A Reference Model and Patterns for Production Event Data Enrichment

Mark van der Pas[1,3][0000−0001−6091−9340], Remco Dijkman[1][0000−0003−4083−0036], Alp Akçay[2][0000−0003−2000−6816], Ivo Adan[1][0000−0002−4493−6367], and John Walker[3]

[1] Eindhoven University of Technology, Department of Industrial Engineering & Innovation Sciences, Eindhoven 5600MB, The Netherlands
[2] Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, United States
[3] Semaku B.V., Eindhoven 5617BC, The Netherlands
m.c.a.v.d.pas@tue.nl

Abstract. With the advent of digital transformation, organisations are increasingly generating large volumes of data through the execution of various processes across disparate systems. By integrating data from these heterogeneous sources, it becomes possible to derive new insights essential for tasks such as monitoring and analysing process performance. Typically, this information is extracted during a data pre-processing or engineering phase. However, this step is often performed in an ad-hoc manner and is time-consuming and labour-intensive. To streamline this process, we introduce a reference model and a collection of patterns designed to enrich production event data. The reference model provides a standard way for storing and extracting production event data. The patterns describe common information extraction tasks and how such tasks can be automated effectively. The reference model is developed by combining the ISA-95 industry standard with the Event Knowledge Graph formalism. The patterns are developed based on empirical observations from event data sets originating in manufacturing processes and are formalised using the reference model. We evaluate the relevance and applicability of these patterns by demonstrating their application to use cases.

Keywords: Data Enrichment · Inference Pattern · Event Knowledge Graph · Production Trace

1 Introduction

There are often many different information systems that capture data about processes executed in an organisation. With digital transformation, the number of systems and the volume of data is growing further [25]. The data come in many different forms, ranging from master data describing a company's assets to quality measurements of a product/service delivered to a customer. Additionally, there is a high variation in sources, for example, a human manually entering a text or numeric value in a system, or the fully automated capturing of high-frequency sensor readings. All these data together can be used to describe the execution of processes, which can be used to analyse performance and quality issues and identify areas for improvement.

There is often a big gap between the raw data captured by the information systems and the relevant insights, or information, useful to a company [32]. Cleaning the data and deriving the 'hidden' information is commonly done in a data pre-processing or data engineering activity, which can be challenging and time-consuming, as it typically requires ad hoc and manual effort [6,12]. It also requires domain knowledge, especially when analysing event data [34].

To aid in this task, we propose a collection of patterns to derive 'hidden' information from data. Patterns are common solutions to frequent problems [2], in this case the problem of extracting information from event data. Patterns are successfully adopted in several other domains. Well-known work is the work by the 'Gang of Four' [19] and Fowler [18] about patterns related to software design. Closer to the business process community are patterns in workflows [1] and event logs [35]. Patterns can be used as a source of learning how to easily extract information and as a step towards automating the extraction. Ultimately, our research aims to improve the quality of the extracted information and the speed with which it is extracted. This paper is a first step towards that goal. The patterns described in this work are inspired by practical use cases, so we adopt the definition by Fowler [17]: "A pattern is an idea that has been useful in one practical context and will probably be useful in others."

We extract the patterns from observations on data sets from four different industrial production environments. To describe and formalise the patterns, we defined a reference model. The reference model combines the well-known standard ISA-95 [31] and the Event Knowledge Graph formalism [14,15]. The benefit of using this reference model is that it provides both a uniform terminology to describe the patterns and a way to automatically apply the patterns to the data stored according to the reference model. Subsequently, we define each of the patterns in terms of the reference model and motivate them by providing use cases. We also show in which of the four different industrial production environments studied the patterns are observed.

In summary, the contributions of this work are as follows:

- Collected generic patterns for enriching event data from multiple discrete manufacturing data sets;
- Defined a reference model that describes the terminology;
- Formalized the patterns and illustrated their applications;
- Provided template queries to apply the patterns and an example data set in a public repository[4].

The remainder of this work is structured as follows. First, in Section 2 we present related work on patterns and enrichment of event data. Section 3 gives an overview of the approach that we used to arrive at the patterns presented

[4] https://github.com/gitmpje/production-trace-patterns/

in this work, followed by a description of the patterns in Section 4. Finally, we provide an evaluation in Section 5 and concluding discussion in Section 6.

2 Related work

In this section, we present previous work related to patterns for deriving insights from data, specifically data originating from manufacturing processes. First, we give some background information on the usage of patterns. Subsequently, we discuss the work related to the application of patterns on event data in general. We position our work within the event processing and reasoning literature, where the generation of new insights from event data is also of interest. Finally, we discuss several ontologies for the manufacturing domain. Those ontologies describe common data patterns in manufacturing environments and provide the basis for the concepts used in the patterns presented in this work.

Suriadi et al. [35] describe patterns to detect imperfections in event data. The proposed patterns can be used in the first place to detect common data quality issues in event data sets and subsequently remove them. Those patterns are therefore also part of the data pre-processing step. Based on experts' knowledge and common sense, it is also possible to define inference rules that can be used to derive missing data (identifiers) from an incomplete event data set [36]. Another approach is to use process models to infer missing or unobserved events [16]. In addition to defining patterns or rules with the help of experts, it is also possible to learn operators that can be used to complete event data [29]. Lee et al. [26] propose a model and a system to combine event data with information on the bill of material to reconstruct the production trace of products. In contrast to the approaches presented above, the patterns proposed in this work are not targeted to detect imperfections in the data, but rather to enrich the data with initially hidden information.

Several works propose event processing and stream reasoning methods to enrich event data. Anicic et al. [4] propose a graph query language for event processing, which can also be used to express patterns on top of multi-dimensional event data. Bonte et al. [7] propose a reasoning approach for complex event processing to derive information from an event stream annotated with semantics from an ontology [39]. Teymourian [37] presents an extensive study of use cases and approaches for knowledge-based event processing, including the enrichment of event streams. Those works show the relevance of deriving information from event data and propose several general methods to retrieve new information, but do not provide the patterns to retrieve domain-specific information. We contribute to this literature stream by describing a set of patterns specifically for the manufacturing domain.

Within the manufacturing domain, there have been several efforts to model knowledge and enable data access using ontologies. Some of the first ontologies were proposed by Lemaignan et al. [27] and Borgo et al. [8]. More recently, with the emergence of Industry 4.0, sensors become an important concept [21,3]. Furthermore, there are efforts to develop a foundational ontology for the industrial manufacturing domain [24] and ontologies for specific manufacturing settings [10]. Yang et al. [42] build on previous work to define an ontology for industrial production workflows. Byun et al. [9] propose a graph-based model for describing events for traceability in manufacturing and the supply chain. Previous work defining manufacturing ontologies provides an overview of the data structures and information that is relevant in manufacturing environments. However, they do not describe how to derive the information, which is what we aim to achieve with the patterns collected in this work.

