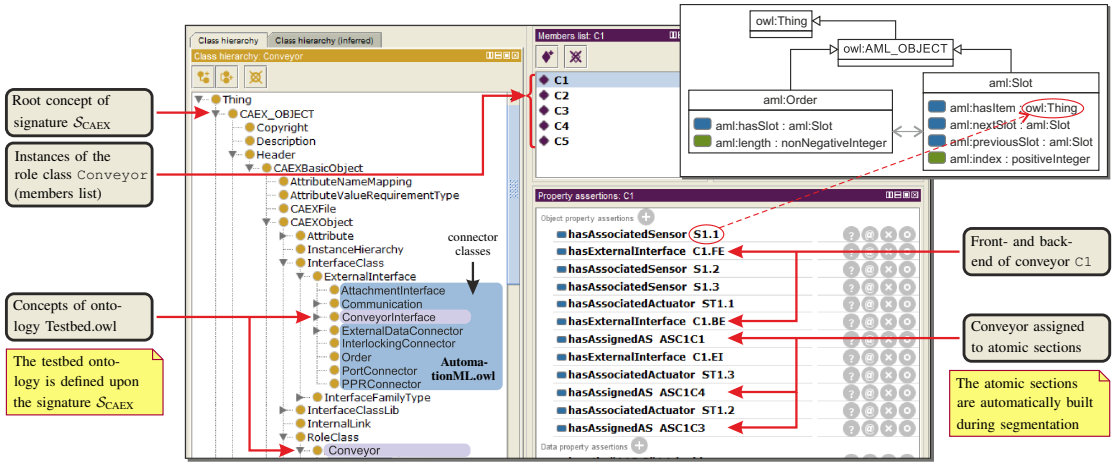

Fig. 4. Implementation example of the ontology in Protègè for the laboratory transportation system as described in Sec. IV-A. The concepts for describing the transportation system are defined upon the CAEX ontology concepts and extend the standardized and generic concepts.

`Order`, which holds the ordering of the attached sensors and actuators, the limiting sensors can easily be retrieved from the instantiated ontologies. For instance, the SPARQL query in Listing 1 retrieves the first sensor of the conveyor `C1` of the transportation system as depicted in Fig. 5. Since the last sensor can be retrieved accordingly, the resulting logical transportation segment is added as an instance to the testbed ontology via SPARQL insert / construct. The built logical segments for conveyor `C1` are denoted as `ASC1C*` and marked in Fig. 4 as atomic sections. Additionally, the physical properties like the dimensions of the new logical segments are computed based on the properties assigned to the underlying physical segments (conveyors).

Beside structural rules to build the logical segments out of the physical model stored in the ontology-based integration framework, there also exist procedural rules. Those rules are responsible for assigning sub-processes to the generated logical segments as well as instantiating the corresponding FDD procedure. The benefit of this modular approach is that the complexity of the FDD procedure assigned to a logical segment is minimized, which represents the basis for a flexible and hierarchical FDD composition. Here, one of the challenges is to identify the possible process which can be executed on the automatically built logical segments.

Listing 1. SPARQL QUERY TO RETRIEVE THE FIRST SENSOR.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX tb: <http://www.acin.ac.at/mst/testbed.owl#>
PREFIX aml: <http://www.acin.ac.at/mst/automationml.owl#>
SELECT DISTINCT ?first ?conveyor
WHERE { ?conveyor rdf:type tb:Conveyor .
        ?conveyor tb:hasAssociatedSensor ?sensor .
        ?slot aml:hasItem ?sensor .
        OPTIONAL { ?slot aml:previousSlot ?prevSlot }
        FILTER( !bound( ?prevSlot ) )
        BIND (?sensor AS ?first) }
```

To cope with the above-mentioned issue, several process descriptions for possible logical segments have been developed and implemented. Each process is described as PackML state machine [12, pp. 214] in order to provide a standardized interface to the logical segment and the process executed on it. With this approach it is possible to automatically identify the corresponding PackML state machine of a generated logical segment. For example, the transportation segment as depicted in Fig. 5 only requires two different parameterizable PackML state machines to describe the entire possible transportation process that can be executed. One type is instantiated for the linear segments with only two physical connections to other conveyors, which also includes the built segments at the corners of the transportation loop, and the second one is applied for the segments with three physical connections. Thus, the appropriate PackML state machine can be easily determined during the segmentation process according to the information in the ontologies.

