Mathias Weske, Judith Michael (Hrsg.): Modellierung,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2024 103

Maximizing Reuse and Interoperability in Industry 4.0 with a Minimal Data Exchange Format for Machine Data

Leah Tacke Genannt Unterberg[1], István Koren[1], Wil M.P. van der Aalst[1]

Abstract: Data interoperability in Industry 4.0 is a continuous challenge for industry and research. Many organizations face the challenge of managing data lakes that, without proper governance, risk becoming disorganized 'data swamps' with disparate data models and formats. This heterogeneity leads to inefficient data utilization. Standardization efforts have produced suites of extensive models as they try to accommodate diverse requirements while still being comprehensive. Their complexity has hindered their adoption. To address this, we propose a minimal intermediate meta model for a frequently considered type of data in smart manufacturing, namely *Machine Data*. This type of data is central to industrial IoT platforms and research efforts on Digital Shadows & Twins. It encompasses raw time series and event data from sensors and digital controllers. This *model-in-the-middle* is intended to bridge the gap between heterogeneous source systems and highly structured and semantically clean input for data science techniques. To be broadly applicable, it has to be minimal and favor abstraction over details. We equip it with a standardized exchange format based on CSV, which reduces friction in data sharing. Furthermore, we provide a precise mathematical formalization that connects it to the language of data science methods. This enables the generic implementation of methods that can easily be reused and combined. Finally, we validate the model together with initial tool support in the large-scale cluster of excellence Internet of Production (IoP). We conclude that it is possible and feasible to accelerate the realization of the ambitions for the future of manufacturing using such minimal models.

Keywords: Industry 4.0; machine data; exchange format

1 Introduction

The pursuit of smart manufacturing in Industry 4.0 has introduced and amplified numerous requirements for digital infrastructure. One of these core requirements is the capability for collection, management and utilization of various kinds of production-related data. This data is typically collected from very heterogeneous sources and eventually consumed by separately developed smart solutions. However, many organizations find themselves with data lakes (or even swamps [ML16]) filled with disparate data models and formats. In these cases, lack of interoperability complicates the utilization of available data. This situation is exacerbated in multi-disciplinary engineering environments [BLG17]. Even within a domain, use-case-specific tools are often custom developed instead of reusing or configuring generic tooling and methods. This slows down organizations and wastes resources.

[1] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany
{leah.tgu,koren,wvdaalst}@pads.rwth-aachen.de

Past and present standardization efforts like the open manufacturing platform[2] (OMP), the industrial digital twin association[3] (IDTA), or even commercial projects like the Azure Common Data Model typically yield powerful but complex and verbose specifications. As data model collections are designed to be comprehensive and able to cover any specific requirement, they quickly end up growing unwieldy, inhibiting their broad adoption. Particularly the lack of reference implementations makes it difficult for potential adopters to make use of them without incurring high development costs.

The cluster of excellence Internet of Production (IoP) [Br23] at RWTH Aachen University is a prime example of a multi-disciplinary engineering environment. The DFG-funded project is a testbed for the development of technologies for realizing the aspirations of Industry 4.0. It connects leading engineering and computer science expertise of over 25 institutes and employs around 200 researchers. Most relevant manufacturing processes like milling, hot/cold rolling, 3D printing, injection molding, textile processing, etc. are represented in the project with actual shopfloors of machines.

As the project also faces the challenge of growing data-silos resulting in stifled cross-domain collaboration, a survey of the data owners within the project was conducted. Among 80+ collected data models, though almost all of them were unique, a categorization effort revealed that many models clearly shared enough concepts to be represented by abstracted meta models. In this paper, we consider the type of data that was most prevalent in the survey and is central to smart manufacturing in general: *Machine Data*. This is raw time series and events produced by sensors and digital controllers of cyber-physical (production) systems [LBK15; Le08]. The number of commercial industrial IoT platform product offerings by top tech companies like Google (Google Cloud for Manufacturing), Amazon (AWS IoT), Microsoft (Azure Industrial IoT) and others, along with the amount of research in this field [Ch22; Mo18], clearly demonstrates its importance.

To address the aforementioned lack of a common data model, we propose a minimal intermediate meta model for this Machine Data to serve as an exchange format. It bridges the gap between application-specific (raw) data and the highly structured and semantically enriched input for data science (AI, ML, statistical, etc.) techniques. In other words, it is a so-called *model-in-the-middle*. Fig. 1 presents these contrasting situations. The arrows represent realizations of use-cases that require data transformations. Currently, even routine visualizations require custom and often ad-hoc transformations. Intermediate models enable definition of generic algorithms that can be easily reused via configuration. Equipping them with concrete file format specifications reduces friction when sharing data intra- and inter-organizationally. This accelerates the interdisciplinary development of methods for Industry 4.0, e. g., Digital Shadows & Twins [Be21].