3 Methodology

We identified the patterns based on observations about existing data sets and subsequently formalised them using a reference model that was constructed based on existing standards. This section describes the way in which the patterns are derived from the data sets and the way in which the reference model was constructed from the existing standards.

3.1 Reference Model

The reference model that will be presented in Section 4.1 is a consolidation of a general model and relevant ontologies from the manufacturing domain (Section 2). General concepts from the Event Knowledge Graph (EKG) model [14,15] are used as the basis of our model. EKG is proposed as a graph-based solution to deal with multi-dimensional event data and can be applied to any event log, including events generated from manufacturing processes [36].

As the production trace patterns target a manufacturing environment, we included more detailed manufacturing entities in the model, which are in line with the existing ontologies referred to in Section 2. Next to the general concepts, more detailed concepts are included in the model. Those detailed concepts are derived from the data sets and are used to motivate and describe the production trace patterns in more detail. The exact data structures, data sources, and terminology will differ between organisations, but the concepts can be used as inspiration and a guideline for applying the patterns.

Figure 1 gives an overview of the notation that is used to define the reference model and the pattern examples. The main objects are the events (blue) and entities (green); other objects are represented in yellow. In addition, we differentiate various types of relationships between objects. In the taxonomy, the subclass relation is used and the relation between event and entity is important. Other relations are represented using a dashed line. Finally, several annotations are used in the examples of the patterns. In those examples, the objects of interest are marked with a red dashed outline and the derived information or relations are represented in orange.

Note that the main goal of the model presented in this work is to reach a common understanding of the manufacturing concepts that are used to define and describe the production trace patterns. Therefore, our goal is to define a reference model and not a comprehensive ontology for the manufacturing domain.

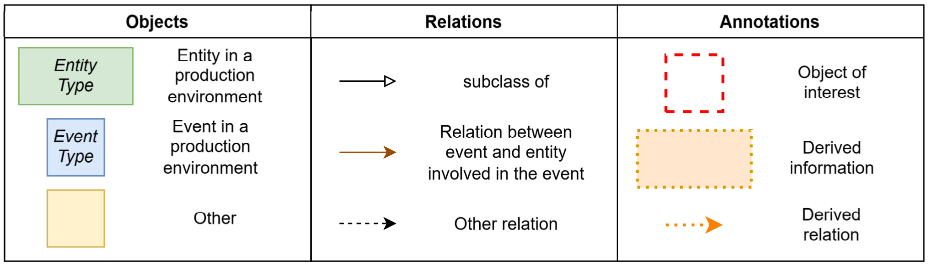

Fig. 1. Notation that is used to define the reference model and the pattern examples.

3.2 Patterns

The data sets from several (discrete) manufacturing companies in different domains inspired the patterns described in this work. The data sets contain production traces, defined as events and associated entities illustrating the production process of items (e.g., batches or lots). Although the use cases for the patterns presented in this work are scoped to the production trace in one organisation (internal traceability [33]), the same concept can be applied to a group of organisations within a supply chain. Production traces may include contextual data that describe events, entities, and their interrelations.

The first data set comes from semiconductor back-end operations, handling assembly and testing steps in a batch process flow shop [28]. The second data set is from the automotive domain, specifically a shop floor simulation where AGVs (Automated Guided Vehicles) transport components between workstations. Another set is from a contract manufacturing company, which produces to order with diverse equipment and human-driven operations (e.g., welding). he final set is from an industrial automation equipment manufacturer, detailing an assembly line where multiple workstations are managed by operators. The process includes several quality inspections that take place during and at the end of the process. Due to confidentiality, the data sets cannot be shared, but a representative dummy data set is available on Github repository[5].

Table 1 gives an overview of the data sets, describing the type of manufacturing setting, the level of automation, the level of digitisation (i.e., to what extent the data are automatically collected and communicated using digital systems), and the ISA-95 [31] level (at which event data are collected). The levels considered are:

level 3: Data from MES or systems on the same control/activity level.
level 2: Data registered at a resource (e.g. machine) level.
level 1: Data concerning sensing of the production process.

The use cases that led to the patterns were derived from the data sets and discussions with practitioners. Subsequently, the patterns observed in the use cases were generalised to arrive at the patterns that we present in this work.

[5] https://github.com/gitmpje/production-trace-patterns/tree/main/event-data

Table 1. Overview of the manufacturing settings that provided inspiration and motivation for the production trace patterns.

Data set	**Manufacturing setting**	**Level of automation**	**Level of digitization**	**ISA-95 level**
Semi-conductor back-end	Flow Shop	high	high	1-3
Automotive	Job Shop	high	high	2-3
Contract manufacturing	Job Shop	low	low	2-3
Industrial equipment	Assembly Line	low	high	1-3

4 Production trace patterns

In this section, we present the production trace patterns, but first we introduce the reference model that is used to define the patterns.

4.1 Reference Model

In this section, we describe the main concepts of the reference model. An overview of our model can be found in Figure 2.

The high-level model covers the concepts that are at the core of business processes and their execution. In business processes typically two types of entities can be distinguished. On the one hand, there are resources that execute a (production) process, like machines and operators, and on the other hand, there are entities that are the subject of a (production) process, for example the products and orders. Additionally, sensors are important in observing and monitoring today's processes. Those concepts, or variants thereof, can also be found in manufacturing environments [27,8,41,40,21]. A subset of concepts is selected such that all entities used in the patterns are represented. The subset consists of the following concepts, which are subclasses of `Entity`:

- `Resource`: resources execute the production processes by performing manufacturing operations [27,8,41,40];
- `ProductionEntity`: other entities involved in the production process, like components, products [40] and orders [8];
- `Sensor`: sensors are used to measure/observe processes, which is important in Industry 4.0 manufacturing environments [21].

