Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 93 (2020) 1346–1351

www.elsevier.com/locate/procedia

53rd CIRP Conference on Manufacturing Systems

Combining multiple data sources and enriching the dataset using Industrial Edge Devices

Stefan Dumss [a,*], Manfred Grafinger [a], Martin Hennig [a], Patrick Rosenberger [a]

[a]*TU Wien, Institute for Engineering Design and Product Development, Getreidemarkt 9/307, Vienna 1060, Austria*

* Corresponding author. Tel.: +43-1-58801-3072167; fax: +43-1-58801-30798. E-mail address: stefan.dumss@tuwien.ac.at

Abstract

Industrial edge devices starting to play a major role as API gateway for data consumers or cloud applications. To fulfill this task industrial edge devices are mostly build from single board computer or small computers with an operation system. Considering the increasing processing power of these computers there is room for applications that enable data preparation for data consumers on the shop floor. This paper proposes a lightweight approach to structure data with the SAX Algorithm and add semantic information's to the dataset. Additionally, an example using machine data gathered from a manufacturing system with MQTT is provided.

Keywords: Industry 4.0; Time Series Preprocessing; Industrial Internet of Things; Industrial Edge Device; Data Streams

1. Introduction

Data and its interconnection represent the backbone of the 4th industrial revolution. In manufacturing, the data originates from diverse sources, ranging from simple electronic sensors to machines with OPC-UA and data from industrial information systems to complex cyber physical systems. Interconnecting and combining this data is of great importance and highly challenging in Industry 4.0 and has so far mostly been done in local industrial information systems or in cloud solutions [1]. A large part of this data is generated on the edge of the network of production machines and supported workplaces. Following the current trend, this data is converted on so-called industrial edge devices or industrial internet of things gateways from analog or digital interfaces to TCP / IP-capable protocols. Based on the increasing processing performance of single board computers and their cost efficiency, there is the possibility of performing increasingly demanding tasks on an industrial edge device. While there is already a large number of standards for the communication of edge devices to the industrial machinery, the cloud and each other, there are only a few guidelines on how the data could be combined and enriched on the edge of the network [2]. Therefore, this paper presents an approach using Symbolic Aggregate approXimation (SAX) combined with data retrieved from industrial information systems (IIS) like enterprise resource planning to enrich the dataset with additional context while at the same time reducing the size of time series (TS) needed for saving or transmitting the representative data. The approach present makes a significant contribution to the interconnection and processability of the data with focus on time series retrieved from manufacturing systems., since all available machines data and measurements are processed on the edge device with low latency. The outline of this publication is as follows: Section 2 covers the related work and shortly summarizes SAX. Afterwards, section 3 introduces the approach taken for data preparation and combining. Section 4 presents a use case in which the proposed approach has been applied and tested. Finally, section 5 finishes with a conclusion of the findings and an outlook on future work.

10.1016/j.procir.2020.03.081

Nomenclature

TS	time series
PAA	piecewise aggregate approximation
SAX	Symbolic Aggregate approXimation
C	original TS
w	dimension of the aggregated TS
Δt	time interval
Δt_d	desired time interval
n	amount of values in the original TS

2. Related Work

2.1. SAX

SAX is an approach that converts TS into a dimensionally reduced series of symbols [3]. The approach is based on assigning letters to a discretized piecewise aggregate approximation (PAA). SAX consists of three steps, first the TS is normalized, secondly the dimensionality of the TS is reduced, and finally the data is discretised by assigning symbols to intervals. SAX uses PAA for aggregating a TS (C) with n-values into a representation with w amount of values ($\overline{C}$) where each element ($\overline{c_i}$) is calculated with equation 1 [3].

$$\overline{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{1}$$

Therefore, $\overline{C}$ represents the dimensionally reduced time series C. For the third step in the original SAX paper, the gaussian distribution was used, and the discretisation breakpoints adjusted to the probabilities. In other words, if the TS is to be divided into three parts, the breakpoints are chosen so that each value has the same probability in one of the three areas. In order to distinguish them from others more easily in the following, these areas are named bins. Figure 1 shows an example of how letters are assigned to a TS in PAA representation.

