

Toward IoT-based Process Analytics: Extending Event Knowledge Graphs with Ambiguity

Marco Franceschetti^{1✉}, Dominik Manuel Buchegger¹, Ronny Seiger¹, and Barbara Weber¹

Institute of Computer Science, University of St.Gallen, St.Gallen, Switzerland
{marco.franceschetti@unisg.ch}

Abstract. Digital traces of processes executed in the physical world without Information Systems (IS) support are composed of high-level events derived from low-level physical world events (e.g., from IoT). Due to this, frequently these high-level events are ambiguous, i.e., they yield multiple interpretations. As current IS lack ambiguity-awareness, ambiguity in digital traces can compromise process analytics. Motivated by this and current trends toward multi-dimensional analytics, we introduce an ambiguity-aware object-centric representation of digital traces by extending Event Knowledge Graphs with ambiguity. We integrate its construction into a framework to enable analytics for ambiguous IoT-based digital traces. A prototype implementation shows the applicability of the construction from ambiguous process event streams in online settings.

Keywords: event knowledge graphs · ambiguity · iot-based analytics

1 Introduction

Two significant emerging research directions arise in the current Business Process Management (BPM) landscape: multi-dimensional process analytics [2] and the integration of the Internet of Things with BPM for process monitoring [14,6].

On the one hand, multi-dimensional analytics represent a shift toward a more comprehensive understanding of business processes by incorporating diverse perspectives such as actor behavior, resource utilization, and entity interactions, beyond the traditional case-centric focus. This shift has led to the development of novel analysis paradigms, such as object-centric process mining [1], and advanced representation formalisms, such as Event Knowledge Graphs (EKGs) [9], able to model and analyze rich interdependencies in business processes [15,16].

On the other hand, the integration of IoT technologies into process monitoring has significantly expanded the scope of BPM. Specifically, the BPM-IoT integration enables the real-time generation of digital traces of processes executed in the physical world without Information System (IS) support [33]. This requires the observation of the physical world via sensors that generate low-level IoT events—usually without process semantics—and to abstract these IoT events into high-level events with process semantics [27]. However, this is challenged

by the fact that IoT event streams are frequently characterized by noise, incomplete or missing sensor data, and limited process context-awareness [19]. This leads to uncertainty about how to interpret IoT events, i.e., to *ambiguity* [10]. Ambiguity-awareness in the event abstraction step, i.e., the ability to explicitly consider the multiple event interpretations when generating process events, is critical to avoid misrepresenting the actual process executions. Therefore, ambiguity originating from the IoT events must be explicitly propagated to the process events to avoid compromising the subsequent process analytics.

Here, we set our focus on two specific challenges posed by ambiguity. A first challenge (**C1**) arises from the ambiguity in IoT-based events during the sensing and abstraction [19]. IoT sensors often lack the process awareness or contextual information needed to clearly correlate observed events with specific process instances or entities in the situational context. This results in multiple possible interpretations of the same event and its correlations with the contextual entities [10], which can compromise process analytics outcomes [2]. A second challenge (**C2**) stems from the ambiguity in entity roles, which is also rooted in the IoT event abstraction. IoT data often provides limited information about the roles entities play in events (e.g., whether an agent is the performer or target of an activity; cf. the concept of endurants in [3]), leading to multiple possible role interpretations that also compromise process analytics outcomes [24].

To overcome these challenges with the object-centric paradigm in mind, we address the research question: *How to explicitly represent ambiguities in process events derived from the IoT-driven monitoring of processes that lead to C1–C2?*

As our overarching goal is to enable IoT-based multi-dimensional process analytics, we first set requirements and outline a software framework to support such analytics. We integrate within this framework the main contribution of this paper: an extension of EKGs *with ambiguity* (EKGA) for the object-centric representation of digital traces with ambiguity, and an EKGA construction method. EKGA allow modeling the uncertainty in interpreting ambiguous events and the different roles with which entities participate in events, overcoming challenges **C1–C2**. We show the applicability of EKGA construction in online settings by testing a prototype implementation on a stream of healthcare process events.

Our main contribution facilitates the creation of an object-centric representation of process executions from IoT data, explicitly modeling ambiguity that leads to uncertainty about event-entity correlations and the roles entities play in events. The EKGA bridges the current gap between object-centric digital trace representation and explicit modeling of event ambiguity. Integrating our contribution with an information system paves the way to the development of novel ambiguity-aware systems capable of performing multi-dimensional analytics of digital traces with ambiguous events. It further stimulates the development of new methods of online event disambiguation to support online process monitoring and conformance checking driven by IoT technologies.

Paper structure. Sect. 2 motivates our work setting requirements. Sect. 3 presents a framework integrating EKGA construction. Sect. 4 presents a prototype implementation. Sect. 5 discusses related work. Sect. 6 concludes the paper.



Fig. 1: Lab setup to simulate the IoT-monitored blood donation process, with two donor stations (red boxes) and one shared hand hygiene station (green box)

2 Motivation and Requirements

2.1 Motivating Scenario

As a motivating scenario, we take the IoT-driven online conformance checking of a blood donation process, for which we can simulate the execution of two concurrent, co-located instances in our lab setup, shown in Fig. 1, as detailed in [11]. As a normative blueprint to check the conformance, we take the process specification extracted in [11] from the WHO guidelines in [20,26] and validated by healthcare experts. We translated the validated specification into constraints and derived the simplified process model shown in Fig. 2 as a DCR graph [13]. We assume one process instance to be associated with one *donor* (a patient donating blood), and that multiple co-located instances can run simultaneously.