The (manufacturing)process [21,40] is not included in the model, but the `Events` are a result of the processes executed by resources on production entities. Those five concepts (`Event`, `Entity`, `Resource`, `ProductionEntity`, and `Sensor`) define the high-level model, together with the `entity` relation between `Event` and `Entity`. In the model, we do not include the specific attributes that an object can have. However, an `Event` should at least have a timestamp at which the event

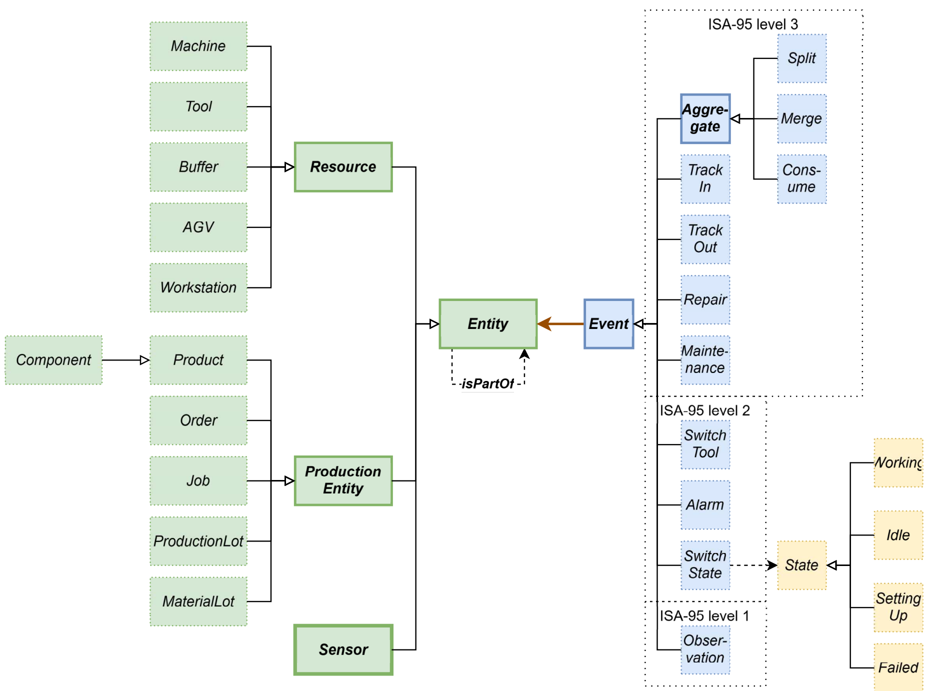

Fig. 2. Reference model describing the different event and entity types, and their relations, that are used to describe the production trace patterns. The concepts in ***bold*** define the high-level model.

occurred. A description of the more specific concepts can be found in the GitHub repository[6]. Note that the more specific concepts are a collection of entities and events that are of interest in the data sets and therefore not complete.

Next to the taxonomy of entities and events, several patterns leverage the notion that a certain entity is either logically or physically included in another entity. For example, a component can be *part of* a product, and a product can in turn be *part of* a production lot. This is a very common relation and is known as the `isPartOf` (or 'part-whole') design pattern [20,30,13].

4.2 Patterns

Inspired by the data sets we extracted the production trace patterns that can be used to enrich data sets with hidden information. The patterns are grouped by the type of information they retrieve. We distinguish the following types of patterns to enrich the production trace: 'aggregation over an interval' (Section 4.3), 'calculating the time that elapsed between events' (Section 4.4), 'deriving relation between entity and event' (Section 4.5), and 'deriving relation between entities' (Section 4.6). Table 2 gives an overview of the patterns, in which section they are discussed, what type of operation is used, and which of the classes from the model are represented.

Section	**Pattern**	**Operation**	**Class**		
			`Event`	`Resource`	`ProductionEntity`
4.3	1	Count	x	x	x
	2	Aggregate	x	x	x
4.4	3	Subtract	x	x	
	4	Subtract	x	x	
	5	Subtract	x	x	x
4.5	6	Relate	x	x	
	7	Relate	x	x	x
	8	Relate	x		x
	9	Relate	x		x
4.6	10	Relate	x	x	x

Table 2. Overview of the production trace patterns

In line with Suriadi et al. [35], the following components are used to describe each pattern:

- *Definition*: each pattern at least includes the following elements:

[6] `https://github.com/gitmpje/production-trace-patterns/blob/main/production-trace-patterns.ttl`

  - - An operation, like aggregate values or relate objects;
  - - One or two target object types, in terms of the reference model;
  - - A condition or constraint to target the relevant objects, for example a timing relation between events or the relation between an entity and an event.
- *Description*: a textual description of the pattern and in what type of manufacturing settings it is typically observed.
- *Use cases*: a set of use cases obtained from the different data sets (Table 1). Next to a description of the use case, the application of the pattern is illustrated through:
  - - A semi-formal rule or instantiation of the pattern for each of the listed use cases, using the terminology as defined in the reference model.
  - - A visualization of the pattern applied to one of the provided use cases. We also describe the parameter values that can be used to instantiate the pattern for the visualized use case.
- *Lightweight rule*: 'Lightweight' formal specification using SPARQL (query templates), which are shared in a Github repository[7]. Next to the formalized patterns, the repository also contains a dummy data set and a script to apply the patterns on the data set for some of the provided use cases.

The patterns that are presented in this work are derived from experiences analysing data sets (Section 3.2).

4.3 Enrich trace by aggregation over an interval

The following patterns occur frequently in production environments with various resources executing different steps in the production process. In those environments, data are typically captured at different levels. On the one hand, there are many events that mark the start and end of an interval. On the other hand, there are various events taking place during that interval. Aggregating the events in the interval will reveal new information. For example, the events describing the track in and out (start and completion of manufacturing) of a job at a machine are typically captured by the MES, while events like machine alarms are captured by a different system, e.g. an alarm monitoring system. For analysis of the performance of the machine and root cause analysis of incidents, a useful insight is the number of machine alarms that took place during the processing of the job. Similarly, there are often sensors monitoring the machine, but data from a sensor often provide valuable insights only after aggregating them over a certain time window. Below we describe the patterns that can be used to derive different types of information over an interval.

Pattern 1 *Count `Events` of a given type in an interval marked by two `Events` of given types, related to the same `Resource` and `ProductionEntity`*

[7] `https://github.com/gitmpje/production-trace-patterns/tree/main/sparql`

Description This pattern captures all use cases where new information can be derived by counting the events of a certain type in the interval between the start and finish of processing a production entity (like a job or batch) by a resource, such as the number of alarms that occurred while a machine was processing a job.

Use cases

1-1 In the interval between the start and finish of processing a job by a machine: count all alarms.
Rule: Count the number of `Alarm`s between a `TrackIn` and `TrackOut` for a `Job` on a `Machine`.

