

# Towards a Simple and Extensible Standard for Object-Centric Event Data (OCED) — Core Model, Design Space, and Lessons Learned

Dirk Fahland<sup>1</sup>, Marco Montali<sup>2</sup>, Julian Lebherz<sup>3</sup>, Wil M.P. van der Aalst<sup>4</sup>, Maarten van Asseldonk<sup>5</sup>, Peter Blank<sup>6</sup>, Lien Bosmans<sup>7</sup>, Marcus Brenscheidt<sup>8</sup>, Claudio di Ciccio<sup>9</sup>, Andrea Delgado<sup>10</sup>, Daniel Calegari<sup>10</sup>, Jari Peepkorn<sup>11</sup>, Eric Verbeek<sup>1</sup>, Lotte Vugs<sup>5</sup>, and Moe Thandar Wynn<sup>12\*</sup>

<sup>1</sup> Eindhoven University of Technology, the Netherlands,  
{d.fahland,h.m.w.verbeek}@tue.nl

<sup>2</sup> Free University of Bozen-Bolzano, Italy, montali@inf.unibz.it

<sup>3</sup> Standard Chartered Bank, Singapore, julian.lebherz@sc.com

<sup>4</sup> Process and Data Science Chair, RWTH Aachen University, Aachen, Germany,  
wvdaalst@pads.rwth-aachen.de

<sup>5</sup> Konekti, Eindhoven, the Netherlands, {maarten,lotte}@getkonekti.io

<sup>6</sup> PwC, Switzerland, peter.blank@pwc.ch

<sup>7</sup> Randstad Digital, Leuven, Belgium, lienbosmans@live.com

<sup>8</sup> Mbrenscheidt@outlook.de

<sup>9</sup> Utrecht University, The Netherlands, c.diciccio@uu.nl

<sup>10</sup> Instituto de Computación, Facultad de Ingeniería, Universidad de la República,  
Uruguay, {adelgado,dcalegar}@fing.edu.uy

<sup>11</sup> Research Center for Information Systems Engineering (LIRIS), KU Leuven,  
Belgium, jari.peepkorn@kuleuven.be

<sup>12</sup> Queensland University of Technology, Australia, m.wynn@qut.edu.au

**Abstract.** Process mining is shifting towards use cases that explicitly leverage the relations between data objects and events under the term of *object-centric process mining*. Realizing this shift and generally simplifying the exchange and transformation of data between source systems and process mining solutions requires a standardized data format for such *object-centric event data* (OCED). This report summarizes the activities and results for identifying requirements and challenges for a community-supported standard for OCED. (1) We present a proposal for a *core model* for object-centric event data that underlies all known use cases. (2) We detail the limitations of the core model wrt. a broad range of use cases and discuss how to overcome them through conventions, usage patterns, and extensions of OCED, exhausting the *design-space for an OCED data model* and the inherent trade-offs in representing object-centric event data. (3) These insights are backed by *five independent OCED implementations* which are presented alongside a series of lessons learned in academic and industrial case studies. The results of this report provide guidance to the community to start adopting and building

\* Moe Thandar Wynn and Julian Lebherz coordinated the efforts of the OCED working group 2021-2024 which led to the results presented in this report.

new process mining use cases and solutions around the reliable concepts for object-centric event data, and to engage in a structured process for standardizing OCED based on the known OCED design space.

**Keywords:** event data, process mining, standardization

## 1 Introduction

**Rationale behind OCED** In the past decade, Process Mining has not only seen tremendous growth in the academic arena, but also started to establish itself as one of the predominant approaches to improving processes for larger companies of virtually any industry. Process Mining and related services have become a sizeable business - for software vendors, professional service firms, and commercial end user alike.

Such a trajectory naturally spurs substantial investment and advancement; however, it is also typical that – in an attempt to safeguard intellectual property – many new features, products and services are being shielded off from usage by other players in the ecosystem. Especially when this affects areas that are of concern to all market participants, such silos impede competition, innovation and the pace of further development. In many industries that are heavily reliant on the exchange of something, standardizing terms and conditions of such exchange led to a great leap forward for the entire ecosystem – think containers for global trade, internet protocol for global communication.

Process Mining itself is heavily reliant on the exchange of data, which typically originates from systems that were not designed around this use case and hence requires substantial transformation. Market participants have created different approaches to reduce the effort required for data transformation, but so far no data exchange format has seen enough adoption to be nominated as a de-facto standard. The relevance and magnitude of this bottleneck as one of the predominant effort drivers in process mining projects has been reconfirmed by the *IEEE Task Force on Process Mining (TFPM)*<sup>13</sup>.

With the *IEEE eXtensible Event Stream (XES)* [20,4,21] initiated in 2010, academia has established a data exchange format, which fueled tremendous growth in process mining research. Standardizing how tools capture, transfer, load, and interpret event data continues to pay dividends. XES is supported by several commercial tools, and all tools support the main concepts identified in the XES meta-model (e.g., concepts like event, trace, timestamp, and attribute). However, the specific XML file format is not widely used in practice. Moreover, process mining adoption by business entities and scientific progress in the past decade changed the requirements towards an up-to-date standard substantially. The discipline is moving from case-centric event data to object-centric event data [1], making XES less relevant. Therefore, the IEEE TFPM decided in 2021 to initiate a community process to co-design the XES successor

<sup>13</sup> <https://www.tf-pm.org/>

**Object-Centric Event Data (OCED)** as a data exchange format to standardize and thereby facilitate system interoperability within process mining and adjacent areas, like process automation, process simulation, and Business Process Management. OCED shall:

1. **spur innovation and competition** by lowering the barrier for new ideas and market participants to access and enter the eco-system
2. **improve the Return on Investment (ROI)**
  - (a) **for all market participants** by improving the security of investments with standardized eco-system access,
  - (b) **for commercial end users and professional service firms** by allowing them to focus more on business value creation and less on data transformation,
  - (c) **for software vendors** by allowing them to focus on differentiating functionality rather than developing yet another data transformation engine, and
3. **create a new marketplace** for source system-specific adapter modules, translating source data into/from OCED.

**Balancing Simplicity and Expressivity** This report summarizes the community efforts and results in developing proposals for establishing a new standard for Object-Centric Event Data. Section 2 briefly documents the events and activities organized by the IEEE Task Force on Process Mining in eliciting requirements and proposals for OCED. A key challenge emerging from this process was that OCED has to be, both:

1. conceptually simple to facilitate implementation and adoption in practice, and
2. conceptually expressive to allow supporting a large number of use cases (considering both forms and types of data in source systems and analysis objectives over this data)

Feedback on initial proposals for OCED (included in Appendices A and B) shared with the community led to the consensus that any standard model for OCED should start from a *core* of essential concepts necessary in all use cases. The core in turn should be easily extensible to cover a wider range of use cases.

Section 3 documents the proposal for such a core model for OCED, that can be seen as the common denominator of the initial proposals for OCED, as well as its limitations.

Section 4 summarizes the arguments made around the need for extending OCED beyond its core as well as the implications and challenges when one attempts to do so. These challenges concern basic principles of conceptual modeling and data modeling as well as implementation considerations. Overcoming these challenges requires to establish further conventions and best practices around OCED.

Section 5 complements these challenges with reports on concrete results and experiences gathered from four independent implementations of OCED (its core

and extensions). Besides highlighting several practical challenges in implementing OCED and how these have been overcome, this section also provides pointers to resources for working with OCED and its further development.

The concluding Section 6 summarizes how the core model for OCED forms a reliable basis for building an eco-system of process mining around object-centric event data but also the potential sources of ambiguity that still hinder interoperability. It also outlines the open challenges and a possible roadmap to make OCED more robust for advanced use cases.

This report thus summarizes the lessons learned on identifying core concepts of OCED and its extensions, but does not formulate a standard<sup>14</sup> for OCED. Rather, its contents shall inform the next steps in the process of developing a community standard for OCED.

## 2 The Path to OCED

The community process initiated by the IEEE TFPM on co-designing OCED involved several steps.