Our initial proposal can serve as a blueprint for other domain-specific meta models. A successful example for this comes from the adjacent discipline of process mining (PM),

[2] https://openmanufacturingplatform.github.io/

[3] https://industrialdigitaltwin.org/

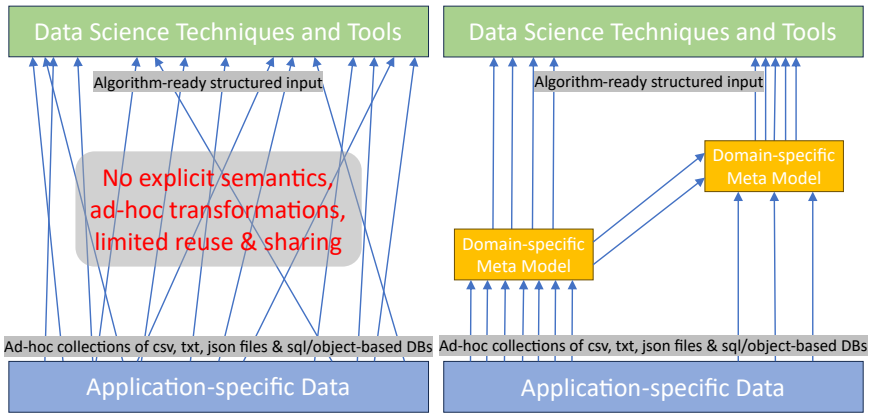


Fig. 1: Connecting source data and data science (AI, ML, statistical, etc.) approaches. The status quo with mostly one-to-one pipelines on the left versus the intended state on the right where approaches developed for meta models can be effectively reused.

where the XES (eXtensible event stream) [GV14] exchange format connects information systems like enterprise resource planning systems to PM algorithms. Connections between such models can further increase data utility.

The structure of this paper is as follows. In Sect. 2, we discuss related work before introducing the basic concepts of our *Machine Data Model* in Sect. 3. We further present a precise mathematical formalization in Sect. 4 that equips the meta model with clear semantics. Sect. 5 details our concrete CSV-file based exchange format and initial tool support. We then present a validation of the model and provided implementations in Sect. 6 and conclude the paper in Sect. 7.

2 Related Work

Related work lies in very different adjacent areas due to the interdisciplinary nature of this challenge. On the one hand, there are industry initiatives for standardization. Their focus mostly lies on comprehensive representation. Then, there is concrete research on IoT data models which focuses more on connecting such data to algorithms. Lastly, there is more general research on knowledge representation-based approaches, like semantic web technologies (SWT), to make heterogeneous data algorithm-accessible via ontologies.

Among the industry initiatives for data model standardization mentioned before are OMP and IDTA, the latter has been dissolved in 2020. Their efforts on so-called *semantic data structuring* had been continued in the eclipse semantic modeling framework[4] (ESMF). It offers a very general semantic aspect meta model which is tailored to general IoT sensor data.

[4] `https://projects.eclipse.org/projects/dt.esmf`

The IDTA has so far published 79 sub models for their asset administration shell. Their *time series data* specification has 39 pages and supports, e. g., arbitrary user-defined "notions of time". This is an artifact of "trying to model it all" and complicates the automated algorithmic usage.

AutomationML e.V.[5] has specified *AML* as a data exchange format for the representation of hierarchical object topologies, relations and properties over the various stages of a production system's lifecycle. While explicitly extensible, AML is not suitable for at-runtime sensor data representation.

Given its role as a key enabling technology for the digitalization of machine control and monitoring, a discussion of OPC UA[6] is warranted. The data modeling framework of this protocol is, in the end, one of the languages of the truly raw data. In particular, the companion specifications, of which there are currently 123, guide how the considered type of data is collected. Research such as [Le17; Sc19] which connect UML and SWT modeling to OPC UA data models, helps data owners to efficiently work with meta models such as ours. The OPC foundation, along with the aforementioned IDTA and AutomationML e.V. have also collaborated on a vision paper [Dr23] where they conclude that there will not be *the one model* but rather that it is their task to combine, connect and reference appropriate domain-specific standards.

All of the above are targeted at (industrial) IoT data in general and thus at the same time outscope und underscope basic and generic *Machine Data exchange*. Furthermore, they have a strong focus on comprehensive representation, rather than connecting data to method, which is the gap we want to close.

Next, the work in the process mining domain on data models and exchange formats (XES [GV14] and OCEL [Gh21]) could serve as a valuable example. The XES standard enabled interoperable algorithm development and still serves as *the* format to use when exchanging event logs in research. Current developments on integrating IoT data into PM [BDS22; Ma23] can be inspiration but the models themselves cannot be used directly, as they include higher level concepts like *business processes*.

