A Process-Aware Model for Industrial IoT Integration and Analytics based on Knowledge Graphs

Claudia Diamantini[1], Alex Mircoli[2], Domenico Potena[1] and Emanuele Storti[1,*]

[1]Department of Information Engineering, Università Politecnica delle Marche, via Brecce Bianche, 60131, Ancona Italy
[2]Department of Economic and Social Sciences, Università Politecnica delle Marche, Piazzale Martelli 8, 60121, Ancona Italy

Abstract

The integration of the huge data streams produced by the Industrial Internet of Things (IIoT) can provide invaluable knowledge in the context of Industry 4.0/5.0, but is also an open research issue. The present paper proposes a semantic approach to this issue, centered around the notion of process as the backbone. The fundamental elements involved in IIoT and their relations are represented in an ontology, serving as a schema for a Process-aware IIoT Knowledge Graph where raw sensor data are enriched with information about process activities and the physical production environment. On top of that, a framework is developed for process-aware analytics and exploration of the IIoT data.

Keywords

Industrial Internet of Things, Data Integration, Business Process, Ontology, Knowledge Graph

1. Introduction

The last decade has been marked by a progressive integration of digital and advanced technologies into manufacturing processes. A key technological enabler in this transformation is the Internet of Things (IoT). The IoT is a network of devices capable of collecting and sharing data over the Internet [1]. IoT includes a wide range of application domains, each with unique and sometimes conflicting requirements. In particular, when it is applied to industrial settings, it is referred to as the Industrial Internet of Things (IIoT).

Unlike consumer-oriented IoT applications, where human interaction plays a central role in enhancing user awareness of their surroundings, IIoT focuses primarily on interconnecting machines and control systems. The goal is to enable self-organizing, self-optimizing, and self-healing manufacturing processes, promoting increasingly autonomous and intelligent factories. Within this context, the emphasis lies on optimizing the production line, where real-time data collection and processing allow for immediate reactions to events. In particular, the goal is to integrate data up to the highest possible level, i.e. the factory or even the entire enterprise, in order to enable global monitoring and optimizations. The architecture supporting this shift is the edge/fog-cloud computing model [2], which balances local data processing with cloud-level integration.

One of the key challenges in this integration process is handling the vast streams of raw data generated by IIoT devices and transforming them into structured, meaningful insights. In order to do that, it is necessary to bridge the abstraction gap between sensor observations and higher-level knowledge in the organization. Model-based approaches have been recognized as valuable tools for facilitating data integration by providing structured reference frameworks. Among these, business processes may play a fundamental role in integration. A process is a structured sequence of interrelated activities designed to produce an output, which may be a product or a service, based on an organization's objective. Multiple business units may participate in the same process, making coordination and resource integration essential for achieving efficiency and consistency across operations. In this sense, the process is therefore an element of homogenization for a company, as reported in [3]. In this paper, which draws inspiration

SEBD 2025: 33rd Symposium on Advanced Database Systems, June 16-19, 2025, Ischia, Italy
*Corresponding author.
c.diamantini@univpm.it (C. Diamantini); a.mircoli@univpm.it (A. Mircoli); d.potena@univpm.it (D. Potena); e.storti@univpm.it (E. Storti)

from a previous work published in [4], we connect low- and high-level information by proposing a process-centric semantic model that formalizes and interconnects key elements of IIoT to enable advanced analytics based on a global view of the company. These elements are: (i) sensors, (ii) business processes, (iii) associated performance indicators, and (iv) the factory, intended as the set of machinery and the environment that contain them. The paper introduces a comprehensive OWL ontology which acts as a conceptual knowledge layer that provides the structure for the *Process-aware IIoT Knowledge Graph*. The work also proposes a framework for querying the Knowledge Graph, enabling advanced analytics on integrated information derived from the four above-mentioned elements of the organization. A case study of the proposed methodology related to the production of metal accessories can be found in [4].