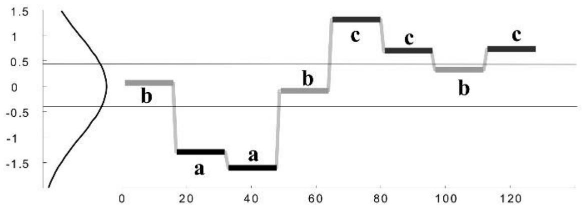

Fig. 1. PAA representation with added symbolic representation: baabccbc [3]

To measure the quality of such approaches, the difference between the original TS and its representation is measured by a distance metric [4]. The most common one is the Euclidean Distance. Since SAX is lower bounding, it is known how large the maximal error is that is done through the symbolic representation. Despite its lower-bounding property, SAX mainly reflects the overall trend of a time series instead of its exact values [5]. For TS issued from manufacturing machines, trends are often more relevant as additional information, since intermediate responses to certain thresholds are already addressed at a lower level on the machine.

Since SAX was invented in 2002, there were numerous improvements suggested to the algorithm [5]. Wang, L., et al. have identified that these improvements could be grouped into, “Improvement of Adaptive Segmentation and Similarity Calculation” and “Enhanced Sequence Representation” [5]. Enhanced sequence representation are improved SAX methods that focus on substituting the PAA with a more advanced option. One of the notable works in this area is SAX-TD which includes trends in the aggregated segments [6], and performs well on classification compared to SAX [5]. In the area on enhanced sequence representation ESAX targets the same idea. The approach uses three symbols to represent a segment with two of the symbols representing the minimum respectively the maximum value [7]. A different target with a similar approach was done with ISAX, where the symbol representing the aggregated sequence is substituted with a binary number that allows comparability of ISAX representations with different cardinalities [8].

There are also improvements for the computational scaling of SAX like approaches. In the area of edge devices, the most notable work is the distributed approach DPISAX from Yagoubi, D.E., et al. [9]. By distributing the computational needs to several edge devices, a large amount of values incurring at a plant could be processed at the edge of the network.

2.2. Role of edge devices in data processing and distribution

Within the field of industry 4.0 many different people and systems consume data created at manufacturing machine level. M. Hermann, et al. identified four general design principles: technical assistance, interconnection, decentralized decision and information transparency [10]. All these principles rely on data being available in the according system. Usually the data is connected bottom up with the cloud on the top, whereas each system connects several systems below [11]. Data models of industrial information system are optimized to fulfil the task they are designed for. Every additional interconnection the system must provide is leading to an extension of the data model, this can cause an unnecessary overhead. Even worse, some vendors use the dependence on their system to a certain extend for vendor lock in.

Cloud applications have been in the focus for years but have issues such as high latencies, lack of context awareness, and data transmission overhead with bandwidth limit, global centralization of data processing, and storage [12,13]. Additionally, real time behaviour is especially important in the manufacturing domain, whereas information may have to be exchanged within milliseconds or even microseconds [12]. Tasks like synchronization, control storage, assign computation and machine learning are placed on the edge device [12,14]. Industrial edge devices could also help to compute in-situ intermittent batch jobs and continuous data streams, such as to eliminate duplicate copies, compress logs, or normalize data formats [15].

The approach presented here focuses on providing a workflow that allows reducing the size of the TS, putting it into context and offering support for later use for machine learning and users. For example, shop floor workers could be assisted with real time notifications and warnings to prevent risk of task failure [16] or dashboards providing data for visualizations specific to role, task and level of decision makers [17]. TS are the main source for machine learning applications in the manufacturing context. They are important in making predictions on the process state depending on the assessment of a current or previous state, showing presence of anomalies, and finding TS trends [18]. To interconnect all these systems, common communication standards like MQTT, REST and AMQP are vital, because these enable flexible combination of the data [11]. These standard does not define any payload and therefore do not define ways to reduce the size of the transmitted data. Currently a publish / subscribe approach is commonly used, especially MQTT plays a major role in industrial internet of things systems [19]. Since the presented approach does not focus on comparing the data reduction methods, only the savings between SAX and non-alternated data are provided.