Additionally, research on the representation of cyber physical systems (CPS) is of interest in this domain [LBK15; Le08]. It is a perfect fit for Machine Data, as it treats discrete events/states and continuous measurements appropriately, being less event-centric than the typical IoT data perspective. Most relevant is the specialized consideration of production systems, e. g., by Monostori [Mo14] and Biffl et al. [BLG17]. The latter authors refer to SWT as a potential solution for heterogeneous data access. Among this angle one may categorize ontology based data integration (OBDI). One example is the extensive review by Ekaputra et al. [Ek17] on OBDI in multi disciplinary engineering environments. In their terms, our work could be likened to the *global as a view (GAV)* approach where a global

[5] https://www.automationml.org/

[6] https://opcfoundation.org/about/opc-technologies/opc-ua/

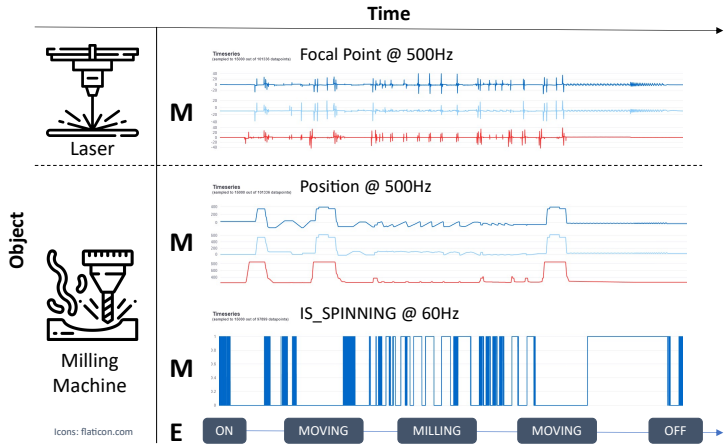


Fig. 2: Schematic example of a *Machine Data* instance. There are focal point measurements (sampled at 500Hz) for the object laser, while the object CNC-mill has two measurements and additionally logged discrete events.

ontology (here, a meta model) is used to transform and view independently created base data models. Work in this field is mostly of a higher-order nature, i. e., gives guidance on how to perform modeling, and lacks concrete specifications.

In contrast to holistic data modeling attempts like the above-mentioned IoT initiatives, or the more methodological descriptions of SWT, we propose a concrete, minimal model that is equipped with mathematically usable semantics and comes with a reference implementation and tool support.

3 Conceptualization of Machine Data

The concepts introduced into our model are carefully chosen to be as generic as possible while still providing enough semantic structure to enable automated analyses. We intend the model to be extensible, positioning possible extensions as compositions of this model with new concepts.

The minimal information we consider viable in the modeling of Machine Data is such that we can answer the following *W*-questions about each data point: *when?*, *what?* and (abstractly) *where?*. In addition to that, we enforce consistent *shapes* for data points by introducing *specifications* that define how to interpret the data points, essentially typing them. We call data points *observations* and differentiate between discrete *events* and continuous *measurements*. The conceptual *where?* is answered by an *object* identifier. Refer to Fig. 2 for an example instance. It shows how measurement time series as well as discrete events can be associated to various objects.

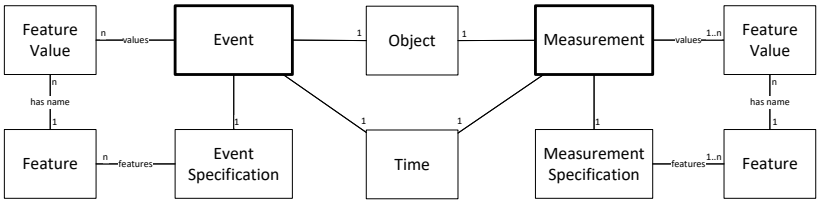


Fig. 3: The proposed meta model for *Machine Data*. Events and Measurements are syntactically similar but kept separate to reflect their distinct semantics.

The main concept is *Time*. It answers the *when?*. *Machine Data* captures the dynamic behavior of systems, not a static ontology of machine properties or production system configurations. The latter is indispensable in many applications but its inclusion would dilute the focus of this model.

The second concept are *Observations*. That is, a piece of information that was digitally recorded at some point in time. This relates to the SOSA ontology [Ja19] concept of the same name. Observations contain values and have an associated and fixed *specification*. The specification determines the interpretation of the values as an ordered vector of features. Thus, grouping by specification enables viewing a collection of observations as a multivariate sequence. Together, this is the *what?*.

We differentiate between two different types of observations to reflect their natural semantic difference. *Events* and *Measurements*. Events are discrete observations that emerge from a system/process. They typically indicate a state change or transition. Regardless of included values, even just the fact that a certain type of event occurred, reveals something about the process that produced it. For example, a *machine overheated* event, even without further attributes, provides information about the observed system. Measurement observations on the other hand are expected, even actively and regularly created —*sensed*. It is purely their *values* that describe the emergent behavior of a physical system. An empty *position* measurement may tell us on a meta level that something is wrong, e. g., the sensor may be broken, but it is not inherently valuable.

Lastly, we introduce the concept of *Objects* as an identifier to which observations are related. This concept is intended to associate measurements and events not just to a time but also to an additional orthogonal dimension, the abstract *where?*. This additional indexing dimension allows observations related to different objects, e. g., machines, to be contained in one *Machine Data* instance. It enables comparative and correlative analyses. Finally, it provides a clue to human analysts about the typical object of interest, which often serves as the subject of analysis.

Fig. 3 depicts the aforementioned concepts using an ER modeling approach. An observation consists of a timestamp, an object of reference, an observation specification and a variable

number of values which are viewed through the spec. Syntactically, events and measurements are mostly analogous, as they are both specializations of observations. However, we separate them here, as discussed before. Fig. 2 exemplifies how this differentiation is useful for visualizations. Treating events as time series by default is not appropriate.

