# NICE: The Native IoT-Centric Event Log Model for Process Mining

Yannis Bertrand[1][0000−0002−6407−7221], Silvestro Veneruso[2][0000−0002−2164−5954], Francesco Leotta[2][0000−0001−9216−8502], Massimo Mecella[2][0000−0002−9730−8882], and Estefanía Serral[1][0000−0001−7579−910X]

[1] Research Centre for Information Systems Engineering (LIRIS), KU Leuven
Warmoesberg 26, 1000 Brussels, Belgium
{firstname.lastname}@kuleuven.be
[2] Sapienza Università di Roma
Rome, Italy
{lastname}@diag.uniroma1.it

**Abstract.** More and more so-called IoT-enhanced business processes (BPs) are supported by IoT devices, which collect large amounts of data about the execution of such processes. While these data have the potential to reveal crucial insights into the execution of the BPs, the absence of a suitable event log format integrating IoT data to process data greatly hampers the realisation of this potential. In this paper, we present the Native Iot-Centric Event (NICE) log, a new event log format designed to incorporate IoT data into a process event log ensuring traceability, flexibility and limiting data loss. The new format was linked to a smart spaces data simulator to generate synthetic logs. We evaluate our format against requirements previously established for an IoT-enhanced event log format, showing that it meets all requirements, contrarily to other alternative formats. We then perform an analysis of a synthetic log to show how IoT data can easily be used to explain anomalies in the process.

**Keywords:** Process Mining · Event Logs · IoT · Standard Format.

## 1 Introduction

As the utilisation of Internet of Things (IoT) devices in support of business processes (BPs) becomes more frequent, there is a growing recognition of the potential to leverage the data collected by these devices for process mining (PM). Most current PM methods that can incorporate IoT data follow a similar approach: the IoT data are preprocessed with event abstraction and event-case correlation techniques to be translated into an event log in XES format (see [4, 14, 23, 25]).

Although this is an interesting initial approach to integrate IoT data into PM and it allows for the application of existing control-flow and data-aware techniques, this method does not fully exploit the potential of IoT data. Often, the resulting high-level event log lacks contextual information (i.e., properties that can influence process execution, as explained in [4, 24]) that could be derived from the IoT data, or it has limited capability to incorporate such context information. Furthermore, by separating the

abstraction phase from the analysis phase, the true potential of advanced algorithms to optimise both abstraction and model discovery together cannot be harnessed. For example, the development of an IoT-enhanced decision mining algorithm requires direct access to lower-level IoT data to learn the most relevant features directly from the source data, instead of relying on an error-prone event abstraction step which might leave important information behind, at a lower granularity level [5].

This shortcoming of existing approaches is to a large extent due to limitations of the most common event log standards, i.e., the eXtensible Event Stream [10] (XES) and the Object-Centric Event Log [8] (OCEL), and has been acknowledged in the IoT PM literature [12] and beyond [2]. In a previous work [5], the authors listed ten requirements for the storage of IoT-enhanced event logs and showed that both XES and OCEL failed to meet more than half of these requirements. Building upon this paper, we present a new data model based on the conceptual model described in [4], which meets the requirements mentioned in [5]. We then describe an event log created following this format, and show that this new format greatly facilitates more in-depth analyses of IoT-enhanced BPs.

The paper is organized as follows. Section 2 introduces the previous research results that drove the development of the proposed event log format. Section 3 describes the proposed format. Section 4 validates the format by confronting it to theoretical requirements and applying it to a smart space case study. Section 5 introduces relevant related works and compare them to the proposed format. Section 6 concludes the paper and outlines future works.

## 2 Background

The definition of an event log suitable for IoT builds upon research and standardization efforts carried on independently in the separated fields of BPM (especially PM) and IoT communities. In this section we provide the background which influenced the proposed model.

### 2.1 Existing standards for event logs

XES [10], the current standard event log model, is an XML-based model that mainly consists of the notions of event, case, and log. It proposes standard attribute types to contextualise the events, e.g. the resource executing an activity, the cost of an activity, etc. A standard activity lifecycle is defined together with XES, based on which the status of an activity can be mapped with events relating to this activity. XES also allows the definition of new data attribute types through extensions.