3. Approach: Combining multiple data sources

The approach consists of five major parts. Beforehand connectors to the systems which should be included in the interconnected data on the edge device have to be build up. This is a challenging task, since not a lot of information system offer easily accessible interfaces. Following the clean architecture guide from M. Robert, interfaces, entities, use cases, and frameworks & divers should be separated from each other [20]. For edge devices a micro service architecture could be used. This is also beneficial since this architecture could be the starting point for a fog computing strategy in the future due to its promise for efficient data processing [21]. The presented approach consists out of four process steps and the interface to other systems. The process flow is shown in Figure 2. The services on the edge device should consists of connectors, interfaces (part I), data processors (part II, III, IV), and data distribution (part V).

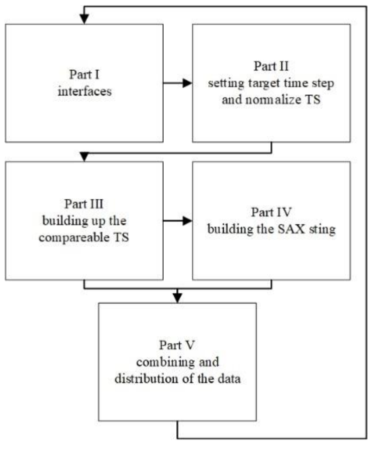
Part I
interfaces
Part II
setting target time step and normalize TS
Part III
building up the compareable TS
Part IV
building the SAX sting
Part V
combining and distribution of the data

Fig. 2. Process flow of the presented approach

3.1. Interfaces to other systems

Since the connection between an industrial information system and the edge device is a highly system specific task, a complete description is out of the scope of this paper. Further, the machine control is not covered, since this requires specific process knowledge. Rather, this paper focuses on determining how the data should be organized at the interface level. The interfaces should lead to three types of representation of data:

- Static information that never or barely changes, like machine name etc.
- Event based information that changes on a certain change of an information, like a new production order
- Information that can be displayed as times series or can be converted to TS, like sensor data or type of tool used

The static information are collected form IIS on the edge device but are not processed by data processor parts of the approach. In the last part of the approach, these changes of static data are added as meta information so that they can be processed by other applications. It may be that static information cannot always be clearly assigned a specific time on the edge device, how the approach presented here can help to connect this information follows in the part V.

Event-based information are easier assigned to time series, since events also need a point in time at which they have been triggered or occurred. Regardless of the selected system architecture on the edge device, it can be important for events that they are forwarded in real time. Therefore, this should be considered when designing the interfaces.

Time series reflect the core of the data considered in this approach. It is therefore essential that these are available with sufficient accuracy and allocation in the smallest possible time increments. Systematic errors can still be taken into account in the data preparation, but unknown deviations can only be corrected through cross-correlation, process models or machine learning. All TS (C_i) are used as input for the data processors, but TS with any flaw that cannot be corrected cannot be combined with the others and may use a higher resolution for better retrospective inspection.

3.2. Setting of target time step and normalize TS

The second part starts with the determination of which time step is relevant and how many time steps can be summarized in the data to be aggregated. In the optimal case, it is possible for all IIS that have subscribed a certain update interval ($\Delta t_{o,i}$) to find the largest common divisor as desired time interval (Δt_d) so that all $\Delta t_{o,i}$ are a $w_{o,i}$ multiple of Δt_d. As a consequence, all $\Delta t_{o,i}$, $w_{o,i}$, and Δt_d should be natural numbers ($\mathbb{N}$), for the most applications milliseconds are sufficient. After $w_{o,i}$ and Δt_d have been determined, any flaw in the TS (gaps, NaN, ...) must be corrected. The respective strategy can be determined depending on the process, or simply by linear interpolation between the neighbouring values.