Next, we introduce our formalization that is the basis for the expression of semantics and the data format definition.

4 Specification of Machine Data

In this section, we give a precise specification in terms of a mathematical formalization of the previously presented concepts. It connects this data model unambiguously to the vocabulary of data science and machine learning approaches. Conversely, this association extends toward classical mechanical engineering formulas and methods.

4.1 Definitions

We first define universe sets for the main concepts. All non-universe sets used in this section are assumed to be finite (as is reasonable in real-life settings).

Definition 1 (Universes)**.** We define the following universes as sets of valid identifiers and values.

- $\mathbb{T}$ is a totally ordered universe of timestamps.
- Σ^* is the universe of all finite strings.
- $\mathbb{U}_{label} \subseteq \Sigma^*$ is the universe of observation specification identifiers.
- $\mathbb{U}_{spec}$ is the universe of observation specifications.
- $\mathbb{U}_{obs}$ is the universe of observations.
- $\mathbb{U}_{feat} \subseteq \Sigma^*$ is the universe of feature identifiers.
- $\mathbb{U}_{val}$ is the universe of feature values. We use $\perp \notin \mathbb{U}_{val}$ to explicitly denote missing values and write $\mathbb{U}_{val}^{\perp} := \mathbb{U}_{val} \cup \{\perp\}$ for the extended value domain.
- $\mathbb{U}_{obj} \subseteq \Sigma^*$ is the universe of object identifiers.

We first define the mathematical objects corresponding to the individual concepts before giving a definition of a complete Machine Data instance. While the ER diagram in Fig. 3 does not explicitly show it, here, we make use of the inheritance/specialization structure between the concepts.

Definition 2 (Observation Specification)**.** An observation specification $s \in \mathbb{U}_{spec}$ has a type $type(s) \in \{E, M\} =: OT$, a label $label(s) \in \mathbb{U}_{label}$ and features $\mathcal{F}(s) = (f_1, \ldots, f_k) \in$

$(\mathbb{U}_{feat})^k$ specifying $k \in \mathbb{N}_0$ distinct features, i. e., $f_i \neq f_j$ for $i \neq j$. The identifier of an observation spec is $id(s) = (type(s), label(s)) \in OT \times \mathbb{U}_{label} =: \mathbb{U}_{specid}$. Specifications with $type(s) = E$ are *event specifications*, those with $type(s) = M$ are a *measurement specifications* and are required to specify at least one feature, i. e., $|\mathcal{F}(s)| \geq 1$.

Intrinsically, we regard observations as *raw* timestamped data. The spec and object reference are added on in the Machine Data instance.

Definition 3 (Observation)**.** An observation $o \in \mathbb{U}_{obs}$ has a timestamp $time(o) \in \mathbb{T}$ and partial mapping of features $\vartheta(o) \in \mathbb{U}_{feat} \nrightarrow \mathbb{U}_{val}$ with a finite natural domain. Undefined values are treated as missing, i. e., $\vartheta(o)(f) = \perp$ for $f \in \mathbb{U}_{feat} \setminus dom(\vartheta(o))$.

The observation value domain $\mathbb{U}_{val}$ is left generic here, as we do not specify concrete types like integers, floating-point numbers, etc. In the end, it is implementation-specific which particular data types are supported and can benefit from special handling. We elaborate on this in Sect. 5.

Observations and their specifications can be related to each other as follows. Let $o \in \mathbb{U}_{obs}$ be an observation and $s \in \mathbb{U}_{spec}$ be an observation spec.

- o is *consistent* with s, if o does not define any values not included in the features of s, i.e., $dom(\vartheta(o)) \subseteq set(\mathcal{F}(s))$.
- o is *complete* for s, if o defines all values of the features of s, i.e., $dom(\vartheta(o)) \supseteq set(\mathcal{F}(s))$.

We write the application of $o \in \mathbb{U}_{obs}$ to $s \in \mathbb{U}_{spec}$ with $\mathcal{F}(s) = (f_1, \ldots, f_k)$ as

$$s[[o]] \coloneqq \vartheta(o)(\mathcal{F}(s)) = (\vartheta(o)(f_1), \ldots, \vartheta(o)(f_k)) \in (\mathbb{U}_{val}^{\perp})^k.$$

Note that the resulting tuple may contain missing values $\perp$. This notation is reminiscent of applying a variable mapping to a logical formula in formal logic.

Given these basic elements, we can define Machine Data as a combination of specifications, observations, and their connection.

Definition 4 (Machine Data)**.** A Machine Data instance is a tuple $MD = (S, O, spec\_id, object)$ consisting of a set of specs $S \subseteq \mathbb{U}_{spec}$, a set of observations $O \subseteq \mathbb{U}_{obs}$ and two mappings $spec\_id : O \to \mathbb{U}_{specid}$ and $object : O \to \mathbb{U}_{obj}$. Additionally, the following has to hold.

$$\forall s, s' \in S : \qquad id(s) = id(s') \implies s = s' \tag{1}$$

$$\forall o \in O \; \exists s \in S : \qquad spec\_id(o) = id(s) \tag{2}$$

That is, (1) there may not be any duplicate spec identifiers and (2) all referenced specs need to be present in S. This makes the following definition of an observation to specification mapping unique and well-defined:

$$spec : O \to S, o \mapsto s \text{ such that } spec\_id(o) = id(s)$$

Lastly, we require observations to be consistent with their specs, i. e., $\forall o \in O : o$ *is consistent with* $spec(o)$.