Recently, the gain in maturity of the PM field has increased the urge to create alternative models. Multiple propositions that relax some assumptions of XES and allow for more flexibility in event data storage have been presented, e.g., in [21, 8]. Among them, the OCEL [8] was designed to be more suitable for storing event logs extracted from relational databases and is widely considered as the main challenger of XES today. It introduces the concept of object, which generalises the notion of case by allowing one event to be linked with multiple objects instead of a single case.

This removes the necessity to "flatten" the event log by picking one case notion from the several potential case notions that often coexist in real-life processes.

### 2.2 Process mining using IoT data

Challenges related to the application of PM to IoT data, and in particular to how event logs are represented are reported in [5, 18]. The vast majority of the PM literature involving IoT data has focused on mining high-level events of the process from low-level IoT data to create XES event logs. Various frameworks to extract high-level logs from IoT data have been presented, e.g., [29, 26, 23, 27, 16, 15, 14, 6, 19]. Traditional PM techniques can then be applied to these event logs to, e.g., discover control-flow models of the processes. A recent survey of process discovery to smart spaces, which represents a large chunk of the literature of PM applied to IoT, is provided in [3].

Although most of the existing literature is in IoT event abstraction, some other possible techniques have also been investigated. Banham et al. [1] proposes to perform data-aware process discovery with IoT-based attributes. The framework proposed requires abstracting the IoT data to integrate them in an XES event log. A second work is proposed by Rodriguez-Fernandez et al. [22], who present an approach for IoT-enhanced deviation detection in the time series data directly (in a so-called *time-series log*). Remark that all these papers bumped into the limitations of traditional event logs and had to abstract the data first or to use the raw sensor data.

Another important research direction is represented by the automated segmentation of logs. Here, statistics-based techniques have been proposed [17], as well as technique based on the structure of possibly mined processes [7].

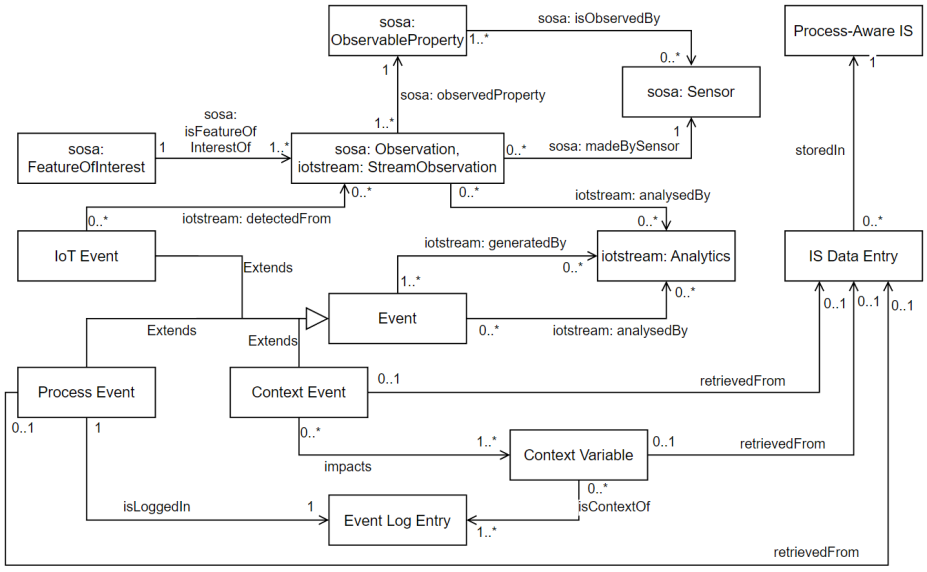



Fig. 1: Core of the bridging model for IoT and PM [4].

### 2.3 Existing models bridging IoT and process mining

A first model for bridging IoT and PM was presented in [4] (see Figure 1). This model integrates IoT ontologies (like SOSA [13]) with typical PM concepts around the central notion of event, which is decomposed in three types of events: 1) IoT events, which correspond to events as understood in IoT systems; 2) Process events, which are events as understood in PM; 3) Context evens, which correspond to changes in a parameter of the context of the process. For more information, see Section 3.1 or [4].