The remainder of this paper is structured as follows: Section 2 discusses the knowledge model of the framework, in terms of relevant knowledge artifacts and the ontological model serving as a schema for the Enterprise Knowledge Graph. On top of it, the framework is described in Section 3, focusing on the graph construction approach and on services for graph exploration and querying. Finally, Section 4 summarizes the conclusions and outlines directions for future work.

2. Knowledge model

This section discusses the relevant types of information in an industrial IoT environment and the proposed ontology that serves as the schema for the Enterprise Knowledge Graph.

2.1. Knowledge artifacts

Information, or knowledge artifacts, in the IIoT is categorized based on its typology and the object it represents. Firstly, we recognize two main perspectives that are intertwined in the IoT environment: a *process perspective* focusing on the definition, planning and execution of business processes, including all the process-related information, and a *sensor perspective* focusing on the characterization of sensors, their deployment on machines or in the environment and their observations, corresponding to retrieved values. The two perspectives are further described in terms of layers, according to the goal of the representation:

- the *design* layer includes all knowledge artifacts that are considered to be stable across multiple process executions. A first type of information is related to (a) process models (e.g. BPMN, UML, Petri net diagrams) in terms of activities, gateways and connections among them. The layer also includes (b) the detailed description of sensors, including their characterization in terms of typology, input/output, capabilities, machine in which they are deployed. Conversely, IoT resources, i.e. industrial machines/platforms, are characterized in terms of the type of activity they can perform (e.g., die casting, pressing, washing), the set of sensors they host and where they are located in the enterprise premises. In particular, the organization topology provides a spatial context by describing how the physical space of the enterprise is structured.
- The *planning layer* focuses on dynamic information which is related to the specific executions. This includes the description of the schedule for a given process instance, i.e. information on what activities are planned to be performed at a given time, on a given (set of) machine(s) and by whom. Additional information related to the specific production output are included, depending on the process type.
- The *execution layer* includes information on event logs, i.e. past executions of a process instance, and the related observations, namely monitored values from the sensors. From the process perspective, information on process execution is produced by workflow management systems, and similar platforms, in terms of event logs. They can be viewed as a multi-set of traces, where each trace describes the execution of a particular process instance as a totally ordered sequence of recorded events. Events typically refer to the starting or the ending of an activity, with further information including the resource (i.e. the person or the machine) which initiated or executed

the activity, the timestamp of the event and possible data elements recorded with the event, such as the size of the order, the number of input parts processed or output parts produced. In this perspective, performance indicators are aimed to measure quantitative parameters related to the whole execution of a process instance, to a given sub-process or even to single activities, e.g., the *Cycle time* measuring the end-to-end time needed to complete a (sub)process, or the *Number of defective products* that are discarded after performing the activity. On the other hand, from the sensor perspective, indicators correspond to (or are derived from) the physical parameters that are monitored, e.g., low-level simple metrics, such as *temperature* or *relative humidity*. Such indicators are typically not associated to specific processes, although the value of the monitored parameters may significantly affect some process phases. To make an example, in the wood industry, relative humidity levels should range from 55% to 60%. Levels below 40% reduce the moisture content in the wood and may lead to variations in its size (such as shrinking or swelling, warping and cracking), which may be irreversible.

2.2. Industrial IoT ontology

The ontological schema for the Industrial IoT Knowledge Graph is able to completely represent the knowledge artifacts introduced in the previous section and highlight the relations among them. It is represented as an OWL ontology which reuses and integrates, according to the best practices in ontology engineering, a number of existing ontological models. The Sensor, Observation, Sample, and Actuator ontology (SOSA, prefix *sosa*) [5], derived from the SSN ontology [6], is used for the representation of sensors, their capabilities and related platforms in which they are hosted, and observations made by sensors. The representation of smart environments in terms of rooms, buildings and mutual containment relations relies on the DogOnt ontology [7] (prefix *dog*), while KPIOnto [8] (prefix *kpi*) is adopted for the representation of indicators (i.e., measures) and their properties. Finally, event logs, corresponding traces and included events, are represented by the Event ontology [9] (prefix *onp*).