The definitions we introduce here give a fixed structure on how to interpret Machine Data instances mathematically. This structure maps closely to the conceptual structure given in the previous section as well as to the concrete implementation in the following section. As mentioned before, this connects data to data-based methods in an unambiguous manner. One such example are methods for time series analysis.

4.2 Time Series Extraction

Subsequently, we provide a formalized example illustrating the extraction of concrete time series data.

Let $MD = (S, O, spec\_id, object)$ be a Machine Data instance. For a spec $s \in S$, its corresponding observations are $O_s \coloneqq spec^{-1}(s) = \{o \in O | \, spec(o) = s\}$. The referenced objects for s are $objects_s(MD) = \{object(o) | \, o \in O_s\}$. Per object $obj \in objects_s(MD)$, the observations related to it are $O_{s,obj} = \{o \in O_s | \, object(o) = obj\}$.

Such a set of observations can be ordered by time, yielding a sequence $seq(O_{s,obj}) = o_1, o_2, \ldots, o_n$ with $t_i = time(o_i) \leq time(o_j) = t_j$ for $1 \leq i < j \leq n = |O_{s,obj}|$. In case of duplicate timestamps, such a sequentialization may not be unique. We assume a deterministic choice to select one in this case. Applying the sequence to the spec yields a sequence of value tuples corresponding to the features of s,

$$seq_s(O_{s,obj}) \coloneqq s[[seq(O_{s,obj}))]] = s[[o_1]], s[[o_2]], \ldots, s[[o_n]].$$

Note that $s[[o_i]] \in (\mathbb{U}_{val}^{\perp})^{|\mathcal{F}(s)|}$, so there may be explicit missing values $\perp$ in these tuples, unless all o_i are *complete* for s.

For measurements, this sequentialization may be regarded as a k-dimensional multivariate time series of the features of s, sampled at points $t_i \in \mathbb{T}$. Formally, $ts_s(O_{s,obj}) \colon \mathbb{T} \nrightarrow (\mathbb{U}_{val}^{\perp})^k$ and

$$ts_s(O_{s,obj}) : t_i \mapsto (seq_s(O_{s,obj}))_i$$

with $t_i = time(seq(O_{s,obj})_i)$ and $1 \leq i \leq n$. The t_i may not be regularly spaced, i. e., $t_{i+1} \neq t_i + t$ for some $1 \leq i < n$ for any fixed time step t. However, further post-processing in the form of resampling and interpolation [LAC17] enables transforming such irregular time series to regular ones. Lastly, for numerical features, interpolation (not necessarily linear) may allow $ts_s(O_{s,obj}) \restriction_{[t_1,t_n]}$ (the restriction to the time interval between the first and last observation in $O_{s,obj}$) to be treated approximately as a total and continuous function. This makes specialized techniques such as *signal processing* applicable.

5 Data Model File Format

We propose a concrete representation of the specification based on comma separated value files. CSV files can be easily viewed and edited on any computer and without incurring (essentially) any tool requirement. This main benefit is important for making the format practical among potential users who are not native data and computer scientists. Our experience with working in the inter-disciplinary environment of the IoP project strongly informs this decision to favor low application/technology requirements. Following the concrete specification, we briefly introduce our initial tool support which forms the basis of our validation in the following section.

5.1 File Format Specification

To facilitate frictionless automated parsing, we first address some technicalities pertaining to the loose specification of CSV itself.

Special characters The CSV delimiters are a commas '**,**', decimal separators are periods '**.**' and the quotechars are double quotes '**"**'.
File encoding Shall be UTF-8.
Field count Equal across all rows. Missing values ($\perp$) are represented with empty fields.
Timestamps Represented as specified in **ISO 8601**, i. e., YYYY-MM-DD "T"hh:mm:ss.SSSSSSSSSZ with up to nanosecond precision.

As CSV is a text-based format, $\mathbb{U}_{val}$ is technically limited to string representable value types. This is more a limitation on storage and parsing efficiency and automatable semantics rather than a conceptional one. In practice, we recommend only working with the following common data types, which are readily inferrable by standard software: *floating-point numbers*, *integers*, *booleans* (represented by **true/false**), *datetime* (**ISO 8601**) and general *strings* (quoted if necessary).

Many more complex data types, like JSON dictionaries with key-value pairs, can be "unpacked/flattened" into multivariate observations. For example `{'x': 5.0, 'y': -3, 'uncertainty': [0.2, 0.5]}` can be flattened to the feature tuple `(x, y, uncertainty_x, uncertainty_y)` and value tuple `(5.0, -3, 0.2, 0.5)`. This makes the features more accessible for automated algorithmic usage. Otherwise, they would have to be manually re-interpreted at analysis-time and are thus most heavily discouraged.