The first step for the healthcare worker (HCW) is to perform a hand hygiene, then HCW proceeds with the preliminary operations (identify the donor, check the drawing machine, disinfect the injection site). Then, HCW again performs a hand hygiene, after which HCW performs the venipuncture and initiates the blood drawing. During the blood drawing, HCW closely monitors the donor. Once enough blood is drawn, HCW ends the blood drawing, removes and disposes the needle, and performs a hand hygiene. Lastly, HCW performs final operations (dismiss the donor, store the blood sample), and a hand hygiene. Any other activity performed by HCW during the process (e.g., signing a document, touching another donor) is considered as a hand contamination (cf. “Contaminate hands” in Fig. 2) and requires a hand hygiene before continuing.

Like many healthcare processes, the blood donation process is not driven by an IS and is mostly manual. To monitor its execution and check its conformance we rely on IoT technologies for sensing, aggregating, and abstracting multiple low-level IoT events into high-level process events, as proposed in [6,27], with the ultimate goal of providing online feedback about process conformance to the users. In our lab setup, we leverage sensors such as proximity, motion, and touch sensors; we deliberately do not use cameras due to privacy concerns in healthcare settings. Additionally, healthcare settings are inherently noisy environments characterized by the occurrence of unpredictable events such as medical emergencies and concurrent treatments [36]. Digital traces of events generated via

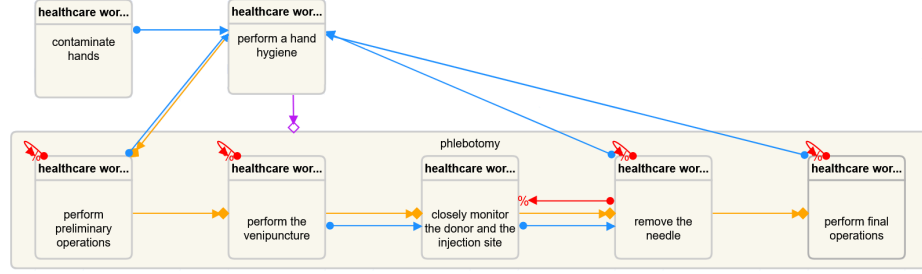


Fig. 2: Blood donation process as a DCR graph [13]. Orange arrows: condition constraints; blue arrows: response constraints; red arrows: exclusion constraints; purple arrows: milestone constraints

the IoT are likely affected by ambiguity [10], which results in uncertain event-entity correlations and uncertain entity participation roles in events. We describe examples in relation to the challenges outlined in the introduction.

Example for challenge C1. Assume that one HCW is attending simultaneously two donors $D1$ and $D2$ in proximity of each other (cf. the two donor stations in Fig. 1). When we observe a “Perform hand hygiene” event by HCW at a hand hygiene station shared by $D1$ and $D2$, ambiguity arises regarding which donor the hand hygiene is for, i.e., which entity correlates with the event. This means that we are uncertain whether to interpret the hand hygiene as part of the blood donation by $D1$ or $D2$. This event could be retrospectively disambiguated by observing a subsequent event revealing more information. For example, if the next event is “Perform preliminary operations” on $D1$, we can confidently establish the event-entity correlation of the hand hygiene with $D1$. Still, without an explicit representation of this ambiguity, we would either treat both interpretations as certain events, or keep only one of the two, potentially discarding the correct one—in either case risking to get incorrect analysis results.

Example for challenge C2. In healthcare processes like the blood donation process, HCW’s hand hygiene is of utmost importance. Specifically, a HCW’s hands must be always sanitized before treating a patient (donor); this must be done also when shifting between patients to avoid cross-contamination. So, after “Perform preliminary operations” on $D1$, HCW must perform a hand hygiene if the next step is a treatment on $D2$. As such, “Perform preliminary operations” on $D1$ is not an event in the treatment process of $D2$. However, it still has an influence on it because its occurrence adds a new precondition for the next step in $D2$ ’s process—executing “Perform hand hygiene”. Here, we must be able to distinguish between two participation roles in treatment events: either a donor is the *target* of a treatment (the treatment is done on her as part of her process) or is *affected by* another donor’s treatment (the treatment is not done on her, but it still affects her process). For the former, the event has the semantics of a normal process step; for the latter, the event has the semantics of a hand contamination. Depending on the role, the respective process instances should have different continuations. Knowing about the specific role allows us to establish how the respective instance should continue and to conclude about process conformance.

In the blood donation process, adherence of the HCW to the hand hygiene regulations is crucial to prevent infections spreading from contaminated surfaces to the donors or from one donor to another donor. The conformance of the process executions has to be checked in near real-time to inform the HCW about potential contamination and missed hand hygiene events so that they can take counter-measures through additional disinfection steps. A post-mortem analysis would only be able to reveal non-conformance but would be too late to take effective counter-measures. The conformance checking has to be aware of potential ambiguities emerging from the IoT sensor processing to identify the unclear roles of the donors in the treatment activities mentioned above, which need to be resolved in a next step to inform the HCW.

2.2 Requirements

Our overarching goal is to define a software framework enabling multi-dimensional analytics and ambiguity resolution of digital traces of processes executed without IS support and monitored via IoT. Here, we focus on a specific component of this framework: the explicit modeling of ambiguity that results in uncertain event-entity correlations and uncertain entity participation roles in events. Aligning with [12], we assume uncertainty to be known in stochastic terms.