**Requirements gathering.** The requirements for OCED were gathered through an online survey with 289 participants and a XES 2.0 workshop co-located with the 3rd International Conference on Process Mining (Eindhoven, 2021) [22]. This led to the following three observations

1. the single-case-notion is very limiting, leading to a disconnect between reality and the represented events,
2. the XES standard is too complex and many of its extensions are rarely used, and
3. XES is associated with a particular XML storage format making it impractical for many real-life use cases.

Learning from the problems associated with XES, OCED needs to be

1. object-centric (i.e., an event may refer to any number of objects instead of a single case) [1,2,12],
2. as simple as possible, and
3. have a meta-model decoupled from a particular storage format.

**Proposal development.** A core *OCED working group* of eight experts from academia and industry was formed to develop an initial proposal for the *OCED Meta-Model* (OCED-MM). Diverse views and opinions were solicited and discussed among the core team in striking the right balance between expressivity of the model and its simplicity. The resulting meta-model captures the concepts

<sup>14</sup> This document deliberately does not refer to any of the data models or meta-models for OCED presented in this report as “standard” as these currently do not meet the associated requirements.

that have a majority vote. The meta-model resulting from this discussion, consisting of a *Base Model* and a *Full Model*, is included in Appendix A.

It was circulated for feedback in two runs: a first run on August, 2nd 2022 to respondents of the 2021 survey, registered attendees of the 2021 XES Workshop as well as the TFFPM steering committee and advisory board, and a second run on September, 9th 2022 to all subscribers of the TFFPM newsletter.

As part of the 4th International Conference on Process Mining (Bolzano, 2022) an XES Symposium was held, in which the OCED-MM was presented and discussed. Many interesting thoughts and ideas had been brought to the attention of the OCED working group during this meeting; specifically, symposium participants stated that OCED-MM was, both,

- not expressive enough as it does not natively support a number of use cases considered relevant by the participants, essentially asking to extend OCED-MM further, and
- being too complex in the concepts that need to be considered and implemented, hindering adoption.

This feedback underlined the difficulty of striking the right balance between expressivity and simplicity. Yet, the core concepts of the OCED-MM Base Model remained in consensus.

**Call for Reference Implementations** In order to get further clarity and feedback on the effort and challenges in implementing an object-centric event data model and exchange format the OCED working group issued on March 10, 2023 a *Call for Reference Implementations* of the OCED-MM Based Model or the OCED-MM Full Model.

As part of the 5th International Conference on Process Mining (Rome, 2023), an OCED Symposium was held at which 4 independent implementations were presented and discussed:

- an implementation of the *OCED-MM Base Model* by the company Konekti<sup>15</sup>, further described in Sect. 5.1;
- an implementation of the *OCED-MM Base Model* called OpenOCED by Delgado et al. [8], further described in Sect. 5.2;
- an implementation of the *OCED-MM Full Model* by Swevels et al. [18], further described in Sect. 5.3; and
- an implementation of a variation of the OCED-MM Full Model called *Object Centric Event Log V2 (OCEL 2.0)*<sup>16</sup> by Koren et al. [5,14]; the core ideas of the OCEL 2.0 model are shown in App. B, the implementation is further described in Sect. 5.4.

A fifth independent implementation has recently been published:

<sup>15</sup> <https://getkonekti.io/>

<sup>16</sup> <https://www.ocel-standard.org/>

- an implementation of the *OCED-MM Full Model* and *OCEL 2.0* by Bosmans et al. called *Stack't*, further described in Sect. 5.5

Alongside the implementations, several datasets in OCED, OCEL 1.0, and OCEL 2.0 format were made available to the community for exploration and adoption (see Sect. 5).

While the independent implementations confirmed that the core ideas of OCED-MM are viable, the discussion among the presenters and participants of the OCED symposium noted a need for, both,

- clarifying the relations and compatibility between the different implementations, and
- clarifying the core concepts of the OCED-MM Base Model and the extensions in the OCED-MM Full Model and OCEL 2.0,

before the proposal can enter a formal standardization process.

In order to facilitate this clarification and pave the way towards standardization, the OCED working group concluded to:

1. derive a minimal *OCED-MM Core Model* which lies at the intersection of all prior proposals and implementations while supporting all essential core concepts, and
2. clarify towards the community the lessons learned and open challenges in extending this OCED-MM Core Model to more expressive OCED models and their implementation.

The remainder of this document presents the results of these efforts and discussions. Section 3 describes the OCED-MM Core Model. Section 4 details the challenges in implementing and extending the OCED-MM Core Model. Section 5 summarizes the lessons learned of implementing OCED in the independent implementations.

### 3 OCED Meta-Model - Core Model

The OCED-MM Core Model shown in Fig. 1 is the simplest<sup>17</sup> model for object-centric event data that the OCED working group could identify. Section 3.1 describes the model which is illustrated by an example in Section 3.2. Section 3.3 highlights non-trivial aspects in the interpretation of the *object* concept in OCED.

Section 3.4 lists the known limitations of this minimal model, also highlighting the interests of the process mining community for more expressivity in an OCED model.

---

<sup>17</sup> in terms of number of concepts and relations included in the model

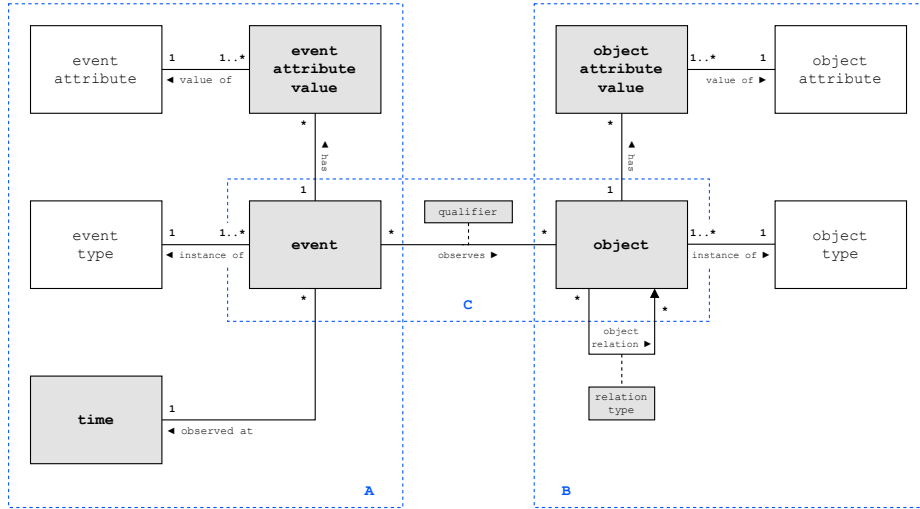


Fig. 1. OCED-MM Core Model

### 3.1 Core Model Concepts

In order to present the OCED-MM Core Model concepts in manageable chunks, Figure 1 is structured into three groups highlighted in blue.

- **Group A** describes events, associated attributes and the time construct, which can be closely related to the XES standard [21] and many non-object centric process mining solutions in the market.
- **Group B** describes the objects, associated attributes, as well as object relations. Inclusion of these concepts directly stems from real-world requirements collected during the development of the proposal, see Sect. 2.
- **Group C** connects the concepts of events and objects in groups A and B.

**Events.** Starting with **Group A**, the following event-centric concepts are captured.

**1. event** — An event describes the occurrence of an observable phenomenon. An event is atomic, meaning it refers to an observation taking place at exactly one point in time rather than having a duration. Every event has, viz. is **instance of**, exactly one **event type** (e.g., *[purchase order approved]*; captured as string). It denotes the kind of observation described by the event. In most use cases, the event type is the process *activity* that was performed, though other types of observations can be described as well (e.g., sensor recordings). Each defined **event type** has at least one event that instantiates it (e.g., *[purchase order approved]* was observed in at least one event *[purchase order ID#298374 approved]*).

**2. time** — Time describes the moment in time where the event has been observed at. It captures both a timestamp conforming to ISO 8601-1:2019 and its resolution (ref. to the precision in which the timestamp was recorded). At a minimum, the following precisions are to be differentiated: date, hour, minute, second, millisecond. If the time zone is omitted, all timestamps are treated as UTC. While the meta-model does not prescribe a method to represent the timestamp-resolution-pair, the ISO standard 8601-2:2019 proposes uncertainty classifiers as a way to store both components in one string.