To further ease tooling development and data usage, we recommend $\mathbb{U}_{label}$ and $\mathbb{U}_{feat}$ to be restricted to strings following the conventions of valid identifiers of common programming languages. That means not using spaces, punctuation or non-ascii symbols.

A Machine Data instance $MD = (S, O, spec\_id, object)$ is represented by a header together with a data CSV file. The header file has the following columns to represent observation

Tab. 1: Schematic of a Machine Data header file. We refer to specs as $s_i \in S$ for $1 \leq i \leq m = |S|$.

type	label	f_1				f_k
$type(s_1)$	$label(s_1)$	$\mathcal{F}(s_1)_1$	...	$\mathcal{F}(s_1)_i$		
		⋮			...	
$type(s_m)$	$label(s_m)$	$\mathcal{F}(s_m)_1$	...	$\mathcal{F}(s_m)_j$		

specs: *type*, *label* and a variable number of feature columns *f_1* to *f_k* where k is the maximal number of features over all specs, i. e., $k = \max_{s \in S} |\mathcal{F}(s)|$. As we strictly specify the order of columns, the header row is optional; it does not have to appear in the CSV file. Refer to Tab. 1 for a schematic of this encoding structure. In addition, Tab. 2 shows a brief example. The observations are stored similarly, that is, all observations are saved as rows

Tab. 2: Example Machine Data header table.

type	label	f_1	f_2	f_3
E	machining_started	spdl_speed		
M	force_sensor	fx	fy	fz
M	temp_sensor	temperature	stddev	

into one CSV file. The first columns hold the fixed attributes *time*, *object*, *type* and *label*, while the values are spread out to a variable number of value columns. They are ordered according to the feature tuple of the corresponding spec. Tab. 3 gives a schematic of the table structure which is almost analogous to the header file. The rows are not required to be ordered by time. As duplicate timestamps are typically resolved by first-occurrence, a deliberate sorting may be advisable. The header row with the defined column names is again optional.

Tab. 3: Schematic of a Machine Data data file. We refer to observations as $o_i \in O$ and their spec $s_i = spec(o_i)$ for $1 \leq i \leq n = |O|$.

time	object	type	label	f_1				f_k
$time(o_1)$	$object(o_1)$	$type(s_1)$	$label(s_1)$	$s_1[[o_1]]_1$	...	$s_1[[o_1]]_i$		
				⋮			...	
$time(o_n)$	$object(o_n)$	$type(s_n)$	$label(s_n)$	$s_n[[o_n]]_1$		...	$s_n[[o_n]]_j$	

The data file employs a non-normalized representation of all observations, characterized by a heterogeneous usage of columns. We acknowledge that a proper database schema would naturally separate the observations into separate tables (one per specification). However, the reason for selecting this file representation as canonical is to simplify the use as an *exchange format* that is supposed to be very accessible. Though the text-based file format may seem verbose and prone to data duplication, this design choice is a deliberate trade-off to enhance

readability and usability for users. The simplified structure accommodates those who may not be database experts, making it easier for a broader range of professionals to engage with the data. Data collection/editing in Excel is surprisingly common-practice in the industry, based on our experience. In contrast to having one CSV file per logical table, this fixed number of two files makes it less likely that a data set is transferred incompletely. Managing a dynamic number of files, which can quickly grow unwieldy, is prone to (human) error.

Two alternatives that we considered are SQLite and HDF5[7]. Both are single-file databases and could store the separate tables *cleanly*. SQLite is relational while HDF5 (hierarchical data format) uses hierarchical key-value pairs (this induces a tree structure like typical OS file systems). The former is not optimized for potentially high-volume time series data whereas the latter was specifically engineered for such uses and, e. g., comes with compression and storage optimization. However, as emphasized before, the requirement of having to interact with a data set via additional programs made both options infeasible as the canonical format.

5.2 Tool Support

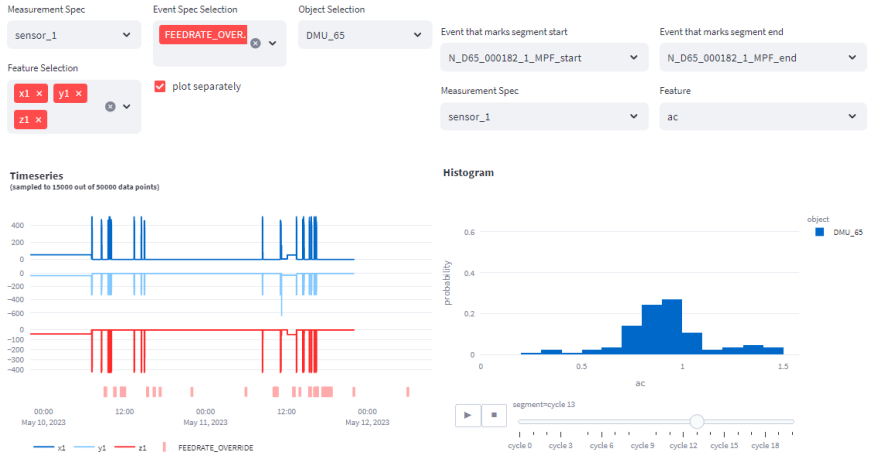
This proposed file format is supported by python tooling and example files on `https://git-ce.rwth-aachen.de/machine-data/mdata`. The python package is also available on PyPi via `pip install mdata`. Further, there is a web-based application that provides an interactive user interface for some of the capabilities of the former. The development repository along with prebuilt docker containers is available at `https://git-ce.rwth-aachen.de/machine-data/mdata_app`. An instance of this app is additionally publicly hosted at `https://mdata.cluster.iop.rwth-aachen.de/`. The most important goals of our tooling are providing support for data ingestion into the format and, on the other end, creation of usable data science algorithm input.