1-2 In the interval between the start and finish of processing a production lot by a machine: count all machine repairs.
Rule: Count the number of `Repair`s between a `TrackIn` and `TrackOut` for a `Job` on a `Machine`.

Figure 3 gives an example of use case 1-1, in this example two alarms occur in the interval between track in and out of a job on a machine. The (SPARQL) pattern template can be instantiated for this example as follows:

- Interval start event type: *IntervalStartType* = `TrackIn`;
- Interval end event type: *IntervalEndType* = `TrackOut`;
- Type of event that should be counted: *EventType* = `Alarm`.

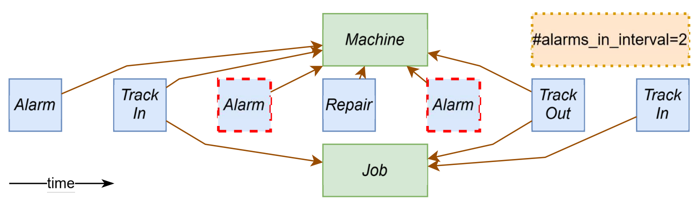

Fig. 3. Illustration of Pattern 1 applied to a data set containing alarms.

Pattern 2 *Aggregate* ***`Event`*** *attribute in an interval marked by two* ***`Events`*** *of given types, related to the same* ***`Resource`*** *and* ***`ProductionEntity`***

Description This pattern captures all use cases where new information can be derived by aggregating an attribute of the event in the interval between the start and finish of processing a production entity by a resource. In contrast to Pattern 1, this pattern aims to derive information by looking at specific attributes of the events that occur in the interval. A common example is the aggregation of sensor measurements, for example to detect if a certain parameter was on average higher when processing a job.

Use cases

2-1 Calculate average sensor measurement while processing a job (Figure 4).
Rule: Calculate the average measured value (`v`) of all `Observation`s between a `TrackIn` and `TrackOut` for a `Job` on a `Machine`.

2-2 Count the number of times a sensor value crosses a certain threshold while processing a job.
Rule: Count the number of times the value of an `Observation` between a `TrackIn` and `TrackOut` for a `Job` on a `Machine` is above a given threshold.

2-3 Sum the number of entities that are rejected between the start and finish of processing a production lot.
Rule: Sum the `quantityRejected` values between the `TrackIn` and `TrackOut` of that `ProductionLot` at a `Workstation`.

Figure 4 gives an example of use case 2-1, in this example two observations are made, with an average value of 11. The (SPARQL) pattern template can be instantiated for this example as follows:

- Interval start event type: $\mathit{IntervalStartType}$ = `TrackIn`;
- Interval end event type: $\mathit{IntervalEndType}$ = `TrackOut`;
- Event type of interest: $\mathit{EventType}$ = `Observation`;
- Attribute of event that should be aggregated: $\mathit{attribute}$ = `value` (`v`).

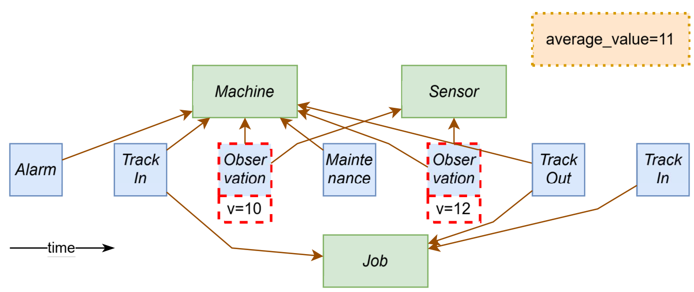

Fig. 4. Example of the application of Pattern 2 on a data set with sensor observations.

4.4 Enrich trace by calculating the time that elapsed between events

Next to defining an interval based on two events, the time that elapsed between certain events is useful information that can be derived from manufacturing event data sets. An obvious use case is to calculate the time between the start and finish of processing a job by a resource, which gives information about the performance of the resource and can be compared with the expected or planned processing time. Likewise, for the time between the start and end time of a machine breakdown. In other cases, it could be interesting to know the time between when a machine started processing a job and the closest preceding repair/maintenance activity, which might have impacted the machine's performance (in terms of throughput and quality).

Pattern 3 *Calculate the time between an `Event` and the closest preceding `Event` of given types related to a `Resource`*

Description This pattern captures all use cases where new information can be derived by comparing an event and the closest preceding event for a resource.

Use cases

3-1 Calculate the time (`t`) it took to process a job on a machine.
Rule: Calculate the time between a `TrackOut` and the closest preceding `TrackIn` for a `Job` on a `Machine`.

3-2 Calculate the time since the last maintenance before start processing a job.
Rule: Calculate the time between a `TrackIn` for a `Job` on a `Resource` and the closest preceding `Maintenance` for that `Resource`.

3-3 Calculate the time between processing two jobs (setup time).
Rule: Calculate the time between a `TrackOut` and the closest preceding `TrackIn` for a `Resource`.

3-4 Calculate the buffer time of a product in front of a resource.
Rule: Calculate the time between a `TrackOut` and the closest preceding `TrackIn` for a `Product` on a `Buffer`.

3-5 Calculate the transportation time of a product by an AGV.
Rule: Calculate the time between a `TrackOut` and the closest preceding `TrackIn` for a `Product` on a `AGV`.

Figure 5 gives an example of use case 3-2, in this example the time since the last maintenance can be derived from the track-in event with timestamp 12 and the maintenance event with timestamp 10. The (SPARQL) pattern template can be instantiated for this example as follows:

– Event type of interest: $EventType$ = `TrackIn`;
– Preceding event type of interest: $PrecedingEventType$ = `Maintenance`.

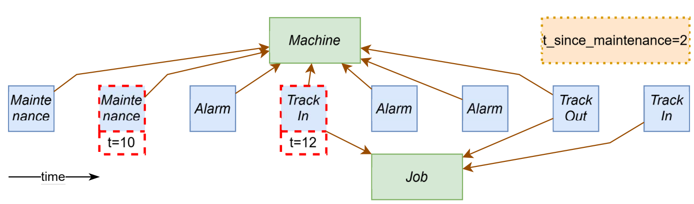

Fig. 5. Illustration of applying Pattern 3 to a data set containing maintenance events.

Pattern 4 *Calculate the time between an `Event` and the closest succeeding `Event` of the same type related to a `Resource`*

Description This pattern captures all use cases where new information can be derived by calculating the time that elapsed between two subsequent events of the same type. An example application is the calculation of the downtime of a machine, which can be derived from the time between when the machine switched to a failed state and the following event where it switches back to another state.