B. Parametrization of the PackML State Machines

Since the defined PackML state machines represent generic process descriptions that can be used to describe a set of elementary processes, these process descriptions have to be

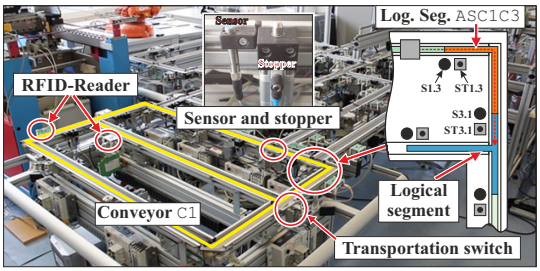

Fig. 5. Laboratory transportation system as testbed for distributed control.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter, "A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247705

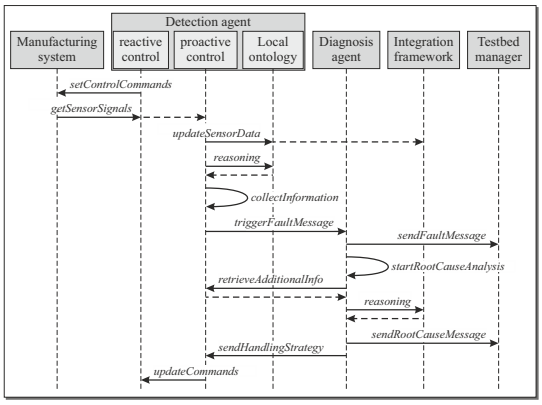

Fig. 6. UML sequence diagram describing the fault detection and identification process of the proposed knowledge-based approach.

parametrized when they are instantiated and assigned to a specific logical segment. The parametrization process is encoded in additional procedural rules which, for instance, insert the physical parameters or transportation time for the detection of a stuck pallet between two sensors.

C. Online Fault Detection and Diagnosis

As depicted in Fig. 2, the ontology-based integration framework instantiated with the a-priori system knowledge is additionally populated with the runtime data during runtime of the underlying manufacturing system. This enables the real-time fault diagnosis of the system based on the formalized system knowledge and the process descriptions. However, the bottle-neck is the execution time of the reasoner in order to process the FDD rules on the implemented ontologies. To enable a fast detection of occurred faults, the relevant concepts and relations which are assigned to a logical segment built during the segmentation process are stored in an additional ontology model. This separated ontology model, denoted in Fig. 2 as local ontology of each detection agent, only contains the knowledge about the specific part of the underlying manufacturing system which is relevant for the FDD process. It is assigned to a detection agent responsible for the fault detection of the according logical segment. Since this information set is reduced to a minimum, the reasoning and execution of the fault detection procedure can be carried out within reasonable time.

In contrast, the diagnosis agent, which is running on a centralized component and acting as the global governor in the FDD process, has access to the entire knowledge of the integration framework. The reason is, that the basic system faults have to be detected very quickly while the diagnosis (e.g., identifying the root cause of the occurred faults) usually has a higher time constant since an entire root cause analysis represents a computationally expensive task. While the ontologies in the integration framework are only updated with the current runtime data, the historical runtime data is stored in an additional data base. The benefit is that the ontologies only hold the current state of the underlying manufacturing system and decisions can be made very fast based on the current state. To conduct an extensive root cause analysis the historical data is also often required to identify the specific root cause of occurred faults. The entire FDD process and the interaction and communication between the involved components is summarized in the sequence diagram depicted in Fig. 6. Therein, it can also be seen, that the diagnosis agent can, beside identifying complex system faults and conducting root cause analysis, also send fault handling strategies to the underlying detection agents. This hierarchical approach also enables the fast adaption of the FDD system, since changes of the underlying manufacturing system can be propagated to the integration framework and, for instance, new segments with assigned detection agents can be easily integrated.

