Designing a recommendation function for semantic merging of middle-of-life data streams

Abou Eddahab-Burke, F.

DOI
10.1080/21681015.2025.2481302

Publication date
2025

Document Version
Final published version

Published in
Journal of Industrial and Production Engineering

JOURNAL OF INDUSTRIAL AND PRODUCTION ENGINEERING
https://doi.org/10.1080/21681015.2025.2481302

Taylor & Francis
Taylor & Francis Group

Designing a recommendation function for semantic merging of middle-of-life data streams

Fatima-Zahra Abou Eddahab-Burke

Faculty of Technology, Policy and Management, Delft University of Technology, Delft, The Netherlands

ABSTRACT

This paper addresses the challenge of handling large and diverse data streams from connected products during middle-of-life use. Current data analytics tools struggle with such data, necessitating the development of a crucial function in next-generation data analytics tools that semantically merges and analyzes these streams collectively. The proposed function, recommendation for semantic merging of middle-of-life, acquires, pre-processes, and merges data from various streams, providing designers with enhanced product information. Tested on simulated data streams of a washing machine enhancement case, the function offers more comprehensive product insights than individual sensor analysis. Implementation of such a function could improve fidelity in reflecting product conditions, reduce sensor analysis time and effort, and deliver an actionable plan for product improvement. Being design-focused, this article addresses the functional validity and the proof of concept of the recommendation function. Full computational analysis is out of this paper's scope and will be addressed in future research.

ARTICLE HISTORY
Received 24 January 2024
Revised 17 February 2025
Accepted 14 March 2025

KEYWORDS
Data analytics; middle-of-life data; data merging; semantic interpretation; product designers

1. Introduction

Traditional data analytics tools cannot be applied directly to big data [1] or to managing big data to extract practical knowledge from them [2]. This points to the need for novel and sophisticated data analytics tools. In a broader sense, what is actually needed is to add smartness and learning capabilities to a wide range of computer systems (including data analytics tools and toolboxes) [3]. To cope with the knowledge gap found in the literature concerning smart data analytics toolboxes (SDATB), we integrated the findings of a questionnaire-based interrogation (QBI) presented in [4] and the synthetic theory devised using the axiomatic theory fusion (ATF) methodology presented in [5] whose combination provides requirements needed to build a SDATB. While the QBI explored what white goods designers missed related to the use and outputs of the marketed data analytics tools in the context of possible product improvements, its objective was to determine the data analytics approaches and tools used by the participants and for what purpose, as well as which of these tools they found useful. The study also investigated designers' expectations related to new data analytics tools and the extra technical support they would like to have via these tools. The ATF, a practical approach to theorizing in a deductive manner [6] was used to merge five independent theories needed to build a comprehensive conceptual basis for a knowledge platform for a next-generation data analytics toolbox for white goods designers. These theories are: a theory describing white goods designers' needs [4], a theory describing advanced technological enablers [7], a theory explaining the evolution of data analytics [8], a theory of combined creative problem solving [9] and decision-making [10], and a theory about functional and structural interoperability [11]. This integral body of knowledge from the QBI and the theory deducted from the application of the ATF was used to formulate operational requirements for a next-generation data analytics toolbox.

On the one hand, the QBI revealed that product designers need SDATB that allow (i) step-by-step assistance, (ii) advice in selecting means, (iii) multifold data visualization, (iv) multichannel data management, (v) blending middle-of-life (MoL) datasets, (vi) combining qualitative and quantitative data, (v) permanent accessibility, (vi) adaptation to users, (vii) case-based reasoning, and (viii) learning from applications. On the other hand, the devised synthetic theory was more concrete about the needs and the requirements for the SDATB. This theory suggested that the SDATB should include (i) context-driven decision-making, (ii) proactive decision-making, and (iii) algorithms able to process complex data. In addition, it should (iv) allow semantic interpretation, (v) blend MoL data (MoLD) and datasets, (vi) merge multiple MoL data streams (MoLD-S), (vii) allow high-speed and high-volume storage, (viii)

CONTACT Fatima-Zahra Abou Eddahab-Burke f.aboueddahab-1@tudelft.nl Faculty of Technology, Policy and Management, Delft University of Technology, Building 31, Jaffalaan 5, Delft 2628 BX, The Netherlands

provide permanent accessibility, (ix) deliver advice to designers based on their work context, (x) allow case-based reasoning, (xi) process structured, semi- structured, and multi-structured MoLD, (xii) propose solutions to solve difficult design problems, (xiii) predict future outcomes, and (xiv) derive actionable insights.

The listed needs should all be included in the design of an all-embracing SDATB that is tailored to designers needs. In this perspective, we decided to start with targeting the needs related to semantic interpretation of data analytics outputs, merging of multiple MoLD-Ss from different sources, predicting outcomes, and deriving actionable insights. All of these needs can be supported by one function of the SDATB that allows semantic merging of middle-of-life data streams. This function should not only computationally fuse MoLD-Ss but should also provide recommendations to the designer on what to do with the to-be-merged and merged data streams and how to do it. Therefore, we call it *"recommendation for merging MoLD-Ss"*, symbolized by F_{SB1}, as it is the first conceptualized function. The script "SB" stands for "smart basic" function. The basic functions are considered operational functions that are directly related to data analytics. Accordingly, they are derived from fundamental requirements that need to be fulfilled by the SDATB.

During recent decades, data fusion has evolved rapidly in various application fields [12,13]. Data fusion is a synthesis of incomplete information about environmental features provided by multiple data sources. The goal is to establish a relatively consistent and complete description through a more complete and accurate set of information [14]. Processing multiple concurrent data streams is an obvious task for the SDATB. By including a multisource data fusion technology, an SDATB can (i) eliminate redundant and contradictory data obtained from various sources, (ii) reduce the uncertainty of provided information, (iii) develop a nearly complete description of the monitoring environment, and (iv) enhance the accuracy of decision-making by intelligent systems [14]. It can also build better situational awareness and reasoning capabilities, as well as reduce its response time [15].

2. Literature study: collecting knowledge for designing the recommendation function for merging MoLD-Ss

Data merging is one aspect of data management [16]. Traditionally, it is interpreted as the total of theories, techniques, and tools used to combine sensor data into a common representational format [17]. Its main purpose is to combine data from heterogeneous data sources [18]. It is widely used in many application domains, such as robotics, industrial manufacturing systems, smart buildings, and healthcare [19]. Data merging is a wide-ranging subject that gave root to many different terminologies, which are often used interchangeably [19]. It can be found in the literature under different names, such as data fusion, data consolidation, or entity resolution [20].

The ultimate objective of data merging is to improve the performance of a system by merging complementary and/or redundant information to reduce the uncertainties of measurements and to obtain information that cannot be perceived within one data source [21]. In some publications, the term information fusion is considered a synonym for data merging or fusion [22], whereas in other sources a clear distinction is made between them. Data merging is employed for raw data (not processed), and information fusion is used to define processed data [23]. Accordingly, information implies a higher semantic level than data [24]. Several types of data merging and fusion have been the focus of many research projects, such as decision fusion, data combination, data aggregation, sensor fusion, and multi-sensor data fusion [25].

In our research, we are interested in benefiting from processing MoLD, since research has shown interest in analyzing such rich use, service and maintenance data [26]. This data is collected when the product is currently in use by the consumer. It includes failure, performance, product age, operating environment, usage intensity, refund and replacement data as well as maintenance reports [27]. Such data allows observation of conditions and behaviors of products during use by the costumer [28]. Collecting MoLD encourages a lifecycle-oriented approach to incremental product design that continuously enhances products and services [29]. Collecting MoLD allows for improvement of a product or product operations in various ways, such as design enhancements and the optimization of maintenance operations [30]. Consequently, the merging concerns data collected from product sensors while the product is in use. This type of merging is called multi-sensor data fusion [31] or multi-sensor data merging (MSDM) [32], which is part of multi-source data fusion [33]. MSDM is rapidly evolving [33]. It refers to the process of integrating and combining information from multiple sensors to achieve a more comprehensive and reliable insights than what individual sensors can provide on their own [25]. Accordingly, its essence is combining data from multiple sources, and it helps provide access to information that cannot be provided by a single sensor or whose quality exceeds that of the information drawn from a single sensor [34]. The core principles of MSDM include sensor integration, data preprocessing, feature extraction, and fusion algorithms [35]. Sensor integration involves employing diverse types of sensors concurrently, each contributing distinct information about the target system. The combination of data from multiple sensors enables a more comprehensive

understanding of the system, thereby enhancing accuracy and reliability across different applications [36]. Data preprocessing is about preparing the data collected from the sensors for the merging by applying cleaning, filtering, then normalization techniques. This allows removal of noise, outliers and inconsistencies in the data [37]. Feature extraction involves determining relevant features from the preprocessed sensor data. These features represent informative inputs for the fusion algorithms, that support pattern recognition and decision-making tasks [38]. Finally, the fusion/merging algorithm that operates on 3 levels, data-level also called signal or sensor-level, feature level and decision-making level [39]. These levels refer successively to low, middle and high levels [40]. Low-level merging applies to raw data coming directly from sensors and is used for knowledge construction or for cooperating with other nodes on complementary activities. It can be realized by a low-level abstraction or by performing local operations in a temporal domain [41]. It's primary advantage is its ability to fully utilize the source data and mitigate information loss [42]. However, the processing of large volumes of raw data can be computationally demanding, leading to long processing time. If the aim is to analyze real-time data, then this low-level merging should not be chosen due to the extensive processing requirements of large datasets. Also, having a variety of sensors with potentially different characteristics, data types etc. makes the fusion task even more challenging. Methods of sensor-level fusion include Kalman filter [43], weighted average [44], machine-learning based fusion schemes [45], and deep-learning-based fusion schemes [46]. Middle-level merging works with the features of datasets and flows [47]. It is performed on preprocessed data or on information obtained by low-level merging of data. It can be realized by implementing feature extraction, pattern matching, or redundant computation operations [48]. This type of merging extracts relevant features from raw data obtained by individual sensors and the best ones are then merged as input to the output layer. Examples of techniques used for this type of merging are principal component analysis [49], canonical correlation analysis [50], and deep learning-based fusion methods [51]. In comparison with the previous type of fusion, this one allows the loss of less information. This makes feature-level fusion a favorite approach when it comes to situations where accurate and informative features are crucial for faults identification and characterization [52]. However, merging various features can result in a feature vector with high dimensionality, which may significantly increase the demand for computational resources, making the merging process resource intensive [53]. Also, not all features contribute equally to the merging outcome, potentially causing information loss. Accordingly, it is important to employ careful feature selection and weighting methods to optimize the merging outcomes and minimize information loss [35]. High-level entails unifying decisions obtained from various sensors to achieve a final decision or inference [54]. This type of merging is a sophisticated process that is implemented by performing semantic inference, executing complex reasoning, learning from and making decisions on sensor data, or exploiting cooperative patterns [55]. It provides significant flexibility, which stands out as one of its primary advantages [56]. It facilitates the use of different merging rules and algorithms based on specific application requirements, allowing the merging process to adjust to various scenarios and accommodate the different sensors characteristics and data sources [35]. Furthermore, it has robust anti-inference performance as it can handle conflicting information extracted from various sensors. By combining decisions from various sources, it can derive relatively reliable conclusions even when individual sensors produce inconsistent or contradictory outputs [57]. Approach of decision-level fusion includes fuzzy logic [58], neural networks [59] and Dempster-Shafer theory [60]. High-level data merging is computationally challenging and is difficult to realize for two reasons [61]. First, inferring semantic knowledge requires transforming a low-level representation of data and information into a higher-level representation. This transformation, however, typically suffers from the so-called information deficit. Second, having a system understand semantics assumes the system has (i) some manifestation of consciousness, (ii) a purpose, and (iii) an awareness of its surrounding and the state of knowledge. These characteristics are strongly related to human beings. Moreover, in such fusion, the decision-making system or the classifier are not allowed to train on the entire data simultaneously, as it relies only on the decisions provided from the individual sensors. This limitation can lead to a less-optimal performance if the decision-making systems has no access to the full dataset [62].