## 3 Format specification

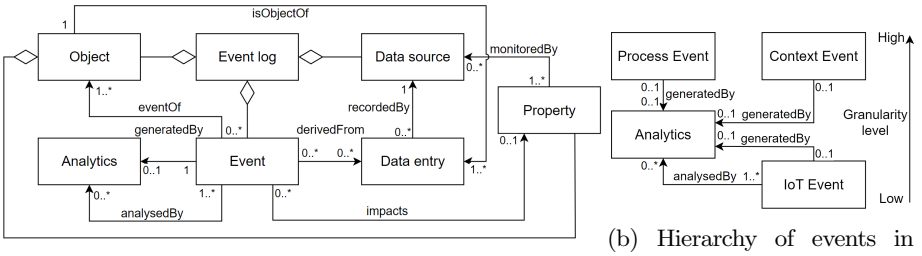
In this section, we outline the format we propose for IoT-enhanced event logs and explain how we operationalised it to generate XML-based NICE logs. It builds further upon the conceptual model presented by the authors in [4], which aimed at integrating IoT ontologies and event log models. To do so, multiple events types were defined to bridge the gap between the (physical) IoT world and the (mostly digital) PM world. In this research, in line with design science research [11], we adopted an iterative approach, moving from between design to evaluation through prototyping in order to continuously improve the quality of the event log format. The evaluation step verified 1) that all constructs and relationships of the metamodel were transcribed correctly in the XML and 2) that the format was capable of representing event logs from various types of IoT-enhanced BPs.

### 3.1 Metamodel

At its core, our data format consists of lists of Data Sources, Objects and Events (see Figure 2a). An Event is related to one or several Objects, which can be digital (Digital Object, e.g., an order) or physical (Feature Of Interest, e.g., a fridge), or both (e.g., a parcel). All Objects can have a collection of Properties, which can also be digital or physical (e.g., the total amount of the order, the temperature in the fridge, the weight and the value of the parcel), and represent context parameters of the process. Events are derived from one or several Data Entries or lower-level Events which are logged by a given Data Source, which can either be an information system or a Sensor. We distinguish between three types of Events, which follow the hierarchy shown in Figure 2b:

- IoT Event: an instantaneous change in a real-world phenomenon that is monitored by a Sensor, or derived from lower-level IoT Events. E.g., the temperature in a fridge decreasing;
- Process Event: an instantaneous change of state in the transactional lifecycle of an activity (corresponding to the usual notion of event in PM). This type of event is deduced from one or more IoT Events or taken from an IS Data Entry. E.g., a product is taken from a fridge and loaded in a truck;
- Context Event: an instantaneous change in a real-world phenomenon or a digital property that has an impact on the execution of a specific process instance (i.e., it impacts a Property of an Object) but does not change its control-flow state. Such an Event typically influences the path followed in a process instance, how an

activity is executed, etc. This type of event is deduced from one or more IoT Events or taken from an IS Data Entry. E.g., the pressure in a tank exceeds a maximum threshold, prompting a human intervention in a chemical production process.




(a) Core of our data model.

(b) Hierarchy of events in NICE logs (taken from [4]).

Fig. 2: Metamodel for our log format.

Higher-level Events can be derived from lower-level Events via the notion of Analytics, which represents any technique used for event abstraction (such as e.g., rule-based reasoning techniques like CEP, data-driven models). IoT Events can cascade until a deduced event has a direct relationship with the process, i.e., it is a Process Event or a Context Event.

### 3.2 Implementation

In this section, we describe how we translated the model in Figure 2a into an XML format for NICE logs. XML was chosen for its flexibility and its popularity, though the model could also be transcribed in other languages such as JSON or YAML. The XML schema of the processed log is structured as follows: the `EventLog` tag is the root and encloses the whole log. Inside, it contains three main elements, identified respectively by the tags:

1. `ObjectList`. This element contains a list of Objects. In IoT-enhanced BPs, different Objects interact and an Event can involve a combination of Objects (e.g., users, locations). The Properties of Objects are represented as attributes, which can be updated by Context Events.
2. `DataSourceList`. This element contains a list of Data Sources available within the observed environment. We distinguish between two types of data sources: *(i)* physical Data Sources, typically sensors, that monitor physical Properties, e.g, the opening of a door or the temperature; and *(ii)* digital Data Sources, like process-aware information systems (PAISs), recording digital Properties and Process Events.
3. `EventList`. This element contains a list of Events. There are three types of events: `IoTEvent`, `ContextEvent` and `ProcessEvent`. `IoTEvent` is used to represent low-level types of Events, such as raw sensor measurements, and, if relevant, mixed-level types of Events, such as the aggregation of lower-level `IoTEvents` into *actions*, i.e., atomic interactions with the environment or a part of it (e.g., sitting on a chair, opening the fridge). A `ContextEvent` represents a change

```
...
<IoTEvent id="66" timestamp="2020-01-01T00:31:58" objectID="objID_1,objID_2">
    <Observation id="obs_66" resultTime="2020-01-01T00:31:58"
    ↪ value="Go_bathroom_sink" featureOfInterest="objID_24"/>
    <Analytics itGenerates="eventID_66" itAnalysesEvents="eventID_62,eventID_65">
        <Methods> <Method name="CEP"/> </Methods>
    </Analytics>
</IoTEvent>
...
<IoTEvent id="86" timestamp="2020-01-01T05:36:18" caseID="objID_1,objID_2">
    <Observation id="obs_86"
    ↪ resultTime="2020-01-01T05:36:18" value="Go_bed" featureOfInterest="objID_26"/>
    <Analytics itGenerates="eventID_86"
    ↪ itAnalysesEvents="eventID_68,eventID_70,eventID_75,eventID_78,eventID_80,eventID_85">
        <Methods> <Method name="CEP"/> </Methods>
    </Analytics>
</IoTEvent>
...
<IoTEvent id="88" timestamp="2020-01-01T05:36:22" objectID="objID_1,objID_2,objID_23">
    <Observation id="obs_88" resultTime="2020-01-01T05:36:22"
    ↪ value="OFF" sensor="s3" featureOfInterest="objID_23"/>
</IoTEvent>
<IoTEvent id="89" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2,objID_23">
    <Observation id="obs_89" resultTime="2020-01-01T05:40:52"
    ↪ value="ON" sensor="s3" featureOfInterest="objID_23"/>
</IoTEvent>
<IoTEvent id="90" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2">
    <Observation id="obs_90"
    ↪ resultTime="2020-01-01T05:40:52" value="Go_in_bed" featureOfInterest="objID_23"/>
    <Analytics itGenerates="90" itAnalysesEvents="88,89">
        <Methods><Method name="CEP"/></Methods>
    </Analytics>
</IoTEvent>
<ProcessEvent id="91" timestamp="2020-01-01T05:40:52" objectID="objID_1,objID_2">
    <Activity value="Sleeping"/>
    <LifecyclePhase value="complete"/>
    <Analytics itGenerates="91" itAnalysesEvents="66,86,90">
        <Methods><Method name="CEP"/></Methods>
    </Analytics>
</ProcessEvent>
...
```

Fig. 3: Example of Events connected by the `Analytics` element. In particular, in this portion of the log, there are five `IoTEvent`s and one `ProcessEvent`. The mixed-level `IoTEvent` with ID 90, that represents the atomic action *Go_in_bed*, is generated by the analysis of the low-level `IoTEvent`s with IDs 88 and 89, with the `Method` called CEP. The high-level `ProcessEvent` with ID 90, that completes the Event *Sleeping*, is generated by the analysis of the mixed-level `IoTEvent`s with IDs 66, 86, and 90, with the `Method` called CEP. The derivation of Events with IDs 66 and 86 follows the same principle, but the lower-level `IoTEvents` from which they are derived are not shown for brevity.

in the value of a Property of an Object which impacts the execution of the process (e.g., influence a decision). It is related to an Object and the Property it updates, and contains this Property's new value. Finally, `ProcessEvent` is used to represent high-level types of Events, e.g., groups of human atomic interactions with the environment that are performed with a common goal. It represents an execution of a business activity (e.g., prepare dinner, register an order).