Use cases

4-1 Calculate the downtime of a machine (Figure 6).
Rule: Calculate the time between a `SwitchState` to `Failed` state and the closest succeeding `SwitchState` for a `Machine`.

Figure 6 gives an example of use case 4-1. In this example the downtime can be derived from the switch to failed state at time 10 and the switch back to working state at time 14. The (SPARQL) pattern template can be instantiated for this example as follows:

– Event type of interest: *EventType* = `SwitchState`.

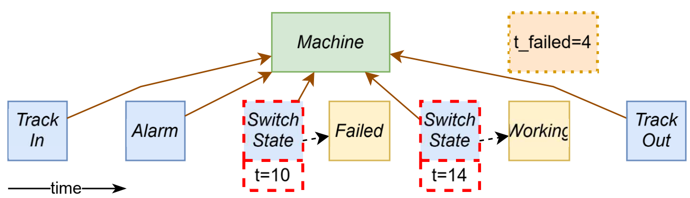

Fig. 6. Illustration of applying Pattern 4 to a data set containing switch state events.

Pattern 5 *Calculate the maximum time between two* ***`Events`*** *of given types related to an* ***`Entity`***

Description Next to the time between two subsequent events, it can also be of interest to look at the maximum time between two events of a certain type. This pattern can for example be applied to calculate the throughput time of a product, which can be derived from the first and last event where this product was tracked at a resource.

Use cases

5-1 Calculate the throughput time of a product.
Rule: Calculate the maximum time between a `TrackIn` and `TrackOut` of a `Product`.

Figure 7 gives an example of use case 5-1. In this example the throughput time can be derived from the first track in at time 3 and the last track out at time 15. The (SPARQL) pattern template can be instantiated for this example as follows:

– Interval start event type: $\mathit{IntervalStartType}$ = `SwitchState`;
– Interval end event type: $\mathit{IntervalEndType}$ = `SwitchState`.

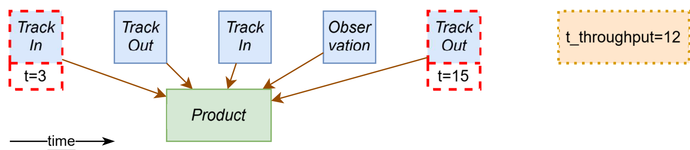

Fig. 7. Illustration of applying Pattern 5 to a series of events for a product.

4.5 Enrich trace by deriving a relation between an event and entity

When data from different sources are combined there are often hidden relations that can be derived. One typical use case is that it is interesting to know what tool is used by a resource to process a job. This relation can be derived by combining the switch tool events for the resource with the events describing the processing of a job by that resource. In other cases, it is possible to derive information about the higher aggregation level entities, like a batch, from events related to lower level entities, like a product, when it is known that this product was part of the batch. In that case, the relation between the lower-level events and the higher-level entity can be derived. This information can be used to generate a more complete view of the production process for a batch, and can for example be used to calculate the total processing time of the batch.

Pattern 6 *Relate* ***`Event`*** *(e1) to a* ***`Resource`*** *that is related to the closest preceding* ***`Event`*** *(e2) of a given type*

Description This pattern captures all use cases where new information can be derived by finding an event closest preceding the start processing at a resource and its related entities.

Use cases

6-1 Derive the relation between the event that marks the start of processing a job on a machine and the tool related to the closest preceding switch tool event.
Rule: Derive the relation between a `TrackIn` of a `Job` on a `Machine` and the `Tool` related to the closest preceding `SwitchTool` for that `Machine`.

Figure 8 gives an example of use case 6-1, where the machine switched tools directly before a job was tracked at that machine. The (SPARQL) pattern template can be instantiated for this example as follows:

– Event type of interest: *EventType* = `TrackIn`;
– Preceding event type of interest: *PrecedingEventType* = `SwitchTool`.

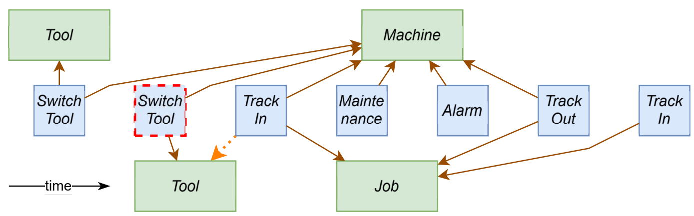

Fig. 8. Illustration of applying Pattern 6 to a data set containing switch tool events.

Pattern 7 *Relate* `Event` *to an* `Entity` *based on a* `isPartOf` *relation between* `Entitys`

Description This pattern captures all use cases where new information can be derived by finding all events related to a production entity that is part of another production entity. Discovering the relation between lower-level entities and production events is useful for traceability, it can for example be used to discover the production trace for a specific product, instead of the production lot this product is part of. Next to that, it can give insights into the performance of resources, for example the number (and type) of products that a resource worked on during a certain interval.

Use cases

7-1 Derive the relation between an event on the production lot level and the product that is part of that lot.
Rule: Derive the relation between an `Event` related to a `ProductionLot` and the `Product` which `isPartOf` that `ProductionLot`.
7-2 Derive the relation between an observation made by a sensor and the resource where this sensor is located.
Rule: Derive the relation between an `Observation` by a `Sensor` and the `Resource` that this `Sensor` `isPartOf`.

Figure 9 gives an example of use case 7-1, where there are events captured on different aggregation levels. The (SPARQL) pattern template can be instantiated without specifying parameters.

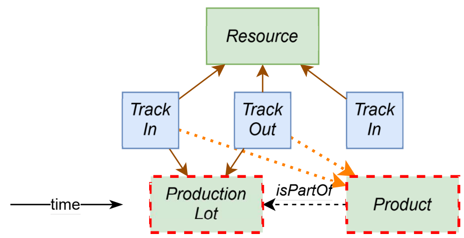

Fig. 9. Application of Pattern 7 on product and production lot level events.

It is common in batch/lot manufacturing environments that lots are split into smaller lots and/or merged into bigger lots, for example to distribute the load over the different machines optimally [11]. The meaning of a split or merge event can be used to relate an event to other entities it is (possibly) related to. For example, if some entities are split from a lot into a new lot, then the events related to this new lot are also related to (entities that were part of) the initial lot. Note that this can be applied recursively, to create the relevant relations along the complete chain of aggregation (split/merge/consume) events. This use case generally applies to aggregation events [22].