IV. Evaluation

The first evaluation of the proposed and implemented knowledge-based fault detection and diagnosis approach is carried out by application to a laboratory transportation system.

A. Description of the Target System

The target system is a laboratory conveyor system as depicted in Fig. 5. It consists of more than 40 Festo control units CPX-CEC-C1, each of them controlling one individual segment. A segment typically consists of a conveyor, an identification station (RFID-reader, inductive proximity switch, and stopper), and a gripper station to fix the pallet at a certain position for manipulation tasks. The system is characterized by a high redundancy of simple transportation structures, which makes the transportation system suitable to illustrate the applicability and scalability of the proposed approach.

As depicted in Fig. 7, each component of the proposed FDD system is connected to an integration layer. This integration layer represents the communication interface for all components and is realized based on MQTT (message queue for telemetry transfer). Therefore, the message broker Mosquitto [13] is applied in this framework, which receives messages from the connected clients and forwards the respective message to all clients that subscribed to the specific topic.

B. Implementation of the Individual Components

The hardware-nearest components are represented by the reactive control part of the detection agents. Therefore, each controller is running the IEC 61499 runtime environment Forte as part of the 4diac (*Framework for Distributed Industrial Automation and Control*) framework [14]. Process descriptions represented by the PackML state machines are encoded into the *event control chart* (ECC) of a basic *function block* (FB) in 4diac. The reactive control has access to the individual sensors and actuators via service-interface FBs and communicates with

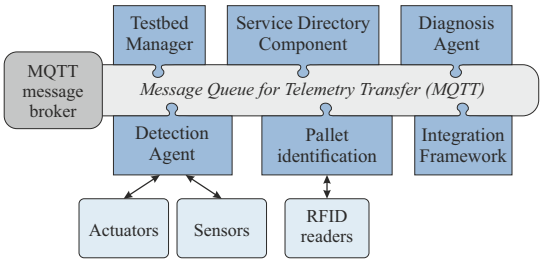

Fig. 7. Communication architecture of the transportation system. All components are connected via an integration layer based on MQTT.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter, "A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247705

the other components via the integration layer. The proactive part of the detection agents are implemented in the *Java agent development framework* (JADE) on a centralized computer.

The superior component in the proposed modular and hierarchical FDD system structure is represented by the diagnosis agent. This agent is also implemented in JADE [15] and has access to the ontologies via the integration layer. Therefore, the diagnosis agent can send queries directly to the ontologies of the integration framework and thus conduct a global fault diagnosis or root cause analysis.

C. Use Case: Stopper Sensor Fault

The considered sub-part of the conveyor system for evaluation is represented by the logical segment `ASC1C3` depicted in Fig. 5, which is built during the segmentation process and contains sensors and actuators of the underlying conveyors `C1` and `C3`. This logical segment is terminated by the sensors `S1.3` of conveyor `C1` and `S3.1` of conveyor `C3`. Thus, the assigned detection agent is focused on the events on the entry and exit sensor of the logical segment.

When the incoming pallet is detected by sensor `S1.3`, the stopper `ST1.3` should be opened when no other pallet is currently passing the logical segment. The reactive control, as part of the assigned detection agent, sends the open signal to the corresponding stopper as depicted on the left-hand side of Fig. 8. Since the position feedback from the sensor attached to the stopper does not change from state *false* to *true* as expected, the reactive control immediately informs the superior proactive control about the deviation of the two signals. The proactive control in turn triggers a warning message in the Testbed Manager and further monitors the process. At this state of the process it can not be distinguished between a sensor fault and mechanical fault of the stopper due to the restricted process feedback. The monitoring of the proactive control is based on constantly executing queries on the local ontology in order to retrieve the cause of the occurred signal deviation. After the sensor signal `S1.3` changes from *true* to *false*, the probability of a fault of the position feedback sensor increased but could still have other reasons within the process. However, after the outer sensor `S3.1` has detected a pallet, the proactive control can identify the sensor of `ST1.3` as the faulty component and triggers the fault message via the diagnosis agent in the superior supervisory and control tool (Testbed Manager). In this case, either the process should be stopped as it is usually required in typical industrial processes or, if the fault does not affect the correct routing of the pallet in this case, an immediate replacement of the sensor should be requested.