MSDM have replaced traditional information fusion systems, which involved user-owned and controlled sensor networks. In addition, they established systems and information architectures that are used for sensor tasking, data acquisition, fusion, dissemination, and decision-making [63]. It has been proved that using MSDM approaches is the only way to get the required amount of information with an expected level of intelligence [19]. MSDM approaches allow (i) an increased probability of detection, (ii) extension of spatial and temporal coverage, (iii) reduction of ambiguity, and (iv) improvement of system reliability and robustness [64]. Despite its multiple advantages, MSDM presents several challenges related to data imperfection, outliers and counterfeit data, conflicting data, data modality, data correlation, data alignment and registration, data association, processing framework, operational timing,

static versus dynamic phenomena, data dimensionality [65].

In our research, the objective was to elaborate a support function (F_{SB1}) for merging MoLD-Ss. It should merge computationally, semantically interrelated MoLD-Ss obtained from different sensors. In doing so, it helps the user gain additional information and knowledge from the streams to support decision-making in various contexts of product enhancement. After merging, the synthesized MoLD-S can be used to infer additional semantic information that initially was not conveyed by any one of the separate MoLD-Ss. Streams may complement each other and may provide additional semantic information when appropriate inference techniques are applied. The proposed function, F_{SB1}, adopts the principles of high-level MSDM. It contextualizes the information conveyed by MoLD-Ss and analyzes the information's meaning in that context. In fact, it extends to one level higher since it generates recommendations about possible enhancements. This knowledge can be deduced by analyzing the merged data streams in a specific context and can be delivered to the designer as a displayed message. Such a function is a genuine enabler of the smartness of the SDATB.

3. Conceptualization of the recommendation function for merging MoLD-Ss

The computational implementation of the function F_{SB1} for merging data streams has three elements. Symbolically, they can be expressed as

$$F_{SB1} = F_{SB1}(DI_{SB1}, CP_{SB1}, SO_{SB1}), \quad (1)$$

where F_{SB1} is the basic function providing recommendations for merging MoLD-Ss, DI_{SB1} are the inputs by the designer (MoLD-Ss), CP_{SB1} is the computational mechanism, and SO_{SB1} represents the outputs expected from the SDATB after execution of F_{SB1} (i.e. messages displayed to the designer about the results of the MoLD-Ss merging). The necessary computational procedures can be defined if F_{SB1}, as a main basic function, is decomposed to lower-level functions and related requirements are considered. The intermediate lower-level functions of F_{SB1} were already presented in the previous section. The underlying process is as follows. First, the SDATB acquires the MoLD-Ss selected for merging from the corresponding sensors in real time. Then, after the designer choses streams for further analysis, those streams are preprocessed individually based on their data. The preprocessed MoLD-Ss are then fused together. The following step focuses on detecting anomalies in the merged data streams and determines what might be wrong with the product based on data. Once the meaning is given to the fused MoLD-Ss, the SDATB derives recommendations on what should be done with the product (such as enhancement possibilities). Finally, this recommendation is sent to the designer as a message appearing on the computer screen.

The procedural steps of merging MoLD-Ss (function F_{SB1}) are shown in Figure 1. This function decomposes to five sub-functions. A lower-level functional decomposition of F_{SB1} is summarized in Figure 2. We have assumed that the designer specifies for the SDATB what data streams will need to be merged. Another assumption is that only temporally finite data streams are handled by the SDATB. This latter assumption facilitates the application of machine learning. The sub-function $F_{SB1,1}$ locates the considered sensors on the product and forwards the data streams provided by them to the SDATB. Our assumption is that the forwarded MoLD-Ss may be stored not only on the background storage devices of the SDATB host computer but also on a separate storage device. To get a reconfirmation from the designer, sub-function $F_{SB1,2}$ presents the data streams to the designer using various means to visualize the MoLD-Ss (for example, plots or histograms). In addition, the sub-function preprocesses the single-modality data streams by selecting particular processing rules. As an example, some rules can eliminate parts or the whole of redundant data streams that are not likely to affect the merging. To avoid the need to transfer and process vast amounts of idle information, the rules may operationalize up/down sampling of values, value transformation, and reducing noise in the data to decrease unnecessary variance of the data to be processed. The sub-function $F_{SB1,2}$ applies a kind of configured data processing, which is required because of the time-consuming nature of processing the data. In this context, "configured" indicates that, for complicated data streams with unknown patterns, comprehensive structural preprocessing (filtering or ordering) is applied, whereas for less complicated data streams, preprocessing is simply data normalization.

The computational merging of data streams is done by sub-function $F_{SB1,3}$. The principle of fusion is correlation based on the time stamps of data in the streams. First, the sub-function generates intermediate representations to reduce time-dependent data to

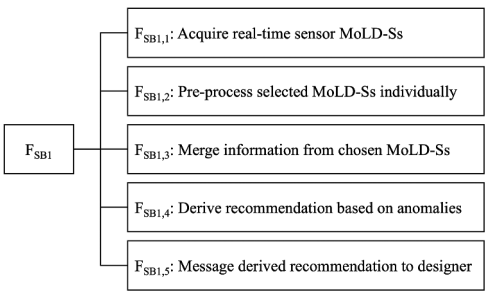

Figure 1. High-level functional decomposition of F_{SB1}.

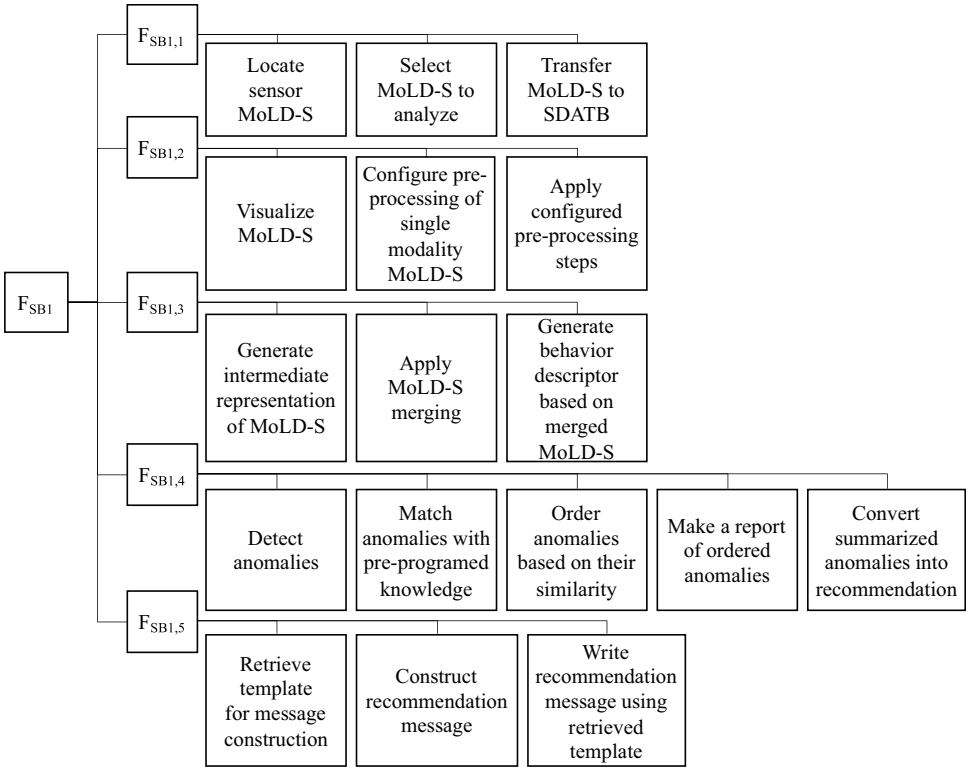

Figure 2. Low-level functional decomposition of F_{SB1}.

a compact fixed-length vector. Then it combines the data streams and generates a behavior descriptor based on the merged MoLD-Ss. To facilitate the application of machine learning, subfunction $F_{SB1,3}$ embeds the fused sensor data streams into a so-called latent space (also called a hidden space). In this space, data is mapped in such a manner that similar data points are close to each other. In the case of neural network-based machine learning, features are extracted through a number of layers of the network architecture, and the operation (function) that maps the input before the last layer projects into the latent space. In other words, the features lie in the latent space. The latent space representations can be used to transform complex forms of raw data into simpler forms that are easier to analyze. Mapping to the latent space also helps in clustering similar cases.

The data streams may contain anomalies regarding the operation of the product in question. The sub-function $F_{SB1,4}$ (i) detects anomalies in the merged data streams, (ii) matches the anomalies to pre-programed knowledge in the SDATB, (iii) orders the anomalies based on their similarity, (iv) makes a report on all of the ordered anomalies based on the merged MoLD-Ss, and (v) converts the outcome into a specific recommendation. The last sub-function, $F_{SB1,5}$, (i) retrieves a template for message construction, (ii) constructs a message for the designer according to the recommendation, (iii) uses the retrieved template to construct the message to be delivered to the designer, and (iv) communicates the message to the designer relating what is improper with the product according to the merged data. In the case of the SDATB, this can also be followed by a recommendation for actions to take to solve detected anomalies in product operation, although this step is not indicated in Figure 2.

To realize the function of $F_{SB1,1}$, the SDATB needs to (i) locate the sensors producing the MoLD-Ss, (ii) identify and access the data streams to be merged, and (iii) import these streams from their storage place (for example, the cloud) to the SDATB. Moving MoLD-Ss from external storage into the SDATB is a common procedure (several commercial tools allow retrieval of data streams from external storage). However, current software tools do not allow the collection of real-time data streams. Consequently, realization of $F_{SB1,1}$ requires the development of new algorithms. Since we had no opportunity to have access to an appropriate sensor network and the multiple data streams generated by its nodes, we provided the necessary data files using computational simulation. The constructed files were used both in the algorithm development stage and in the validation of the algorithms. What it means is that we are not dealing with function $F_{SB1,1}$ here, since the data streams have been included in the toolbox database directly. We assumed that, in a real-life situation, the SDATB would have access to sensors and would be able to receive multiple MoLD-Ss. These data streams are checked before they are used for merging. The SDATB offers the option of visualizing all data streams received, and the designer can select varying numbers of the MoLD-Ss for analysis

and merging by the SDATB. The basis of merging is the "internal" affordance (semantic cohesion potential). Anomalies detected in the fused streams are identified and included in the results.