**3. event attribute value** — Each event has an arbitrary number of event attribute values (e.g., *[USD]* and/or *[Emirates Airlines]*) and corresponding event attribute names (e.g., *[transaction currency]* and/or *[merchant name]*) further describing the observation captured by the event as *attribute-value pairs* (e.g., *[transaction currency] = [USD]* and/or *[merchant name] = [Emirates Airlines]*). Each event attribute value is captured as string, boolean, integer, real, date, time or timestamp (the latter three in accordance with ISO 8601-1:2019). Each event attribute value is related to exactly one event (i.e., if two different events have attributes with the same value, then each event has its “own copy” of the value). Each defined event attribute name relates to at least one event attribute value (e.g., *[user type] = [manual]* or *[RPA]* or *[automated]*), and every event attribute value is value of exactly one event attribute name (i.e., if two different attributes have the same value, then each attribute name has its “own copy” of the value). Some information is typically represented with value-unit-pairs (e.g., price and currency) describing parts of the same logically connected information. In such cases, it is good practice to indicate their relation by choosing the unit’s event attribute name as the value’s event attribute name suffixed with “\_unit” (e.g., *[price] = [48.76]* and *[price\_unit] = [USD]*). Please note that this convention is not prescribed by the meta-model.

**Objects and Relations.** In [Group B](#), the following object-centric concepts are captured:

**4. object** — An object either represents something tangible or abstract. Examples of tangible objects are persons, locations, machines, documents, document line items. Examples of abstract objects are legal entities, organizational constructs, and electronic documents. To represent objects in accordance with the model, it is not required to classify them into either of the two. Every object has exactly one object type (e.g., *[sales order]*; captured as string), while each defined object type relates to at least one object (e.g., *[sales order ID#12345]*).

**5. object attribute value** — Each object has an arbitrary number of object attribute values (e.g., *[blue]* and/or *[1897]*) and corresponding object attribute names (e.g., *[color variant]* and/or *[weight]*) further describing the object as *attribute-value pairs* (e.g., *[color variant] = [blue]* and/or *[weight] = [1897]*).



Each **object attribute value** is captured as string, boolean, integer, real, date, time or timestamp (the latter three in accordance with ISO 8601-1:2019). Nested object attribute values are not supported. Each **object attribute value** is related to exactly one object (i.e., if two objects have the same attribute value, then each object has its “own copy” of this value). Every object attribute value is **value** of exactly one object attribute name, while each defined object attribute name relates to at least one object attribute value (i.e., if two different attributes have the same value, then each attribute name has its “own copy” of the value). Some information is typically represented with value-unit-pairs describing parts of the same logical information. In such cases, it is good practice to indicate their relation by choosing the unit’s object attribute name as the value’s object attribute name suffixed with “\_unit” (e.g.,  $[weight] = [1897]$  and  $[weight\_unit] = [kg]$ ). Please note that this convention is not prescribed by the meta-model.

**6. object relation** — An **object relation** represents a link between a pair of **objects** and is represented as a *directed relationship* from one **object** (reflecting the relation’s origin) pointing to one **object** (reflecting the relation’s target). Every object relation has exactly one **object relation type** (e.g., *purchase order line items* being related to their parent *purchase order*; their object relation type is  $[child\ of]$ ; captured as string). Possible relation types are CHILD\_OF and PARENT\_OF but these are not prescribed by the core model and additional object relation types can be introduced as part of the data capture and used to reflect semantics of the relation.

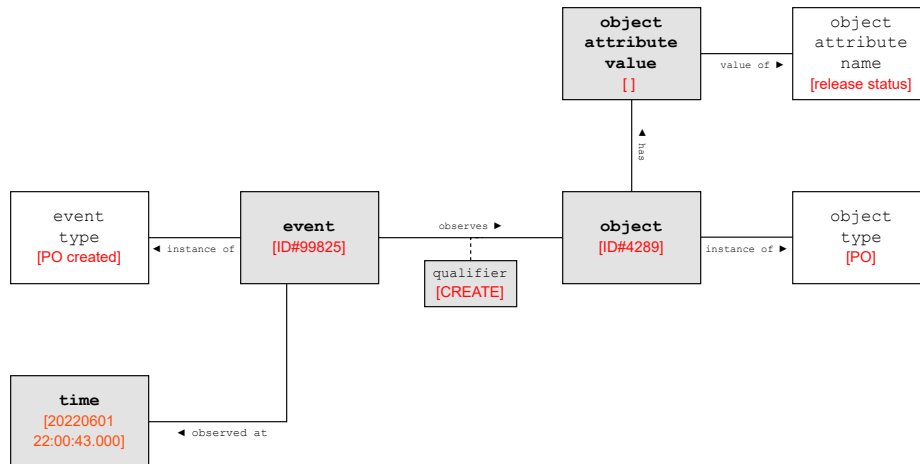
**Event-to-Object Relations.** In [Group C](#), events and objects are connected.

**7. observes relation from events to objects** — In an object-centric setting, an **event** not only observes an activity, viz. **event type**, but also explicitly **observes** (changes to) **objects**. An **event** and an **object** can be related in a **qualified** (i.e., association class) manner, meaning their type of relationship is denoted. Possible qualifiers are CREATE, MODIFY and DELETE but these are not prescribed by the core model and additional qualifiers can be introduced as part of the data capture and used to reflect the semantics of the relationship. Each object can be observed by an arbitrary number of events, while each event can observe an arbitrary number of objects. This means there can be events without objects and vice-versa.

### 3.2 Example

The following shall exemplify how the core meta-model can be applied. For this, real-world scenarios in the context of a purchase-to-pay process have been selected.

**A) PO creation** — A purchase order document is created. In this context we focus on the PO header information only. Hence a PO object is created alongside a number of object attributes. For simplicity we solely present the PO’s release status, which is created as non-released. The respective values are marked with red font in Figure 2, sparing unused parts of the meta-model.



**Fig. 2.** Example Scenario A - PO created

**B) PO release** — The purchase order is released. Data-wise this is reflected in an updated release status. The object itself does not change, but its object attribute value does. Please refer to Figure 3 for details.

**C) Invoice Receipt** — An invoice is received in relation to the PO created in scenario A. The invoice, one of its invoice line items, as well as their object relation get recorded. Another object relation is recorded for the link to the PO. Further PO details (i.e., object attribute values) of the purchase order are omitted in Figure 4 since they do not change. As per standard practice this invoice gets recorded with an active payment block. Since this object attribute value is associated to the invoice (header), it implicitly applies to its children, i.e., invoice line items, as well.