A framework achieving this goal should meet the following functional requirements, drawn from our experience with event sensing and abstraction, ambiguity, and the IoT contexts that the system should operate in. **F1**: it should support the online sensing of low-level event data from the process environment. Our goal is to enable analytics of processes monitored via IoT; specifically in our lab setup we aim to check and provide feedback about process conformance in near real-time. Thus, we need to consider data from IoT sensors as streams of low-level events. **F2**: it should be able to abstract low-level event data into the correct high-level process events recording activity name, timestamp, and information about all entities possibly involved in each process event to facilitate the application of (online) process mining techniques in later steps. **F3**: it should emit all events when multiple interpretations of the low-level events as process events, and multiple interpretations of the entity-event correlations, are possible. **F4**: it should allow for the online translation of process events from an event stream in a data structure suitable to support multi-dimensional process analytics. This structure should include all possible interpretations of ambiguous events and correlations, allowing the explicit representation of ambiguities. We also define **N1**: performance as a non-functional requirement, as the framework should be able to handle input with latencies acceptable for online settings.

3 Framework to Enable Multi-dimensional Analytics

In this section, we outline a four-step framework (depicted in Fig. 3) that addresses the requirements listed in Sect. 2 and contextualizes our main contribution, i.e., an extension of EKG with ambiguity and a method for its construction.

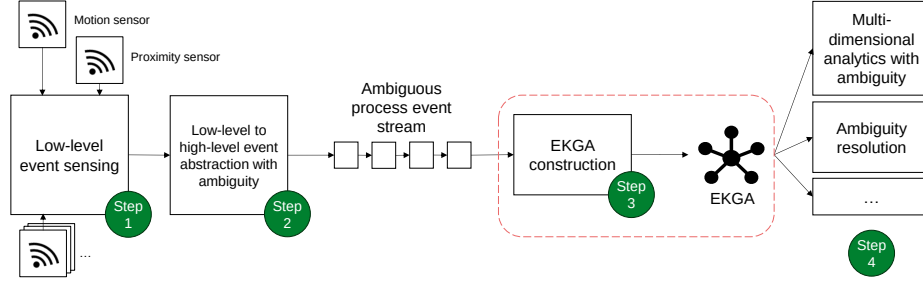


Fig. 3: Framework to enable multi-dimensional analytics of processes executed without IS support in the presence of ambiguity based on EKGA; the focus of this paper is highlighted with the red box

3.1 Step 1: Low-level Event Sensing

Analyzing digital traces of physical world processes executed without IS support requires these processes to be observable. This can be achieved using IoT technologies, specifically sensors able to emit *low-level* event streams describing the events happening in the physical world [14]. Here, a challenge is selecting the right sensors to describe the physical world events that are relevant in the process, i.e., that correspond to process activity executions. Depending on the specific process, a domain expert selects and positions different sensors in the process environment to sense these physical world events (cf. requirement F1), which can then be related to process activity executions (how this is done is out of this paper scope). Still, even with the deployment of all permissible sensors within the process environment, there remains the possibility that these sensors cannot unambiguously capture and describe the targeted physical world events. Beyond the limitations posed by constrained brownfield integration scenarios, this could be attributed to the sensors' inability to provide adequate contextual information during sensing, or the insufficiency of the generated low-level events in distinguishing between physical world phenomena. Factors such as limited sensor precision, which might also be an intentional choice to safeguard user privacy [19], further contribute to this challenge. This results in multiple physical world events being represented by the same low-level events, which are undistinguishable. We assume that, when positioning the sensors, the expert is aware of which physical world events correspond to which low-level events and of any potential ambiguities. In contrast to related approaches, we leverage this awareness to explicitly model ambiguity in the following event abstraction step.

Example. In our lab setup (cf. Fig. 1), to be able to detect hand hygiene activity occurrences, we chose a proximity sensor and a scale, and positioned them in the hand hygiene station. The proximity sensor continuously emits events describing how close *something* is from the sensor at each point in time. The scale is positioned under the hand sanitizer dispenser, and it continuously emits events describing the weight it measures at each point in time. The reason is that, to perform a hand hygiene, a HCW must be in proximity of the hand hygiene station and press the sanitizer dispenser (causing an increase in the weight)

to release the sanitizer, and these sensors help us to detect these events. However, the sensor configuration in the setup does not provide any contextual data about which donor is the target of the hand hygiene activity, if multiple donors are present. Thus, hand hygiene activities done for a donor in the left station and for a donor in the right station are represented by the same low-level events. Similarly, in our lab setup, we use in conjunction a motion sensor, a proximity sensor, and an ambient light sensor (sensing a donor’s arm presence on the arm-rest) to detect three distinct process activities: *apply the tourniquet on the arm*, *disinfect the arm*, and *insert needle in the arm*. These activities are represented by the same low-level events from these sensors. Since no additional sensors can be introduced and we cannot use cameras to disambiguate, the low-level events generated by these sensors can be interpreted as either of the three activities.

3.2 Step 2: Low-level to High-level Event Abstraction with Ambiguity

Low-level events emitted by sensors in Step 1 do not yield process semantics. For example, events generated by a proximity sensor only state how close *something* is from the sensor, but do not specifically inform about any process activity. The event abstraction step fuses and translates low-level events into *high-level* events with process semantics (cf. requirement F2) [14]. Typically, this is achieved by matching specific low-level event patterns signifying the occurrence of a process activity. In this step, we distinguish between a design-time and a run-time phase.