Pattern 8 *Relate `Events` preceding an `Aggregate` event to an indirectly related `ProductionEntity`*

Description This pattern captures all use cases where new information can be derived by finding all events preceding an event where some production entities are aggregated. For example, consider two production lots that first follow distinct production traces on different machines, are subsequently merged into one production lot, and then continue as one entity through the remaining production steps. In this scenario, Pattern 8 can be applied to relate all events from the distinct production lots to the merged production lot and in that manner discover the complete production trace for the products in the merged production lot.

Use cases

8-1 Derive the relation between events related to a lot, and the lots split from that lot.
Rule: Derive the relation between `Events` preceding a `Split` related to a `ProductionLot` and another `ProductionLot` that is also related to this `Split`.

8-2 Derive the relation between events related to a lot that was merged into another lot, and the merged lot.
Rule: Derive the relation between `Events` preceding a `Merge` related to a `ProductionLot` and another `ProductionLot` that is also related to this `Merge`.

8-3 Derive the relation between events related to a component that is consumed to assemble/produce to produce a product.
Rule: Derive the relation between `Events` preceding a `Consume` related to a `Component` and a `Product` that is also related to this `Consume`.

Figure 10 gives an example of use case 8-1, where there is one event related to the lot before two lots are split from that lot. The (SPARQL) pattern template can be instantiated for this example as follows:

– Type of entity for which new relations should be derived: *RelatedEntityType* = `ProductionLot`.

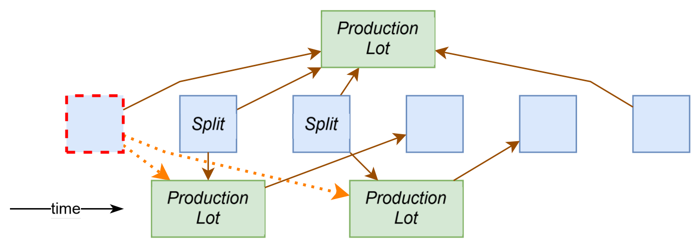

Fig. 10. Illustration of applying Pattern 8 to a data set containing an event describing the split of a production lot into smaller lots.

Pattern 9 *Relate* ***`Events`*** *succeeding an* ***`Aggregate`*** *event to an indirectly related* ***`ProductionEntity`***

Description This pattern captures all use cases where new information can be derived by finding all succeeding events related to a production entity that is involved in the same event as another production entity.

Use cases

9-1 Derive the relation between events related to a lot that was split from a lot and the original lot.
Rule: Derive the relation between `Events` succeeding a `Split` related to a `ProductionLot` and another `ProductionLot` that is also related to this `Split`.

9-2 Derive the relation between events related to a merged lot, and the lots merged into that lot.
Rule: Derive the relation between `Events` succeeding a `Merge` related to a `ProductionLot` and another `ProductionLot` that is also related to this `Merge`.

Figure 11 gives an example of use case 9-1, where there are two events related to lots that were split from another lot. The (SPARQL) pattern template can be instantiated for this example as follows:

– Type of entity for which new relations should be derived: *RelatedEntityType* = `ProductionLot`.

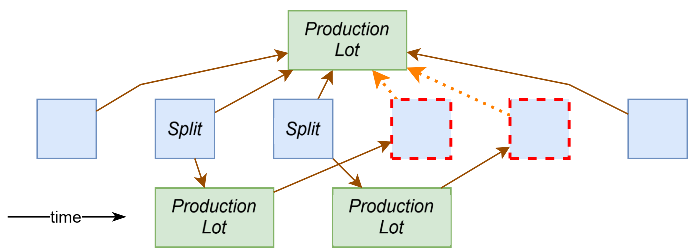

Fig. 11. Illustration of applying Pattern 9 to a data set containing an event describing the split of a production lot into smaller lots.

4.6 Enrich trace by deriving a relation between two entities

In an environment where data from various aggregation levels are collected by isolated systems, it might remain unknown whether a specific product is part of a batch. For example, the batch-level events will be captured by the MES, while there might be another system that registers operations on individual products by a machine. These data can be combined to derive to what batch the different products belong.

Pattern 10 *Relate* `ProductionEntities` *(*`partOf` *relation) based on* `Events` *between* `TrackIn` *and* `TrackOut` *at a* `Resource`

Description This pattern captures all use cases where new information can be derived by finding all production entities that are part of another production entity based on events that occurred in a certain interval. Deriving the 'part of' relation is in the first place useful for traceability, where it is necessary to know which product belonged to which batch(es). The relation can also be used for monitoring the performance, in which case we are interested in the number of products that are part of a batch.

Use cases

10-1 Derive the logical 'part of' relation between a product and a batch from (lower level) events related to the product in the interval between the start and finish of processing the batch on a resource.

Rule: Derive the `isPartOf` relation between a `Product` and a `Batch` from an `Event` related to the `Product` that occurs in the interval between `TrackIn` and `TrackOut` of the `Batch` at a `Resource`.

10-2 Derive the physical 'part of' relation between a component and a product from consume/assembly events in the interval between the start and finish of processing the product.
Rule: Derive the `isPartOf` relation between a `Product` and a `Component` from a `Consume` related to the `Component` that occurs in the interval between `TrackIn` and `TrackOut` of the `Product` at a `Resource`.

Figure 12 gives an example of use case 10-1, where a specific product is observed at a resource in the interval where this resource was processing a certain production lot. In this case it can be derived that the product is part of the production lot. The (SPARQL) pattern template can be instantiated for this example as follows:

- Interval start event type: *IntervalStartType* = `TrackIn`;
- Interval end event type: *IntervalEndType* = `TrackOut`;
- Type of entity (on a lower aggregation level) for which to derive the part of relation: *PartEntityType* = `Product`.

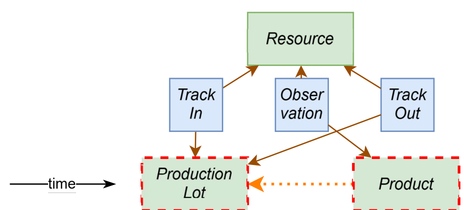

Fig. 12. Application of Pattern 10 on product and production lot level events.

4.7 Combining patterns

The patterns listed above can also be combined and sequentially applied to derive further (aggregated) information. Below we describe two example use cases and what patterns can be used to derive the information.

Average downtime of a resource: a typical piece of information that is of interest for monitoring the performance of resources is the average downtime of the resource. Use case 4-1 already showed how Pattern 4 can be used to compute the duration of one downtime period of a machine. Subsequently, Pattern 2 can be used to aggregate the downtime of the machine over a certain interval, for example the average or variation.