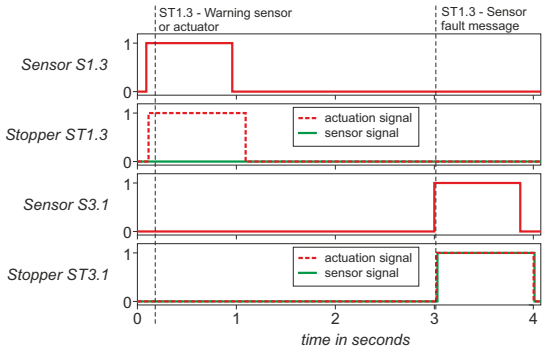

Fig. 8. Automatic detection of a sensor fault in the stopper `ST1.3`.

It should be noted, that each local fault detection agent also takes into account the transportation time for the underlying logical segment. This parameter is currently set with a default value but could also be computed based on the information contained in the ontologies and/or the runtime data (current motor speed if accessible).

D. Future Work

The current evaluation focused on the generation process itself, verifying the mapping of the engineering data to the developed ontologies as well as the correct automatic instantiation and parametrization of the developed state machines for the local fault detection. It was shown, that the automatic setup of the individual detection agents responsible for local fault detection is feasible, based on the developed ontology-based framework. However, in order to evaluate the interaction of the individual (local) fault detection methods, especially in cases of multiple faults occurring at the same time, more complex use cases are required. Considering a logical segment with three connections (see right side of Fig. 5), there exists possible faults where the decision on the root cause cannot be made locally. For instance, if the pallet should follow a desired path through the transportation system, it could be possible that either the transportation switch or the outer sensors of the logical three-port segment (or even all at the same time) are not working as expected. In such cases, the superior diagnosis agent has to increase the reasoning scope by additionally taking into account the information gathered from the adjacent logical segment in order to identify the root causes. However, the current implementation represents the foundation for a modular, scalable, and also easily configurable fault detection and diagnosis system.

V. Conclusions

This paper presents a framework for the knowledge-based *fault detection and diagnosis* (FDD) in industrial transportation system. The approach is based on the formalized knowledge about the underlying manufacturing system gathered during the engineering process. This system knowledge is stored in predefined ontologies and is additionally enriched with runtime data from the sensors, which enables the online reasoning for occurred faults in the system. To tackle the complexity and scalability issues, the initial step of the proposed method consists of a (semi-)automatic segmentation of the ontology model of the underlying manufacturing system into logical segments. Here, the entire segmentation process is defined by manually defined SPARQL rules which are executed on the instantiated ontologies by a reasoner. Afterward, predefined process and FDD descriptions based on the PackML state machine are automatically instantiated and parametrized based on the information stored in the ontologies. These descriptions are individually assigned to the previously generated logical segments (mapping to different physical segments).

The proposed approach has two main advantages. First, repetitive tasks like the programming and parametrization of the FDD methods for each segment can be automatized by encoding them once into SPARQL rules. These rules can

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter, "A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247705

automatically extract or also compute the required parameters from the underlying knowledge base, represented by the instantiated ontologies. Here, the main task is to define the required SPARQL rules, which can then be executed for each logical segment of the manufacturing system. However, the main challenge here is to extract the required data from the engineering documents elaborated during the engineering phase of the manufacturing system, and to instantiate the ontologies with this data. The second advantage is that the probability of forgotten changes on changing system structures can be reduced due to the possibility to automatically trigger the FDD generation process on each change in the ontologies.