Each Event is associated with an identifier, a timestamp, and one or more objectIDs. An `IoTEvent` has an additional child element called `Observation` that contains the raw sensor value or the action, if available. While a `ProcessEvent` has two child elements that refer to the name of the related `Activity` and its `LifeCyclePhase` (e.g., start, complete).
These three levels of Events are connected through the `Analytics` element that references the lower-level Events that generate a higher level of Event, and the `Method` used. For instance, the mixed-level `IotEvent` called "open the fridge" in its `Analytics` has references to the lower-level Events related to the opening of the fridge door and the triggered motion sensors near the fridge. The higher-level `ProcessEvent` related to the activity "cook", in its `Analytics`, has the references of the mixed-level Events that all together compose the activity "cook", including the mixed-level `IotEvent` "open the fridge". Figure 3 shows examples of `Analytics` elements used to derive mixed-level IoT Events and Process Events.

For our experiments, the simulator presented in [28], which was originally designed to produce logs in the XES format, has been adapted to generate NICE logs. Using a Python script, the synthetic log produced was then processed to fit the meta-model proposed in this article, following the XML structure described above. The Python script and the resulting log are available in this repository: `https://github.com/silvestroveneruso/NiceParsingTool.git`.

## 4 Format validation

In this section, we evaluate our new format from two angles: 1) theoretically, we compare it with the requirements for IoT-enhanced log formats outlined in [5]; 2) in practice, by showing how it can be used to easily represent links between different types of events and gain a better understanding of a process.

### 4.1 Theoretical requirements fulfilment

In this section, we confront our model with the requirements outlined in [5]:

- R1 (Store high-level events): Obtained with process and context events
- R2 (Store low-level events): Obtained with IoT events
- R3 (Store intermediary events): Obtained with IoT events derived from other IoT events
- R4 (Enable traceability between high-level and low-level events): Achieved with the concept of Analytics, which links derived events with the events they are derived from
- R5 (Represent context at event level): Done with the properties of the objects linked to an event
- R6 (Represent context at activity level): Done with properties of the objects linked to the events of an activity; an additional activity object could link events together for convenience (link the events to the activity object, and the context to the activity object)

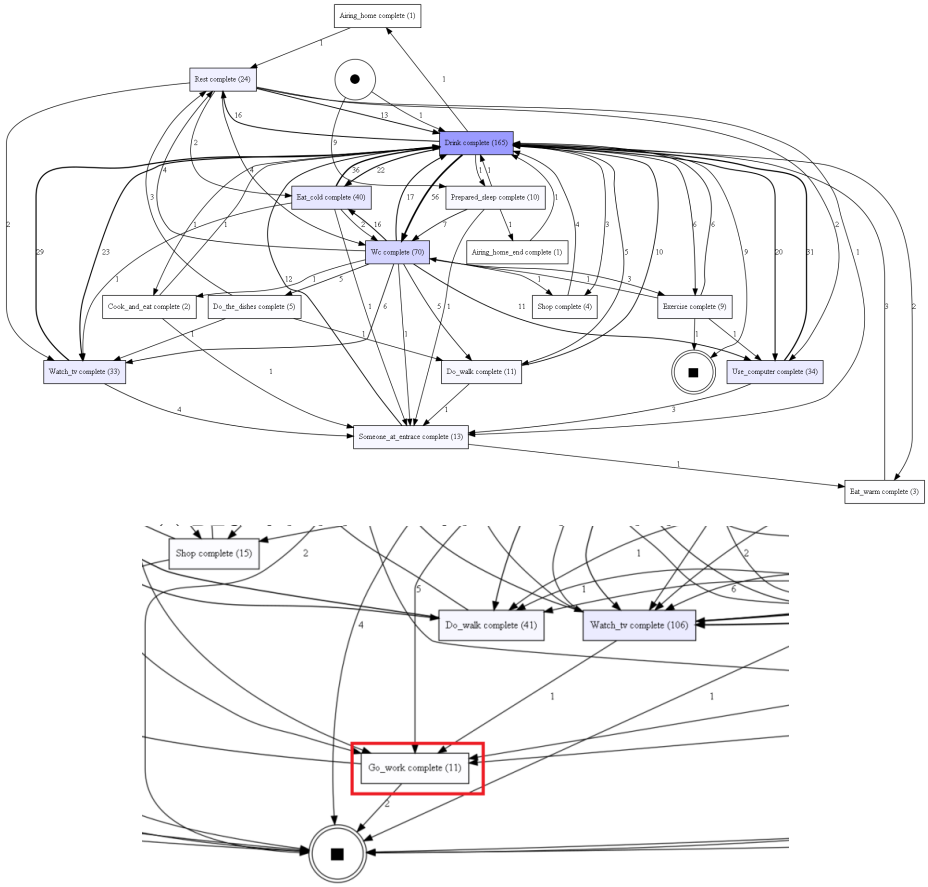
- R7 (Represent context at case/object level): Context is represented at object level with the object properties, case is a special object type
- R8 (Represent context at process level): This requirement can be completed with a process-wide object (e.g., the house in a smart home log)
- R9 (Update context parameters independently from process events): Context parameters (Properties) are updated by context events and not by process events
- R10 (Update context parameters at a higher frequency (than process events)): This requirement is achieved together with R9, as context events can happen at any rate, distinguished from process events