The last step in part II consists of making the TS easier comparable in a sense of similarity measurements. Since TS can be compared better if they are normalized [22]. All TS with all their elements $C_i \in$ R and in the broadest sense gaussian distributed, should be statistical normalized with the z-score (μ=0 and σ= 1), this preserves the characteristics of the TS [23]. Of all other TS, their elements are viewed the same way as non-numerical values.

3.3. Building the comparable TS

In the third part the focus lies on retrieving the PAA representation of the TS. In the last step, the approach distinguishes if the TS values are real numbered or not. For the calculation of the PAA value equation 1 is used. For the ease of understanding the nomenclature is extended with the time step of a TS (C_i) as Δt_i and w_i as the TS specific representational amount of values out of the TS length n.

For all TS with $C_i \in \mathbb{R}$ and $\Delta t_d > \Delta t_i$:

- If $\Delta t_d = \Delta t_i,\ w_i = n$, the PAA can be directly calculated
- If $\Delta t_d \ mod\ \Delta t_i = 0$, w_i can be calculated with $w_i = \Delta t_d \cdot n/\Delta t_i$ as input for the PAA calculation
- If $\Delta t_d \ mod\ \Delta t_i \neq 0$, the values of the TS have to be recalculated with a new time step ($\Delta t_{i,new}$) and the constraints of $w_i = \Delta t_d \cdot n/\Delta t_{i,new}$ with $w_i \in \mathbb{N}$ and $\Delta t_d > \Delta t_{i,new} > \Delta t_i$ fulfilled. The calculated TS and w_i are the input for the PAA calculation

For the TS with $C_i \in \mathbb{R}$ and $\Delta t_d < \Delta t_i$, it is recommended to use process knowledge and recalculate the TS. There are various options for the recalculation like interpolation.

For the TS where either $C_i \notin \mathbb{R}$ or C_i is not a number, the values on each of the last time step in the viewed time interval should be used. The value at time step t_j is calculated with equation 2. With the assumption that the values are not a number, PAA could not be used and the values should be either directly assigned with letters or mapped to letters after an aggregation.

$$c_{i,j} = C_i\left(\left\lfloor \frac{\Delta t_d \cdot t_j}{\Delta t_i} \right\rfloor\right) \tag{2}$$

As already mentioned, TS with uncorrectable flaws are a special case, therefore the resolution of the values should be higher, the Δt_d should be its integer multiple of the new time step.

3.4. Building the symbolic representation

The fourth part is about discretising the PAA of the TS and applying a letter to each interval. Our tests with that approach on several machine TS have shown that not for all TS it is feasible to just follow the SAX approach from Lin, J., et al. [3]. If the distribution of the TS is skewed the z-score will also be skewed [24], this leads to high Euclidean distance errors on the extreme values of the TS. There are two ways to approach this:

- Test the TS values on highly gaussian distribution before PAA. If they are skewed either use a method to address the skew and recalculate or use a custom discretisation.
- Discretise the TS, convert it back to a TS and check the error. If the error is too high, use a custom discretisation

For the custom discretisation, a uniform approach could be used, where the bins have identical width over the whole range of possible values. As with original SAX, the lowest area is represented with the first letter. When comparing SAX strings, it is important to take into account that normally distributed bin and uniformly distributed bins lead to different SAX-strings. If the majority of the TS are skewed it may be better to use uniform for all TS. The easiest solution would be to add the minimum and maximum value to the breakpoint list and then take the midpoint between two values as the numerical representation of an interval in the bins. It is recommended to not use a high variety of custom discretisation approaches since all have to be decoded before distribution. Further, changing the breakpoints makes comparability harder.

The last step of the data processing is to gather all needed parameters so that the SAX string could be decoded in the future. If the original TS is saved, for example in case of the TS with time offset, a temporarily download link to the TS could be added. For normalized TS it is recommended to calculate the values of the bins on the edge device, since it is still known how they were calculated.