The current implementation of the proposed framework enables the automatic segmentation of the entire conveyor system into logical segments, the assignment of sub-processes, as well as the automatic parametrization of the FDD modules. However, future work will address the combination of the individual instantiated FDD modules assigned to the automatically defined logical segments, in order to achieve global fault detection and handling as discussed in Sec. IV-D.

Acknowledgments

Financial support by Festo AG (Esslingen, Germany) is gratefully acknowledged.

References

[1] G. Bai, T. Kajiwara, and J. Liu, "The cost of manufacturing disruptions," *Strategic Finance Magazine*, 2015.

[2] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis part II: Qualitative models and search strategies," *Computers and Chemical Engineering*, no. 27, pp. 313–326, 2003.

[3] R. Isermann, "Model-based fault-detection and diagnosis - Status and applications," *Annual Review in Control*, vol. 29, no. 1, pp. 71–85, 2005.

[4] S.-H. Liao, "Expert system methodologies and applications - A decade review from 1995 to 2004," *Expert Systems and Applications*, vol. 28, no. 1, pp. 93–103, 2005.

[5] L. Abele, M. Anic, T. Gutmann, J. Folmer, M. Kleinsteuber, and B. Vogel-Heuser, "Combining knowledge modeling and machine learning for alarm root cause analysis," in *IFAC Conf. on Manufacturing. Modeling, and Control*, 2010.

[6] L. Christiansen, A. Fay, B. Opgenoorth, and J. Neidig, "Improved diagnosis by combining structural and process knowledge," in *IEEE Int. Conf. on Emerging Technologies Factory Automation*, 2011, pp. 1–8.

[7] W. Hu, A. G. Starr, and A. Y. T. Leung, "Operational fault diagnosis for manufacturing systems," *Journal of Materials Processing Technology*, no. 133, pp. 108–117, 2003.

[8] M. Melik-Merkumians, T. Moser, A. Schatten, A. Zoitl, and S. Biffl, "Knowledge-based runtime failure detection for industrial automation systems," in *Workshop Modelsrun.time*, 2010, pp. 108–119.

[9] M. Steinegger, M. Melik-Merkumians, J. Zajc, and G. Schitter, "Automatic generation of diagnostic handling code for decentralized PLC-based control architectures," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2016, pp. 1–8.

[10] R. Drath, Ed., *Datenaustausch in der Anlagenplanung mit AutomationML*, 1st ed. Springer, 2010.

[11] M. Steinegger and A. Zoitl, "Automated code generation for programmable logic controllers based on knowledge acquisition from engineering artifacts: Concept and case study," in *IEEE Int. Conf. on Emerging Technologies and Factory Automation*, 2012, pp. 1–8.

[12] W. Hawkins, D. Brandl, and W. Boyes, Eds., *Applying ISA-88 in Discrete and Continuous Manufacturing*, ser. The WBF Book Series: Vol. 2. Momentum Press, LLC, New York, 2010.

[13] R. Light and I. Craggs, "Mosquitto – an open source MQTT broker," https://mosquitto.org/, 2016, accessed: 2016-09-20.

[14] T. Strasser, M. Rooker, G. Ebenhofer, A. Zoitl, C. Sünder, A. Valentini, and A. Martel, "Framework for distributed industrial automation and control (4diac)," in *6th IEEE International Conference on Industrial Informatics*, 2008, pp. 283–288.

[15] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE: A FIPA2000 compliant agent development environment," in *Proc. of the 5th Int. Conf. on Autonomous Agents*, 2001, pp. 216–217.

Post-print version of the article: Michael Steinegger, Martin Melik-Merkumians, Johannes Zajc, and Georg Schitter, "A Framework for Automatic Knowledge-Based Fault Detection in Industrial Conveyor Systems," *2017 IEEE 22st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017. DOI: 10.1109/ETFA.2017.8247705