Form a software engineering point of view, the main functions of the SDATB are provided by various modules. Specification of the modules and determining the computational algorithms included in the modules is the task of architecting. For instance, two external components are needed for the architecture of the main function F_{SB1}. One of them is a system user interface, which enables communication between the designer and the SDATB. It also transfers the inputs and outputs to and from the toolbox. Another component is the database, also referred to as the knowledge warehouse. In addition to data, it stores the rules and conditions for analyses, as well as the results of merged data streams.

Figure 3 shows the overall conceptual architecture of the MoLD-Ss merging module of the SDATB. The main constituents are (i) the search engine, (ii) the database, (iii) the preprocessing unit, (iv) the merging unit, (v) the recommendation unit, (vi) the explorer, (vii) the query manager, and (viii) the user interface. The lower-level components of the units are shown in Figure 3. The MoLD-Ss explorer, used for exploring the data streams to be analyzed, is a kind of entry point to this module. The MoLD-Ss preprocessing unit communicates with the designer and receives and processes the individual MoLD-Ss in the toolbox. The MoLD-Ss manager visualizes the data streams stored in the database and makes them available for the search engine. The preprocessing configurator determines the preprocessing rules and conditions to be applied to the individual streams by the preprocessing executor. These two components use knowledge already existing in the database. The preprocessed MoLD-Ss are transferred to the merging unit, which is composed of four components: (i) the merging executor, (ii) the anomalies detector, (iii) the semantic similarity calculator, and (iv) the anomalies organizer. These components are closely related to the knowledge stored in the database. The semantic similarity calculator compares the explored anomalies with those stored to determine resemblances. The anomalies organizer manages the weights and filters and organizes the anomalies to be used by the recommendation unit. The recommender agent converts the information generated by the above components into recommendation contents. The message generator produces messages to the designer using the recommendation contents. Finally, the query manager converts the produced message to human language and communicates it to the designer as a recommendation via the user interface.

4. Algorithm-level specification of the recommendation module for merging MoLD-Ss

The recommendation module for merging MoLD-Ss is reasonably novel. To realize it, three algorithms are needed for its sub-function $F_{SB1,1}$. The first (algorithm A_1 "request list of sensors") is responsible for requesting from the designer the list of sensors to be analyzed by the SDATB. The second (algorithm A_2 "request a subset of devices supporting provided sensors") requests access to data streams and their locations. The third (algorithm A_3 "fetch MoLD-Ss to the SDATB") is responsible for acquiring MoLD-Ss from remote storage (for example, a cloud environment) and moving them to the SDATB and its local storage. For sub-function $F_{SB1,2}$, two algorithms are needed. One is responsible for providing means for visualizations (plotting) to comprehend data despite their raw format (algorithm A_4 "plot sensors' data streams as time series for selected data streams"). The second one (algorithm A_5 "apply time-series

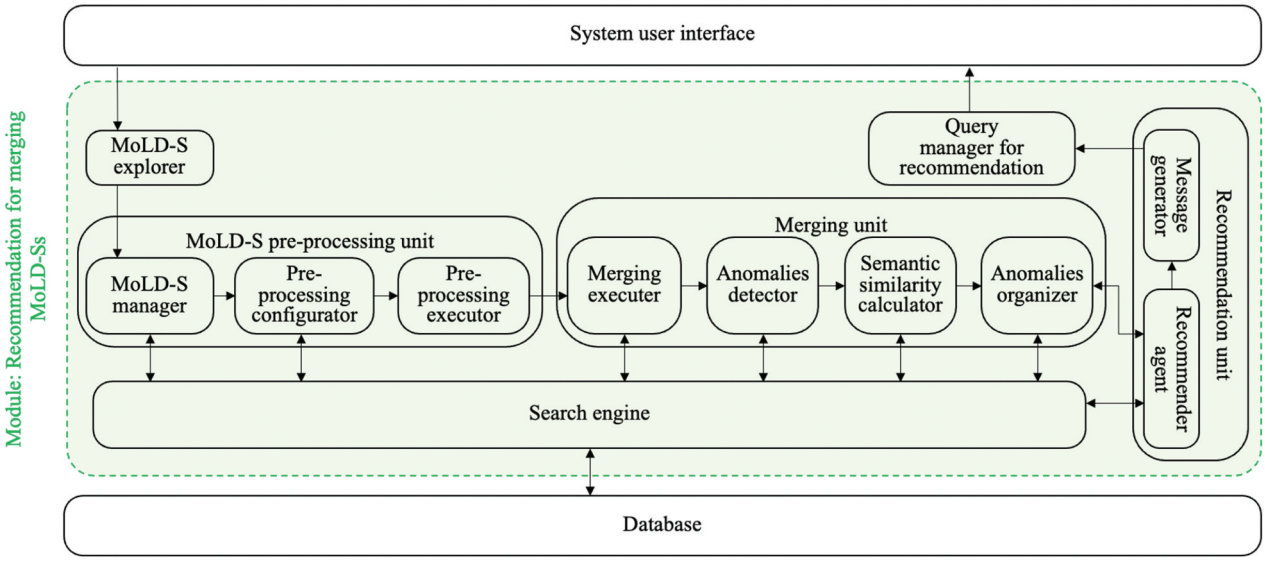

Figure 3. Overall conceptual architecture of the recommendation module for merging middle-of-life data streams.

normalization for each MoLD-S") is responsible for the normalization of MoLD-Ss so the data streams can be properly used for further analyses. This algorithm is needed to remove anomalies that might complicate the analysis, such as by (i) deleting data (e.g. removing correlated time series), (ii) inserting more information (e.g. applying one hot encoding for categorical features), or (iii) updating existing information (e.g. clipping outliers).

For sub-function $F_{SB1,3}$, four algorithms are needed. Algorithm A_6 "process single stream time series with statistical model" is responsible for processing normalized single MoLD-Ss time series using a statistical model. This is needed to generate length-invariant representations of MoLD-Ss to reduce computation costs in the upcoming steps. Algorithm A_7 "estimate sensors' importance" calculates or estimates the importance of the sensors to analyze. This is only needed when a large number of sensors are to be merged and analyzed. By considering a lower number of relevant data streams, the interpretation of predictions is improved. The outcomes of this steps are used in Algorithm A_8 "Merge MoLD-Ss based on fusion weights," which is an algorithm run in Matlab. This algorithm has been constructed to merge data streams that are obtained from various sensors, but which are captured in the same time frame. The developed algorithms are for semantic fusion based on estimating anomalies and performing similar descriptor searches in the database. Algorithm A_8 considers the weights allocated to sensors and selects only those with the highest weight values for merging. This means we order the sensors according to their estimated fusion weights and consider a portion of the most relevant MoLD-Ss in the merging. Algorithm A_9 "estimate behavior descriptor based on merged MoLD-Ss representation" processes the MoLD-S jointly and embeds information into a new latent space (or representation). In such a space, a distance reflects the degree of semantic similarity. The behavior descriptor is sensor independent and describes the behavior independently from the source.

To realize sub-function $F_{SB1,4}$, six algorithms are needed. Algorithm A_{10} "estimate probability of anomaly" is responsible for estimating anomalies probability. It is a preliminary step to a more thorough search through the knowledge database containing a list of pre-programmed anomalies. It consists of calculating the distance to knowledge anomalies in the database. Algorithm A_{11} "perform similar descriptor search in database" gathers similar past anomalies from the database. It performs a search for similar descriptors, iterating through the pairs of the detected anomalies and the past ones. Algorithm A_{12} "calculate distance between anomalies" calculates the pairwise distance between the detected and the past anomalies. These anomalies are ranked via Algorithm A_{13} "rank anomaly descriptors by their distance from a requested one" and then retrieved using Algorithm A_{14} "retrieve relevant anomalies based on ranking as well as their corresponding sensors." Algorithm A_{15} "merge relevant anomalies into an action plan (recommendation)" executes the semantic merging of the retrieved anomalies and generates a recommendation, which contains an action plan detailing what needs to be done with the product. Realizing sub-function $F_{SB1,5}$ requires five algorithms. Algorithm A_{16} "select recommendation message template" selects the template for the recommendation message from the database. Algorithms A_{17} "convert individual anomalies into recommendation message component" and A_{18} "convert the action plan into recommendation message component" successively convert the detected anomalies and the action plan into components of the recommendation message. Algorithm A_{19} "order recommendation message components" executes the ordering of the appearance of individual anomalies and includes an action plan in the recommendation message. Algorithm A_{20} "integrate recommendation message components according to template" integrates the ordered components of the message into the template to provide the recommendation message to be presented to the designer.

5. Implementation of the recommendation module for merging MoLD-Ss

To realize the merging MoLD-Ss recommendation module, the algorithms presented previously need to be implemented. Some of these algorithms (A_1, A_2, A_3, A_6, etc.) are either developed by software tools such as Matlab, Python, or SPSS, or they are described in detail in the literature. To avoid redundances, in this paper, we are detailing only the new algorithms that we designed to realize the proposed function.

Algorithm 4. Plot sensors data streams as time series for selected data streams.

Inputs: I1 = D
I2 = SensorNames function
I3 = RequestIds function. UI method to request a subset of sensors
I4 = PlotTimesseries

Outputs: PlotTimeseries(A, SensorNames)

1: $sel \leftarrow RequestIds(SensorNames)$; % obtains sensors selected by user
3: **if** $numel(sel) == 0$
4: *break*;
5: **end**
6: $A \leftarrow zeros(numel(sel), 256)$; % output matrix to pass into UI method
7: **for** $i \leftarrow 1 : numel(sel)$
8: $t18 \leftarrow randi([1, 2])$; % randomly select either time series are faulty or not
9: $t17 \leftarrow randi([1, S])$; % randomly select one of S time series instances
10: $A(i, :) \leftarrow squeeze(D(i, t18, t17, :))$;
11: **end**
12: **Return** PlotTimeseries(A, SensorNames)

Algorithm A_4 has been developed to generate an interface for MoLD-Ss visualization. For this algorithm, we need to define the following: (i) Matrix D of M × 2× S x T dimension with instances of sensors' time-series data. The first dimension M corresponds to sensor number, the second dimension corresponds to either normal behavior (−1) or faulty behavior (−2), and the third dimension corresponds to S = 256 instances of different windows of sensor data, each of which has $T = 256$ time steps (representing the fourth dimension). (ii) SensorsNames is a function providing sensor descriptions for each of the M available sensors. (iii) The RequestIds function is responsible for visualizing a chosen set of sensors at the same time. Once the visualization is finished, the function returns an empty set (nothing to visualize). (iv) The PlotTimeseries function is a user interface method to display multiple sensor data streams within the same window in a certain time range.