Aggregated average processing time: similarly, Pattern 2 can be combined with Pattern 5 to compute the average processing time of a resource, which is another useful insight for monitoring the performance.

5 Implementation and evaluation

In this section, we examine in what manufacturing environments the presented patterns can be applied. We consider the use cases presented in the previous section and the manufacturing data sets presented in Section 3.2. The results of the study are summarized in Table 3, which gives an overview of the use cases for the patterns and the data sets in which they are observed.

The patterns are implemented as SPARQL [38] query templates[8]. Those queries can be used to extract information from the event data and enrich it. The repository also contains a dummy data set and a script to apply the patterns for a selection of the use cases.

As can be seen in Table 3, most of the use cases are observed in multiple data sets. If we aggregate the results over the patterns, we can see that most of the patterns apply to at least three data sets, except for Pattern 4, 6, and 9, which appear to be more specific for a certain data set.

Table 3. Overview of the use cases and in what manufacturing setting they are observed.

Use case	Semi-conductor back-end	Automotive	Contract manufacturing	Industrial equipment
1-1	x			
1-2	x	x		x
2-1	x	x		x
2-2	x			x
2-3	x			
3-1	x	x	x	x
3-2	x	x	x	
3-3		x		
3-4		x		x
4-1		x		
5-1	x	x	x	x
6-1		x		
7-1	x			x
7-2	x	x		
8-1	x			
8-2	x			
8-3	x	x		x
9-1	x			
9-2	x			
10-1	x			
10-2	x	x	x	x

[8] https://github.com/gitmpje/production-trace-patterns/tree/main/sparql

6 Conclusion and Discussion

The digital transformation has led to an increase in data collection across various systems. When data from these systems are combined, they yield new and valuable insights. Knowledge about the context and data of the process is required to extract this hidden information. Therefore, this information is often extracted ad hoc during data pre-processing. In examining event data sets from four production environments, we identified recurring themes in the information of interest. This paper introduces production trace patterns, generic patterns to uncover hidden information. These patterns are defined using a reference model and are illustrated through industry use cases. To our knowledge, this is the first work in which patterns are described for deriving hidden information from event data sets from manufacturing processes.

We recognise that restricting our study to data sets and use cases in the (discrete) manufacturing field is a significant limitation of this work. This is because EKGs, which underpin our reference model, are primarily used for multi-dimensional event data from manufacturing processes. Nevertheless, this work serves as an initial step, and we anticipate the applicability of the abstract patterns in other fields. For example, Pattern 1 could be used in a data centre, where a computing `Resource` runs a programme to handle a job (`ProductionEntity`), with alarm `Event`s triggered by exceptions.

We do not assert that the patterns gathered here are complete. Each manufacturing setting is unique, necessitating distinct patterns to uncover hidden information. Thus, there is room for adapting and expanding patterns across diverse manufacturing environments, not limited to discrete manufacturing. Collecting patterns is a step forward towards systematically enriching event data. We hope the community will embrace and apply these patterns, growing the collection over time. The broader the pattern set, the more engineers and practitioners can benefit from new insights in their event data.

Future work can take several paths. First, as mentioned previously, additional patterns can probably be defined to derive hidden information from manufacturing event data. Collaborating with peers and using diverse data sets could help to gather and document these patterns. Furthermore, it would be beneficial to establish a set of checks to evaluate and validate the derived information, for example using SHACL [23]. This work outlines patterns, but does not specify the application steps. Thus, future research can be done on a process and a tool to configure the patterns, such that they can be tailored to specific use cases and data sets [5].

Acknowledgments. This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 957204, the project MAS4AI (Multi-Agent Systems for Pervasive Artificial Intelligence for assisting Humans in Modular Production). Additionally, it is part of the project AIMS5.0 supported by the Chips Joint Undertaking and its members, including the top-up funding by National Funding Authorities from involved countries under grant agreement no. 101112089.

References

1. Van der Aalst, W.M., Ter Hofstede, A.H., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases **14**(1), 5–51 (7 2003). https://doi.org/10.1023/A:1022883727209/METRICS
2. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language : Towns, Buildings, Construction. Oxford University Press (1977)
3. Alvanou, G., Lytra, I., Petersen, N.: An MTConnect Ontology for Semantic Industrial Machine Sensor Analytics. In: SWeTI: Semantic Web of Things for Industry 4.0. pp. 57–80. Heraklion (2018)
4. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In: Proceedings of the 20th international conference on World wide web. pp. 635–644 (2011)
5. Benevolenskiy, A., Roos, K., Katranuschkov, P., Scherer, R.J.: Construction processes configuration using process patterns. Advanced Engineering Informatics **26**(4), 727–736 (10 2012). https://doi.org/10.1016/J.AEI.2012.04.003
6. Bilal, M., Oyedele, L.O., Qadir, J., Munir, K., Ajayi, S.O., Akinade, O.O., Owolabi, H.A., Alaka, H.A., Pasha, M.: Big Data in the construction industry: A review of present status, opportunities, and future trends. Advanced Engineering Informatics **30**(3), 500–521 (8 2016). https://doi.org/10.1016/J.AEI.2016.07.001
7. Bonte, P., Tommasini, R., Valle, E.D., De Turck, F., Ongenae, F.: Streaming MASSIF: Cascading Reasoning for Efficient Processing of IoT Data Streams. Sensors **18**(11), 3832 (11 2018). https://doi.org/10.3390/S18113832
8. Borgo, S., Leitão, P.: Foundations for a Core Ontology of Manufacturing. In: Sharman, R., Kishore, R., Ramesh, R. (eds.) Ontologies: a Handbook of Principles, Concepts and Applications in Information Systems, vol. 14, pp. 751–775. Springer, Boston, MA (2007)
9. Byun, J., Woo, S., Kim, D.: Efficient and privacy-enhanced object traceability based on unified and linked EPCIS events. Computers in Industry **89**, 35–49 (8 2017). https://doi.org/10.1016/J.COMPIND.2017.04.001
10. Cao, Q., Beden, S., Beckmann, A.: A core reference ontology for steelmaking process knowledge modelling and information management. Computers in Industry **135**, 103574 (2 2022). https://doi.org/10.1016/J.COMPIND.2021.103574
11. Cheng, M., Mukherjee, N.J., Sarin, S.C.: A review of lot streaming. International Journal of Production Research **51**(23-24), 7023–7046 (11 2013). https://doi.org/10.1080/00207543.2013.774506
12. Dogan, A., Birant, D.: Machine learning and data mining in manufacturing. Expert Systems with Applications **166**, 114060 (3 2021). https://doi.org/10.1016/J.ESWA.2020.114060
13. Dublin Core: DCMI: Is Part Of (10 2023), https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/isPartOf/
14. Esser, S., Fahland, D.: Multi-Dimensional Event Data in Graph Databases. Journal on Data Semantics **10**(1-2), 109–141 (2021). https://doi.org/10.1007/s13740-021-00122-1
15. Fahland, D.: Process Mining over Multiple Behavioral Dimensions with Event Knowledge Graphs. In: Process Mining Handbook, vol. 448, pp. 274–319. Springer Science and Business Media Deutschland GmbH (2022). https://doi.org/10.1007/978-3-031-08848-3{_}9
16. Fahland, D., Denisov, V., Van Der Aalst, W.M.P.: Inferring Unobserved Events in Systems with Shared Resources and Queues. Fundamenta Informaticae **183**(4), 203–242 (2021). https://doi.org/10.3233/FI-2021-2087