Besides the SAX string, the bins values, and how they were created, the mean and the standard deviation in the subscribed time interval of the TS before the normalization are needed. For some cases the minimum and maximum values that occur in the TS are also interesting. Furthermore, to distinguish between the TS, labels should be added.

3.5. Combining and distribution of the data

In part V the SAX strings and additional information have to be combined together in a representation model of the relevant machine data and distributed to systems (e.g. IIS) interested in the data. Events and TS have to be distributed depending on the aimed usage. In addition, limit values from time series can trigger events. It has been shown that a publish/subscribe messaging protocol like MQTT or AMQP are widely established in IoT applications for data distribution [19, 25].

As part I has already distinguish the data type incoming into the edge device, the active outgoing type is mostly event based. Events can be triggered by subscribing to certain time intervals or events that have to be forwarded. The content of this to publish messages consists of combining the static, event and TS data.

The SAX strings from part IV can easily be summarised as all have been normalized and have the same Δt_d. The output from part IV already described can be used directly, adding the data and time when the TS started. The other TS that are not considered gaussian distributed and have the same Δt_d can be added with providing equivalence list of assigned letters to values or areas. If areas are used there may also be a temporary download link added to the original values.

To minimize the overhead of data transmission, static data has only to be included if it is changed between two publish periods. Static data is normally of a type that has to be interpreted either by a human or encoded for machine learning. Therefore, a short and descriptive string should be chosen.

The events that only occur once could be added in the same way as static information. All the other events that occur more often should be converted the same way as the discrete TS in part III.

In summary, the message published to the subscribers on the selected time interval should consist of the following information:

- Message identifying values: machine id, time interval covered, type of message
- Updated static values: updated value, time of change, possible additional information where the change was issued
- Data processing information: desired time interval (Δt_d), number of values
- A list of TS aggregated with SAX and gaussian distributed bins: SAX string, mean, standard deviation, minimal and maximal value
- A list of TS with SAX and other than gaussian distributed bins: SAX string, mean, standard deviation, already decoded list of letters and assigned values
- A list of TS data with simple discrete later representation: representative string, letters representations list, minimal and maximal value
- Events that don't have to be forwarded immediately: message, time of occurrence, origin of the event

The exact representation on the data could be system depended, there are several established options for that, for example a lightweight approach is UNIDE an eclipse project which proposed a Production Performance Protocol (PPMP) that defines rules for combining multiple data sources. Other industry standards are also feasible as long as the support aggregation of information in as few as possible messages.

4. Use Case

The presented approach was validated using datasets from the TU Wien pilot factory [26]. Since SAX already has been proven to work well on a variety of TS compared to other representations [4], targeting the SAX representation quality is not the aim of this work. To illustrate the approach presented, a small extract of TS from the pilot factory is used. Two small sample of these normalized TS are shown in figure 3 and 4. As the approach does not depend on the datatype or source, the real valued parameters are named value I and II. Following the approach presented here, the interfaces for additional information have to be provided. The use case assumes that there exists an interface to an IIS that delivers parameter A. In addition, there are two data consumers (1,2) in place which require data every five respectively ten time steps. Therefore, every five timesteps a SAX string is build. The SAX string with uniformed bins for the value I is: "EEFFF, FFEED, DCCCC, CBBBB, BBCBB, BBAAA" and for value 2 it is:

"FFFFF, EDCCD, CCCCC, DCCDC, BCDCC, CBAAA" To visualize the SAX string is decoded in figure 3 and 4.