To provide a recommendation based on a multi-stream dataset D,' we specify annotations to past anomalies. Descriptions of past anomalies need to be specified. Let us consider a window of aligned multimodal features X = {X(t, k)}, where t = [a, b] and k = [1, M]. M is the total number of selected sensors. The interval [a, b] represents the time boundaries of the anomalous behavior of the historical data of some device. Since we considered the triplet loss function for the ANN used for clustering a predefined set of classes, we assigned unique labels to the anomalies. Furthermore, we defined a set of incidents for each anomaly to allow the model to have sufficient data during the training and to avoid overfitting. The triplet loss training can fit a dataset of 8 million unique labels and achieving >95% classification accuracy [66]. The neural networks architecture that was considered for this purpose can be described as the algorithm responsible for sensor importance weight predictions A_7 (see Appendix) for forward pass (which refers to the calculation process and values of the output layers from the input data). To build the algorithm, we needed to define a real = valued matrix X with B x M x T size, where M represents multimodal features of each window of frames (sliding window), T represents the time frame, and B is the batch size.

To train the model, we used a specific triplet loss algorithm, A_9 (see Appendix), known in the literature as a hinge triplet loss algorithm [67]. This algorithm uses a hinge function to create a fixed margin between the anchor-positive difference and the anchor-negative difference. The following inputs had to be defined: (i) H, a real-valued matrix of B x 3 × L dimension, where B is the batch size, 3 represents two triplets of same label behavior representations and one outlier, and L is a latent representation dimension. H must be constrained within the boundaries [−1, 1]; otherwise, either a *tanh*(.) activation function can be applied, or rescaling of the vector values can be considered. (ii) B', a separation margin to control how much nonrelevant behavior should be embedded in the latent space according to cosine similarity distance. The triplet loss presented in Algorithm A_9 has been optimized using stochastic gradient descent (SGD) [68]. This algorithm optimizes the triplet loss by changing the parameters of the neural network. During the stochastic gradient descent procedure, we sampled a batch of triplets to perform the optimization step.

After introducing the behavior descriptors of multiple sensor data streams, we developed Algorithm A_{10} (see Appendix) to select potential candidates for an anomaly. For this purpose, three inputs have to be defined: (i) h, a matrix of size $N \times L_3$ of behavior descriptors to analyze; (ii) q, a matrix of size $M \times L_3$ of behavior descriptors in the database of past anomalies; and (ii) *tau*, the upper bound of the confidence interval. Given the vector p, we select as candidates only those entries for which $p_i > \tau$, where τ is the upper bound of confidence interval for normal behavior. Algorithm A_{10} filters out normal cases based on the large number of descriptors, which are generated by the sliding window approach working on time-series data. A_{10} was also intended to detect anomalies. Another algorithm was developed for similarity-based searching A_{11}, which is based on similarity estimation. This was done because, in addition to detecting an anomaly, we must also retrieval a ranked list of relevant anomalies for the computational processing. To develop Algorithm A_{11} (see Appendix), the following inputs were defined: (i) h, a matrix of size $N \times L_3$ of behavior descriptors to find similar past cases; (ii) q, a matrix of size $M \times L_3$ of behavior descriptors in the database of past anomalies; and (iii) tau, the distance threshold for descriptor retrieval.

After determining the possible anomaly candidates, we used Algorithm A_{12} (see Appendix) to calculate the distances between these anomalies. This algorithm requires the following inputs: (i) h, a matrix of size $N \times L_3$ of behavior descriptors for anomaly candidates; (ii) q, a matrix of size $M \times L_3$ of behavior descriptors in the database of past anomalies; (iii) index, identifiers of past anomalies; (iv) the offset of the first entry for each of the N anomalies; and (v) the number of relevant past cases discovered for each of the N anomalies. Given the distances of past cases, they can be sorted to generate a ranked list of anomalies. This is achieved with Algorithm A_{13} (see Appendix). This algorithm needs four inputs: (i) d, distances between anomalies (expressing the degree of similarity between anomalies and the past cases relevant to them); (ii) index, identifiers of past anomalies; (iii) offset of the first entry for each of the N anomalies; and (iv) the total number of relevant past cases found for each of the N anomalies. It represents the similarity between anomalies and past cases relevant to them.

To generate a recommendation, we need to obtain the top K anomalies per descriptor using a ranked list of their identifiers. This can be done with Algorithm A_{14} (see Appendix). The inputs for this algorithm are as follows: (i) r, a ranked list of the identifiers of relevant anomalies; (ii) index, identifiers of anomalies in the distance vector d; (iii) offset of the first entry of each of the N anomalies; (iv) C, an $M \times L_4$ causality matrix of past anomalies related to L_4 sensors; and (v) K, the total number of the (most) relevant anomalies to be found for each of the N candidate anomalies. The database contains "if … then" type rules, which are used in mapping between anomalies and possible recommendations. Algorithm A_{15} (see Appendix) is used to determine the best match and what to extract. This algorithm requires the following inputs: (i) d, distance between anomalies; (ii) sensors, sensor identifiers for each past anomaly; (iii) sensor_offset, offset of each past anomaly sensors list; (iv) sensors_amount, number of each past anomaly sensors, (v) anomaly, anomaly identifiers with up to K entries for each of N anomaly candidates; (vi) anomaly_new_index, anomaly identifiers within retrieved distances of vector d; (vii) anomaly_offset, offset of each anomaly group; (viii) anomaly_amount, size of each anomaly group; and (ix) sensors_importance, matrix of size $N \times L_4$ of importance weights extracted from attention layer for each anomaly candidate. The other algorithms not presented in this section are used during the functional validation of the merging MoLD-Ss recommendation module, discussed in Section 6.

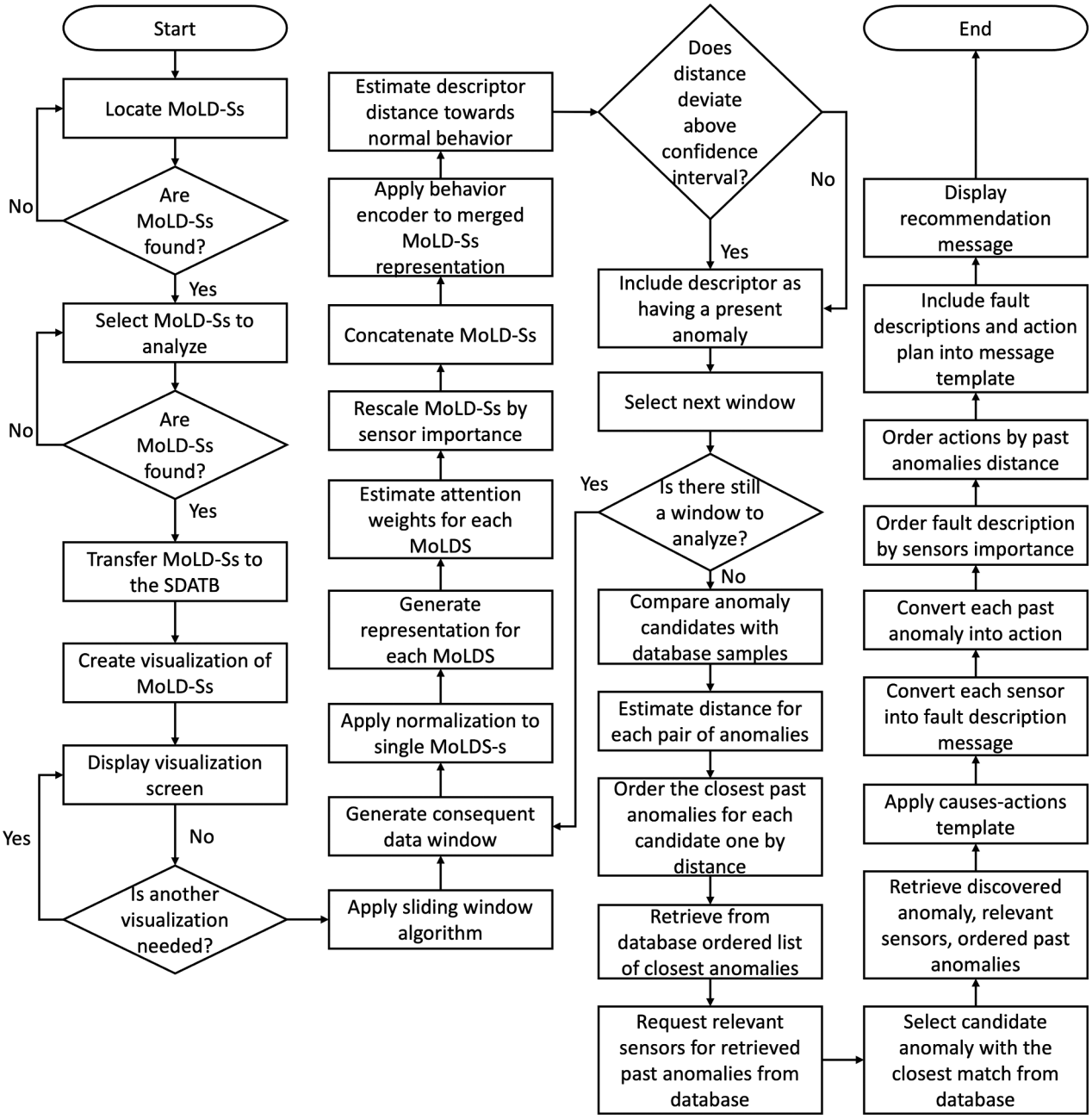

Figure 4. Computational workflow of the merging of middle-of-life data streams recommendation module.

In addition to the developed algorithms, the computational workflow (CWF) is also an important characteristic of this recommendation module for merging MoLD-Ss. Ordering all computational steps, the CWF of this module is shown in Figure 4. After the sensors are located and the data streams for analysis and merging are selected, the data contents of the MoLD-Ss are transferred to the SDATB, as a step that completes the analysis and the merging. In the next step, the data contents are visualized and presented in various plotted forms to the designer. The designer is given the opportunity to visualize the stream plots more than once. Towards the merging, the sliding window algorithm is used to iterate over the MoLD-Ss. The algorithm selects a consequent time frame of data and normalizes the data along the time axis. After this, the single-stream encoder part of the used neural network is applied, and single-sensor latent representation is generated in the attention layer of the neural network.

In the next step of the data processing, the single-sensor representation is rescaled according to the importance weights. These rescaled representations are concatenated into a two-dimensional matrix, and the behavior encoder part of the neural network is applied. Furthermore, the toolbox queries the database to find the past anomalies that are closest to the current descriptor. If the distance to past anomalies stored in the database is small, then a confidence interval including the current time window and its descriptor is selected as an anomaly candidate. Otherwise, it is skipped. When the algorithm finds no additional windows to analyze, it starts a similarity search. In this context, the descriptors are compared to those stored in the SDATB database. The distances between the anomaly pairs are estimated, and the matches are sorted according to the distances.