At design-time, a domain expert chooses an appropriate event abstraction approach, defines patterns of low-level events (e.g., based on domain knowledge [28]), and mechanisms for matching these patterns to process events. The discussion of details on the pattern specification is out of scope of this work. Suitable approaches may involve, for instance, developing Complex Event Processing (CEP) applications, like in [27], or training Machine Learning (ML) models, like in [18]. Leveraging process and domain knowledge, each high-level event to be generated upon pattern matching is specified. Each high-level event comprises an activity name, a timestamp attribute, and additional attributes that describe the entities correlated with the event and whose values are derived from the low-level events. Current event abstraction approaches do not account for the existence of multiple high-level interpretations of a given low-level event pattern—they may either assume all interpretations as correct and generate all respective high-level events as they all occurred, or select a single interpretation and generate the respective event, e.g., [18]. To explicitly model ambiguity, we propose leveraging the ambiguity-awareness from the first step and augmenting the event abstraction procedure with a wrapper that generates each possible high-level event interpretation of a pattern, adding a representation of ambiguity to it (cf. requirement F3). Specifically, we model ambiguity in terms of stochastically quantified uncertainty, i.e., a *probability* representing the confidence value of an interpretation [29]. While a domain expert can leverage domain knowledge to specify how probabilities are determined during event abstraction, a domain-agnostic naive strategy is also possible. This strategy simply distributes the

```

<event>
  <string key="concept:name" value="Hand hygiene"/>
  <date key="time:timestamp" value="2024-09-13T11:36:50.000+00:00"/>
  <string key="perform:hcw" value="HCW0001"/>
  <string key="location:station" value="Hygiene station"/>
  <container key="uncertainty:discrete_weak">
    <container key="uncertainty:entry">
      <string key="target:donor" value="D035"/>
      <double key="uncertainty:probability" value="0.5"/>
    </container>
    <container key="uncertainty:entry">
      <string key="target:donor" value="D036"/>
      <double key="uncertainty:probability" value="0.5"/>
    </container>
  </container>
</event>

```

Fig. 4: Ambiguous event example, with uncertainty regarding the specific correlated entity in the role of activity target. The event is represented using the XES extension for uncertain event data [23].

probabilities equally across all interpretations of one ambiguity, for a total mass of 1. In general, however, the total probability mass for one ambiguity might be lower than 1—which is the case of indeterminate events discussed in [22].

At run-time, the event abstraction approach is deployed (e.g., as a CEP app or service, like in [27]) to fuse and analyze low-level events being produced by the sensors. Upon the detection of a defined low-level event pattern that has a single interpretation, the respective high-level event is generated. If a pattern can be interpreted as multiple high-level events, a high-level event per interpretation is generated with the respective probability attached, as defined at design-time. If a pattern can be interpreted as one high-level event but the event-entity correlations have multiple interpretations, one high-level event is generated with one attribute per interpretation with the respective probability (cf. requirement F3). Note that the run-time phase can help to inform the design-time phase by discovering low-level event patterns that unexpectedly result in multiple or incorrect high-level events. Here, an analyst can iterate between the two phases to incrementally refine event abstraction with ambiguity, for instance by defining new interpretations or re-assigning the associated probabilities.

Example. In our lab setup, we can detect the occurrence of activity *Perform hand hygiene* by observing a concurrent proximity within 20 cm from the hand hygiene station and a weight on the scale higher than 2 kg. Accordingly, we defined a low-level event pattern for a CEP service that results in the generation of a high-level event with name *Perform hand hygiene* and with an attribute specifying that the location is the hand hygiene station. However, in the presence of two donors, when generating the high-level event we are not able to determine which of the two donors is the activity target. Therefore, since in our setup we follow the naive strategy to distribute the probabilities, one high level event for the hand hygiene activity is generated with both donors indicated as target of the activity, each with a probability of 50%. Fig. 4 depicts this event using the XES extension for uncertain event data [23].

3.3 Step 3: EKGA Construction

Here, we assume that the previous event abstraction step generates a stream of high-level events with ambiguity, e.g., encoded in the XES format extended with uncertain event data [23], as in our lab setup, through the near real-time fusion and analysis of low-level events. In this step, we aim to translate this stream of events with ambiguity into a representation suitable for performing multi-dimensional analytics (cf. requirement F4). To this end, we build upon the work in [9], which introduced Event Knowledge Graphs (EKGs), and extend EKGs to additionally model ambiguity. As here we focus on defining EKGs with Ambiguity (EKGA), we refer the reader to [9] for a formal definition of EKGs.

Informally, we define an Event Knowledge Graph with Ambiguity as a Labeled Property Graph [32] in which the set of relationship labels defines directly-follows relationships and event-entity correlation types, and in which each relationship label has in addition a probability p as a property. The intuition is that a correlation type label states the specific participation role of an entity in an event, and that a probability states the likelihoods of the respective event-event or event-entity relationship. Since correlation type labels are domain-specific, we do not impose restrictions on their set of values. We specifically propose specialized correlation labels instead of reified correlations (i.e., each correlation is transformed into a new entity representing it; this new entity is correlated with the originally correlated entities by a generic relationship [5]). This is because reification introduces additional nodes and edges in the graph, increasing its size and making it more challenging in terms of cognitive demand for analysts to read and understand [35]. Additionally, in an EKGA we always specify probabilities for all relationship labels, even when their value is 1.