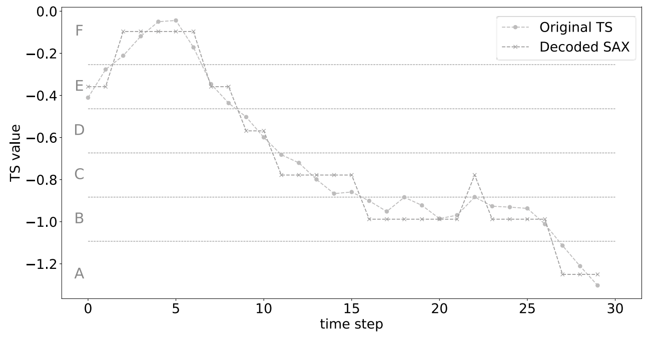

Fig. 3. Graphical representation of value I

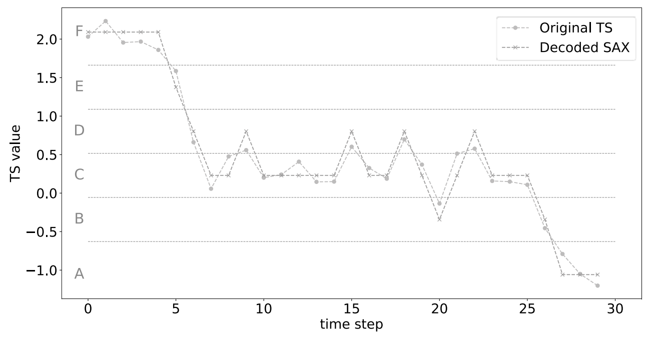

Fig. 4. Graphical representation of value II

To prepare the messages for publishing, the state changes of parameter "A" and the node values have to be added. The visual representation already shows the result of the string decoding for comparison purposes. Since the TS are normalized, they are easily comparable. Both, the TS have about the same trend which may be an indicator that the two are corelated. A check if the parameter "A" has changed around a time step could lead to the cause of the trend.

Regardless of the representation's advantage for trend analysis, this approach can also save storage space or transmission bandwidth compared with the traditional time series storage as CSV, tables or simple compression algorithms like gzip. If the values for machine status, date and time from start and end, and machine ID are added to the two TS mentioned above. The outcome of the presented approach represented in JSON using a gaussian distribution for the bins (bin values are not added to the JSON) and added additional information requires 630 bytes and the original values as two CSV files needs 635 bytes without any additional information. In the case of the uniform distribution of the bins used in figure 3 and 4, the size is 827 bytes, since the bin values are appended to the JSON. It follows that a breakeven point can be reached even in very small TS. For large TS sets this is a significant saving, for example ten TS with 20 000 values need about 2MB as CVS, whereas the representation of the presented approach in JSON needs only 481KB. Therefore, the gains on saving data store increased with the amount of data points. For uncompressed data a saving up to 80% depending on the structure of the data can be achieved. Specialised compression methods may change this ratio, but for common compression methods like rar, zip, or gzip it approximately stays the same.

5. Conclusion

The presented approach has shown the advantages of data processing on the edge of the network. The process flow consists of five parts and establishes a procedure that allows the combination of multiple data sources on the network's edge with respect to the needs of the individual data consumer while reeling on the same data source. This brings several advantages for the internet of things in production systems. A trend analysis using SAX, combined with additional information, allows the rapid detection of unwanted trends and their causes. Further, edge devices are inexpensive to buy and use, so it is possible to equip any machine with them. This enables them to be purchased in addition to the existing infrastructure, as tests have shown that some machines react to too many OPC UA requests with an interface slowdown.

This approach helps clustering the TS by bringing it in a form which can be directly analysed with both TS processing algorithms and text mining approaches. The SAX strings can be analyzed for reoccurring structures as if they were words. These words can be put in context to each other and therefore form a kind of sentence structure, which can be interpreted as a higher structure of the TS.

The approach could be improved, if it targets two of the upcoming applications such as real time data stream processing and machine learning on the edge device. In particular, setting the desired time interval (Δt_d) based on trained machine learning models could highly improve the efficiency of the approach. Since we used JSON over MQTT a letter representation of the aggregated sequence was fine due to JSON's limitation on data types the representation has to be a string. A protocol that supports binary numbers would enable ISAX to further reduce the needed transmission size without the further loss of accuracy.

Although the presented approach has proven to reduce the size of saved and transmitted data, it should be further investigated which alternatives for SAX in Part IV of the approach could be used. Especially lossless reduction algorithms should be part of further work. Furthermore, a long-time study on the overall system impact is needed to quantify the overall system performance impact.