After this step, a ranked list of anomaly candidates is retrieved from the database. In combination with this, the sensors relevant to past anomalies are obtained based on the causality matrix. The anomaly candidate that has the shortest distance to its first relevant past anomaly is selected. In terms of the best candidate, this module of the SDATB presents a ranked list of past anomalies, as well as the sensors related to the past anomalies ordered according to the importance weights of the sensors. As a next step, the module selects a template for generating a recommendation message about the faulty sensors and possible improvement patterns. Then, the fault descriptions for each selected sensor and the improvement (or maintenance) actions for each anomaly are retrieved. These are subsequently arranged according to the importance of the sensors and the anomaly distance values and are used to generate the final recommendation message, which includes both the identified faults and the action plan. As the final computational action, this message is displayed to the designer.

6. Validation of the recommendation module for merging MoLD-Ss

To computationally implement the merging MoLD-Ss module and test its functionality, we used our reference application case of enhancing a connected washing machine by white goods designer. Accordingly, we considered that this device has 13 sensors: S_1 "force gauge on the axle bearings of the washing drum," S_2 "force gauge on transmission belt," S_3 "brake shoes position sensor," S_4 "force gauge on brake spring," S_5 "spinner time control clock," S_6 "washing timer control clock," S_7 "detector of spinning R.P.M setting," S_8 water level indicator, S_9 "inside temperature sensor in the housing," S_{10} "solid deposition indicator in the outlet of the waste water pipe," S_{11} "switch on/off detector counter," S_{12} "heater temperature thermometer," and S_{13} "heating time counter." Since we do not have access to real data streams, we built fake data streams (some streams have anomalies, others do not) that do not assume multidimensional values for single-sensor streams. To improve the performance of the function, a model capable of reasoning on multi-dimensional time series is needed. However, this requires the adoption of a more complex neural network. In addition, we incorporated prior knowledge for product anomalies in the data streams. This directed the focus of the implemented function toward maintenance kind of action plans. This was done to demonstrate how the semantics from different sensors can be captured and worked into an action plan. Developing algorithms able to automatically generate rules and to be aware of the dynamic changes in context and data streams will reduce time and effort dedicated to scenario building and algorithm training.

For the sake of the functionality testing, five anomalies (An_x, where x is the anomaly number) and their possible action plans were built and described. We created a mapping between the anomalies, related sensors, and recommendation messages. Examples of the mapping are presented in Table 1. Regardless of the anomaly type, if a particular sensor must exhibit a faulty signal,

Table 1. Mapping sample between anomalies, sensors, and recommendation messages.

Anomaly code	Description	Related sensors	Recommended action
An_1	Mechanical wear out of most-used components in the washing machine (washing drum, brakes to stop the drum, and related components).	S_1 or S_2 or S_3 or S_4	Mechanical control, adjustment, or replacement of components is needed
An_5	Abnormal temperature values and heating time deviation, with potentially sporadic device terminations. This can be caused by overheating or under-heating issues.	S_{11}, S_{12}, S_{13}	Water heater element should be cleaned or replaced

Table 2. Sample of normal and faulty behaviors for each of some sensors.

Sensor code	Normal behavior	Faulty behavior
S_1	Constant force during the whole washing cycle.	Abnormal force at some moments during the washing cycle.
S_4	Steadily increasing force during the brake application.	A large Gaussian noise is added to the force value. It models a loose contact between the brake shoe and the surface.

we manually engineer anomalous sensor activity. For testing purposes, we generated more complex but consistent MoLD-Ss. The logic behind data stream generation is presented in Table 2. To test the functionality of the merging MoLD-Ss recommendation module, we implemented the mentioned reasoning and learning procedures as hidden operations behind a graphical user interface (GUI) developed in Matlab. We adopted the definition that refers to it as a software platform designed with visual components (icons, windows, menus, etc.) allowing a user to easily navigate and interact with inputs and outputs requirements [69]. We decided to implement a simple GUI to visualize this module for the designer from his or her point of view (of course the interface of the actual data analytics toolbox will be much more sophisticated). The main screen of this module includes two actions (two possible buttons to press by the designer): (i) "Data" containing one option called "Select Sensors" for choosing which sensors to analyze, since our sensors are already located in the platform, and (ii) "About," which displays general information about the function. A designer who clicks on "Select Sensors" is moved to the next screen, which displays available MoLD-Ss with their corresponding codes and a short description of each. At this level, the designer chooses which sensors to merge (the option "Select all" is also available), or chooses one sensor if he or she only wants to analyze a particular sensor, and then presses "OK" to continue with the visualization or "Cancel" to return to the initial screen. After the designer's choice (we assume that the designer selects S_{11}, S_{12}, and S_{13}), the MoLD-Ss are transferred to be analyzed. The following screen is called "Visualization," see Figure 5. Once the inspection of represented plots is completed, the designer needs to

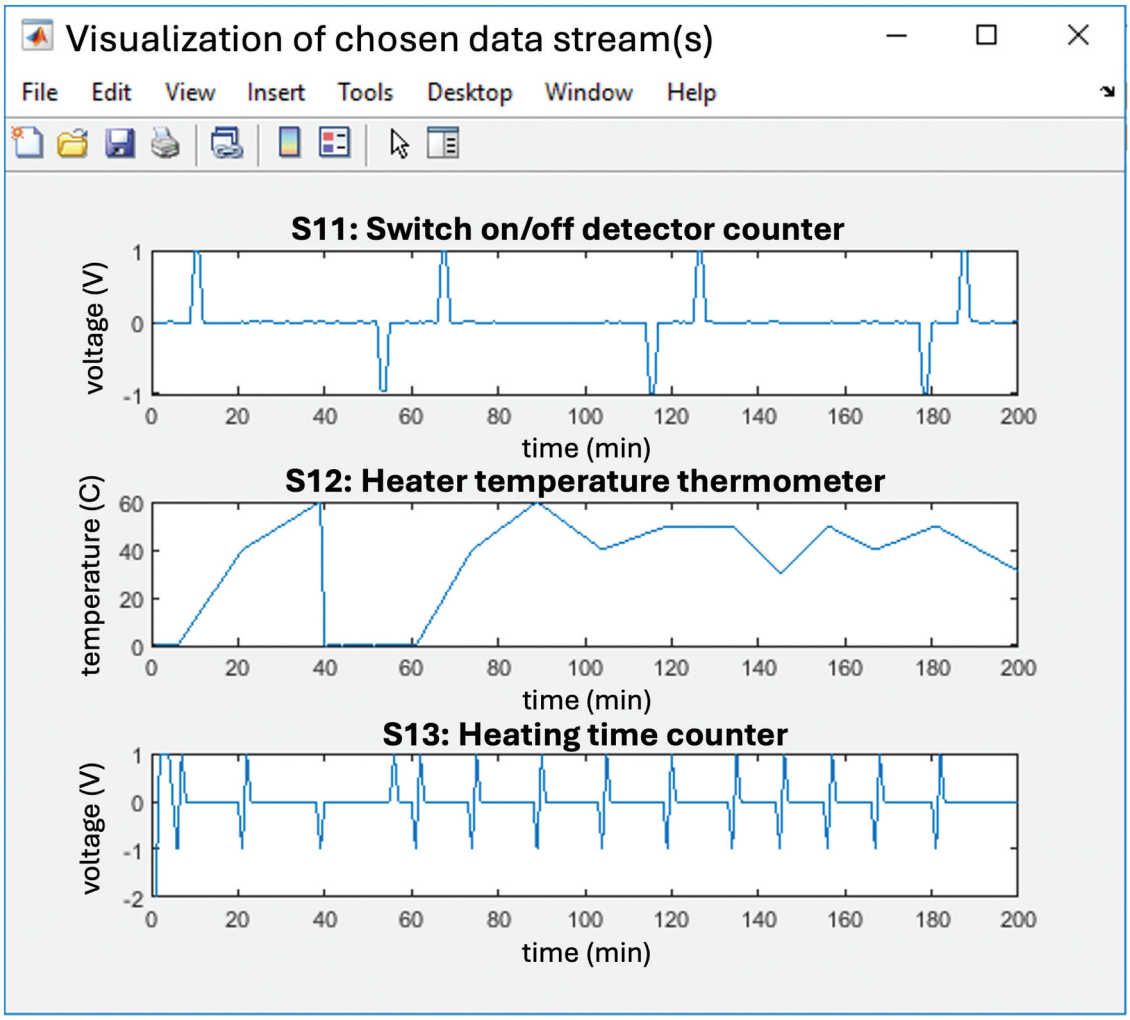

Figure 5. "Select data stream(s)" screen of the recommendation module for merging MoLD-Ss.

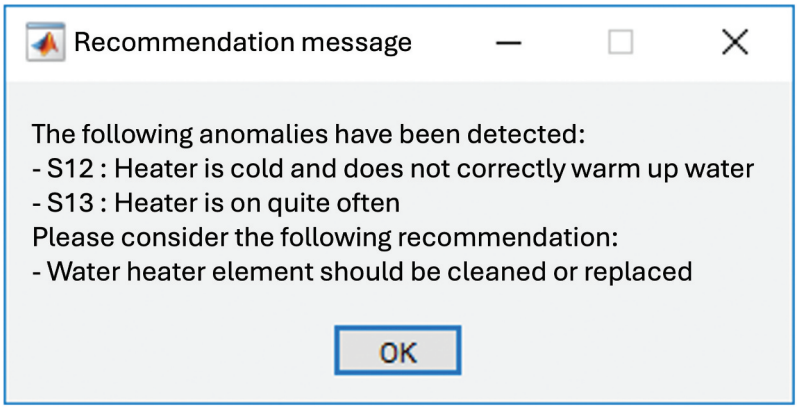

Figure 6. Recommendation message communicated to the designer.

press "x" to return to the previous window to select the button to merge data streams.

The merging is performed in the background of the GUI. The designer is only presented with a final textual recommendation within seconds. The recommendation message contains an explanation of detected anomalies and their sensors, as well as the recommendation (or action plan) semantically related to anomalies from different sensors. The message communicated based on the assumed choices presented above is displayed in Figure 6. As can be seen in this figure, S_{11} is not mentioned in the message. This means that no anomalous behavior was detected related to that specific sensor, but its semantic meaning was used in delivering the recommendation. If the user of the washing machine had turned on and off the device more often than, S_{11} would have reflected that, and consequently serious measures and different recommendation would be advised. Perhaps not only the water heater element is an issue but the whole electrical system of the machine is failing. To check the relevance of the analyses, we repeated the merging three times for the same sensors, and we obtained the same anomalies with the same recommendation.

To sum up, the functionality testing proved that the objective set for this module was achieved. From a computational point of view, the algorithms designed for this function and the ones taken from the literature were able to be converted, showing no computational errors. It was observed that the reasoning and learning from MoLD-Ss played a significant role in the formulation of the recommendation message to be delivered to the designer. The message covers not only the detected anomalies but also recommends certain actions to be considered by the designer. We concluded that, at the beginning of the implementation, the conditions set for the conversion of faulty behaviors of the MoLD-Ss into a concrete action plan for the designer were correctly elaborated.