6 Validation with Domain Experts

Our proposed meta model was extensively discussed with a group of mechanical engineering researchers at a project-wide *Data Modeling Workshop* conducted in the IoP. While there was no consensus about the details on how to model all of the data they use, all 30 participants from different domains agreed with these basic concepts. To validate the specification together with its exchange format, we additionally worked with four data owners in the IoP and one external partner and transformed their data sets into our format.

The data sets were exported from databases that store more than our targeted minimal Machine Data, e. g., including machine configuration and other metadata like *author*. The transformations are thus closer to extractions in some cases. However, the additional

[7] `https://www.hdfgroup.org/solutions/hdf5/`



(a) Measurement time series of selectable features together with events.

(b) Measurement feature histogram generation based on event-defined segmentation.

Fig. 4: Two example generic visualizations based on Machine Data input.

concepts present in the data could be inspiration for an extended model that composes this one with new concepts. Naturally, more *models-in-the-middle* are necessary to cover the entire spectrum of production data.

Conceptually, all transformations are rather straightforward, that is, in terms of finding the concepts of events and measurements in the data. On the syntactic level, i. e., file format and structure, these data sets exemplified great heterogeneity. Collections of CSV files, a heavily nested JSON file, folders of text-based files without filetype (MinIO[8] (object-based) database export) and an entire postgreSQL dump. No two data sets can be loaded let alone visualized with the same code or tool. After transformation into our exchange format, they become valid input for our provided initial tooling. We present two proof-of-concept visualization widgets. Both are generic implementations of basic plotting code that are configurable via a graphical user interface.

The first is a time series view in Fig. 4a that combines measurement time series with discrete events. This is a complete staple and often the first visualization used by domain experts. The second one shown in Fig. 4b produces histograms over feature values per manually configurable segmentation delimited by events. In the data used here, events corresponding to NC program start and end are selected. This feature histogram is particularly of interest in machine condition monitoring where, e. g., particularly high process forces hasten degradation of the machine tool. In general, the domain experts were very interested in such aggregation methods and are currently developing their own. Discussions with the data owners also led to considerations for future work concerning our tool support.

[8] `https://min.io/`

7 Conclusion

A lack of standardization and interoperability due to unique data models impedes the efficient development of Industry 4.0 solutions. This affects industry, as well as research on the future of manufacturing. We identified that existing modeling efforts end up too complex or specific by trying to be comprehensive, which has hampered their widespread adoption. To rectify this, we proposed a minimal meta model for the most widely used type of data: low-level *Machine Data* consisting of events and measurements. To connect it to the language of data science as well as traditional engineering, we equipped it with a mathematical formalization. By additionally providing a concrete exchange file format along with initial tool support, we made it immediately usable. The tool support served as a proof-of-concept for the possibility of creating *generic implementations* that can be easily applied to any data set meeting the spec. Our validation with domain experts and data owners within the IoP research project showed agreement with the concepts and enthusiasm for the simplified data sharing.

As future work, we plan to extend the tool support for Machine Data in two main directions: (1) semi-automatic data extraction/transformation pipelines, and (2) generic implementations of universal time series visualizations/analysis tasks. Easing usage of the data model and at the same time incentivising usage with tooling should drive adoption. Additionally, we plan to release an extended model that supports derived/manually added annotations and more comprehensively facilitates reuse of higher-level aggregation/enrichment algorithms. Lastly, taking inspiration from OBDI, it is possible to define concrete domain-specific ontology-based instantiations of this meta model. By annotating time series features and objects with ontologies, automated semantic-aware approaches may be implemented. For example, marking a 3-dimensional measurement as a `3D-position`, would enable the time series plot shown in Fig. 4a to be plotted as a 3D graph automatically.

8 Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC-2023 Internet of Production - 390621612. We also thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

[BDS22] Bertrand, Y.; De Weerdt, J.; Serral, E.: A Bridging Model for Process Mining and IoT. In (Munoz-Gama, J.; Lu, X., eds.): Process Mining Workshops. Lecture Notes in Business Information Processing, Springer International Publishing, Cham, pp. 98–110, 2022, ISBN: 978-3-030-98581-3.

[Be21] Bergs, T.; Gierlings, S.; Auerbach, T.; Klink, A.; Schraknepper, D.; Augspurger, T.: The Concept of Digital Twin and Digital Shadow in Manufacturing. Procedia CIRP, 9th CIRP Conference on High Performance Cutting 101/, pp. 81–84, Jan. 1, 2021, visited on: 09/28/2023.