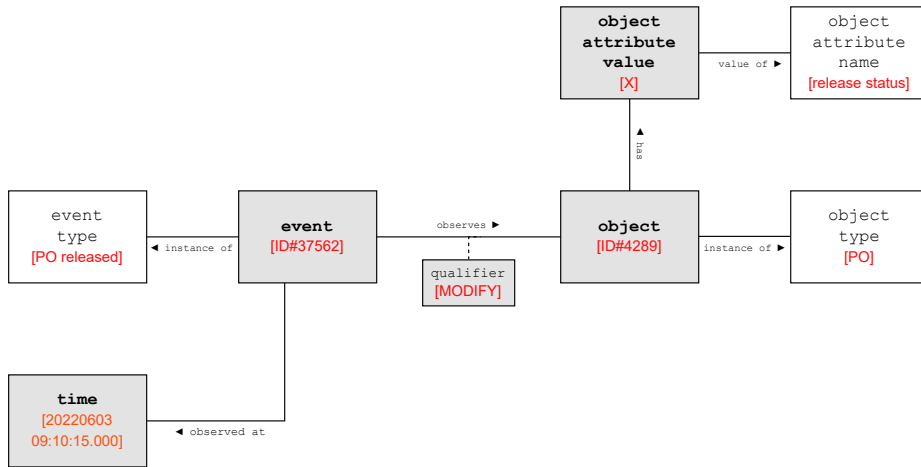


Fig. 3. Example Scenario B - PO released

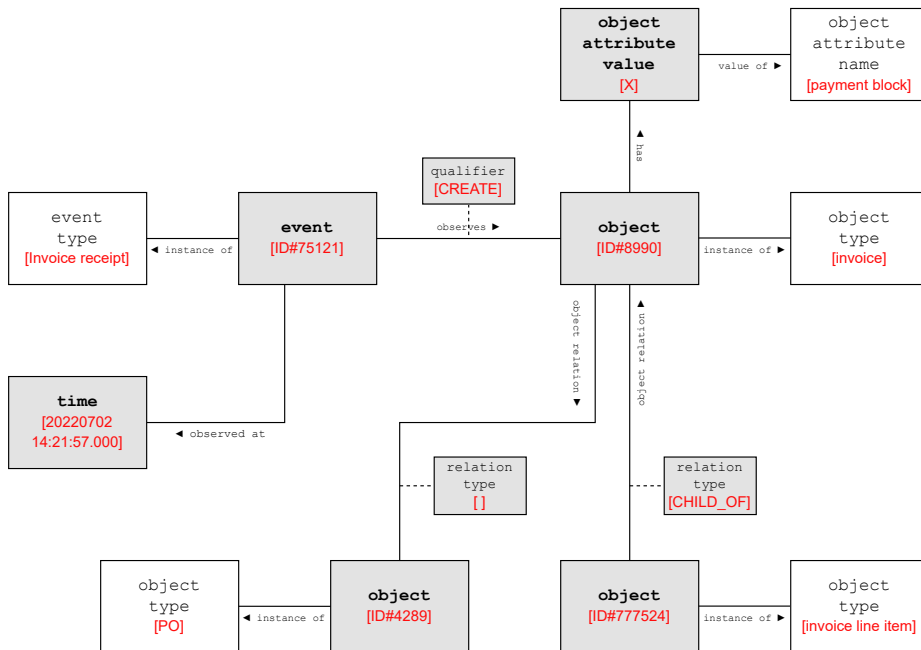


Fig. 4. Example Scenario C - Invoice receipt

### 3.3 Baseline Interpretation of OCED

In OCED, events and objects have distinct roles: an event describes an observation in time<sup>18</sup>, whereas an object describes a tangible or abstract entity that is observed (i.e., via the qualified **observes** relationship).

- Each object needs a *unique object identifier* that events and relations can refer to.
- Likewise, each event needs a *unique event identifier* that objects can refer to, but only events carry a timestamp.
- Attributes are “owned” by their parent concept (object, event) and thus should always be represented/serialized as children of their parent concept that do not carry their own identifier.
- Relations in the OCED-MM Core Model are not distinct entities carrying their own identifier; see Sect. 3.4.

The current OCED-MM Core Model allows some room for interpreting these concepts. The following is a “minimal” *baseline interpretation* of the concepts that can be understood as the basis for OCED and all further extensions.

- In an OCED instance (i.e., a concrete dataset), each object, relation, and attribute is represented once, describing a single state or static view on all objects, relations, and attributes at an *unspecified* point in time. This could be the state of all objects, relations, and attributes at the time of extraction of the data, but this is subject to how the source system provides the data and the type of extraction. Therefore, the OCED Core Model does not enforce this interpretation.
- An **observes** relation from an event  $e$  to an object  $o$  is essentially only a reference of  $e$  to the identifier of  $o$  denoting that  $e$  observed or operated on  $o$  but not denoting what of  $o$  has been observed or changed or in which state  $o$  has been.
- In this representation, the semantics of the event  $e$  wrt.  $o$ , i.e., what  $e$  did with  $o$ , lies in the qualifier of the **observes** relation, e.g., **CREATE** or a domain-specific qualifier, and in domain-knowledge about  $e$ , e.g., the event’s type being *[Create Order]* or *[Change Price]* or *[Re-assign flight]*.

Figure 5 visualizes this baseline interpretation on an OCED instance that combines all three scenarios A-C from Sect. 3.2. Note that this representation shows the latest **value** of attribute *release status* of the PO object (after *PO released*), but not its initial value (before *PO released* occurred). While this baseline interpretation limits which process dynamics OCED can describe without further domain knowledge, it arguably contains the minimum requirements for all interpretations of OCED:

- each object, relation, attribute is described,

<sup>18</sup> While most events describe observations recorded in the past, an event may also describe a “future observation” such as a due date of an invoice

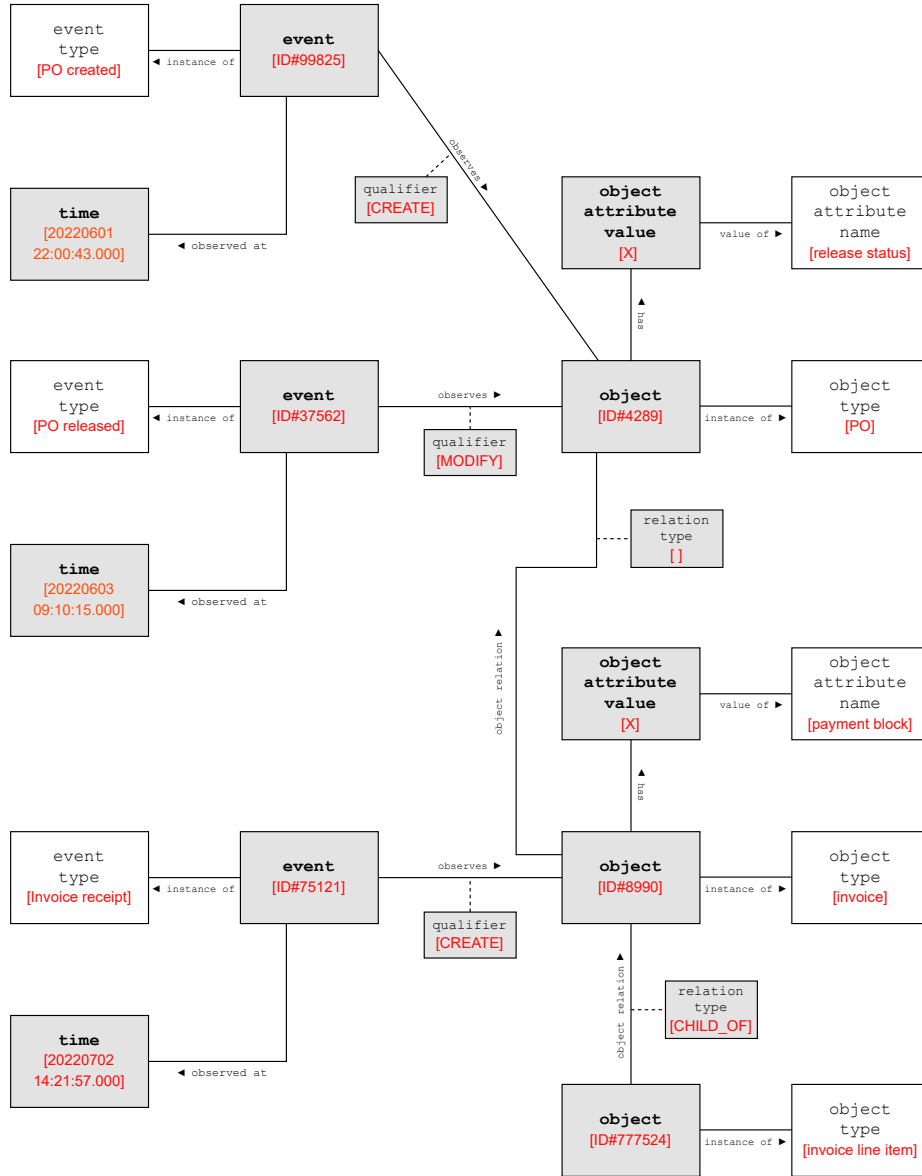


Fig. 5. Baseline representation and interpretation of the Scenarios A-C in a single OCED instance

- each event is described,
- the existence of relations between events and objects is stated

in the minimum required form allowing usage and extension in a variety of use cases.

Note that the OCED-MM Core Model does not prescribe this baseline interpretation, e.g., it does not prescribe that an object is only represented once in the data or that event semantics must be deferred to domain knowledge. However, richer representations and interpretations of data—such as also describing the different values of the *release status* attribute—require more care and consensus as discussed next.