7. Discussion

The functionality testing proved that the computational mechanism was correctly implemented. From a computational point of view, the integration of the newly designed algorithms and the ones taken from the literature did not lead to any inconsistences. Based on the results shown in Figure 6, it was observed that reasoning with and learning from the MoLD-Ss, as semantic operations, played a significant role in the formulation of the recommendation messages delivered to the designer. The message could cover not only the detected anomalies but could also recommend certain actions for the designer. Here, the conditions concerning the conversion of faulty behaviors of the MoLD-Ss into a concrete action plan for the designer were correctly incorporated in the computational mechanism. The function for merging MoLD-Ss (i) provides more information than that is conveyed by the sensors' data individually, (ii) reflects the condition of the product more realistically, (iii) communicates information about the product while it is in use by the customer (iv) reduces the time and effort of sensor analyses, and (v) provides recommendation as an action plan for the product at hand. Offering this function to product designers will allow them to continually analyze the behaviors of their products and to come up with enhancement solutions in a short while.

However, based on the analysis of the research activities and the testing of the implemented function some limitations were recognized: The lack of publications concerning a comprehensive understanding of the procedure of semantic inferring in the context of product enhancement made it difficult to select and deploy the best algorithms and techniques. The need to incorporate prior knowledge about product anomalies resulted in an inclination in the implementation toward maintenance type of action plans. The development of algorithms which are able to automatically generate rules and to be aware of the dynamic

changes in context and data streams, was supported by preprogrammed means. This reduces the time and efforts needed for building scenarios and training the algorithms. Although using simulated MoLD-Ss allowed us to meet the objective of testing the pilot implementation's functionality, the stimulated MoLD-Ss could not fully substitute for or replicate real-life MoLD-Ss, as they did not account for the actual behavior of sensors in their intended environment. Consequently, the performance or robustness of the computation during the functionality testing was not evaluated. This means that the pilot implementation was not exposed to the unexpected data patterns. With real data streams, it is possible that the processing takes longer time or exhibits inconsistencies. Transitioning from simulated to real-world data presents several challenges, primarily due to: (1) dynamic variation in data streams, where differences between simulated and real-world datasets can cause reduced model performance due to sample selection bias [70]; (2) sensor noise and data quality, as real-world data often contains noise and artifacts that are absent in simulations [71]; and (3) patterns relevance, where patterns that are significant in simulated data may lose their importance in real-world settings [71]. To address these issues, fine-tuning pre-trained algorithms with a small subset of real-world data can enhance their ability to generalize while retaining the insights learned from simulations [71]. For sensor noise and missing data, employing preprocessing and noise reduction techniques is crucial [72,73]. To tackle patterns relevance, reevaluating pattern selection and extraction methods is necessary [74]. The literature on bridging the gap between simulated and real-world scenarios highlights the value of using simulated data to build predictive models, especially when real-world data is limited or difficult to collect [71]. Available articles provide a step-by-step guide for transitioning from simulated to real-world data [75,76]. For this research, the transition will be documented in follow-up research.

Another limitation concerns the usage of the deep learning toolbox of Matlab for the implementation of the computational function. The fact of the matter is that it made implementation process more time-consuming in comparison with other computational solutions such as offered by Python in which predefined operations can be adapted or even directly used. The last observed limitation of this work is its focus solely on functionality testing that reflects also on the feasibility of the function, with limited attention to the performance, computational robustness, and efficiency of the chosen algorithms. The algorithms selected were considered adequate for demonstrating functional viability, but their selection did not prioritize metrics such as execution speed, scalability, or resilience under different conditions. Given more time, additional validation dimensions such as structural, performance, applicability and utility could have been considered. In addition, the validation approach could be expanded to incorporate properness or automated validation, further ensuring the reliability, consistency, and replicability of the results across multiple settings.

8. Conclusions

In this paper, we presented the functional, architectural, algorithmic, and computational considerations in implementing the recommendation function for merging MoLD-Ss. This function is one of the functions for a next-generation SDATB that we are busy developing. First, we identified, listed, and detailed the algorithms. We then collected, from the literature and the web, information about the prototype-level implementation and the computational techniques to build the SDATB functions. This set of knowledge was enough to realize and implement these functions. Then, we validated the functionality and execution of the algorithms. We built an application case involving enhancement of white goods (a connected washing machine) by product designers and used it in computationally implementing the function in Matlab software. This not only allowed us to test the functionality of the module but also provided information about the feasibility of the components of the function.

The recommendation function for merging MoLD-Ss offers two levels of semantic inferring (i) the level of merging, and (ii) the level of decision-making. Technically, the former was fulfilled by employing a proper neural network architecture, using its attention layer, and clustering past knowledge with triplet network embedding. The reasoning by the computational function provides opportunities for constructing implicit knowledge graphs, learning the statistical model, and separating related and unrelated behavior patterns. The multidimensional latent space captures the similarity considering multiple criteria, and exploration of knowledge clusters can happen in a non-constrained way. The recommendation function is a data-driven function, capable of capturing semantics. It can be seen as a knowledge construction with the help of behavior encoder. It is useful for helping designers understand unsupervised data and for assessing large volumes of sensor information, and is able of processing vast amounts of data streams to discover unusual behavior in MoLD-Ss.

The implementation of such a function will reduce decision-making time for product maintenance, repair, and enhancement. It not only identifies anomalies related to products but also provides a recommended action plan with the next steps to adjust the product. The implemented function is capable of deriving a simple yet efficient knowledge representation with the assistance of a triplet network. Furthermore, by merging MoLD-

Ss, more comprehensive information can be provided than by each sensor individually. Therefore, the implementation of the proposed recommendation function offers practical benefits. It renders the actual state and condition of the product transparent and communicates that information to the designer while the product is in use by its owner. The realized function provides a semantically correct recommendation to the designer based on product anomalies. It minimizes the time and effort required for processing data streams and facilitates a swift decision-making process for product enhancements.

Being design-focused, this conducted study primarily focuses on the functional validity, and proof of concept of the recommendation function rather than an exhaustive computational performance analysis. The primary goal is to validate the feasibility and the usability of the proposed function, particularly in supporting product designers in decision-making based on sensor data. While computational methods are employed to support the function's operation, an in-depth evaluation of computational efficiency, algorithmic complexity, and optimization aspects is beyond the scope of this paper. These technical aspects will be comprehensively analyzed in future research.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributor

Dr. ir. Fatima-Zahra Abou Eddahab-Burke is currently an Assistant Professor at the Faculty of Technology, Policy and Management, Delft University of Technology, the Netherlands. She received a Bachelor's Degree in Mathematics and Physics from Omer Ibn Abdelaziz Institute, Morocco, in 2009. In 2013, she obtained her Master's Degree in Mechanical Engineering from Mohammadia School of Engineers. In 2014, she obtained a Research Master's degree in Industrial Engineering from Grenoble Institute of Technology, France. in 2020, she obtained her Ph.D. from the Faculty of Industrial Design Engineering at the Deft University of Technology. She was a Postdoctoral research er in Wageningen Univeristy and Research for 2 years in the Farm technology group. Her current research interests include participatory design, designing interactive information systems , designing for the value of human dignity, and (re) designing engineering education systems.

ORCID

Fatima-Zahra Abou Eddahab-Burke http://orcid.org/0009-0007-0474-2641

References

[1] Suraj MV, Singh NK, Tomar DS. Big data analytics of cyber attacks: a review. Proceedings of ICSCA; Kuantan (Malaysia): IEEE; 2018. p. 1–7.
[2] Rizwan A, Zoha A, Zhang R, et al. A review on the role of nano-communication in future healthcare systems: a big data analytics perspective. IEEE Access. 2018;6:41903–41920. doi: 10.1109/ACCESS.2018.2859340
[3] Akbari N, Modarressi M. A high-performance network-on-chip topology for neuromorphic architectures. Proceedings of CSE; Guangzhou (China). IEEE; 2017. Vol. 2. p. 9–16.
[4] Abou Eddahab-Burke F-Z, Horváth I. What do designers miss regarding the outputs of data analytics tools in the context of possible product improvements? Proceedings of TMCE; Las Palmas de Gran Canaria (Spain); 2018. p. 423–438.
[5] Abou Eddahab-Burke F-Z. Towards building next-generation data analytics toolbox: application of the axiomatic theory fusion methodology. Proceedings of SICE; Kochi (Japan); 2024. p. 432–437.
[6] Abou Eddahab-Burke F-Z. Exemplifying smart functions for a next generation data analytics toolbox [Doctoral dissertation]. Delft University of Technology.
[7] Iqbal R, Doctor F, More B, et al. Big data analytics: computational intelligence techniques and application areas. Technol Forecast Soc Change. 2018;153 (2020):119253. doi: 10.1016/j.techfore.2018.03.024
[8] Sruthikha S, Tajunisha N. A study on evolution of data analytics to big data analytics and its research scope. Proceedings of the ICIIECS; Coimbatore (India); 2015. p. 1–6. doi: 10.1109/ICIIECS.2015.7193065
[9] Mishra M, Sidoti D, Avvari GV, et al. A context-driven framework for proactive decision support with applications. IEEE Access. 2017;5:12475–12495. doi: 10.1109/ACCESS.2017.2707091
[10] Kim SH. A formal model of creative decision making. Robot Comput Integr Manuf. 1991;8(1):53–65. doi: 10.1016/0736-5845(91)90007-F
[11] Horváth I, Wang J. Towards a comprehensive theory of multi-aspect interaction with cyber-physical systems. Proceedings of the ASME; Boston (USA); 2015. p. 1–12. doi: 10.115/DETC2015-47243
[12] Ban Y, Jacob A. Object-based fusion of multitemporal multiangle ENVISAT ASAR and HJ-1B multispectral data for urban land-cover mapping. IEEE Trans Geosci Remote Sens. 2013;51(4):1998–2006. doi: 10.1109/TGRS.2012.2236560
[13] Jirsa L, Kuklišová Pavelková L, Quinn A. Bayesian transfer learning between uniformly modelled Bayesian filters. Proceedings of ICINCO; Prague (Czech Republic). Springer; 2021. p. 151–168.
[14] Zhang W, Ji X, Yang Y, et al. Data fusion method based on improved DS evidence theory. Proceedings of the ICBDSC; Shanghai (China). IEEE; 2018. p. 760–766.
[15] Mönks U, Dörksen H, Lohweg V, et al. Information fusion of conflicting input data. Sens. 2016;16 (11):1798–1835. doi: 10.3390/s16111798
[16] Henry KA. Data collection and data management. In: Dubinsky PM, Henry KA, editors. The fundamentals of clinical research: a universal guide for implementing good clinical practice. New Jersey (NJ): Wiley; 2022. p. 265–283. doi: 10.1002/9781119772682.ch20
[17] Hodovychenko M, Antoshchuk S, Lobachev I, et al. Approaches and techniques to improve machine