Formally, we define an Event Knowledge Graph with Ambiguity as follows:

Definition 1 (Event Knowledge Graph with Ambiguity). *An event knowledge graph with ambiguity is a labeled property graph $G = (N, R, \lambda, \#)$ with node labels $\{Event, Entity\} \subseteq \Lambda_N$ and relationship labels $\{df, r_{type}\}$ indicating “directly-follows” and correlation relationship types between entities and nodes, with $r_{type} \subseteq \Lambda_R$ and the following properties:*

1. *Every event node $e \in N^{Event}$ records an activity name $e.act \neq \perp$, a timestamp $e.time \neq \perp$, and records a probability $e.p \in [0, 1]$.*
2. *Every entity node $n \in N^{Entity}$ has an entity type $n.type \neq \perp$ and an identifier $n.id \neq \perp$.*
3. *All correlation relationship types are subsumed by the set $R^{corr} = \bigcup R^{r_{type}}$.*
4. *Every correlation relationship $r \in R^{corr}$, $\vec{r} = (e, n)$ is defined from an event node $e \in N^{Event}$ to an entity node $n \in N^{Entity}$; we write $n \in corr(e)$ and $e \in corr(n)$ as shorthand. Every $r \in R^{corr}$ records a probability $r.p \in [0, 1]$ and a type $r.type$.*
5. *Any directly-follows relationship $df \in R^{df}$, $\vec{df} = (e_1, e_2)$ is defined between event nodes $e_1, e_2 \in N^{Event}$, records a probability $df.p \in [0, 1]$, a pair of correlation relationship types $df.rel \in R^{r_{type}} \times R^{r_{type}}$ and refers to a specific entity $df.ent = n \in N^{Entity}$ and two correlations c_1, c_2 such that:*

- (a) $c_1 = (e_1, n), c_2 = (e_2, n) \in R^{corr}$;
- (b) $e_1.time < e_2.time$;
- (c) $\nexists e' \in N^{Event}$ such that $(e', n) \in R^{corr}$ and $e_1.time < e'.time < e_2.time$;
- (d) $df.p = e_1.p \times c_1.p \times e_2.p \times c_2.p$;
- (e) $df.rel = (c_1.type, c_2.type)$.

The original EKG definition in [9] states that a directly-follows relationship between two event nodes depends on the existing event-entity correlations of the nodes and their timestamp property. It can be established only if the event nodes correlate with the same entity node and there exists no other event node that correlates with the same entity node and whose timestamp is between the timestamps of the two nodes. The definition implies that an event can directly follow multiple events based on different event-entity correlations. The intuition is that a directly-follows relationship models a temporal order relation that is *local* to the entity that the events correlate with. This allows representing the behavioral information relevant to each individual entity separately from other entities' behavioral information [9]. Extending the original EKG definition, Def. 1 states that in an EKGA a directly-follows relationship is established on the basis of a correlation with a common entity node, with the additional assignment of the probability property. The probability of the directly-follows relationship is determined as the product of the probabilities of the events and of the probabilities of the correlation relationships that determine it. Note that, given an event node and an entity node, due to ambiguities there can be multiple relationships between them with a probability < 1 . Therefore, in the presence of ambiguity, there can be multiple directly-follows relationships between two given event nodes due to the same entity node, with probability < 1 . We denote with $\langle l, p \rangle$ a (correlation or directly-follows) relationship with label l and probability p .

The procedure we propose for the online translation of a stream of high-level events into an EKGA is similar to the one for EKG construction described in [9], with the difference that it adds additional roles and probabilities, and computes the directly-follows relations after each new event is added to the graph. For each event in the input stream, a new event node E is added to the EKGA. The new event node is created with properties activity name as node identifier, event timestamp as the node timestamp, and node probability as specified in the event or 1 if not specified. For each context attribute in the event, a new entity node N , if not preexisting, is created in the EKGA (otherwise this preexisting N is fetched from the graph), and a new correlation edge R is created between E and N , along with the role property for R as specified in the event attribute. If a probability is specified for the attribute, this is assigned to the correlation probability property, 1 otherwise. Lastly, the directly-follows relationships for the new event are computed, based on the correlation with the entity nodes. The probability of a directly-follows relationship is computed by multiplying the probabilities of the new event, of its correlation with entity N , of the most recent event correlated with N , and of its correlation with N (cf. Def. 1). In principle, the directly-follows relations can be computed after all events have been added to the graph (cf. [9]). However, when considering an event

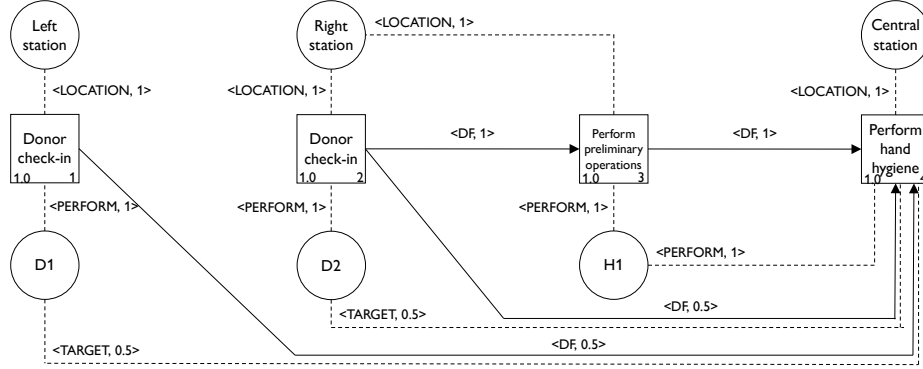


Fig. 5: Example EKGA. Squares = event nodes; circles = entity nodes; dashed lines = correlations; arrows = directly-follows relations. Event probabilities and timestamps are at each event bottom-left and bottom-right corner, respectively

stream as input, this would mean postponing the directly-follows computation indefinitely (cf. requirement F4). Therefore, the proposed procedure computes new directly-follows relations after each new event (with its entities) is translated into the respective EKGA elements. To avoid establishing incorrect directly-follows relations, we require events to be ordered by increasing timestamp value.