In addition to existing ontologies, a further module, namely the Business Process Ontology (prefix *bpo*), was designed as a lightweight model to provide the vocabulary representing BPMN elements, including a process model and its structure in terms of subprocesses, activities, gateways and relations among them. Finally, a set of classes and relations have been defined to build the bridge ontology capable of providing the needed connections among the different modules (prefix *meta*), as shown in Figure 1. They include the *IoTResource* class, which specialized a platform, has some capabilities in terms of *ActivityType*s and is located in a building environment. A *Task* generalizes both a *ScheduledTask* and an *ExecutedTask*, refers to a specific activity of a process, and is executed on an IoTResource. Furthermore, a task refers to the corresponding event and has the *start* and *complete* properties representing the timestamp in which it starts and ends. Furthermore, the task is linked to the causally following task in the same schedule/trace through the *nextTask* property. A *Result* is associated with a task (and then to the corresponding event) or to a trace, e.g. the number of defective products that is measured at the end of the trace. A result has a *Measurement* type, which generalizes an observable property, and a specific value. Finally, a *Trace* is linked to the particular process.

The final ontological model is hence capable to overall represent (i) the process and (ii) the sensor perspectives. As for the former, this is achieved by defining a process model and its relations with the related scheduling and executed process instances. The scheduling and the event log are represented through the same approach, i.e. as a Scheduled/Executed trace including a set of scheduled or executed events. Each event is in relation with the particular scheduled/executed activity in the process model, and with the IoT resource which is in charge of its execution (in case of a scheduled event) or has executed it (in case of an executed event). As for the sensor perspective, the ontology includes information on the domain configuration, in terms of IoT resources that are located at specific places in the organizational environment and may include a number of sensors onboard, characterized in terms of observable indicators and observations that have been collected over time. Each observation implicitly refers to one or more executed events, which in turn refer to particular process instances. This link can be derived by comparing the observation to the event timestamps, as discussed in the next section.

