A Framework for IoT-Based Monitoring and Diagnosis of Manufacturing Systems

I-Ling Yen, Shuai Zhang, Farokh Bastani
Computer Science Department
University of Texas at Dallas
{ilyen, shuai.zhang1, bastani}@utdallas.edu

Yuqun Zhang
Department of Computer Science and Engineering
Southern University of Science and Technology
zhangyq@sustc.edu.cn

Abstract. IoT systems have gained increasing attentions in research community and industry. Tens of billions of devices are now connected to the Internet and quintillion bytes of data are generated from sensing devices every day. One of the important applications of IoT systems in industry is monitoring, fault detection, and diagnosis of manufacturing systems (MFDM). However, current practices in the development of such systems are individualized with each company developing their own solutions. To address this issue, we propose a SaaS-centered framework for manufacturing system health management. The configurability and easy evolution of SaaS can facilitate reuse and sharing of data, processes, and technologies.

Besides the general framework, we also look into the technologies that are important for the framework. The literature in time series data storage and the techniques for mining correlated data are reviewed and the gaps are identified. To bridge the gap, we discuss some potential methods for resolving the problems. We also consider how to incorporate the potential techniques into our framework for effective fault detection and diagnosis.

Keywords: Internet-of-things, cyber-physical systems, smart manufacturing, smart industry, SaaS, health monitoring, fault detection and diagnosis.

1 Introduction

IoT (Internet-of-things) technologies are being developed rapidly in recent years. It is estimated that there are tens of billions of physical things that are connected to the Internet, and the number is still growing rapidly. Various IoT applications are being developed towards the goal of more advanced automation and improved human living.

Due to the importance and popularity of IoT, many application domains have incorporated IoT to improve the application systems. A lot of efforts are currently devoted to the development of smart home, smart building, smart city, smart planet, smart farms, smart agriculture, smart factory, smart manufacturing, smart industry, smart roads, smart parking, smart transportation, smart cyber physical systems, smart grid, etc.

To enable the IoT applications, many IoT technologies have been developed. One category of such technologies is related to the interoperation and integration of the extremely diverse IoT devices. Many standards and tools for different layers of IoT systems, from communication protocols, integration platforms, middleware technologies, to cloud based platforms, have been developed [1-5]. These technologies can facilitate IoT connection to the network, interoperation, wrapping and encapsulation, and rapidly evolve to adopt new technologies and IoT system development and deployment. Also, big data plays a very important role in IoT systems. Most of the IoT systems involve a large number of sensors that collect a huge quantity of data and machine learning methods are applied to analyze them for various goals. Besides existing big data analytics, performance improvements and adapted algorithms are developed to fit specific IoT applications.

Among various applications of IoT systems, a lot of industrial companies are now adopting IoT to improve their operations. One major use of IoT systems in industry is for monitoring, fault detection and diagnosis (MFDD) of the operations of the systems to ensure their proper operations. This can help with early problem detection and mitigation to ensure high quality as well as highly reliable and safe system operations. For manufacturing systems, IoT based monitoring and problem detection can also help reduce the likelihood of producing defective products and potentially improve product quality. However, many companies face the problem of adopting IoT solutions. Due to the potential differences, each company ends up developing its own company-specific IoT system to achieve the monitoring and fault detection and diagnosis goals.

In this paper, we discuss the design of a SaaS-centered MFDM framework for IoT based system health management for manufacturing systems. The goal is to extend system configurability such that manufacturers can easily build the health management capabilities for their production systems using the MFDM framework. The MFDM-SaaS is designed to offer physical system architecture management, data collection infrastructure and management, big data analytics, sensor and diagnosis capability provisioning and scheduling, and data provenance and security assurance.

Besides a SaaS-centered framework, we also look into the desired technologies that should be incorporated in the MFDM framework. We review the state of the art technologies in using IoT and big data for manufacturing system monitoring, fault detection and diagnosis, and investigate the gaps in current research. One major issue we consider is the data correlation analysis. In many physical systems, individual sensor data are not sufficient to detect and

978-1-5090-6320-8/17 $31.00 © 2017 IEEE
DOI 10.1109/SOSE.2017.26

diagnose faults. The sensor readings for some components are correlated. Traditionally, a powerful paradigm for FDD for this type of systems is model-based fault detection and diagnosis [6][7]. However, model-based approaches require a specific model, including the system architecture and sensor data correlation rules, being built to enable fault detection and diagnosis. This information is not always available from some manufacturers and, even if available, it may not be complete. Also, manufacturing systems may evolve to adopt new technologies or upon the demand of new production needs. Moreover, with the new drive of customer-oriented industry, it is expected that manufacturing systems will become more and more agile and configurable. All these factors make it more challenging to maintain the up to date system specification for the manufacturing systems. Thus, it is expected that some sensor data correlation should be automatically derived instead of being prespecified. However, with large-scale systems, there may be hundreds, thousands, or an even larger number of sensors. Mining the data correlation without knowing which sensor data streams are correlated can be highly computation intensive and may even be infeasible.