### 3.4 Known Limitations

The meta-model shall be kept as simple as possible, while retaining relevance in industry and academia. This implies that there are some known limitations (and corresponding workarounds) that are also further discussed in Sect. 4:

1. **Event atomicity & no relations between events** — An event is atomic (i.e. has exactly one timestamp) and it is not possible to directly relate events to each other, hence, amongst others, causality relations, activities with both a start and end event and partial orders cannot be stored explicitly; see Sect. 4.6.
2. **No complex attribute values** — The data type of any event attribute value and object attribute value are consistent throughout the whole model, i.e., all values of the same attribute name can only be of type string, boolean, integer, real, date, time or timestamp (date, time and timestamp in accordance with ISO 8601-1:2019). Complex (or nested) data types are not supported. This affects representing complex values as well as how units of numeric values are stored; see Sect. 4.5.
3. **Limited semantics** — Events, object types, and object and event attribute values may carry semantics specific to selected domains. It is encouraged to establish such conventions, however these are not enforced by the meta-model.
4. **Binary object relations** — Object relations in the meta-model are directed and binary. Even though some of the relationships in real life are tertiary or of higher order, such relationships are less frequent; see Sect. 4.4.
5. **Ambiguous representation of object relations** — The meta-model does not prohibit object attributes to store references to other objects, allowing ambiguous and inconsistent representations of object relations, see Sect. 4.3.
6. **Relations are identified by their objects** — Object relations are defined by their source and target object and their type. While this is sufficient for expressing static data models, i.e., where object relations do not change, changes to object relations (creation, modification, deletion) are not unambiguously expressible, see Sect. 4.4.

7. **Limited types and no data schema** — Each concept (event, object, object relation, attribute) has a *type*, but these types are not related to each other and some domains may ascribe more than one type to them, see Sect. 4.7.
8. **Meta-model vs. reference implementations** — The meta-model does not prescribe how things are stored or syntactically represented. For example, there may be one table per event type and one table per object type. The qualified relations may also correspond to tables. Things will be typed, but this is outside the scope of the meta-model and needs to be detailed for reference implementations of this meta-model.

## 4 Challenges in Standardizing and Extending OCED

**Consistent interpretation between producers and consumers.** The OCED-MM Core Model presented in Sect. 3 outlines the core concepts for representing object-centric event data for exchange and storage. Adoption of OCED in practice is subject to it fulfilling a role as *intermediary* between *data producers* (e.g., source systems, ETL solutions) and *data consumers* (e.g., process mining algorithms, tools, and solutions) for a variety of use cases. Thereby, a standardized OCED format has to codify to both producers and consumers how various aspects of object-centric event data are to be *represented* and how various representations of object-centric event data are to be *interpreted* – ultimately allowing a producer to provide a serialization of OCED that a consumer can unambiguously interpret.

**Transport vs. storage vs. analysis.** The large variety of forms in which object, relations, and event data are stored in source systems, as well as the broad range of analysis use cases pose a series of challenges for standardizing representation and interpretation of OCED. Further, process mining tools consuming OCED must implement OCED-compliant data structures that internally model object-centric event data suitable for querying and algorithms. OCED for data exchange may prioritize fewer concepts to reduce storage footprints (e.g., relying on implicit semantics and conventions). Instead, OCED for storage, querying, or algorithms may prefer more explicit representations using more concepts to improve performance and functionality.

The following sections detail these challenges and also outline how extending the core model introduces further sources of ambiguities that can be resolved through agreeing on conventions in representing and interpreting OCED.

### 4.1 Interpreting and Representing Objects over Time

An essential contribution of OCED is the ability to express how events relate to and operate on objects over time. If observed objects change over time, OCED needs a concept to describe *changes to objects over time* and how different observations of the same object are related to each other. For instance, in Fig. 2

and Fig. 3, PO object *ID#4829* changes the value for attribute *release status* from *[]* to *[X]*.

**Event attributes.** A basic possibility is to encode changes in object attribute value changes as attributes of the event performing the change, e.g., by including event attributes *[release\_status\_old]* and *[release\_status\_new]*. However, this encoding requires consistent interpretation of event attribute names and cannot reliably describe value changes of multiple objects observed by an event.

**Static objects and attribute values changes.** A more reliable possibility is to interpret an **object** as a static description of the entire object (e.g., its full representation at the moment of extraction) while historic changes to the object are represented elsewhere, e.g., in a series of timestamped attribute values that is proposed as an extension to OCED in OCEL2.0 and further discussed in Sect. 4.7).

**Object snapshots.** Alternatively, one could interpret the concepts in **Group B** of Fig. 1 not as a static, singular observation of objects, attributes, and relations, but allow the same object *o* to be observed repeatedly (in different states). In this interpretation, the same tangible or abstract entity *o* can be observed multiple times. Each such observation of *o* is represented as an **object** that is interpreted as a “snapshot” of *o*. The **observes** relation then describes that an event **observes** a specific “snapshot” of *o*. This idea extends to the attributes and relations associated with *o* alike.

Figure 6 illustrates such a “snapshot”-based representation of the events and objects of Figures 2, 3, and 4 of the example in Sect. 3.2 together.

If such an interpretation is assumed, the following considerations arise:

- Multiple “snapshots” of the same object *o* must be relatable to each other. For instance, through an immutable **object attribute** that uniquely identifies the object (e.g., *[POid]*, *[InvoiceID]*, and *[InvoiceLineID]* in Fig. 6) and all “snapshots” of *o* carry the same value (e.g., *[POid=#4829]*).
- At the same time, object identifiers no longer identify objects but snapshots (e.g., *#4289-1* and *#4289-2* are different snapshots of the same *PO* object).
- The moments in time where an object “snapshot” has been observed is described through the events that observe the snapshot; the same snapshot can be observed an arbitrary number of times, i.e., referred to from an arbitrary number of events.
- An object “snapshot” is not required to enumerate all object attributes but only those that are relevant for the observation or event, i.e., typically the changed attribute values.
- An object “snapshot” observed by an event itself may be “empty”, i.e., it only contains the object identifier but no other object attributes etc. For example, when an event **observes** an object but does not change any of its attribute values (e.g., the event of a user loading a website).