### 4.2 Log analysis

**Log description** To showcase the usability of the data model, we generated a log simulating the behaviour of two users living in a smart home for four weeks, executing activities such as *Cook_and_eat, Do_the_dishes, Drink, Eat_cold, Eat_warm, Exercise, Go_work, Sleep, Rest, Shop, Use_Computer, Watch_tv* and *Wc*, recorded as process events, and interacting with 30 motion sensors and a temperature sensor (each change in sensor value being recorded as an IoT event). The simulator was programmed so that during one week, the users show a different behaviour, i.e., they stay at home the whole day instead of going to work in the morning and coming back home in the evening during weekdays. This change in behaviour is due to extremely high outside temperatures during that week, which are tracked by temperature sensors and logged as IoT events.

**Analysis** Based on these IoT events, a context event is derived when the temperature exceeds the maximum acceptable temperature for work (set to 32.5 degrees for office workers in Belgium) and sets a property 'Temperature suitable for work' to False. When the temperature decreases below the threshold again, another context event is generated, which sets 'Temperature suitable for work' back to True. Figure 4 contrasts directly-follows graphs (DFGs) while the 'Temperature suitable for work' property was False (Figure 4a) and over the whole log (Figure 4b). From the figure we can see that the behaviour of the users differs in both circumstances, as the activity Go_work is not performed during the heatwave period and more activities are recorded at home.

These context events make the change in the behaviour of the users easy to explain, as the days of anomalous behaviour are flagged with context events and characterised by a value of 'Temperature suitable for work' equal to True. This way, all relevant information is stored in the log and is easily traceable, as lower-level IoT events are still present. Moreover, this approach makes it possible to set or mine different thresholds for 'Temperature suitable for work', e.g., for users working in different sectors, and to create one context parameter for each user, with different rules, which would be much more difficult in traditional event log formats. Once clearly identified with the context events, deviating behaviour can be removed from the log or treated separately simply by removing the events recorded between the two context events. Finally, storing the low-level data in the log enables the use of, e.g., decision mining or trace clustering techniques to mine the finer-grained rules behind the break from work from the low-level data directly, which is not possible in a XES or OCEL log.




(b) Zoom in on the 'Go_work' activity in the DFG of the behaviour of the users over the whole log.

Fig. 4: DFGs of the behaviour of the users in the simulated log.

## 5 Related works

In the last years, several data formats have been adapted to IoT-enhanced event logs [9, 20, 28, 22]. In this section, we present these new alternatives and compare them to our data format with respect to the requirements identified in [5].

Of these models, the least tailored one to IoT-enhanced logs is D-OCEL [9]. It is an extension of the OCEL [8] which introduces dynamic attributes, thereby solving one of the main issues faced by OCEL for IoT-enhanced event logs.

In [20], authors present the XES DataStream extension, which aims at facilitating the storage of IoT data in XES logs. DataStream introduces, among others, the notions of *point* and *datastream*, which respectively represent individual sensor observations and group together points that belong to a same event or trace. This enables the storage of all IoT data in the log, together with the impacted events or traces.