Another important issue in MFDM is the storage of sensor data streams. Many time series database (TSDB) have been developed and they are still evolving. Since there are many different applications that may require different TSDB support, some desired features are still being identified and incorporated. We discuss the TSDB requirements for manufacturing sensor data that are not well supported in current TSDBs, such as the consolidation methods, the sensor data semantics, etc., and consider some solutions.

In the next section, we review the literature related to IoT systems, manufacturing systems, and system monitoring and fault diagnosis. The architecture and components of the MFDM framework are discussed in Section 3. Specific components in the framework that require in depth research and investigation are discussed in Sections 4 and 5. Section 5 states the conclusion of the paper.

2 Literature Review

2.1 IoT Technologies

IoT related research spans many dimensions. Basic IoT systems research focuses on the communication and interoperation for machine to machine (M2M) and machine to cloud interactions. For example, Bluetooth, Zigbee, AllJoyn are frequently used M2M protocols for M2M communications. 6LowPAN and RPL are popular IP based communication protocols used for large scale mobile device communications. Some more abstract messaging protocols built on top of the publish/subscribe or remote procedure call paradigm have also been developed for IoT communications, such as MQTT, CoAP, and REST protocols.

Some basic integration platforms have also been built to facilitate integration and interoperation of IoT devices. For example, Contiki [1] is an open source IoT operating system that can support several IoT communication protocols including CoAP, 6LowPAN, and RPL. Users of Contiki can communicate with IoT devices with common sockets. Arduino [2], besides offering communication interface, also provides a programming language for controlling the IoT devices. Thus, users can exercise the desired control without needing to know the low level commands of the IoT devices. There are also some higher level platforms such as SenaaS (sensor as a service) system [3] and SOCRADES [4], which support event based service oriented IoT system development. They require IoT device developers to wrap the low level IoT devices as services and the IoT application system users can invoke IoT services through common service interfaces without knowing the device details.

Since many IoT systems focus on sensing and monitoring, data is an important element in this class of IoT systems. It is estimated that everyday a couple of quintillion bytes of data are collected from sensing devices. Thus, a lot of research efforts are devoted to data related topics for IoT systems. Some IoT platforms focus on interacting with IoT devices for data collection. OpenIoT [5] supports easy sensor integration and data collection without needing to worry about how to interact with the sensors. It also is an infrastructure of sensors and collected sensor data. Its sensor data repository is based on the semantic sensor network (SSN) standard. Legitimate users can access semantic sensor data hosted by OpenIoT to obtain the desired information.

Research related to IoT data also investigate the issues of data semantics. Several standards have been developed. OGC (Open Geospatial Consortium) has developed the Observations & Measurements (O&M) standard [8], which defines the XML schema observations and measurements data from sensors. The schema considers specifications of sensor location, sensor functionality, and lineage of observation. Correspondingly, the Sensor Model Language (SensorML) is a standard defined to describe the sensor systems and processes associated with sensor observations [9]. But Sensor ML mainly focuses on geospatial data, and may not be sufficient to provide the semantics for system health monitoring data. SSN ontology [10] is defined by W3C and it provides a generic sensor data semantics model (used by OpenIoT). The ontology describes sensor devices, sensor operations and management, sensor observation and measurement data semantics, etc. Sensor O&M data semantics also uses Quantity Kinds and Units standard [11] for the specification of what kind of data are collected by a sensor. SSN is a powerful ontology and when properly integrated with domain ontology, it can be used for modeling the semantics for manufacturing monitoring data.

2.2 Fault Detection and Diagnosis

Due to the importance of fault detection and diagnosis for system health, a large volume of research works have been

presented in the literature. Over the past, the most prominent solution for FDD is probably the model-based approach [6]. In model-based FDD, a model for the cyber physical system should be built based on the expert knowledge and mathematical abstraction of the system. From the model, the rules and invariants that had to be followed by the system states can be derived and accordingly faults can be detected and diagnosed when the system state violates the model. The model-based FDD has been applied to many application systems successfully [7].

A variation of model-based FDD is the integrated system health management (ISHM), which is renamed from IVHM (integrated vehicle health management) initiated by NASA [12] to general systems. IVHM integrates model based FDD, case based reasoning (CBR) [13], fault signature matrix (FSM) [14], anomaly detection, etc. Model-based FDD relies fully on the system model for fault identification, while CBR relaxes it by the concept of training by examples. In CBR, normal and faulty cases of the system are presented to the learning module, which outputs the learnt model to be used for fault detection. Neural networks are frequently used as the learning module [15]. FSM is one specific method under the model-based FDD concept and focuses on fault diagnosis. It defines the sensor vector and correlates system components with sensors in the vector and, thus, forms a signature vector for each component. Different components have different signatures. When faulty sensor data are detected, the signature helps isolate the faulty component.