Example. Fig. 5 illustrates an EKGA resulting from four high-level events encoded in XES extended with uncertainty (cf. Fig 4) from the blood donation process in our lab setup. The EKGA depicts four event nodes e_1 to e_4 , all with probability 1 ($e_1.act = \text{Donor check-in}$, $e_2.act = \text{Donor check-in}$, $e_3.act = \text{Perform preliminary operations}$, and $e_4.act = \text{Perform hand hygiene}$, with $e_i.time = i$) and six entity nodes (n_1 with $n_1.type = \text{Donor}$ and $n_1.id = \text{D1}$, n_2 with $n_2.type = \text{Donor}$ and $n_2.id = \text{D2}$, n_3 with $n_3.type = \text{HCW}$ and $n_3.id = \text{H1}$, n_4 with $n_4.type = \text{Station}$ and $n_4.id = \text{Left station}$, n_5 with $n_5.type = \text{Station}$ and $n_5.id = \text{Right station}$, and n_6 with $n_6.type = \text{Station}$ and $n_6.id = \text{Central station}$). e_4 is correlated with n_3 with $\langle \text{perform}, 1 \rangle$, meaning that we are certain that H1 is the performer of the hand hygiene. e_4 is correlated with n_1 by $\langle \text{target}, 0.5 \rangle$ and e_4 is correlated with n_2 by $\langle \text{target}, 0.5 \rangle$, meaning that D1, resp. D2, is the donor for which e_4 occurred with probability 50%. Thus, e_4 directly follows e_1 with probability 50% due to n_1 , resp. e_4 directly follows e_2 with probability 50% due to n_2 , signifying that these df-relations are ambiguous.

3.4 Step 4: Multi-dimensional Analytics and Ambiguity Resolution

The EKGA representing a digital trace derived in Step 3 enables multi-dimensional analytics [9]. For example, the EKG-based deterministic analyses of actor behaviors in [16] can provide insights into the interactions between HCWs in the blood donation process (cf. Sect. 2); another example is multi-dimensional conformance checking [8], e.g., supported by the model in Fig. 2. However, to the best of our knowledge, existing approaches to multi-dimensional analytics do not contemplate the presence of stochastic event data. Developing new ambiguity-aware

approaches requires considering the stochastic information associated with the event occurrence and the correlations. An option is to extend the deterministic approach in [16] to deal with stochastically known correlations between activities and the performing actors. Additionally, as interpretations of ambiguities are context-dependent [17], we foresee disambiguation methods to leverage the representation of the events situational context in terms of event-entity correlations. Specifically, knowing about event-entity correlations enables the definition of strategies to infer the admissible and most likely interpretations of ambiguous events, i.e., context-dependent ambiguity resolution. Possible strategies include unsupervised (e.g., LLM-driven) and supervised methods (i.e., domain expert-driven). We plan to investigate different ambiguity resolution approaches. On the one hand, we will leverage domain knowledge integrated with the EKGA, in analogy to the ontology-based approach in [30] to derive missing entity identifiers. On the other hand, we will use the all optimal alignments described in [25] to determine the most likely interpretations. These are only initial examples of the opportunities enabled by EKGAs for analytics and ambiguity resolution: here, an exhaustive and in-depth discussion of these opportunities is out of scope.

4 Implementation and Experiments

Having demonstrated in prior work the capabilities of event sensing and abstraction in near real-time (cf. [27]), we implemented a prototype for the EKGA construction procedure to assess its ability to process input in online settings. The prototype is able to receive in input an event stream in the MQTT-XES format [7]. To support the representation of stochastic uncertainty, we specifically consider the XES extension with uncertainty introduced in [23]. The prototype is implemented in Python; it uses the graph database Neo4j¹ to store the EKGA, with Cypher as the query language. The implementation is available at <https://github.com/ics-unisg/ekga>. Our execution environment is a Win Server 2025 machine with a 12-core CPU@3.80GHz and 48GB of memory.

To assess the applicability of the prototype to an online setting (cf. requirement N1), we measured the latency for translating each new event in an input stream of process events (originating from a log file) into an EKGA. To the best of our knowledge, no event logs with uncertainty describing real process executions are publicly available to be used as input. Therefore, we took an event log describing 36 executions of the blood donation process conducted in our lab setup (cf. Sect. 2) in the period 11–13 September 2024 by a group of local medical students acting as HCWs². Due to our lab setup being limited to two donor stations, a maximum of two blood donors treated concurrently by the same HCW is recorded in the log. The log contains a total of 800 events, of which 328 uncertain, 83 unique entities, and 39 uncertain event-entity correlations. We then artificially extended this log, generating an event log with

¹ <https://neo4j.com/>

² They executed a more realistic version of the process shown in Fig. 2, with a fine-grained specification of the activities: this specification is available in github.