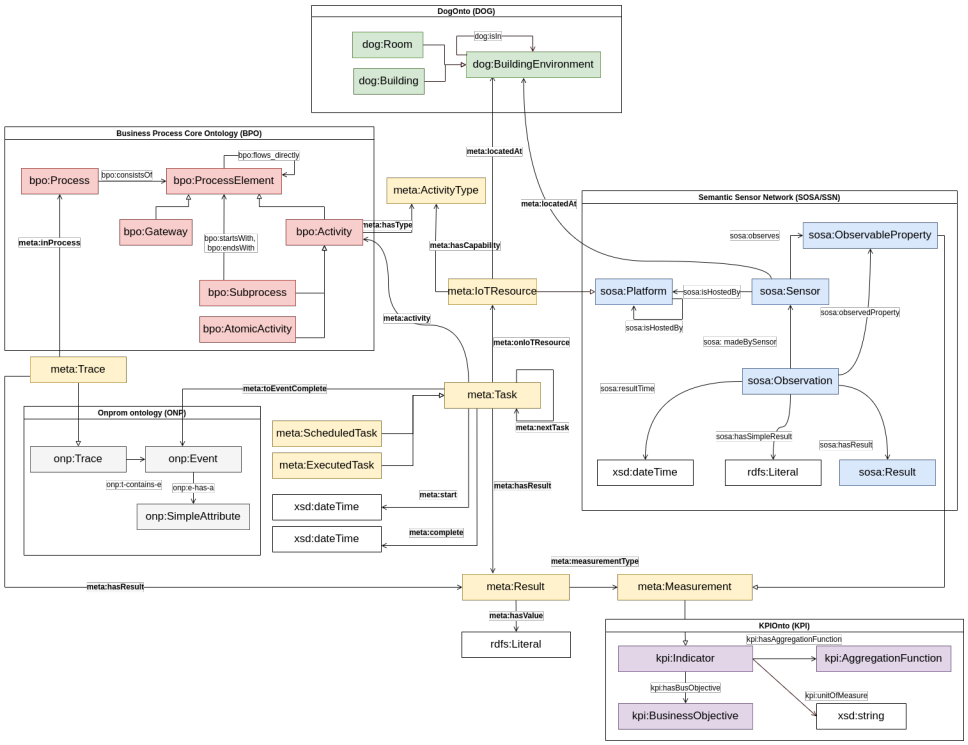

Figure 1: IIoT ontology schema.

3. Framework

In this section, we describe the procedure for building the Process-aware IIoT Knowledge Graph presented in Section 2, as well as the main challenges that may be posed by the integration of sensor-related and process-related information. Moreover, we discuss how to define queries to explore the IIoT Knowledge Graph.

3.1. Graph construction

The ontological model described in the previous section provides the foundation for building the Process-aware IIoT Knowledge Graph, which is structured as an RDF model that integrates all knowledge artifacts across the layers and perspectives introduced in Section 2. Some artifacts, such as process models, can be directly mapped to corresponding classes and properties in the ontology, while elements like *event logs* and *sensor observations* require additional pre-processing operations that include aligning them with the defined schema, extracting implicit relationships, and performing necessary data cleaning. In this subsection, we describe the challenges posed by event logs and sensor data, as well as the operations that need to be carried out to map these elements to the ontological model.

Regarding the event logs, we assume that they may capture multiple events related to each process activity, aligning with the IEEE XES standard[1] - an XML-based language for encoding event logs. This format proposes a life-cycle model for activities that defines different states, such as “start" (when execution begins), “complete" (when it ends), “suspend" (if paused), and “resume" (when restarted). As a minimal assumption, our approach considers event logs that include at least two key events per activity: “start" and “complete". Using these timestamps, we determine the duration of an activity. For example, let the event *e1* be the “start" of the “Coating" activity at “2024-11-20T09:32:10" and the event *e2* be the “complete" event at “2024-11-20T09:36:16": in this case, the activity was active for 4 minutes and 6 seconds. Mapping event logs into the ontology can be complex under particular circumstances, such as in the case of parallelisms. Event logs typically represent linear sequences of events, with each trace

[1] https://www.xes-standard.org/

corresponding to a specific process instance. A process model following a strictly sequential structure can hence be easily mapped. However, if the process includes parallel branches, a direct sequence of events in the log may not necessarily reflect the actual execution order as two consecutive events in the log might belong to parallel activities rather than forming a sequence. To address this challenge, we use the approach proposed in [10] to make causal relations between events in a trace explicit. In particular, a trace is translated into an instance graph [11], i.e. a graph whose nodes represent the events of the trace and edges represent a direct succession between two events defined by a causal relation. Parallel activities are clearly recognizable in the instance graph, as shown in Figure 2.

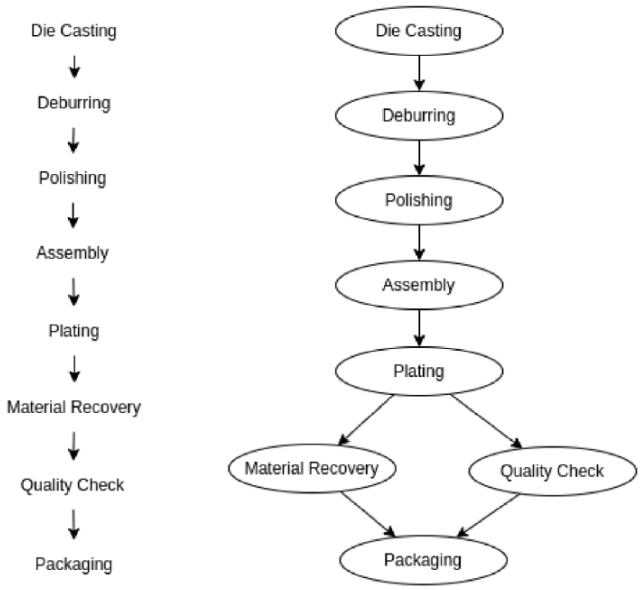

Figure 2: An example of instance graph (right) related to a trace (left). Parallelism is evident in the instance graph.