Table 1: Comparison of different solutions with respect to the Requirements.

| | | XES [10] | OCEL [8] | D - OCEL [9] | DS [20] | Simu-lator [28] | TS log [22] | NICE log |
|---|---|---|---|---|---|---|---|---|
| Requirements | R1: Store high-level events | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ |
| | R2: Store low-level events | | | | ✓ | ✓ | ✓ | ✓ |
| | R3: Store intermediary/mixed-level events | | | | | ✓ | | ✓ |
| | R4: Enable traceability between high-level and low-level events | | | | ✓ | | | ✓ |
| | R5: Represent context at event level | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ |
| | R6: Represent context at activity level | | | ✓ | ✓ | | | ✓ |
| | R7: Represent context at case/object level | ✓ | ✓ | ✓ | ✓ | | | ✓ |
| | R8: Represent context at process level | ✓ | | ✓ | | ✓ | | ✓ |
| | R9: Update context parameters independently from process events | | | | | ✓ | ✓ | ✓ |
| | R10: Update context parameters at a higher frequency | | | | ✓ | ✓ | ✓ | ✓ |

Another format based on XES is described in [28] and represents IoT-enhanced logs generated by the smart home simulator used in this paper. There, multiple files are generated: a segmented high-level event log, an unsegmented high-level event log and a sensor log.

Finally, the TS-log, a model which is completely independent from existing standards like XES and OCEL, was presented in [22]. This model is designed exclusively for TS data which are collected around a process and can be used to derived the execution of activities (i.e., process events). A TS-log therefore contains a set of variable names, a domain function for each variable, an index and a function recording the value of each variable at each timestamp.

Comparing our model to related works, the NICE log format is the most flexible and balanced and the only one fulfilling all requirements expressed in [5] (see Table 1). OCEL does neither allow data attribute updates nor traceability and only has one event type. XES and D-OCEL have the same limitations, except that they allow for data attribute updates, but only with (process) events. XES with DataStream does much better, but still requires all sensor data points to be linked with one process event, and the support for mixed-level events is uncertain. The simulation tool does not support all context representation and enables only limited traceability (and information is scattered in multiple files). Finally, TS logs focus fully on low-level data, and cannot store traditional process data.

Compared to the conceptual model presented by the authors in [4], the format of the NICE logs differs in several aspects. Primarily, NICE logs rely on the concept of object instead of the concept of case. This is because objects offer more flexibility and can be used to precisely define the scope of context parameters as object properties. Then, while the original conceptual model gave a more detailed description of IoT concepts than of process concepts, NICE logs harmonise both perspectives with

overarching concepts. Finally, attention has been given to reducing the redundancy between some concepts and attributes.

## 6 Conclusion

In this paper, we presented a new format for IoT-enhanced event logs. This format combines the different event types presented in [4] with the notion of object from OCEL to form a very flexible format, capable of integrating IoT data with process data with a minimal information loss. The format was evaluated in Section 4, where we gave evidence of its theoretical robustness and its practical usefulness. In Section 5, we also compared our new format with existing alternatives and showed that it is the only one fulfilling the requirements in [5]. Moreover, a first practical application demonstrated how generating a log following our format enables process analyses integrating IoT data.

Some limitations of our work include the still experimental implementation so far and the potential amplifying effect of multiple event abstraction steps on sensor data quality issues. To cope for these limitations, in future works, we first plan to further develop the format by creating more tool support to generate and manipulate logs. Next to this, we would like to design new analysis techniques taking advantage of the capabilities of the new format, e.g., for IoT-enhanced trace clustering, IoT-enhanced predictive process monitoring. As such, looking into the compatibility of NICE logs with the OCEL 2.0 format could help leveraging existing techniques and tools.