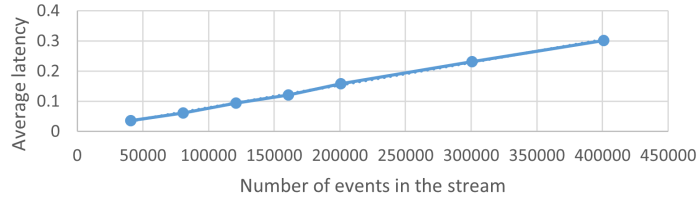


Fig. 6: Average latencies (in s) for translating events from input streams of varying size into EKGA

400800 events, of which 164328 uncertain, and 19539 uncertain event-entity correlations. In the extended log we kept the unique entities the same as in the original log without adding new ones. This is because having events that correlate with always new entities requires only local traversals of the EKGA to compute the directly-follows relationships, which is unrealistic and it would bias the latency measurement. We used this artificially extended log as the input for the prototype by streaming it using MQTT-XES and measured the latency to translate each event. The average time to process each event in the stream was 0.302s, with a standard deviation of 0.195s, and 95th percentile 0.645s. Having measured the latency for each event, we observed a growth in the latency with the total number of processed events. Hypothesizing that the growth is linear, we performed a doubling test, in which we generated additional logs (all available on github) by linearly expanding the original one to measure the respective average translation latencies. With these measurements, we observed a linear trend in the average event translation times, supported by a coefficient of determination $R^2 = 0.99903$, indicating an extremely strong fit to a linear model (cf. Fig. 6).

Overall, the average latencies are below 1.0s, therefore compatible with the online translation of an event stream into an EKGA for an application domain such as healthcare. For application domains where scalability is an additional requirement (not considered here), and safety-critical IoT systems requiring strict real-time capabilities, however, further experiments might need to be conducted.

5 Related Work

Managing ambiguity in digital traces is a recognized challenge. Ambiguity is indicated in [10] to affect multiple artifacts managed in the BPM lifecycle, among which event logs. The work in [10] analyzes sources of ambiguity in event logs such as missing and incomplete data, which frequently happen in IoT settings. The manifesto paper in [14] recognizes the necessity to deal with event log ambiguity as one of the challenges resulting from the integration of IoT and BPM. It specifically points out the problem of establishing event-instance correlations, caused by the lack of a 1:1 mapping between process events and process instances. Among the most pressing challenges in BPM indicated in [4] are challenges related to data quality and adequate data granularity for analysis. Data

quality issues and inadequate data granularity are enablers of ambiguity about event-entity correlations and entity participation roles addressed here.

The work in [30] extends EKGs with entity and activity context, introducing a rule-based inference mechanism to derive missing information in an EKG derived from an event log. The proposed mechanism constitutes a potential alternative to our proposed approach and underscores the potential of ambiguity resolution in EKGAs. It assumes that at least one occurrence of each identifier appears in the original event log to be able to process the whole log at once for the inference of the missing identifiers. In contrast, we assume entity identifiers to be always present: relaxing this assumption would allow us to study how to apply the inference mechanism to our incremental EKGA construction. In [31], authors implement object-centric event data models using EKGs, while in [34], authors propose an approach for extracting object-centric event logs using knowledge graphs. We expect EKGAs to allow the extension of the set of application scenarios of these works to settings with stochastically known uncertainty.

In [21], authors studied uncertainty in event logs, proposing conformance checking methods for logs with uncertainty [22]. With the object-centric representation enabled by EKGAs, these works stimulate further research on how to transfer these methods to EKGAs. The relevance of uncertainty in log data was further highlighted in [12], focusing on challenges related to stochastically known logs. Our stochastic modeling of ambiguity further underlines the significance of these challenges and promotes developing new approaches to tackle uncertainty.

6 Conclusion

The convergence of multi-dimensional process analytics and the BPM-IoT integration brings forth the challenge of modeling ambiguity in IoT-based events to enable ambiguity-aware multi-dimensional analytics. Here, we set requirements and outlined a framework to enable the analytics of IoT-based digital traces with ambiguous events. Within the framework, we focused on the step of explicitly modeling ambiguity in process events derived from IoT data as this work main contribution. The extension of EKG with ambiguity and the definition of the EKGA online construction method are critical constituents of this step and pave the way to developing multi-dimensional ambiguity-aware analytics. Having focused on modeling ambiguity, in future work we will refine the remaining steps, develop and evaluate an end-to-end framework implementation including analytics, and evaluate its scalability. We will also conduct a user-centered study of EKGA understandability. Moreover, we plan to study ambiguity resolution methods in EKGAs based on graph transformation rules and, alternatively, optimal alignments to pinpoint the most likely interpretations of ambiguous events.

Acknowledgments

This work has received funding from the Swiss National Science Foundation under Grant No. IZSTZ0_208497 (*ProAmbitIon* project).