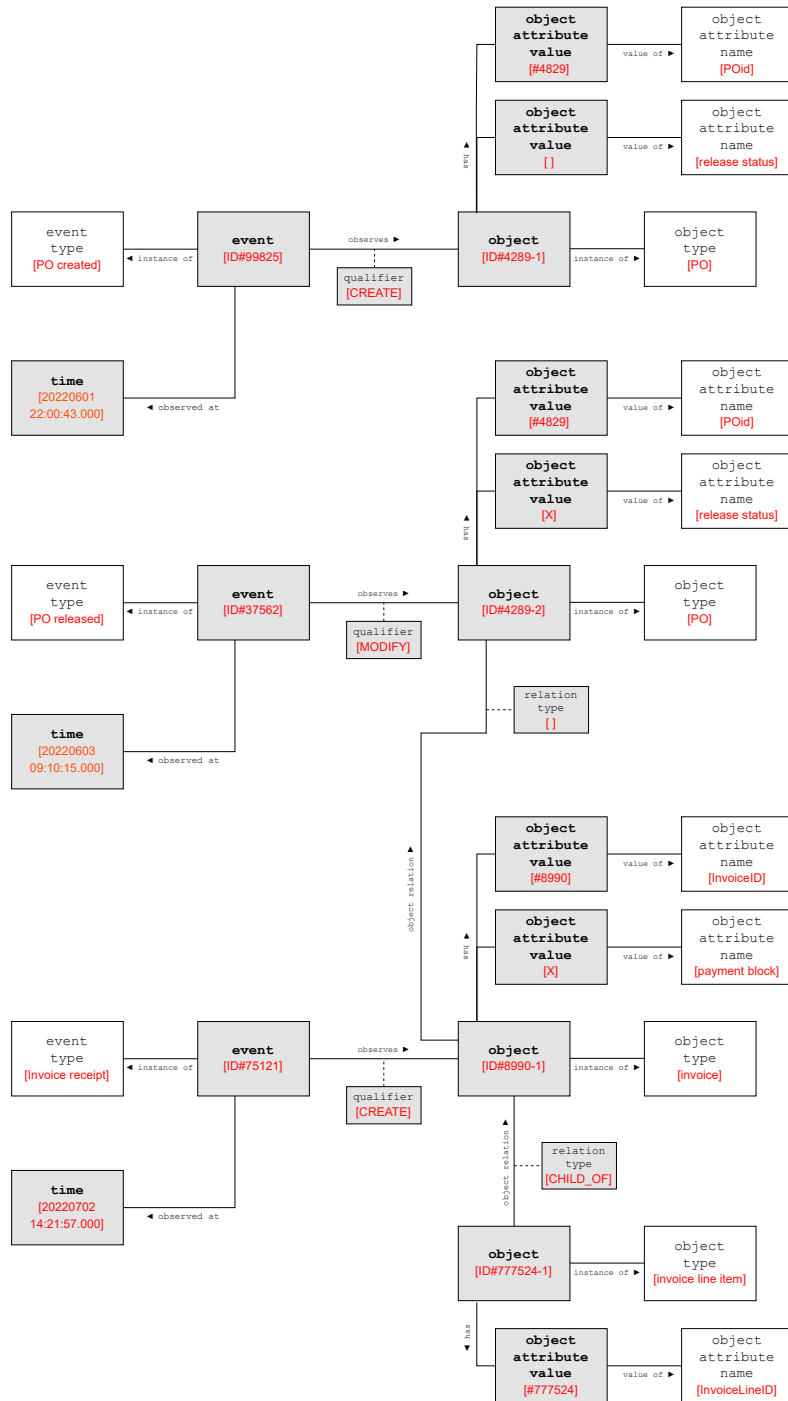


Fig. 6. Snapshot interpretation of the example of Sect. 3.2

- Conversely, each object “snapshot” theoretically could be a complete representation of the entire object every time it is observed. While the OCED concepts do not dictate otherwise, common sense will render this approach non-viable given the storage footprint.
- This interpretation also includes (and hence generalizes) the minimal interpretation of OCED in Sect. 3.3: there is one “full snapshot” for each object that events can observe.

With these considerations in mind, Figure 6 describes the creation of a *PO* object with [*POid*=#4829] and attribute [*release status* = []] at time 2022-06-01 22:00:43.000; the subsequent *PO release* event changes the *PO*’s attribute to [*release status* = *X*] at time 2022-06-03 09:10:15.000. The later *Invoice receipt* event at time 2022-07-02 14:21:57.000 creates an *Invoice* object with [*InvoiceID* = #8990] and [*payment block* = *X*] having an *Invoice Line Item* with *Invoice-LineID* = #777524. At this time (2022-07-02 14:21:57.000), *Invoice* #8990 is related to *PO* #4829 (in its most recently observed state).

**Conclusion.** The OCED-MM Core Model described in Sect. 3 does not prescribe how to interpret an **object** allowing to use it also to describe auxiliary and proxy objects as discussed in Sect. 4.4, 4.5, 4.6. Thus, a snapshot interpretation of an object is possible. However, the baseline interpretation of an **object** describing *the* object is arguably less ambiguous and complex in comparison.

Generally, describing changes to an object over time either requires an extension of the OCED model, i.e., timestamped attribute values, or a more involved interpretation of the **object** concept. Notably, different use cases may require different choices.

## 4.2 Semantics for Qualifiers and Relation Types

OCED allows to *qualify* the **observes** relation and giving a *relation type* to an **object relation**. The purpose of defining qualifiers and relations is to provide semantics to events and object relations that allow consistent interpretation of object-centric event data across various tools and solutions. For this purpose, it is beneficial to agree on a set of qualifiers and relation types with agreed on semantics.

**Agreeing on standard qualifiers and relation types.** Suggested qualifiers that are likely to occur in most use cases are **CREATE**, **MODIFY**, and **DELETE**, while suggested object relation types that are likely to occur in most use cases are **CHILD\_OF** and **PARENT\_OF**. The following semantics have been proposed for these in the past.

**Agreeing on standard qualifier semantics.** Agreeing on qualifiers allows to agree on how to interpret how an event changes objects and associated relations.

For instance, the semantics of a **CREATE** or **DELETE** qualifier on an **observes** relation between an **event** and an **object** implicitly extends to **object relations** the **object** can be interpreted as follows:

1. Any **object relation**, if not created explicitly (i.e., after both objects are in existence), is created implicitly with the **CREATE** of the second **object**.
2. Any **object relation**, if not deleted explicitly (i.e., while both objects are remaining in existence), is deleted implicitly with the **DELETE** of either **object**.

For example, applying this interpretation on Fig. 4 and Fig. 6 implies that the relation from *Invoice #8990* to *PO #4829* is implicitly created by event *Invoice receipt* at time *2022-07-02 14:21:57.000*. Agreeing on such an interpretation allows to omit further **observes** relations between the event and other objects.

**Standard relation type semantics wrt. qualifiers.** Likewise, agreeing on the semantics of relation types allows more extensive interpretations of events. For instance, **CHILD\_OF** defines an **object** relationship, in which an arbitrary number of child **objects** are related to exactly one **parent** object (N:1); the opposite applies to **PARENT\_OF**. The predefined object relation types (**CHILD\_OF**, **PARENT\_OF**) result in further implicit semantics between objects and events:

1. When a child **object** gets **DELETED** by an **event**, its **object relation** to the **parent object** is deleted in unison, however, the **parent object** itself is not affected.
2. When a child **object** gets **CREATED** by an **event**, its **object relation** to the **parent object** is created in unison, however, the **parent object** itself is not affected.
3. When a **parent object** gets **DELETED** by an **event**, all its child **objects** and their **object relations** to the **parent** are deleted in unison.
4. When a **parent object** gets **CREATED**, its initial child **objects** (i.e., the ones without a dedicated related **CREATE** event) are created in unison (e.g., when a *purchase order* is created with ten *line items*, solely the *purchase order's* link to the event needs to be captured).

For example, applying this interpretation on Fig. 4 and Fig. 6 implies that *Invoice Line Item #777524* is implicitly created as a **CHILD\_OF** *Invoice #8990* by event *Invoice receipt* at time *2022-07-02 14:21:57.000*.

Assigning such semantics allows to omit certain relations in the data, especially the **observes** relations from events to a large set of objects, reducing the data footprint for transmission and storage, while allowing the omitted relations (and event semantics) to be unambiguously reconstructed.

### 4.3 Relations between Objects

In the core model, the **object relation** from **object** to **object** only models the *existence* of a relation of a particular type between two objects. The relation

itself does not bear<sup>19</sup> any identifier and thus cannot be referred to. Any **object relation** is implicitly and uniquely identified by the pair (source **object**, target **object**) while **relation type** is only a *qualifier* for the relation, i.e., describes the nature of the pair (source **object**, target **object**).

A strict interpretation of this concept allows that two objects are related by at most one kind of relation, but two objects may not be related in two different ways. For example, assume *[Order o]* is owned by *[Person p]*, and also *[Order o]* has been issued by the same *[Person p]*. The OCED-MM Core Model can only represent the binary relation  $(p, o)$  – which can exist only once – and qualifies it, either by “owned by” or by “issued by”. Most source systems do not have this restriction and identify a relation by the triple (source **object**, target **object**, **relation type**). This challenge has to be addressed through conventions or extensions of OCED, e.g., materializing relations as objects (see Sect. 4.4).

Independently of this uniqueness constraint, further aspects of relations have to be considered. While the OCED meta-model shall be decoupled from any particular storage format (see Sect. 2), general constraints of serializing OCED in a representation for storage to data exchange need to be considered. Specifically wrt. relations, implementations of OCED have to be aware of the following considerations.

**Serializing References.** When serializing any actual instance of OCED, an **object relation** can be serialized as follows:

- As an explicit triple (source, target, type). However, a relation in such a representation is inherently “static”, i.e., it denotes that such a relation exists between the source and target **object** but no event can refer to such a tuple to express changes to the relation. Sect. 3.1 discusses in which cases the semantics of observing **CREATE** or **DELETE** of an object implies creation and deletion of associated **object relations**. But OCED cannot express, for instance, changes to relations due to **MODIFY** events (e.g., reassigning a *[package]* object from one *[delivery]* object to another *[delivery]* object). Expressing such dynamics requires to “objectify” the relation, see Sect. 4.4.
- As an implicit triple stored as attribute at one of the objects, i.e., similar to a foreign-key attribute, an **object attribute** (both, name and value) is explicitly marked as “reference attribute” of the source object having as
  - **object attribute name** the relation type
  - **object attribute value** the identifier of the target **object** (see Sect. 3.3).