## References

1. Banham, A., Leemans, S.J., Wynn, M.T., Andrews, R., Laupland, K.B., Shinners, L.: xpm: Enhancing exogenous data visibility. AI in medicine **133**, 102409 (2022)
2. Beerepoot, I., Di Ciccio, C., Reijers, H.A., Rinderle-Ma, S., Bandara, W., Burattin, A., Calvanese, D., Chen, T., Cohen, I., Depaire, B., et al.: The biggest business process management problems to solve before we die. Computers in Industry **146**, 103837 (2023)
3. Bertrand, Y., Van den Abbeele, B., Veneruso, S., Leotta, F., Mecella, M., Serral, E.: A survey on the application of process discovery techniques to smart spaces data. EAAI **126**, 106748 (2023)
4. Bertrand, Y., De Weerdt, J., Serral, E.: A bridging model for process mining and iot. In: International Conference on Process Mining. pp. 98–110 (2021)
5. Bertrand, Y., De Weerdt, J., Serral, E.: Assessing the suitability of traditional event log standards for iot-enhanced event logs. In: International Conference on Business Process Management. pp. 63–75 (2022)
6. Dimaggio, M., Leotta, F., Mecella, M., Sora, D.: Process-based habit mining: Experiments and techniques. In: UIC 2016. pp. 145–152. IEEE (2016)
7. Esposito, L., Leotta, F., Mecella, M., Veneruso, S.: Unsupervised segmentation of smart home logs for human habit discovery. In: IE'22. pp. 1–8. IEEE (2022)
8. Ghahfarokhi, A.F., Park, G., Berti, A., van der Aalst, W.M.P.: OCEL: A Standard for Object-Centric Event Logs, CCIS, vol. 1450, p. 169–175 (2021)
9. Goossens, A., De Smedt, J., Vanthienen, J., van der Aalst, W.M.: Enhancing data-awareness of object-centric event logs. In: ICPM 2022. pp. 18–30. Springer (2022)
10. Günther, C.W., Verbeek, E.: Xes standard definition (Mar 2014)

11. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly pp. 75–105 (2004)
12. Janiesch, C., Koschmider, A., Mecella, M., Weber, B., Burattin, A., Di Ciccio, C., Fortino, G., Gal, A., Kannengiesser, U., Leotta, F., et al.: The internet of things meets business process management: a manifesto. IEEE Systems, Man, and Cybernetics Magazine **6**(4), 34–44 (2020)
13. 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**, 1–10 (May 2019)
14. Janssen, D., Mannhardt, F., Koschmider, A., van Zelst, S.J.: Process model discovery from sensor event data. In: ICPM 2020. pp. 69–81 (2020)
15. Koschmider, A., Janssen, D., Mannhardt, F.: Framework for process discovery from sensor data. In: EMISA. p. 8 (2020)
16. Koschmider, A., Mannhardt, F., Heuser, T.: On the Contextualization of Event-Activity Mappings, LNBIP, vol. 342, p. 445–457. Springer (2019)
17. de Leoni, M., Pellattiero, L.: The benefits of sensor-measurement aggregation in discovering iot process models: a smart-house case study. In: BPM 2021. pp. 403–415 (2021)
18. Leotta, F., Mecella, M., Mendling, J.: Applying process mining to smart spaces: Perspectives and research challenges. In: CAiSE 2015 Workshops. pp. 298–304. Springer (2015)
19. Leotta, F., Mecella, M., Sora, D.: Visual process maps: A visualization tool for discovering habits in smart homes. JAIHC **11**(5), 1997–2025 (2020)
20. 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), 109 (2023)
21. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. arXiv:1303.2554 [cs] (Mar 2013)
22. Rodriguez-Fernandez, V., Trzcionkowska, A., Gonzalez-Pardo, A., Brzychczy, E., Nalepa, G.J., Camacho, D.: Conformance checking for time-series-aware processes. IEEE TII **17**(2), 871–881 (2021)
23. Seiger, R., Zerbato, F., Burattin, A., Garcia-Banuelos, L., Weber, B.: Towards iot-driven process event log generation for conformance checking in smart factories. In: EDOCW. p. 20–26 (Oct 2020)
24. Serral, E., De Smedt, J., Vanthienen, J.: Making business environments smarter: a context-adaptive petri net approach. In: UIC 2014. pp. 343–348 (2014)
25. Soffer, P., et al.: From event streams to process models and back: Challenges and opportunities. Information Systems **81**, 181–200 (Mar 2019)
26. Trzcionkowska, A., Brzychczy, E.: Practical aspects of event logs creation for industrial process modelling. MAPE **1**(1), 77–83 (Sep 2018)
27. Valencia-Parra, A., Ramos-Gutierrez, B., Varela-Vaca, A.J., Gomez-Lopez, M.T., Bernal, A.G.: Enabling process mining in aircraft manufactures: Extracting event logs and discovering processes from complex data pp. 166–177 (2019)
28. Veneruso, S., Bertrand, Y., Leotta, F., Serral, E., Mecella, M.: A model-based simulator for smart homes: Enabling reproducibility and standardization. JAISE **15**(2), 143–166 (2023)
29. van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granular Computing **6**, 719–736 (2021)