17. Fowler, M.: Analysis patterns: reusable object models. Addison-Wesley, Massachusetts (1997)
18. Fowler, M.: Patterns of Enterprise Application Architecture. Addison Wesley (2002)
19. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (11 1994)
20. Gerstl, P., Pribbenow, S.: A conceptual theory of part-whole relations and its applications. Data & Knowledge Engineering **20**(3), 305–322 (11 1996). https://doi.org/10.1016/S0169-023X(96)00014-6
21. Giustozzi, F., Saunier, J., Zanni-Merk, C.: Context modeling for industry 4.0: An ontology-based proposal. Procedia Computer Science **126**, 675–684 (2018)
22. GS1: EPCIS Standard. Tech. rep., GS1 (6 2022), https://ref.gs1.org/standards/epcis/
23. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL) (2017), https://www.w3.org/TR/2017/REC-shacl-20170720/
24. Kulvatunyou, B.S., Wallace, E., Kiritsis, D., Smith, B., Will, C.: The industrial ontologies foundry proof-of-concept project. In: APMS 2018: Advances in Production Management Systems. Smart Manufacturing for Industry 4.0. vol. 536, pp. 402–409. Springer New York LLC (2018). https://doi.org/10.1007/978-3-319-99707-0{_}50/TABLES/2
25. Lee, C.H., Liu, C.L., Trappey, A.J., Mo, J.P., Desouza, K.C.: Understanding digital transformation in advanced manufacturing and engineering: A bibliometric analysis, topic modeling and research trend discovery. Advanced Engineering Informatics **50**, 101428 (10 2021). https://doi.org/10.1016/J.AEI.2021.101428
26. Lee, D., Park, J.: RFID-based traceability in the supply chain. Industrial Management and Data Systems **108**(6), 713–725 (2008). https://doi.org/10.1108/02635570810883978/FULL/PDF
27. Lemaignan, S., Siadat, A., Dantan, J., Semenenko, A.: MASON: A proposal for an ontology of manufacturing domain. In: IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06). p. 195–200. Prague (2006)
28. Lin, J.T., Chen, C.M.: Simulation optimization approach for hybrid flow shop scheduling problem in semiconductor back-end manufacturing. Simulation Modelling Practice and Theory **51**, 100–114 (2 2015). https://doi.org/10.1016/J.SIMPAT.2014.10.008
29. Müller, G., Bergmann, R.: Case Completion of Workflows for Process-Oriented Case-Based Reasoning. In: Case-Based Reasoning Research and Development: 24th International Conference, ICCBR 2016. pp. 295–310. Springer International Publishing (2016)
30. Ontology Design Patterns (ODP): Submissions:PartOf - Odp (3 2010), http://ontologydesignpatterns.org/wiki/Submissions:PartOf
31. OPC Foundation: OPC UA Online Reference (2019), https://reference.opcfoundation.org/v104/ISA-95/docs/4.2.3/
32. Rowley, J.: The wisdom hierarchy: representations of the DIKW hierarchy. Journal of information science **33**(2), 163–180 (2007)
33. Schuitemaker, R., Xu, X.: Product traceability in manufacturing: A technical review. Procedia CIRP **93**, 700–705 (2020). https://doi.org/10.1016/J.PROCIR.2020.04.078
34. Schuster, D., van Zelst, S.J., van der Aalst, W.M.: Utilizing domain knowledge in data-driven process discovery: A literature review. Computers in Industry **137**, 103612 (5 2022). https://doi.org/10.1016/J.COMPIND.2022.103612

35. Suriadi, S., Andrews, R., ter Hofstede, A.H., Wynn, M.T.: Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. Information Systems **64**, 132–150 (3 2017). https://doi.org/10.1016/J.IS.2016.07.011
36. Swevels, A., Dijkman, R., Fahland, D.: Inferring Missing Entity Identifiers from Context Using Event Knowledge Graphs. In: International Conference on Business Process Management. pp. 180–197. Springer Nature Switzerland, Cham (9 2023). https://doi.org/10.1007/978-3-031-41620-0{_}11/TABLES/3
37. Teymourian, K.: A Framework for Knowledge-Based Complex Event Processing. Ph.D. thesis, Freie Universität Berlin, Berlin (11 2014). https://doi.org/10.17169/REFUBIUM-11103
38. The W3C SPARQL Working Group: SPARQL 1.1 Overview (2013), https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/
39. Tommasini, R., Bonte, P., Della Valle, E., Mannens, E., De Turck, F., Ongenae, F.: Towards ontology-based event processing. In: International Experiences and Directions Workshop on OWL. vol. 10161 LNCS, pp. 115–127. Springer Verlag (2017). https://doi.org/10.1007/978-3-319-54627-8{_}9/TABLES/1
40. Usman, Z., Young, R.I.M., Chungoora, N., Palmer, C., Case, K., Harding, J.A.: Towards a formal manufacturing reference ontology. International Journal of Production Research **51**(22), 6553–6572 (2013)
41. W. Long: Construct MES Ontology with OWL. In: ISECS International Colloquium on Computing, Communication, Control, and Management. pp. 614–617. IEEE, Guangzhou (2008)
42. Yang, C., Zheng, Y., Tu, X., Ala-Laurinaho, R., Autiosalo, J., Seppänen, O., Tammi, K.: Ontology-based knowledge representation of industrial production workflow. Advanced Engineering Informatics **58**, 102185 (10 2023). https://doi.org/10.1016/J.AEI.2023.102185