[BLG17] Biffl, S.; Lüder, A.; Gerhard, D., eds.: Multi-Disciplinary Engineering for Cyber-Physical Production Systems. Springer International Publishing, Cham, 2017, ISBN: 978-3-319-56344-2 978-3-319-56345-9.

[Br23] Brecher, C.; Padberg, M.; Jarke, M.; van der Aalst, W. M. P.; Schuh, G.: The Internet of Production: Interdisciplinary Visions and Concepts for the Production of Tomorrow. In. Pp. 1–12, July 12, 2023, ISBN: 978-3-030-98062-7.

[Ch22] Christou, I. T.; Kefalakis, N.; Soldatos, J. K.; Despotopoulou, A.-M.: End-to-End Industrial IoT Platform for Quality 4.0 Applications. Computers in Industry 137/, p. 103591, May 1, 2022, visited on: 10/04/2023.

[Dr23] Drath, R.; Mosch, C.; Hoppe, S.; Faath, A.; Barnstedt, E.; Fiebiger, B.; Schlögl, W.: Interoperabilität Mit Der Verwaltungsschale, OPC UA Und AutomationML, Apr. 11, 2023, preprint.

[Ek17] Ekaputra, F.; Sabou, M.; Serral, E.; Kiesling, E.; Biffl, S.: Ontology-Based Data Integration in Multi-Disciplinary Engineering Environments: A Review. Open Journal of Information Systems (OJIS) 4/, pp. 1–26, Oct. 1, 2017.

[Gh21] Ghahfarokhi, A. F.; Park, G.; Berti, A.; van der Aalst, W. M. P.: OCEL: A Standard for Object-Centric Event Logs. In (Bellatreche, L.; Dumas, M.; Karras, P.; Matulevičius, R.; Awad, A.; Weidlich, M.; Ivanović, M.; Hartig, O., eds.): New Trends in Database and Information Systems. Communications in Computer and Information Science, Springer International Publishing, Cham, pp. 169–175, 2021, ISBN: 978-3-030-85082-1.

[GV14] Günther, C. W.; Verbeek, H. M. W.: XES - Standard Definition, 1409, BPMcenter.org, Jan. 1, 2014, visited on: 10/09/2023.

[Ja19] Janowicz, K.; Haller, A.; Cox, S. J. D.; Le Phuoc, D.; Lefrançois, M.: SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators. Journal of Web Semantics 56/, pp. 1–10, May 1, 2019, visited on: 09/28/2023.

[LAC17] Lepot, M.; Aubin, J.-B.; Clemens, F. H. L. R.: Interpolation in Time Series: An Introductive Overview of Existing Methods, Their Performance Criteria and Uncertainty Assessment. Water 9/10, p. 796, Oct. 2017, visited on: 09/28/2023.

[LBK15] Lee, J.; Bagheri, B.; Kao, H.-A.: A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems. Manufacturing Letters 3/, pp. 18–23, Jan. 1, 2015, visited on: 09/26/2023.

[Le08] Lee, E. A.: Cyber Physical Systems: Design Challenges. In: 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC). 2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC). Pp. 363–369, May 2008, visited on: 09/26/2023.

[Le17] Lee, B.; Kim, D.-K.; Yang, H.; Oh, S.: Model Transformation between OPC UA and UML. Computer Standards & Interfaces 50/, pp. 236–250, Feb. 1, 2017, visited on: 10/06/2023.

[Ma23] Mangler, J.; Grüger, J.; Malburg, L.; Ehrendorfer, M.; Bertrand, Y.; Benzin, J.-V.; Rinderle-Ma, S.; Serral Asensio, E.; Bergmann, R.: DataStream XES Extension: Embedding IoT Sensor Data into Extensible Event Stream Logs. Future Internet 15/3, p. 109, Mar. 2023, visited on: 08/08/2023.

[ML16] Madera, C.; Laurent, A.: The next Information Architecture Evolution: The Data Lake Wave. In: Proceedings of the 8th International Conference on Management of Digital EcoSystems. MEDES, Association for Computing Machinery, New York, NY, USA, pp. 174–180, Nov. 1, 2016, ISBN: 978-1-4503-4267-4, visited on: 10/06/2023.

[Mo14] Monostori, L.: Cyber-Physical Production Systems: Roots, Expectations and R&D Challenges. Procedia CIRP, Variety Management in Manufacturing 17/, pp. 9–13, Jan. 1, 2014, visited on: 10/05/2023.

[Mo18] Moura, R.; Ceotto, L.; Gonzalez, A.; Toledo, R.: Industrial Internet of Things (IIoT) Platforms - An Evaluation Model. In: 2018 International Conference on Computational Science and Computational Intelligence (CSCI). 2018 International Conference on Computational Science and Computational Intelligence (CSCI). Pp. 1002–1009, Dec. 2018, visited on: 10/05/2023.

[Sc19] Schiekofer, R.; Grimm, S.; Brandt, M. M.; Weyrich, M.: A Formal Mapping between OPC UA and the Semantic Web. In: 2019 IEEE 17th International Conference on Industrial Informatics (INDIN). 2019 IEEE 17th International Conference on Industrial Informatics (INDIN). Vol. 1, pp. 33–40, July 2019, visited on: 10/06/2023.