All the above methods require the knowledge about the system model and some of them require more specific rules for fault detection. This may not be feasible in some systems due to the heavy cost and efforts to generate the model and the flexibility of the model is limited. More recently, big data analytics are being applied to FDD with the benefit of not needing detailed system models. Many such applications consider individual sensor data streams and use anomaly detection, timed automata learning, etc., methods to detect faults [16]. When a more complex system with a large number of sensors is considered, the multivariate statistical analysis such as principle/independent component analysis (PCA/ICA) have been applied to discover the correlation between the large number of sensor data streams [17]. However, PCA and ICA can only derive linear correlations while many physical systems have nonlinear relations among their components. Nonlinear multi-variate algorithms have scalability problem. Thus, further research is required to support more powerful MFDD data analytics.

3 SaaS-Centered Framework for IoT-Based MFDD

In this section, we discuss a SaaS-centered configurable framework for IoT-based monitoring and diagnosis of manufacturing systems (MFDM framework). The MFDM framework is not fully a SaaS system because it requires on-site hardware/software components to perform data collection, on-the-fly data analysis and problem detection, and activation of mitigation schemes when problems are detected. All the other functions that do not have local dependency are incorporated in the SaaS. In Subsection 3.1, we discuss the SaaS design in the MFDM framework (MFDM-SaaS). The MFDM framework design for the manufacturing site (MFDM-Manufacture) is discussed in Subsection 3.2.

3.1 MFDM SaaS

Figure 1 shows the architecture for the SaaS in the MFDM framework. The corresponding technical issues and potential solutions for each component are discussed below.

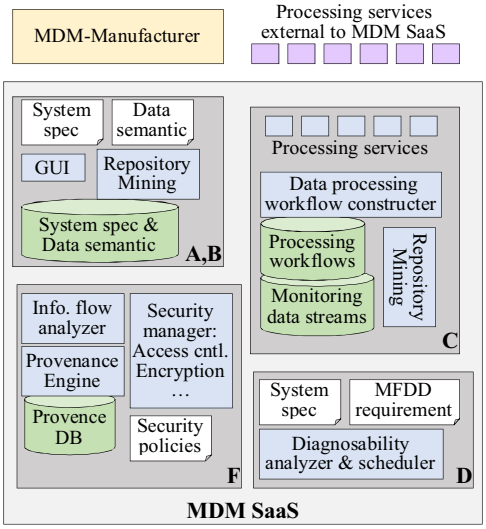

Figure 1. MFDM SaaS.

A. Knowledge of the Target System

In order to achieve fault diagnosis of a physical system, it is necessary to know the system architecture and workflow as well as the sensors placement in the system. System architecture provides the information regarding system components and their correlations, and the relations between the sensors and system components. System workflow provides the information regarding the sequences and timings that system components or subsystems are activated upon various events or stimuli. From the sensor data, one can observe the behaviors of the system. In case some anomalies are detected from the data, the fault diagnosis techniques can be used to identify the faulty components based on the relation of the sensors with faulty data and the system [6][18].

In traditional practice, the architecture and workflow of the system are created by human experts and interpreted into machine understandable model to facilitate fault diagnosis. However, creating these models are time consuming and costly. Some smaller companies may even lack the expertise for generating proper system models. With system evolution, it is unlikely these system specifications can be kept consistent with the real physical system architecture and

workflow. Modern manufacturing systems are shifting toward configurability to fit customized production needs, which makes the specification of the system model even more challenging. Thus, it is desirable that the system architecture can be discovered automatically.

In the MFDM SaaS, multiple approaches are used to help with system spec construction. Users can choose to directly submit system spec through the GUI. The repository maintains a large number of ready specifications. In case assistance is needed, the "System spec Miner" is designed to mine the repository for similar systems and use the mining results to help construct the spec of the target system.

Semantic models and specification languages are needed to support well defined system specifications. Existing specification techniques, such as UML, BPMN, timed automata can be combined to support the specifications.

B. Knowledge about the Collected Data

Effective system monitoring and fault diagnosis requires not only knowledge about the physical system, but also about the sensor data. Big data analytics can be applied to the sensor data for system fault detection and diagnosis. With the sensor data repository in the MFDM framework, it is also possible to cluster the sensor data streams and perform cross stream analysis to help more accurately detect and diagnose problems. Knowledge about the sensor and the data collected can help with better mining of similar data streams and perform cross data stream analysis. It is also possible to mine the similar data streams and obtain the most effective data processing workflows.