In order to map the trace to the Knowledge Graph, for each activity (identified by a pair of “start" and “complete" events e1 and e2 in the trace), an instance ET_i of the class *ExecutedTask* is created and linked to e2 through the *meta:toEventComplete* property. Such an instance also has a property *meta:start* equal to the timestamp of the event e1, and a property *meta:complete* with a value equal to the timestamp of event e2. The sequence of activities is defined through the property *meta:nextTask*: in particular, the instance ET_i is then linked to the causally following ExecutedTask ET_{i+1}. Information on the process activity and the particular related resource is available in the corresponding event as attributes. As such, a property *meta:activity* is added to connect the task to the proper instance of *bpo:Activity*. Similarly, a property *meta:onIoTResource* is added between the task and the related IoTResource. Finally, values of possible measurements that are related to the event are explicitly represented as instances of the *meta:Result* class, which includes the value and the type of measurement (as an instance of *meta:Measurement*).

Regarding sensor observations, when sensors are deployed on a machine, it is necessary to map them to the corresponding running activities (if any). In fact, observations generated by a sensor only include a recorded value corresponding to a specific observable property at a particular timestamp, without the contextual information about the executed task. The mapping process involves two key steps:

- Identifying the sensor’s deployment platform: this determines whether the sensor is installed on a machine or positioned in the factory environment.
- Associating the observation with an executed task: this is achieved by identifying the specific activity being performed on the machine at the time of the observation.

The mapping can be performed through the following SPARQL[12] query:

```
CONSTRUCT <obs> meta:observedOn ?e
WHERE {
  <obs> a sosa:Observation;
  sosa:resultTime ?t;
  sosa:madeBySensor ?s.
  ?s sosa:isHostedBy ?p.
  ?e a meta:ExecutedTask;
  meta:onIoTResource ?p;
  meta:start ?start;
  meta:complete ?end.
  FILTER (op:dateTime-less-than(?start,?t)).
  FILTER (op:dateTime-greater-than(?end,?t)).
}
```

where <obs> represents the considered observation.
For sensors installed in the factory environment rather than on machines, a similar query is used. However, in this case, the observation is linked to all tasks that were running, at the time of recording, on machines located in the room. As such, the approach allows to link an observation to multiple events belonging to different traces.

Constructing and maintaining the Knowledge Graph depends on the availability of information in digital format. Some data streams are natively in digital format: for example, event logs and production schedules are typically stored in information systems, such as Enterprise Resource Planning (ERP) or Manufacturing Execution System (MES). Similarly, sensor observations are collected automatically in Industrial IoT environments. In addition to that, some static data — such as factory layouts, machine configurations, and sensor characteristics — can be manually curated. However, explicit process models may not always be readily available. In such cases, process discovery techniques can be used to infer process models from event logs.