learning performance in distributed transducer networks. Proceedings of ISDMCI; Zalizniy Port (Ukraine); Springer; 2022. p. 511–524.
[18] Khaleghi B, Khamis A, Karray FO, et al. Multisensor data fusion: a review of the state-of-the-art. Inf Fus. 2013;14 (1):28–44. doi: 10.1016/j.inffus.2011.08.001
[19] Bleiholder J, Naumann F. Data fusion. ACM Comput Surv. 2009;41(1):1–41. doi: 10.1145/1456650.1456651
[20] Bakhtouchi A. Data reconciliation and fusion methods: a survey. Appl Comput Inf. 2022;18(3/4):182–194. doi: 10.1016/j.aci.2019.07.001
[21] Albahri AS, Duhaim AM, Fadhel MA, et al. A systematic review of trustworthy and explainable artificial intelligence in healthcare: assessment of quality, bias risk, and data fusion. Inf Fusion. 2023;96:156–191. doi: 10.1016/j.inffus.2023.03.008
[22] Gutiérrez R, Rampérez V, Paggi H, et al. On the use of information fusion techniques to improve information quality: taxonomy, opportunities and challenges. Inf Fusion. 2022;78:102–137. doi: 10.1016/j.inffus.2021.09.017
[23] Rivera-Castillo J, Lindner L, Basaca-Preciado LC. Applying optoelectronic devices fusion in machine vision. In: Information Resources Management Association, editor. Natural language processing: concepts, methodologies, tools, and applications, New York (NY): IGI Global. 2019; p. 184–213. doi: 10.4018/978-1-7998-0951-7.ch010
[24] Castanedo F, Takama Y, Ursino D. A review of data fusion techniques. Sci World J. 2013;2013(1):1–19. doi: 10.1155/2013/704504
[25] Kong L, Peng X, Chen Y, et al. Multi-sensor measurement and data fusion technology for manufacturing process monitoring: a literature review. Int J Extrem Manuf. 2020;2(2):022001. doi: 10.1088/2631-7990/ab7ae6
[26] Jun H-B, Kiritsis D, Xirouchakis P. Research issues on closed-loop PLM. Comput Ind. 2007;58(8):855–868. doi: 10.1016/j.compind.2007.04.001
[27] Xhafa F, Barolli L. Semantics, intelligent processing and services for big data. Future Gener Comput Syst. 2014;37:201–202. doi: 10.1016/j.future.2014.02.004
[28] Bufardi A, Kiritsis D, Xirouchakis P. Generation of design knowledge from product life cycle data. Methods and tools for effective knowledge. Life-Cycle-Management: Springer; 2008. p. 375–389. doi: 10.1007/978-3-540-78431-9_21
[29] Ericson A, Müller P, Larsson T, et al. Product-service systems–from customer needs to requirements in early development phases. Proceedings of CIRP-IPS2; Linkoping (Sweden); 2009. p. 62–67.
[30] Shin JH, Jun HB, Catteneo C, et al. Degradation mode and criticality analysis based on product usage data. J Adv Manuf Technol. 2015;78(9–12):1727–1742. doi: 10.1007/s00170-014-6782-7
[31] Zhao Q, Ma G, Wang Q, et al. Generation of long-term InSAR ground displacement time-series through a novel multi-sensor data merging technique: the case study of the Shanghai coastal area. J Photogramm Remote Sens. 2019;154:10–27. doi: 10.1016/j.isprsjprs.2019.05.005
[32] Yang X. Progress in urban remote sensing: An overview. In: Yang X, editor. Urban remote sensing: monitoring, synthesis, and modeling in the urban environment. New Jersey (NJ): Wiley; 2021. p. 3–13. doi: 10.1002/9781119625865.ch1
[33] Chen G, Liu Z, Yu G, et al. A new view of multisensory data fusion: research on generalized fusion. Math Probl Eng. 2021;2021:1–21. doi: 10.1155/2021/5471242
[34] Hall DL, Llinas J. Multisensor data fusion. In: Liggins ME, Hall DL, Llinas J, editors. Handbook of multisensor data fusion. Boca Raton (FL): CRC Press; 2017. p. 21–34. doi: 10.1201/9781420053098
[35] Kibrete F, Woldemichael DE, Gebremedhen HS. Multi-sensor data fusion in intelligent fault diagnosis of rotating machines: a comprehensive review. Meas. 2024;232(2024):114658. doi: 10.1016/j.measurement.2024.114658
[36] Zu Westerhausen SM, Schneider J, Lachmayer R. Automatic reliability assessment of data paths in component-integrated sensor networks. Proceedings of the ESREL 2023; Southampton (UK); 2023. p. 2223–2229. doi: 10.3850/978-981-18-8071-1_P224-cd
[37] Moon Y, Lee Y, Hwang Y, et al. Long short-term memory autoencoder and extreme gradient boosting-based factory energy management framework for power consumption forecasting. Energies. 2024;17(15):3666. doi: 10.3390/en17153666
[38] Zheng Y. Scalable big data analysis and processing technology for intelligent traffic control. Proceedings of AIC; Gwalior (India): IEEE; 2024. p. 448–453.
[39] Castanedo F, Takama Y, Ursino D. A review of data fusion techniques. Sci World J. 2013;2013(1):704504. doi: 10.1155/2013/704504
[40] Dautov R, Distefano S, Buyya R. Hierarchical data fusion for smart healthcare. J Big Data. 2019;6 (1):1–23. doi: 10.1186/s40537-019-0183-6
[41] Li X, Guo Z. Multi-source information fusion model of traffic lifeline based on improved DS evidence theory. Proceedings of Geoinfromatics; Kunming (China): IEEE; 2018. p. 1–6.
[42] Shao H, Lin J, Zhang L, et al. A novel approach of multisensory fusion to collaborative fault diagnosis in maintenance. Inf Fusion. 2021;74:65–76. doi: 10.1016/j.inffus.2021.03.008
[43] Demirci M, Gözde H, Taplamacioglu MC. Improvement of power transformer fault diagnosis by using sequential Kalman filter sensor fusion. Int J Electr Power Energy Syst. 2023;149:109038. doi: 10.1016/j.ijepes.2023.109038
[44] Aguileta AA, Brena RF, Mayora O, et al. Multi-sensor fusion for activity recognition—A survey. Sens. 2019;19(17):3808. doi: 10.3390/s19173808
[45] Meng T, Jing X, Yan Z, et al. A survey on machine learning for data fusion. Inf Fusion. 2020;57:115–129. doi: https://doi.org/10.1016/j.inffus.2019.12.001
[46] Zhang Y, Morel O, Blanchon M, et al. Exploration of deep learning-based multimodal fusion for semantic road scene segmentation. Proceedings of VISAPP; Prague; Czech Republic: HAL; 2019. p. 1–8.
[47] Zhang L, Xie Y, Xidao L, et al. Multi-source heterogeneous data fusion. Proceedings of ICAIBD; Chengdu (China): IEEE; 2018. p. 47–51.
[48] Kulmukhametov A, Rauber A, Becker. Improving data quality in large-scale repositories through conflict resolution. Int J Digit Libr. 2021;22(4):365–383. doi: 10.1007/s00799-021-00311-0
[49] Masnan MJ, Zakaria A, Shakaff AY, et al. Principal component analysis - a realization of classification success in multi sensor data fusion. In: Sanguansat P, editor. Principal component analysis - engineering

applications. London (UK): IntechOpen; 2012. p. 1–25. doi: 10.5772/2693
[50] Nemati S. Canonical correlation analysis for data fusion in multimodal emotion recognition. Proceedings of IST; Tehran (Iran): IEEE; 2018. p. 676–681. doi: 10.1109/ISTEL.2018.866140
[51] Teoh JR, Dong J, Zuo X, et al. Advancing healthcare through multimodal data fusion: a comprehensive review of techniques and applications. PeerJ Comput Sci. 2024;10:e2298. doi: 10.7717/peerj-cs.2298
[52] Kim J, Song M, Kim D, et al. An adaptive Kalman filter-based condition monitoring technique for induction motors. IEEE Access. 2023;11:46373–46381. doi: 10.1109/access.2023.3273809
[53] Ross A, Jain AK. Fusion techniques in multibiometric systems. In: Hammoud RI, Abidi BR, Abidi MA, editors. Face biometrics for personal identification: multi-sensory multi-modal systems. Berlin (Germany): Springer; 2007. p. 185–212.
[54] Pu H, Chen Z, Liu J, et al. Research on decision-level fusion method based on structural causal model in system-level fault detection and diagnosis. Eng Appl Artif Intel. 2023;126:107095. doi: 10.1016/j.engappai.2023.107095
[55] Berardinelli L, Cortellessa V, Di Marco A. Performance modeling and analysis of context-aware mobile software systems. Proceedings of FASE; Paphos (Cyprus): Springer; 2010. p. 353–367. doi :10.1007/978-3-642-12029-9_25
[56] Gu X, Shen Y, Xu J. Multimodal emotion recognition in deep learning: a survey. Proceedings of ICCST; Beijing (China): IEEE; 2021. p. 77–82. doi: 10.1109/ICCST53801.2021.00027
[57] Karangwa J, Liu J, Zeng Z. Vehicle detection for autonomous driving: a review of algorithms and datasets. IEEE Trans Intell Transp Syst. 2023;24(11):11568–11594. doi: 10.1109/TITS.2023.3292278
[58] Stover JA, Hall DL, Gibson RE. A fuzzy-logic architecture for autonomous multisensor data fusion. IEEE Trans Ind Electron. 1996;43(3):403–410. doi: 10.1109/41.499813
[59] Barreto-Cubero AJ, Gómez-Espinosa A, Escobedo Cabello JA, et al. Sensor data fusion for a mobile robot using neural networks. Sens. 2021;22(1):305. doi: 10.3390/s22010305
[60] Chong CY, Chang KC, Mori S. Fundamentals of distributed estimation. In: Hall DL, Chong CY, Llinas J, Liggins II M, editors. Distributed data fusion for network-centric operations. Boca Raton (FL): CRC Press; 2013. p. 95–124. doi: 10.1201/b10124-5
[61] Isaac NJ, Jarzyna MA, Keil P, et al. Data integration for large-scale models of species distributions. Trends Ecol Evol. 2020;35(1):56–67. doi: 10.1016/j.tree.2019.08.006
[62] Gunatilaka AH, Baertlein BA. Feature-level and decision-level fusion of noncoincidently sampled sensors for land mine detection. IEEE Trans Pattern Anal Mach Intell. 2001;23(6):577–589. doi: 10.1109/34.927459
[63] Gite S, Agrawal H. On context awareness for multi-sensor data fusion in IoT. Proceedings of IC3T; New Delhi (India): Springer; 2016. p. 85–93. doi: 10.1007/978-81-322-2526-3_10
[64] Abou Eddahab FZ, Horváth I. Recommendation function for smart data analytics toolbox to support semantic merging of middle-of-life data streams. Proceedings of ICCAD; Paris (France): IEEE; 2020. p. 1–7.
[65] Abbas MA, Benblidia N, Bachari N-I. A multi-level fusion approach for climate variation study using multi-source data - case study: Algeria. Proceedings of AICCSA; Abu Dhabi (UAE): IEEE; 2019. p. 1–7.
[66] Zou Y, Dong F, Lei B, et al. Maximum similarity thresholding. Digit Signal Process. 2014;28:120–135. doi: 10.1016/j.dsp.2014.02.008
[67] Bao F, Maier T. Stochastic gradient descent algorithm for stochastic optimization in solving analytic continuation problems. Found Data Sci. 2020;2(1):1–17. doi: 10.3934/fods.2020001
[68] Wojtowytsch S. Stochastic gradient descent with noise of machine learning type part I: discrete time analysis. J Nonlinear Sci. 2023;33(3):45. doi: 10.1007/s00332-023-09903-3
[69] Blake MB, Nowlan MF. A web service recommender system using enhanced syntactical matching. Proceedings of ICWS; Salt Lake City (UT); IEEE; 2007; p. 575–582. doi: 10.1109/ICWS.2007.28
[70] Quiñonero-Candela J, Sugiyama M, Schwaighofer A, et al. *Dataset shift in machine learning*. Cambridge (MA): MIT Press; 2009.
[71] Alberts M, St. John S, Odie S, et al. Transitioning from simulation to reality: applying chatter detection models to real-world machining data. Mach. 2024;12(12):1–25. doi: 10.3390/machines12120923
[72] Widodo A, Yang BS. Support vector machine in machine condition monitoring and fault diagnosis. Mech Syst Signal Process. 2007;21(6):2560–2574. doi: 10.1016/j.ymssp.2006.12.007
[73] Kachhoria R, Mahalle PN, Bhagwat S. Data collection and preprocessing for environmental monitoring using wireless sensor networks. In: Mahalle PN, Takale DG, Sakhare SR, Regulwar GB, editors. Machine learning for environmental monitoring in wireless sensor networks. New York (NY): IGI Global; 2025. p. 39–52. doi: 10.4018/979-8-3693-3940-4.ch003
[74] Wang S, Li D, Song X, et al. A feature selection method based on improved fisher's discriminant ratio for text sentiment classification. Expert Syst Appl. 2011;38(7):8696–8702. doi: 10.1016/j.eswa.2011.01.077
[75] Schön T, Gosswami B, Hvingelby R, et al. Automated defect recognition in X-ray projections using neural networks trained on simulated and real-world data. Proceedings of iCT; Fürth (Germany); 2022. p. 1–7. doi: 10.58286/27732
[76] Lin C, Tian D, Duan X, et al. 3D environmental perception modeling in the simulated autonomous-driving systems. Complex Syst Model Simul. 2021;1(1):45–54. doi: 10.23919/csms.2021.0004