Specifying the data semantics can help with mining cross data streams. The semantics about one sensor data stream should include the sensor specification, the components or the subsystem the sensor is monitoring, the quantity and kind of the data, and the data unit. For each monitoring data, we need to have the time the data is collected and the collected data itself. Data stream correlations can be specified by rules or by simply indicating that a group of data streams are correlated.

C. General and Domain-Specific Monitoring Data Processing Services

Common sensor data stream analysis techniques can be incorporated in the MFDM SaaS as services. Generally, for individual data streams, anomaly detection, pattern mining, episode mining, periodicity analysis, etc., methods may be used. For correlated data, if the rules are specified, then analysis can be performed to ensure the conformance to the rules and determine the normal deviations from the rules in case the rules are not exactly satisfied. For correlated data without rules, the correlation of data can be analyzed and the rules can be derived. Similarly, the normal deviations from the rules can be recorded. We can also mine the data stream repository to obtain similar data streams and use the similar group for enhancing data analysis accuracy.

To improve data analysis performance, incremental mining methods should be considered for time series data. Also, the data quality from different sensors may be different and should be taken into account.

Big data analysis can be done in the cloud, and the model derived from the data streams is passed to the manufacturing site for on-the-fly sensor data validation against the model.

Pictures and videos are generally the generic and effective means for monitoring and fault detection of physical systems. For example, videos can show that in a nut bread production line, the nut dispenser did not dispense nuts for some doughs. Images of the system can be used to examine, for example, eroded parts of some production equipment. Thus, image processing and problem detection from videos are important capabilities to be included as the MFDM SaaS services.

When detecting problems from images and videos, specifications of what to look for could help guide the analysis and achieve better detection accuracy. Proper models for the specifications of faults and associated positive and negative image/video examples should be incorporated.

Other common data processing techniques that may be useful in processing monitoring data for manufacturing systems, such as signal processing, Fourier transformation, wavelet transformation, etc., can be incorporated as the MFDM SaaS services. Also, common mitigation software can be incorporated as the MFDM SaaS services as well.

The MFDM SaaS also incorporates a data processing workflow construction interface to support the description of the fault detection-diagnosis-mitigation workflows and their association to the sensor data streams. Data processing services hosted by MFDM SaaS as well as external data processing services can be discovered, selected, and composed into workflows to facilitate sensor data analysis.

Data streams and workflows from various sources are stored in the repository and the repositories can be mined for knowledge that can be reused by other systems. The data stream repository can be mined to discover similar data and use them to enhance fault analysis results. We can also support reuse of data processing schemes for fault detection and diagnosis by mining the repository for similar systems and data sources and learn the most effective processing methods and workflows for the corresponding sensor data streams.

D. Diagnosability Analysis and Optimal Sensor and Diagnosis Software Scheduling

For some sensors that may be far from power sources, we need to minimize their activation time if they can be dynamically activated. Also, we need to determine when to activate the fault detection and diagnosis software. These scheduling decisions are made to balance the diagnosability of the system, diagnosis timeliness, power consumption, and system overhead. Also, the activation schedule can be an adaptive one. In the normal situation, the schedule is set to ensure basic diagnosability. When there is suspicion of faults but below the identification threshold, then the schedule can be adapted to ensure better diagnosability and timely mitigation. The scheduling can be performed at the manufacturing site or in the cloud and compute an adaptive

scheduling rules and pass them to the manufacturing side

E. Data and Metadata Storage and Management

As shown in Figure 1, MFDM SaaS hosts several data and process repositories. They not only store the historical data and processing information, but also will be mined to get reusable knowledge to facilitate more accurate fault detection and diagnosis. To facilitate cross data stream mining, the semantics of the data are very important. Thus, semantic knowledge for the data and processing schemes should be clearly defined. Also, the storage system for storing the data is very important. Many supporting factors and performance issues should be considered in the data storage. Moreover, since the quantity of data will keep on growing as time goes by, if the storage space is confined, historical data should be compressed. Many time series databases have associated consolidation mechanisms and new consolidation algorithms for the data should be considered.

F. End-to-End Provenance and Security

The MFDM SaaS system hosts the data and processing elements from many manufacturing tenants. Hence, it is very important to protect the security, privacy, and integrity of the tenant assets. Each manufacturer should supply their own security policies. MFDM SaaS system offers data provenance, access control and information flow control mechanisms to protect the security of the data and processing elements of the tenants. The information in the repository that are private to the tenants may be publicly sharable, sharable after sanitization, or not sharable. MFDM SaaS offers a set of privacy preserving mining services to ensure the security policies of each tenant are strictly observed.