3.2. Graph exploration and analysis

In this subsection, we discuss how to query the Process-aware IIoT Knowledge Graph in order to extract and analyze data under a set of user conditions. Quantitative measures in the graph can be split into two main categories: those produced by sensors, described through instances of ObservableProperty (e.g., “Humidity"), and those related to processes which refer to the class Measurement (e.g., “Elapsed time", “Number of defective products"). The two categories of measures are expressed at different levels of granularity. In fact, the former refers to single observations performed by sensors, hence they are expressed at the lowest level of granularity. On the other hand, the latter are at a higher level of granularity as they refer to tasks or to whole traces. In some cases, it may be possible to aggregate sensor observations up to the task level, e.g., by aggregating all observations recorded during a task. Similarly, measures at the task level may be aggregated up to the trace level, e.g., by summing up the elapsed times for all tasks of a trace. Measures can be analyzed and aggregated along different perspectives: e.g., if the analysts are interested in what is happening in a given area (or during the execution of a specific trace), observations of sensors deployed in such an area (or active during the execution of the trace) could be considered. The Knowledge Graph offers a wide range of analytical dimensions, including the process model, process scheduling and execution, IoT resources, activity types, and environments. An important feature is that relationships between these dimensions are explicitly represented within the graph structure. For example, when analyzing a specific environment, such as a warehouse, the system automatically considers only the activity types that can be performed by resources located in that area. The same principle applies to executed tasks, relevant sensors, and recorded observations. Additionally, dimension hierarchies are defined dynamically based on the actual relationships among instances, allowing for flexible exploration of the graph. For instance, starting from a particular environment, the analysis can be moved to the “sub-environments" level using the isIn

relation. Alternatively, if the focus is on a process model, users can explore specific sub-processes for a lower-level analysis. Exploration and analysis can be performed by directly querying the graph through SPARQL queries. In such a way, all the expressivity of the query language can be exploited to select, filter, group and aggregate information combining dimensions in any possibly valid way. Formally, a SPARQL query is a tuple (DS, R, GP, SM) where:

- DS is an RDF Dataset;
- R is a result form;
- GP is a graph pattern;
- SM is a set of solution modifiers.

In the context of this work, DS is the RDF serialization of the IIoT Knowledge Graph, while R is the SELECT modifier, which enables specifying the target result. Regarding GP, the graph pattern of the query is generated from a fixed general pattern discussed in [4], capable of matching the whole graph. The basic graph pattern is then constrained through a set of FILTER conditions allowing to specify dimensions, measures of interest and grouping attributes, as described in the following:

1. selection of the analysis dimensions: this step identifies the relevant subgraph within the broader Knowledge Graph by applying constraints to the initial graph pattern;
2. selection of the measure to be analyzed: this step involves defining a condition to further constrain the graph pattern;
3. application of grouping: a grouping condition is expressed and added to the graph pattern to aggregate data as needed.

Finally, the SM, which represents the solution modifiers, includes both the grouping attributes and the aggregation operator applied to the selected measure.

To give an example, in a production process for metal accessories, if the user aims to check whether the position of workpieces during the $Deburring$ activity was nominal or not on trace number 554, the following query would be generated, where <Q> stands for the fixed general pattern:

```
SELECT ?task ?resource ?obsProp
        (AVG(?obs_simpleResult) as ?value)
WHERE {<Q>
       FILTER (?activityType = :deburring).
       FILTER (?trace = :trace554).
       FILTER (?obsProp = :pos_workpiece_x ||
              ?obsProp = :pos_workpiece_y ||
              ?obsProp = :pos__workpiece_z).
}
GROUP BY ?task ?resource ?obsProp
```

4. Conclusion

The paper presented a semantic approach to integrating data streams within the Industrial Internet of Things (IIoT), offering a more comprehensive view of manufacturing systems and supporting analytical tasks. The novelty of the approach lies in its process-oriented perspective, which is embedded in the design of the ontology and the associated Process-Aware IIoT Knowledge Graph. In particular, the proposed Knowledge Graph bridges the gap between raw sensor data and processes. The paper proposes a methodology for constructing and managing the Knowledge Graph, as well as a querying framework for data analysis. Rather than modeling specific industry-specific tasks (e.g., pressing or washing), the ontology defines generalizable concepts such as production activities and machine capabilities, making it adaptable to numerous industrial contexts. This flexibility allows for modeling different types of machinery, such as multi-function machining centers that handle turning, milling, and drilling.