Appendix. Some of the algorithms used for the recommendation module for merging MoLD-Ss

Algorithm 7. Estimate sensors importance

Inputs: I1 = Matrix X
I2 = B
Outputs: O1= h, latent representation of behavior described by the current window of features
O2 = a, sensors importance
1: $t2 \leftarrow conv1(X)$;
2: $t16 \leftarrow leaky\_relu(t2, 0.2)$;
3: $t15 \leftarrow conv2(t16)$;
4: $t7 \leftarrow leaky\_relu(t15, 0.2)$;
5: $t6 \leftarrow attention(t7)$;
6: $t8 \leftarrow reshape(t6, B, M, 1).*t7$;
7: $t9 \leftarrow behavior\_conv1(t8)$;
8: $t10 \leftarrow tanh(reshape(sum(t9, 2), B, 1, L))$;
9: **Return** *struct('h,' t10, 'a,' t6)*

Algorithm 9. The estimate behavior descriptor based on the merged MoLD-Ss representation

Inputs: I1 = H
I2 = B′
Outputs: O1= J, loss value that is to be minimized with a gradient descent algorithm
O2= Acc, separation accuracy of triplets within specified margin
1: $t1 \leftarrow sum(H(:, 1, :). * H(:, 2, :), 3) - sum(H(:, 1, :). * H(:, 3, :), 3) - B'$;
2: $t2 \leftarrow sigmoid(t1)$;
3: $t3 \leftarrow -log(t2)$; % we maximize likelihood of t2 probability to be equal to 1
4: $t4 \leftarrow mean(t2 > 0.5)$;
5: **Return** struct('J,' mean(t3, 1), 'Acc,' t4)

Algorithm 10. Estimate probability of the anomaly

Inputs: I1 = h
I2 = q
I3 = tau
Outputs: O1 = p, N dimensional vector specifying the probability that N exhibits anomalous behavior
1: **for** $i = 1 : N$
2: $p(i) \leftarrow 0.0$;
3: **for** $j = 1 : M$
4: $cur\_p = \sigma(h_i \cdot q_j{}')$;
5: **if** $cur\_p > p(i)$
6: $p(i) \leftarrow cur\_p$;
7: **end**
8: **if** $p(i) > tau$
9: **break;**
10: **end**
11: **end**
12: **end**
13: **Return** struct('p,' p)

Algorithm 11. Perform search for similar descriptors in database

Inputs: I1 = h
I2 = q
I3 = tau
Outputs: O1 = index, identifiers of relevant past anomalies descriptors
O2 = index, N dimensional vector specifying an offset of descriptors retrieved for a particular anomaly candidate specified by the array index
(Note that Matlab handles every variable as an array that can hold numbers. In order to access selected elements of an array, indexing is used).
O3 = amount, N dimensional vector specifying the number of retrieved the descriptors per anomaly candidate

1: $index \leftarrow []$
2: $offset \leftarrow []$
3: $amount \leftarrow []$
4: **for** $i \leftarrow 1 : N$
5: $offset(i) \leftarrow numel(index) + 1;$
6: $amount(i) = 0;$
7: **for** $j \leftarrow 1 : M$
8: $d \leftarrow \sigma(h_i \cdot q_j')$;
9: **if** $d > tau$
10: $index(offset(i) + amount(i)) \leftarrow j;$
11: $amount(i) \leftarrow amount(i) + 1;$
12: **end**
13: **end**
14: **end**
15: **Return** struct('index,' index, 'offset,' offset, 'amount,' amount);

Algorithm 12. Calculate distance between anomalies

Inputs: I1 = h
I2 = q
I3 = index
I4 = offset
I5 = amount
Outputs: O1 = d, distance between each of N anomalies and past case relevant to them

1: $d \leftarrow []$
2: **for** $i \leftarrow 1 : N$
3: **for** $j \leftarrow 0 : amount(i) - 1$
4: $k \leftarrow index(offset(i) + j);$
5: $cur\_d \leftarrow \sigma(h_i \cdot q_k')$;
6: $d(offset(i) + j) \leftarrow cur\_d;$
7: **end**
8: **end**
9: **Return** struct('d,' d)

Algorithm 13. Rank anomalies

Inputs: I1 = d
I2 = index
I3 = offset
I4 = amount
Outputs: O1 = r, ranked identifiers of past anomalies
O2 = r_index, list of the identifiers of anomalies in the distance vector d
1: **for** $i \leftarrow 1 : N$
2: $a(offset(i) : offset(i) + amount(i) - 1);$
3: $[b, i] \leftarrow sort(a);$
4: $c \leftarrow index(offset(i) : offset(i) + amount(i) - 1);$
5: $r(offset(i) : offset(i) + amount(i) - 1) \leftarrow c(i);$
6: $r\_index(offset(i) : offset() + amount(i) - 1) \leftarrow i;$
7: **end**
8. **Return** struct('r,' r, 'r_index,' r_index)

Algorithm 14. Retrieve relevant anomalies based on their ranking and the corresponding sensors

Inputs: I1 = r_
I2 = r_index
I3 = offset
I4 = amount
I5 = C
I6 = K
Outputs: O1 = sensors, sensor identifiers for each past anomaly
O2 = sensors_offset, offset of each sensor influenced by anomalies (representing what anomalies to remove or to keep)
O3 = sensors_amount, number of sensors influenced by anomalies
O4 = anomaly, anomaly identifiers with up to K entries per each of the N anomaly candidates
O5 = anomaly_new_index, anomaly identifiers within distance d
O6 = anomaly_offset, offset of each anomaly group
O7 = anomaly_amount, size of each anomaly group
1: **for** $i \leftarrow 1 : N$
2: $anomaly\_offset(i) \leftarrow numel(anomaly);$
3: $anomaly\_amount(i) \leftarrow 0;$
4: $t\_1 \leftarrow anomaly\_offset(i);$
5: **for** $j \leftarrow 0 : min(amount(i), K) - 1$
6: $k \leftarrow r(offset(i) + j);$
7: $anomaly\_amount(i) \leftarrow anomaly\_amount(i) + 1;$
8: $t\_3 \leftarrow t\_1 + j;$
9: $anomaly(t_3) \leftarrow k;$
10: $anomaly\_new\_index(t_3) \leftarrow r\_index(offset(i) + j);$
11: $sensors\_offset(t_3) \leftarrow numel(sensors);$
12: $sensors\_amount(t_3) \leftarrow 0;$
13: **for** $l \leftarrow 1 : L_4$
14: **if** $C(k, l) = 1$
15: $t\_4 \leftarrow sensors\_amount(t\_3);$
16: $sensors(t_4) \leftarrow l;$
17: $sensors\_amount(t_3) \leftarrow sensors\_amount(t_3) + 1;$
18: **end**
19: **end**
20: **end**
21: **Returns** struct('sensors,' sensors, 'sensors_offset,' sensors_offset,
22: 'sensors_amount,' sensors_amount, 'anomaly,' anomaly,
23: 'anomaly_offset,' anomaly_offset, 'anomaly_amount,'
24: anomaly_amount);

Algorithm 15. Identification of possible actions (recommendation)

Inputs: I1 = d
I2 = sensors
I3 = sensor_offset
I4 = sensors_amount
I5 = anomaly
I6 = anomaly_new_index
I7 = anomaly_offset
I8 = anomaly_amount
I9 = sensors_importance

Outputs: O1 = faulty_sensors, identifiers of the sensors that most likely cause anomaly
O2 = anomaly_action: identification of possible actions (recommendation) matching the most relevant past anomalies with the smallest distance to the detected anomaly candidate

$best\_match \leftarrow -1;$
$best\_distance \leftarrow +Inf;$
for $i \leftarrow 1 : N$
if $anomaly\_amount(i) = 0$
  continue;
end
$k \leftarrow anomaly\_new\_index(anomaly\_offset(i));$
if $d(k) < best\_distance$
  $best\_match \leftarrow i;$
  $best\_distance(k);$
end
end
if $best\_match = -1$
  Return $struct(faulty\_sensors', \{\}, anomaly\_action', \{\});$
if $sensors\_amount(anomaly\_offset(best\_match)) > 0$
  $t\_1 \leftarrow sensors\_offset(anomaly\_offset(best\_match));$
  $t\_2 \leftarrow sensors\_amoun(anomaly\_offset(best\_match));$
  $t\_3 \leftarrow sensors(t\_1 : t\_1 + t\_2 - 1);$
  $t\_4 \leftarrow sensors\_importance(t\_3);$
  $[t_5, t_6] \leftarrow sort(t_4);$
  $faulty\_sensors \leftarrow t_3(flip(t_6));$
end
$t\_7 \leftarrow anomaly\_offset(best\_match);$
$t\_8 \leftarrow anomaly\_amount(best\_match);$
$anomaly\_action \leftarrow anomaly(t_7 : t_7 + t_8);$
Return $(faulty\_sensors', faulty\_sensors, anomaly\_action',$
$anomaly\_action);$