Additional security mechanisms should be incorporated in the data collection process and data collection channels. Physical system security and integrity protection should also be considered.

3.2 MFDM-Manufacture

The manufacturing site needs to manage the following tasks.

(1) Manage the sensors and the data collection process using existing IoT system management platforms [5]. Control the data stream flows from sensors to intermediate collectors, on-site analyzers, and to the Cloud. Some M2M and M2C standards for communication can be used.

(2) Coordinate with MFDM SaaS to obtain learned models and rules for individual as well as correlated data streams. Based on the predefined rules and/or learned models to exam the newly collected data on-the-fly and to detect faults and ensure that the system is operating correctly. Activate the mitigation system workflows to either confine the faults and protect the system or lead the system to a safe state.

(3) Coordinate with the scheduling unit in MFDM SaaS and determine the optimal scheduling for sensors and diagnosis software according to the current situation of the system. When necessary, adapt the schedule based on the diagnosis results.

Big data analysis and mining tasks are performed in the cloud. The learned models and rules are passed from MFDM SaaS to MFDM-Manufacturing for on-the-fly sensor data validation against the model. Image and video analysis should be done at the manufacturing site and the analysis capabilities can be migrated as virtual machines to the remote site. The historical image and video data as well as analysis results can be passed back to the cloud for storage and mining. The other data processing services can also be performed in one of these mechanisms.

4 Monitoring Data Storage

With the rapid developments in the domain of IoT, a huge amount of time series data are gathered by distributed sensors, which creates a huge demand and presents a great challenge for time series storage. Many time series data bases (TSDBs) have been developed with different features to store and retrieve time series data streams. Various TSDB research works have also been investigating how to efficiently store and process data at scale. The suitability of each TSDB for various IoT based monitoring and fault detection and diagnosis systems needs to be evaluated based on the context and the underlying data. Here, we take a high level look at the design of some popular TSDBs, including RRDtool [19], Graphite [20], BlueFlood [21], OpenTSDB [22], InfluxDB [23] and DalmatinerDB [24] and analyze the important features of these systems for IoT MFDD.

Scalability and data access performance are key features in the development of TSDB. RRDtool is the earliest version of time series database that offers industry level performance with fast disk seeking and fixed storage usage. It requires the data size related specifications in advance in order to pre-allocate required storage space when creating databases. However, RRDtool is not designed to handle a large number of data streams. With large-scale, even multiple-site manufacturing systems, thousands or tens of thousands of data streams flow into the storage at various rates, which makes RRD to exhibit performance bottlenecks. Graphite attempts to improve the bottleneck on random disk IO and handling irregularly arriving data. But this still cannot satisfy the increasingly large scale and high rate of time series data from MFDD systems. Newer TSDBs attempt to use distributed storage to address the IO access bottlenecks. Blueflood attempts to provide high scalability by using Cassandra for data storage and Elasticsearch [25] for indexing. Similar to Blueflood, OpenTSDB is built on top of HBase. Since Cassandra and HBase are both general purpose databases, not specifically designed for time series data, and are not the best performing distributed databases, Blueflood and OpenTSDB place a layer on top of them, which adds an additional cost for data access and management. Several even newer TSDBs try to build databases from ground up with the aim of improving query performance and storage efficiency.

InfluxDB and DalmatinerDB are representative TSDBs in this aspect. Some storage space optimizations, including limited rebuilding of RAID arrays, tuned B+Tree block size against the OS page size, etc., are adopted in these systems to ensure better data compression for storage efficiency. As a result, disk space usage in DalmatinerDB is less than one tenth of that in OpenTSDB for the same amount of data points. Besides storage efficiency, InfluxDB and DalmatinerDB also adopt more advanced storage engines for customization, such as RocksDB [26], LMDB [27] and ZFS [28], and have specific designs to work with them to achieve high performance as well as high scalability.

Due to the importance of TSDB performance and scalability, one critical task is the evaluation of the TSDBs for MFDD processing. We plan to develop a benchmark suite to represent the TSDB accesses in MFDD operations and study the tradeoffs and identify new requirements in TSDBs for MFDD of manufacturing systems.