Thereby, usual principles of data serialization apply.

<sup>19</sup> While strictly speaking the relation may have a technical identifier, the relation is already fully identified by the pair of source and target node. This is due to a non-trivial and relevant detail of conceptual modeling: a conceptual model as in Fig. 1 describes a family of graphs. Concepts describe the allowed nodes, relations describe the allowed edges that may be present between two nodes in this graph, i.e., pairs of nodes. As such there cannot be two distinct edges between the same pair of nodes.

- Serializing a 1:N relation, requires the objects at the N-side to store the reference attribute to the object at the 1-side<sup>20</sup>.
- An N:M relation cannot be serialized in this way, but requires either storage of all relations as triples or reification of the relation into an object (Sect. 4.4).

**Reference Attributes vs Relations: Source of Potential Inconsistencies.** The alternative forms for serializing object relations described above is a source of potential inconsistencies in OCED instances that any implementation of OCED (both producers and consumers of OCED) have to address:

1. Source data may contain object attributes which explicitly or implicitly refer to other objects, i.e., model relations. In OCED these should either be fully expressed and serialized in the way object relations are serialized, or the object reference attribute and the relation must state the same values (i.e., there are no two conflicting descriptions of a relation).
2. Attributes not marked as relation references must not be interpreted as relation references.

**Events referring to multiple objects.** Also events can be a source of inconsistency in relations. Events referring to multiple objects may (implicitly) suggest relations between these objects that are not explicitly represented in the data. Consider the following example: “Event [*#e17 Clear Invoice*] observes object [*Payment P1*] and objects [*Invoice I1*] and [*Invoice I2*]”, i.e., payment P1 is used to clear Invoices I1 and I2. Depending on the semantics of the event and the involved objects, this may constitute a relation between P1 and I1 and between P1 and I2. The current OCED-MM Core Model does not state whether these relations also have to be described and whether any integrity constraints between object relations and objects observed together at events apply.

Future developments of OCED must provide clear conventions to producers and consumers of OCED wrt. implicit and explicitly represented relations.

#### 4.4 Relations as Objects

In the OCED-MM Core Model, only objects and events are required to be uniquely identified, allowing to describe how objects change over time (see Sect. 3.3 for the associated interpretations and design decisions and Sect. 4.1 for describing changes of objects over time).

**Changes to object relations.** As object relations do not carry identifiers, the OCED-MM Core Model is limited wrt. describing how object relations change over time.

For example, suppose an OCED instance records

<sup>20</sup> Storing a reference to N objects at the 1-side would violate the requirement that attribute values do not have complex structures in themselves

- objects  $[student\ S]$ ,  $[supervisor\ M]$ , and  $[supervisor\ D]$ ,
- an event  $e1$  observing  $[student\ S]$  with a  $[supervises]$  relation to  $[supervisor\ M]$ , and
- a second event  $e2$  observing  $[student\ S]$  with a  $[supervises]$  relation to  $[supervisor\ D]$ .

It is not possible to conclude whether the supervisor of student  $S$  changed from  $M$  to  $D$ , and that the prior relation from  $S$  to  $M$  no longer holds. It may be equally possible that  $M$  and  $D$  both supervise  $S$  but no single event observes this, or that  $M$  and  $D$  alternate supervision.

Generally, an event  $e$  observing an object  $o1$  with a relation  $R$  to an object  $o2$  only states that, at the moment of  $e$ , the relation  $R$  from  $o1$  to  $o2$  has been “observed”. No further interpretation is possible: Observing the relation  $R$  from  $o1$  to  $o2$  by event  $e$  for the first time does not imply that it has been created at that point. Neither does observing it by event  $e$  for the last time imply that it has been deleted.

**Usage pattern or explicit extension.** Reliably and unambiguously tracking object relation across multiple observations in time requires to assign identifiers to object relations. This was proposed in the OCED-MM Full Model (see Sect. A) but can also be achieved within the OCED-MM Core Model by materializing (also called *reifying*) an object relation (*source*, *target*,  $T$ ) into an object on its own. In other words, by introducing an “artificial” object  $R$  of type  $T$  that has a *from* and a *to* relation to the *source* and *target* objects, see Fig. 9. This artificial relation object now carries an identifier and can be explicitly observed by events through qualified relations.

Note that materializing a relation is a potential source of inconsistencies and ambiguities in OCED, see Sect. 4.8.

#### 4.5 Attributes as Objects

Object attributes are limited to basic types and are not required to carry an identifier.

**Complex data types.** Use cases that require to represent complex data types can do this by introducing a proxy object which in turn can have a collection of object attributes (or refer to further objects). For example, an attribute representing a numeric value and a unit can be stored as a proxy object containing solely the numeric value and the unit (as text) and then be related to the original object. Alternatively, such compound values can be represented through a collection of **object attributes** with consistent naming conventions, e.g., *price* and *price\_unit*.

**Object attribute changes.** Some use cases require expressing that an event operated on a specific `object attribute` or in which way.

In some cases, naming conventions for `object attributes` can help express the semantics of an event with respect to object attributes. For instance, if an event *[Price Change]* changed attribute *[price]* of object *[Order#17]*, then naming conventions for `event attributes` can help expressing the semantics of an event without the need for creating proxy objects, e.g., *[price\_old]* and *[price\_new]*.

However, it may be inconvenient to represent a series of multiple attribute-value changes of an object in this way as one rather wants to express that the event directly (and only) operated on the particular attribute. This either requires extending OCED, see (Sect. 4.7) or materializing attributes as objects. That is, turn a particular attribute of an object *o* into a uniquely identifiable object *A* that in itself carries the value as an attribute-value pair. Then, events can explicitly refer to *A* through different qualified relations.

**Fine-grained semantics.** Also, there may be use cases requiring to express event semantics on a more fine-grained level, e.g., when (as a result of more complex database transaction) an event *e* is, both, `READ`-ing an attribute *a* of an object *o*, `DELETE`-ing *o*, and `CREATE`-ing another object *o2* whose values depend on *a*. Also, in this case, materializing the attribute as an object *A* would allow to express the semantics of *e*.

Note that materializing an object attribute is a potential source of inconsistencies and ambiguities in OCED, see Sect. 4.8.

#### 4.6 Other Process Concepts as Objects

Further limitations of the OCED-MM Core Model (listed in Sect. 3.4) can be addressed through introducing artificial objects.

Events may be indirectly related through shared (abstract) objects, however such semantics are not prescribed by the meta-model. Examples:

- duration of an activity can be achieved through an (abstract) proxy object linking to exactly two events: start and end,
- transaction types can be represented through an (abstract) proxy object linking to multiple events (e.g., *[scheduled]*, *[started]*, *[completed]*, *[archived]*),  
or
- grouping of events can be achieved through an (abstract) proxy object linking to multiple events (i.e., groups of events).

The limitation of non-complex data types can be overcome by defining (abstract) proxy objects that are related to the main object via an object relation (e.g., `CHILD_OF`). With object attribute values of the child objects, nesting can be mimicked.

While the OCED-MM Core Model supports the creation of artificial objects and proxy objects to describe more complex structures in event data and objects,

clear conventions must be established to ensure interoperability between OCED producers and consumers. For instance, in case a proxy object  $A$  relates multiple events  $e_1, \dots, e_k$  to each other to describe a long running activity operating on several objects  $o_1, \dots, o_l$ , can each event  $e_i$  refer to any object  $o_j$ , or do all events refer to all objects, or may only  $e_1$  or  $e_k$  refer them, etc.?