References

[1] Chen B., et al., Smart factory of industry 4.0: Key technologies, application case, and challenges, IEEE Access, 6 (2018) pp. 6505-6519.

[2] Serrano M., Gyrard A., A review of tools for iot semantics and data streaming analytics, Build. Blocks IoT Anal, 6 (2016) pp. 139-163.

[3] Lin J., et al., A symbolic representation of time series, with implications for streaming algorithms, Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, ACM, 2003, pp. 2-11.

[4] Wang X., et al., Experimental comparison of representation methods and distance measures for time series data, Data Mining and Knowledge Discovery, 26 (2013) pp. 275-309.

[5] Wang L., et al., Survey of methods for time series symbolic aggregate approximation, in: X. Cheng, W. Jing, X. Song, Z. Lu (Eds.) Data Science, Springer Singapore, Singapore, 2019, pp. 645-657.

[6] Sun Y., et al., An improvement of symbolic aggregate approximation distance measure for time series, Neurocomputing, 138 (2014) pp. 189-198.

[7] Lkhagva B., et al., Extended sax: Extension of symbolic aggregate approximation for financial time series data representation, DEWS2006 4A-i8, 7 (2006).

[8] Shieh J., Keogh E., ISAX: Indexing and mining terabyte sized time series, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, pp. 623-631.

[9] Yagoubi D.E., et al., Dpisax: Massively distributed partitioned isax, 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 1135-1140.

[10] Hermann M., et al., Design principles for industrie 4.0 scenarios, 2016 49th Hawaii International Conference on System Sciences (HICSS), 2016, pp. 3928-3937.

[11] Zuehlke D., Smartfactory—towards a factory-of-things, Annual reviews in control, 34 (2010) 129-138.

[12] Li C., et al., Edge-oriented computing paradigms: A survey on architecture design and system management, ACM Comput. Surv., 51 (2018) pp. 1-34.

[13] Rosenberger P., Gerhard D., Context-awareness in industrial applications: Definition, classification and use case, Procedia CIRP, 72 (2018) pp. 1172-1177.

[14] Chen J., Ran X., Deep learning with edge computing: A review, Proceedings of the IEEE, 107 (2019) pp.1655-1674.

[15] Li C., et al., Towards sustainable in-situ server systems in the big data era, 2015, pp. 26.

[16] Belkadi F., et al., Intelligent assistant system as a context-aware decision-making support for the workers of the future, Computers & Industrial Engineering, 139 (2020) 105732.

[17] Mahmoodpour M., et al., Role-based visualization of industrial iot-based systems, 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), IEEE, 2018, pp. 1-8.

[18] Antipov S., et al., The problem of the anomaly detection in time series collections for dynamic objects, International Conference on Intelligent Information Technologies for Industry, Springer, 2018, pp. 106-117.

[19] Happ D., Wolisz A., Limitations of the pub/sub pattern for cloud based iot and their implications, 2016 Cloudification of the Internet of Things (CIoT), 2016, pp. 1-6.

[20] Martin R.C., Clean architecture : A craftsman's guide to software structure and design, Boston : Prentice Hall, Boston, 2018.

[21] Peralta G., et al., Fog computing based efficient iot scheme for the industry 4.0, 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2017, pp. 1-6.

[22] Keogh E., Kasetty S., On the need for time series data mining benchmarks: A survey and empirical demonstration, Data Mining and Knowledge Discovery, 7 (2003) pp. 349-371.

[23] Mitsa T., Temporal data mining, Chapman and Hall/CRC2010.

[24] Park H., Jung J.-Y., Sax-arm: Deviant event pattern discovery from multivariate time series using symbolic aggregate approximation and association rule mining, Expert Systems with Applications, 141 (2020) 112950.

[25] Naik N., Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http, 2017 IEEE International Systems Engineering Symposium (ISSE), 2017, pp. 1-7.

[26] Hennig M., et al., TU Wien pilot factory industry 4.0, Procedia Manufacturing, 31 (2019) pp. 200-205.