In the domain of MFDD for manufacturing systems, the vast and diverse set of measurement metrics make the semantic modeling of the data streams and data points another important feature in TSDB design. Flexible data models with higher degrees of freedom on data types and schema along with more powerful query languages are the goals to be pursued. RRDtool, Graphite, and Blueflood use RRD-like models where data semantics, such as name of time series data, consolidation method, etc., need to be defined upon creation of each dataset. In OpenTSDB, InfluxDB, and DalmatinerDB, labels can be arbitrary created for each data entry in the form of key-value pairs. However, simple data labels are non-structured concepts and cannot provide high level semantics for the specification of what data is being stored. For example, for one measurement time series, we may need to know which system component it is for in the overall system architecture, what property it is measuring, what unit it is using, etc. Simple key-value pair representation will not be sufficient in describing all these multi-dimensional attributes for the data stream. Descriptor concatenation may help, but is too primitive and still cannot associate related semantics. In [29], we have developed a semantic model and tools for adding semantics to TSDBs with the focus on cloud system monitoring. SSN [10] is an advanced semantic model for sensors and measurement data description. Though these two works can be applied to the manufacturing sensor databases, additional domain specific ontology for manufacturing systems should be developed to bridge the semantic gap in existing TSDBs.

Monitoring data streams continuously flow into the database at a relatively high rate. Generally, it is impossible to expand the storage space continuously to accommodate all the data. Thus, most of the TSDBs provide a data consolidation mechanism to compress the older data so as to free up the space for the new data. RRDtool and Graphite support data consolidation and offer a few primitive consolidation functions, such as average, min, max, last, etc. to compress ancient data. Users can specify how many raw data points should be kept in the database and upon consolidation, which of the given functions should be used to compress data. OpenTSDB, InfluxDB, and DalmatinerDB offer a more flexible approach on consolidation. Users can specify the retention policy for ancient data consolidation. Users can specify the time threshold for data consolidation, i.e., data before the threshold will be "down sampled". Users can also define a continuous query to down-sample ancient data using the consolidation method defined in the query. Continuous query will be activated automatically and periodically by the database to consolidate data. The consolidated data can be interpolated to the original precision when queried. Though existing TSDBs provide data consolidation support, specific consolidation methods that can best preserve the data stream characteristics should be investigated and integrated with the TSDB systems.

Timing management and timestamps are important in TSDB. RRDtool and Graphite only support data series in fixed time intervals and they do not consider events. Also, they only consider numerical data with a min-max range. Their timestamps are at the second level, which is too coarse grained for some senor data where recording is done at a much higher frequency. Blueflood, OpenTSDB and DalmatinerDB work at the millisecond level and InfluxDB can work at the nanosecond level. Also, in InfluxDB, data can be stored as events and the data type of the time series data can be numerical or string.

TSDBs are in the development stage and some of the released systems are continuously enhancing their features. Specific needs in MFDD for manufacturing systems should be more carefully investigated so that the desired features can be incorporated in commodity and open source TSDBs.

5 Data Correlation Issues in MFDD for Manufacturing Systems

Some sensor data may exhibit normal and abnormal patterns and can be used for system anomaly detection. Various big data analysis and anomaly detection algorithms have been developed to achieve fault detection and diagnosis for individual sensor data streams. However, in some systems, some anomalies can only be detected from the correlation of data, not by the data from individual sensors. An example of fault diagnosis based on data correlation rules is shown in Figure 2 [18][30].

In this gas fuel system, sensors p_i and q_i monitor pressures and flows in corresponding locations, sensors *fag*, *fagr*, *fsgr* and *fsg* are the valves' positions, 96*hql* is the water pressure. The normal sensor data should be governed by a set of rules and some of these rules are given below:

$$q_2 = fsg\sqrt{fqg_2 - p_3}$$
$$fsg = fsgr\sqrt{p_1 - fqg_2}$$
$$fsg = f(fag, 96hql)$$
$$fsgr = f(fagr, 96hql)$$

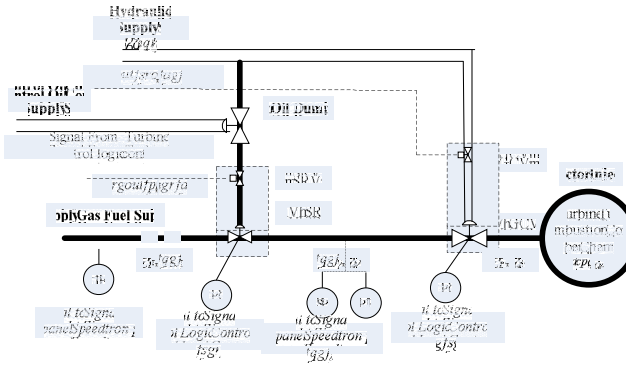

Figure 2. Flow diagram of GFS of GE Frame 6 Turbine (adopted from [18][30]).

Model-based FDD is a typical class of techniques for modeling this type of systems and their sensor data streams and performing fault detection and diagnosis based on the model. However, model-based approaches rely on the specification of the system architecture and sensor data correlation rules. This information is not always available for manufacturing systems. Even if the specification is available, it may not be complete. Also, manufacturing systems may evolve to adopt new technologies or upon the demand of new production needs. Moreover, with the new drive of customer-oriented industry, it is expected that manufacturing systems will be more and more agile and configurable. All these factors make it more challenging to maintain the architecture and activity flows for the systems. Thus, automated discovery mechanisms to derive the knowledge about sensor data correlations is highly desirable.