References

1. van der Aalst, W.M.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Software Engineering and Formal Methods: 17th International Conference, SEFM 2019, Oslo, Norway, September 18–20, 2019, Proceedings 17. pp. 3–25. Springer (2019)
2. van der Aalst, W.M., Carmona, J.: Process mining handbook. Springer Nature (2022)
3. Almeida, J.P.A., Falbo, R.A., Guizzardi, G.: Events as entities in ontology-driven conceptual modeling. In: Int. Conf. on Conceptual Modeling. pp. 469–483. Springer (2019)
4. 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)
5. Benelhaj-Sghaier, S., Gillet, A., Leclercq, É.: Knowledge graph multilevel abstraction: A property graph reification based approach. In: International Conference on Research Challenges in Information Science. pp. 12–19. Springer (2024)
6. Bertrand, Y., Veneruso, S., Leotta, F., Mecella, M., Serral, E.: Nice: the native iot-centric event log model for process mining. In: International Conference on Process Mining. pp. 32–44. Springer (2023)
7. Burattin, A., Eigenmann, M., Seiger, R., Weber, B.: Mqtt-xes: Real-time telemetry for process event data. In: CEUR Workshop Proceedings. vol. 2673, pp. 97–101. CEUR-WS (2020)
8. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. *Expert systems with applications* **65**, 194–211 (2016)
9. Fahland, D.: Process mining over multiple behavioral dimensions with event knowledge graphs. In: Process Mining Handbook, pp. 274–319. Springer (2022)
10. Franceschetti, M., Seiger, R., López, H.A., Burattin, A., García-Bañuelos, L., Weber, B.: A characterisation of ambiguity in bpm. In: Int. Conf. on Conceptual Modeling. pp. 277–295. Springer (2023)
11. Franceschetti, M., et al.: Proambition: Online process conformance checking with ambiguities driven by the internet of things. In: Research Projects Exhibition at CAiSE 2023. pp. 52–59. CEUR-WS.org (2023)
12. Gal, A.: Everything there is to know about stochastically known logs. In: ICPM. pp. xvii–xxiii. IEEE (2023)
13. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Declarative modelling and safe distribution of healthcare workflows. In: FHIES 2011. Revised Selected Papers 1. pp. 39–56. Springer (2012)
14. Janiesch, C., et al.: The internet of things meets business process management: a manifesto. *IEEE Systems, Man, and Cybernetics Magazine* **6**(4), 34–44 (2020)
15. Klijn, E.L., Mannhardt, F., Fahland, D.: Multi-perspective concept drift detection: Including the actor perspective. In: International Conference on Advanced Information Systems Engineering. pp. 141–157. Springer (2024)
16. Klijn, E.L., Tentina, I., Fahland, D., Mannhardt, F.: Decomposing process performance based on actor behavior. In: 2024 6th International Conference on Process Mining (ICPM). pp. 129–136. IEEE (2024)
17. López, H.A.: Challenges in legal process discovery. In: ITBPM@ BPM. pp. 68–73 (2021)

18. Maier, J.B., Gram, J., Weisbarth, M., Hennebold, C., Huber, M.F.: Unsupervised event abstraction for automatic process modeling of plc-controlled automation systems. *Procedia CIRP* **120**, 631–636 (2023)
19. Mangler, J., et al.: From internet of things data to business processes: Challenges and a framework (2024), <https://arxiv.org/abs/2405.08528>
20. Organization, W.H.: Who guidelines on drawing blood: best practices in phlebotomy (2010)
21. Pegoraro, M., van der Aalst, W.M.: Mining uncertain event data in process mining. In: ICPM. pp. 89–96. IEEE (2019)
22. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.: Conformance checking over uncertain event data. *Information Systems* **102**, 101810 (2021)
23. Pegoraro, M., Uysal, M.S., van der Aalst, W.M.: An xes extension for uncertain event data. arXiv preprint arXiv:2204.04135 (2022)
24. Plötzky, F., Britz, K., Balke, W.T.: Shards of knowledge—modeling attributions for event-centric knowledge graphs. In: International Conference on Conceptual Modeling. pp. 259–276. Springer (2023)
25. Rivera-Partida, A., Armas-Cervantes, A., García-Bañuelos, L., Rodríguez-Flores, L.: All Optimal k-Bounded Alignments Using the FM-Index. In: Cooperative Information Systems: 30th International Conference, CoopIS 2024, Proceedings (2024)
26. Safety, W.P., Organization, W.H., et al.: Who guidelines on hand hygiene in health care. Tech. rep., World Health Organization (2009)
27. Seiger, R., Franceschetti, M., Weber, B.: Data-driven generation of services for iot-based online activity detection. In: Int. Conf. on Service-Oriented Computing. pp. 186–194. Springer (2023)
28. Seiger, R., Franceschetti, M., Weber, B.: An interactive method for detection of process activity executions from iot data. *Future Internet* **15**(2), 77 (2023)
29. Seiger, R., Zerbato, F., Burattin, A., García-Bañuelos, L., Weber, B.: Towards iot-driven process event log generation for conformance checking in smart factories. In: 2020 IEEE 24th international enterprise distributed object computing workshop (EDOCW). pp. 20–26. IEEE (2020)
30. Swevels, A., Dijkman, R., Fahland, D.: Inferring missing entity identifiers from context using event knowledge graphs. In: Int. Conf. on Business Process Management. pp. 180–197. Springer (2023)
31. Swevels, A., Fahland, D., Montali, M.: Implementing object-centric event data models in event knowledge graphs. In: International Conference on Process Mining. pp. 431–443. Springer (2023)
32. Webber, J., Eifrem, E., Robinson, I.: Graph databases. O'Reilly Media, Incorporated (2013)
33. Weber, B., Abbad-Andaloussi, A., Franceschetti, M., Seiger, R., Völzer, H., Zerbato, F.: Leveraging digital trace data to investigate and support human-centered work processes. In: International Conference on Evaluation of Novel Approaches to Software Engineering. pp. 1–23. Springer (2023)
34. Xiong, J., Xiao, G., Kalayci, T.E., Montali, M., Gu, Z., Calvanese, D.: A virtual knowledge graph based approach for object-centric event logs extraction. In: ICPM. pp. 466–478. Springer (2022)
35. Yoghourdjian, V., Yang, Y., Dwyer, T., Lawrence, L., Wybrow, M., Marriott, K.: Scalability of network visualisation from a cognitive load perspective. *IEEE transactions on visualization and computer graphics* **27**(2), 1677–1687 (2020)
36. Zavala, A.M., Day, G.E., Plummer, D., Bamford-Wade, A.: Decision-making under pressure: medical errors in uncertain and dynamic environments. *Australian Health Review* **42**(4), 395–402 (2017)