Furthermore, the model supports a hierarchical representation of activities, offering different levels of granularity based on analytical requirements and also addressing the problem of low-level events from process logs. Future research will focus on conducting comprehensive evaluations using real-world case studies to validate and refine the approach. Addressing scalability challenges will be essential, particularly given the large volumes of data produced in industrial environments. Another area of research will be the investigation of the practical difficulties of integrating data from legacy systems like PLCs and SCADA. Moreover, the model will be extended to include all data artifacts produced in an industrial environment, following the evolution of IoT into the Internet of Everything (IoE) [13**?**]. A research direction will involve managing the evolution of production processes, particularly in relation to changes in process models. To address this, we plan to extend the ontology in order to handle process modifications and versioning. Finally, in future work we will also explore the application of this approach to key Industry 4.0/5.0 challenges, including data quality assessment, anomaly detection, and process optimization.

Acknowledgments

Alex Mircoli has received funding from the project Vitality – Project Code ECS00000041, CUP I33C22001330007 - funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - 'Creation and strengthening of innovation ecosystems,' construction of 'territorial leaders in R&D' – Innovation Ecosystems - Project 'Innovation, digitalization and sustainability for the diffused economy in Central Italy – VITALITY' Call for tender No. 3277 of 30/12/2021, and Concession Decree No. 0001057.23-06-2022 of Italian Ministry of University funded by the European Union – Next Generation EU. Emanuele Storti was partially supported by the PRIN 2022 project "HOMEY: a Human-centric IoE-based Framework for Supporting the Transition Towards Industry 5.0", funded by the European Union - Next Generation EU, Mission 4 Component 1 (code: 2022NX7WKE, CUP: F53D23004340006).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (2013) 1645–1660.

[2] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, C. Mahmoudi, Fog Computing Conceptual Model, Technical Report, National Institute of Standards and Technology, 2018.

[3] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, G. Fortino, A. Gal, U. Kannengiesser, F. Leotta, et al., The internet of things meets business process management: a manifesto, IEEE Systems, Man, and Cybernetics Magazine 6 (2020) 34–44.

[4] C. Diamantini, A. Mircoli, D. Potena, E. Storti, Process-aware iiot knowledge graph: A semantic model for industrial iot integration and analytics, Future Generation Computer Systems 139 (2023) 224–238. doi:https://doi.org/10.1016/j.future.2022.10.003.

[5] K. Janowicz, A. Haller, S. J. Cox, D. Le Phuoc, M. Lefrançois, Sosa: A lightweight ontology for sensors, observations, samples, and actuators, Journal of Web Semantics 56 (2019) 1–10.

[6] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al., The ssn ontology of the w3c semantic sensor network incubator group, Journal of Web Semantics 17 (2012) 25–32.
[7] D. Bonino, F. Corno, Dogont-ontology modeling for intelligent domotic environments, in: International Semantic Web Conference, Springer, 2008, pp. 790–803.
[8] C. Diamantini, D. Potena, E. Storti, Sempi: A semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators, Future Generation Computer Systems 54 (2016) 352–365.
[9] D. Calvanese, T. E. Kalayci, M. Montali, A. Santoso, Obda for log extraction in process mining, in: Reasoning Web International Summer School, Springer, 2017, pp. 292–345.
[10] C. Diamantini, L. Genga, D. Potena, W. van der Aalst, Building instance graphs for highly variable processes, Expert Systems with Applications 59 (2016) 101–118.
[11] B. F. van Dongen, W. M. P. van der Aalst, Multi-phase process mining: Building instance graphs, in: P. Atzeni, et al. (Eds.), Conceptual Modeling – ER 2004, Springer Berlin Heidelberg, 2004, pp. 362–376.
[12] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of sparql, in: International semantic web conference, Springer, 2006, pp. 30–43.
[13] M. Arazzi, A. Nocera, E. Storti, The semioe ontology: A semantic model solution for an ioe-based industry, IEEE Internet of Things Journal 11 (2024) 40376–40387.