Data analytics for mining the correlation rules among sensor data streams. Discovery of data correlations may not be straightforward. If knowledge of which data streams are correlated is needed, then existing mining techniques may be able to help derive the relations among data series. Of course, there are still issues. For example, the timing of the data may not be in sync, which may impact data correlation derivation. Also, data quality such as missing data or noise in data may impact the proper discovery of data correlation rules. Some potential techniques, such as using interpolation to obtain missing data or to derive data at the desired synchronization points, may be developed. Also, techniques such as multivariate statistical analysis [17] can help solve data correlations if the data dependencies are linear. But many physical laws are nonlinear. Nonlinear multivariate analysis [31] may not scale well and further study on its scalability for modern manufacturing systems should be investigated. Faster analysis algorithms should also be developed to support such data correlation discovery.

Mining the system model. For a system with a large number of sensors, it will be difficult to rely purely on data analytics to discover sensor data correlations. Thus, we have to go back to discover the system model and derive the potential sensor data correlations from the complete or partial system model.

Some research works attempt to automate the creation of system specifications. In [32], the system workflow is created by analyzing the timing correlations of the sensor data and discovering the potential "rippling effects". When the data stream of sensor x shows the pattern of an event occurrence, if the data stream of sensor y also exhibits data pattern change shortly after x's event, indicating an event occurrence (as shown in Figure 3), then it can be inferred that sensor y is associated with a component that executes after the component which sensor x is associated with. In [33], a case study is given to show the feasibility of the approach in correctly discovering the system workflow.

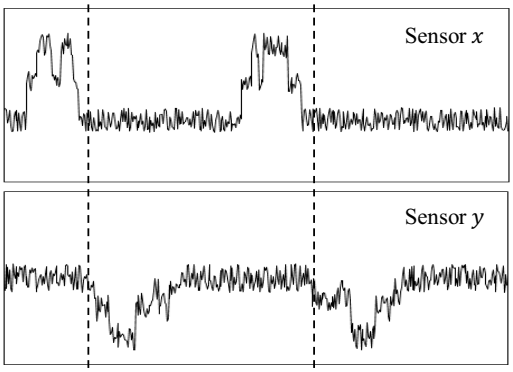

Figure 3. Example sensor data for system spec mining.

Though the "reverse engineering" approach for creating system specifications is very intriguing, the applicability may be limited. First, the timing for the "ripple effects" may not be good enough for such detection. If the "ripple latency" is not much smaller than the event occurrence period, then it is not possible to determine the ordered correlation of the sensors. Second, if the fault diagnosis system places sensors at the subsystem level and the subsystems have overlapping components, then the sensor data have correlations that are beyond the timing effects, making the approach infeasible. Third, the analysis may be able to discover the workflow of simple systems. For systems with many concurrent activities, the approach will not be able to determine whether components are activated concurrently or it is just the timing problem as discussed in the first case.

In our MFDM framework, we consider to develop a "System spec Miner" which mines the "System spec" repository to find similar physical systems based on some informal descriptions of the system. Documents of similar systems can be retrieved and modified to create the desired system spec. In addition to text based descriptions, images of the physical system and videos of the system operations can also be submitted to the spec miner to achieve better mining. The spec miner can also cluster the systems based on their specifications and develop system patterns. These patterns can be reused to help with system specification construction.

Even if the "System spec Miner" can only discover an

incomplete or even distorted system model, it can serve as a guideline to help reduce the complexity with data correlation mining. Validity of the mined system spec can be validated by mining the correlations between the actual data streams under the guideline of the incomplete system model. Also The system spec can be captured in a fuzzy model to indicate its uncertainty.

6 Conclusion

In this paper, we investigate the monitoring, fault detection and diagnosis issues in manufacturing systems using IoT and big data technologies. First, we discuss the design of a SaaS-centered MFDM framework, which aims at offering configurability so that most manufacturing systems can make use of the SaaS provisioning for their MFDD tasks. We also raise two technology gaps that should be considered in realizing MFDM SaaS. First, data storage for time series data is still in the development stage. Issues such as scalability, semantic support, space efficiency, consolidation scheme, etc., should be investigated. Second, how to mine the correlated data when there are a large number of sensors? Better solutions are needed for these aspects.

We plan to identify a few case studies of manufacturing systems and use them to evaluate the feasibility of the MFDM framework design. In the evaluation process, we will identify some previously overlooked issues and improve the framework design to address them. We also plan to perform in depth evaluation of existing TSDBs in supporting MFDM data storage and processing and improve them for achieving better health management of manufacturing systems. In addition, we plan to develop new techniques to cope with the data correlation discovery problem and apply them to real manufacturing data for validation.

7 References

[1] A. Dunkels, B. Gronvall, T. Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors". LCN 2004, pp. 455-462.

[2] Arduino, 2011, http://arduino.cc

[3] S. Alam, M. M. R. Chowdhury, J. Noll. "Senaas: An event-driven sensor virtualization approach for internet of things cloud". NESEA 2010, pp. 1-6.

[4] A. Cannata, M. Gerosa, M. Taisch. "Socrades: A framework for developing intelligent systems in manufacturing". IEEM 2008. pp 1904-1908.

[5] J. Soldatos, N. Kefalakis, M. Hauswirth, et al. "OpenIoT: open source Internet-of-Things on the cloud". InterOSS-IoT Workshop 2014. pp. 13–25.

[6] S. Ding. Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. Springer Science & Business Media.

[7] R. Isermann. "Model-based fault-detection and diagnosis–status and applications". Annual Reviews in Control, Vol.29, No.1, 2005, pp.71-85.

[8] S. Cox. "Geographic information - Observations and measurements", 2013, http://portal.opengeospatial.org/files/?artifact_id=41579

[9] M. Botts, A. Robin. "OGC SensorML: Model and XML encoding standard", The Open Geospatical Consortium Inc., 2014, http://www.opengis.net/doc/IS/SensorML/2.0

[10] M. Compton, P. Barnaghi, L. Bermudez, et al, "The SSN Ontology of the semantic sensor networks incubator group", Web Semantics: Science, Services and Agents on the World Wide Web, 2012, pp. 25-32.

[11] L. Lefort. "Ontology for quantity kinds and units: units and quantities definitions". W3C Semantic Sensor Network Incubator Group, 2005.

[12] E.Baroth, W. Powers, J. Fox, et al. "IVHM (Integrated Vehicle Health Management) techniques for future space vehicles." In 37th Joint Propulsion Conference and Exhibit, p. 3523. 2001.

[13] A. Aamodt, E. Plaza. "Case-based reasoning: Foundational issues, methodological variations, and system approaches." AI communications, Vol.7, No.1, 1994, pp. 39-59.

[14] H. Niemann, N.K. Poulsen. "Active fault diagnosis in closed-loop systems." IFAC World Congress, Volumes 38, No. 1, 2005, pp. 448-453.

[15] T. Sorsa, H.N. Koivo, H. Koivisto. "Neural networks in process fault diagnosis." IEEE SMC, Vol. 21, No. 4, 1991, pp. 815-825.

[16] O. Niggemann, G. Biswas, J.S. Kinnebrew, et al. "Data-driven monitoring of cyber-physical systems leveraging on big data and the Internet-of-Things for diagnosis and control." DX@ Safeprocess, pp. 185-192. 2015.

[17] D. Slišković, R. Grbić, and Ž. Hocenski. "Multivariate statistical process monitoring." Tehnicki Vjesnik-Technical Gazette, Vol.19, No.1, 2012, pp. 33-41.

[18] Y. Zhang, I.L. Yen, F.B. Bastani, et al. "Optimal adaptive system health monitoring and diagnosis for resource constrained cyber-physical systems". ISSRE 2009, pp. 51-60.

[19] RRDtool, http://oss.oetiker.ch/rrdtool/

[20] Graphite, https://graphiteapp.org/

[21] BlueFlood, blueflood.io/

[22] OpenTSDB, opentsdb.net/

[23] InfluxDB, https://www.influxdata.com/

[24] DalmatinerDB, https://dalmatiner.io/

[25] Elasticsearch, https://www.elastic.co/

[26] RocksDB, http://rocksdb.org/

[27] LMDB, http://www.lmdb.tech/doc/

[28] ZFS, https://en.wikipedia.org/wiki/ZFS

[29] S. Zhang, I.L. Yen, F.B. Bastani. "Toward semantic enhancement of monitoring data repository." ICSC, 2016, pp. 140-147.

[30] L. Trave-Massuyes, T. Escobet, R. Milne. “Model-based diagnosability and sensor placement application to a frame 6 gas turbine subsystem,” IJCAI, Vol.1, 2001, pp. 551-556.

[31] B. Schölkopf, A. Smola, K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural computation, 10(5), 1998. Pp. 1299-1319.

[32] O. Niggemann, B. Stein, A. Vodencarevic, et al. "Learning Behavior Models for Hybrid Timed Systems". AAAI 2012, Vol. 2, pp. 1083-1090.

[33] A. Vodenčarević, H.K. Bürring, O. Niggemann, et al. "Identifying behavior models for process plants." ETFA 2011, pp. 1-8.