#### 4.7 Strict OCED extensions

**Timestamped Attribute Values.** Events are considered atomic and have a timestamp. Therefore, event attributes are fixed. Objects may evolve over time and be involved in multiple events. Therefore, object attributes may change as illustrated in Sect. 3.2 in Fig. 2 and Fig. 3. The baseline interpretation of the OCED-MM Core Model cannot describe such changes as explained in Sect. 4.1

One possible way to handle this is to extend object attributes with a timestamp. OCEL 2.0 supports so-called *dynamic attribute values* that can change over time [5,14]. Instead of having a single, fixed value, an object attribute may have a value that changes over time. The smallest timestamp in OCEL 2.0 is 1970-01-01 00:00 UTC. This is the default time of an attribute value. Later values for the same attribute should be seen as updates. For example, if an object attribute *weight* has a value of 80 kilograms with timestamp 2023-01-01 00:00 UTC, a value of 90 kilograms with timestamp 2024-01-01 00:00 UTC, and a value of 85 kilograms with timestamp 2025-01-01 00:00 UTC, then the weight of the object is assumed to be 80 kilograms throughout the year 2023 and 90 kilograms throughout the year 2024. The timestamps may coincide with the timestamps of events, allowing for some form of event correlation. However, this is not mandatory.

Figure 7 illustrates this extension by extending the baseline interpretation of the running example in Fig. 5 with timestamped attributes values: the *PO* object now has *two object attribute values* for the object attribute *release status*. Value `[]` has been **observed at** time *2002-06-01 22:00:43.000* while value `[X]` has been **observed at** time *2002-06-03 09:10:15.000*. Together, they describe that and when the attribute value changed, which is not described in the baseline interpretation in Fig. 5. Note that while these timestamps coincide with timestamps of the event creating and releasing the *PO*, and thereby implicitly are correlated with them, this correlation is not mandatory.

**Data Schemas.** The OCED-MM Core Model describes the existence of type information for events and event attributes, and for objects and object attributes, but does not describe relations between **event type** and **event attribute** and **object type** and **object attribute**. It also does not allow to express which types of **object relations** may exist between which **object types**. Such information, typically documented in a *data schema*, informs tools about which attributes are reliably present in all **instances** of a particular event or object, allowing algorithms and statistics to draw on them. Further, object, events, and relations may in practice carry *more than one type*, for instance when expressing



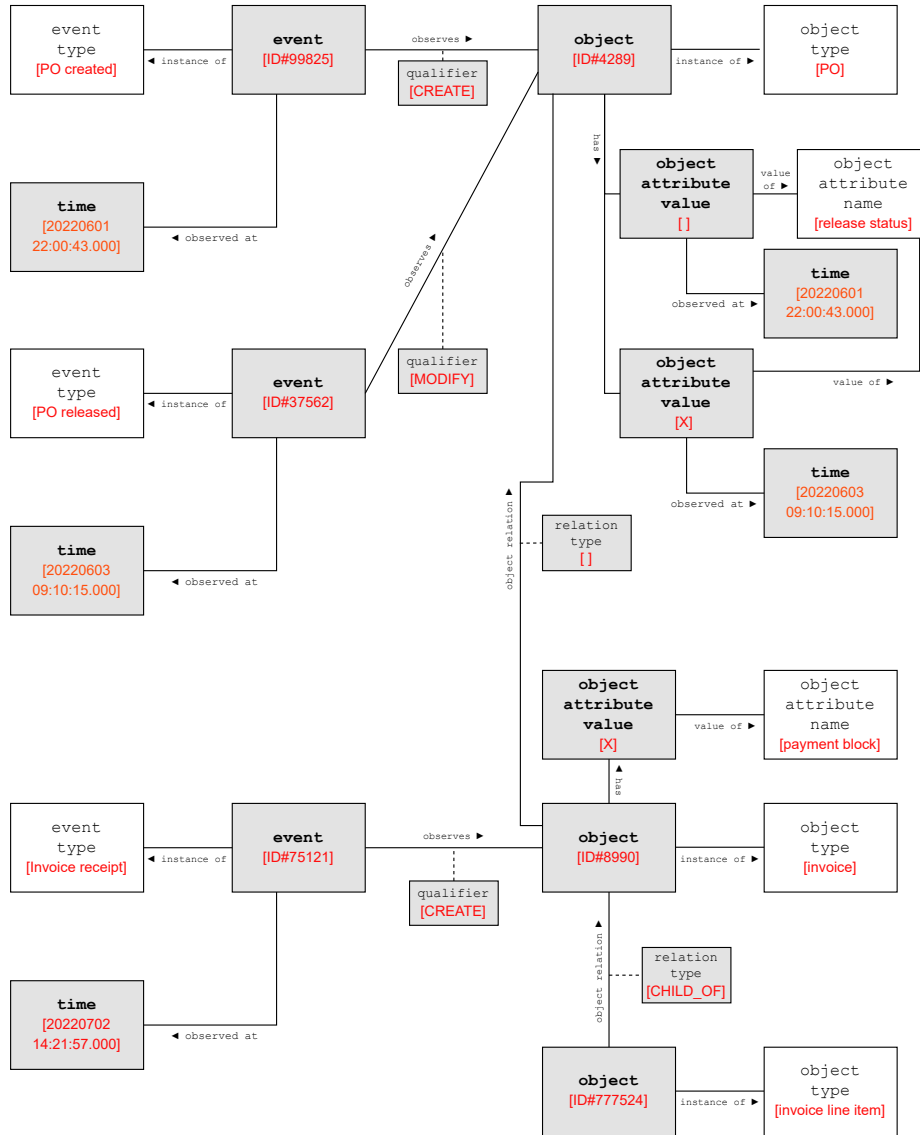


Fig. 7. Running Example with OCEL 2.0 timestamped attribute values

generalization and specialization or when describing different functions of them (e.g., a specific employee can be a *resource* as well as a *role*).

Data schemas cannot be expressed through (repurposing) existing concepts of the OCED-MM Core Model and requires a true extension. Two such extensions have been explored in OCED implementations (see Sect. 5).

- OCEL 2.0 allows for the specification of the possible **event attributes** per **event type** and the possible **object attributes** per **object type**. This is also reflected in the OCEL 2.0 meta-model.
- The *OCED-PG* implementation [18] (cf., Sect. 5.3) of OCED argues that any instance of OCED has to be accompanied by an OCED-compliant domain-specific schema of the data. This schema should specify the specific types of objects, relations, and events present in the data and how they are related to each other. Similar to *OCEL 2.0*, *OCED-PG* extends the OCED Meta-Model to also specify attributes of objects and events, but also includes relations. The extended OCED Meta-Model can then be refined into a domain-specific OCED-compliant schema. But current implementations do not support checking consistency of an OCED instance to a given schema.

#### 4.8 Cycles and Ambiguity through Extensions

Materializing relations or attributes in artificial objects introduces *cycles* in the data model that involve the **observed** relation between events and objects. Also other extensions, such as assigning timestamps to other concepts introduce cycles. These cycles are a possible source of inconsistency in representing and interpreting OCED.

**Overloading the object concept.** The preceding sections presented various use cases for a flexible interpretation of the **object** concept besides the tangible and abstract entities present in the process itself. These interpretations of objects include “snapshots” (Sect. 4.1), relations (Sect. 4.4), attributes (Sect. 4.5), and other process concepts such as activities or transactions (Sect. 4.6). Allowing these interpretations overloads the **object** concept itself. Consistent interpretation across producers and consumers requires conventions and mechanisms to clearly distinguish what exactly an object stands for.

**Materializing relations as objects.** When materializing an **object relation** as an object  $R$ , the attributes and relations of  $R$  have to be consistent with all other representations of this relation in the data which includes (changes to) reference attributes describing the relation (see Sect. 4.3). For example, assume an object  $[package \#5]$  has an **object attribute**  $[assigned-to] = [delivery \text{ tour } \#17]$  which is interpreted as a relation  $([package \#5],[delivery \text{ tour } \#17],[assigned-to])$ . To model changes to this relation, a relation object  $R$  is created representing this relation  $([package \#5],[delivery \text{ tour } \#17],[assigned-to])$ . Subsequently, an event  $e$  changes the delivery tour assigned to an object  $[package \#5]$  by updating  $R$  into  $([package \#5],[delivery \text{ tour } \#23],[assigned-to])$ .

An event  $e$  **observing**  $R$  has to be consistent with observations of all involved objects and their attributes along these changes. Note that this can be expressed in OCED, both, through the “snapshot” interpretation of objects (Sect. 4.1) as well as through timestamped attribute values of reference attributes (Sect. 4.7 and Sect. 4.3), and hence consistency considerations arise either way.

**Materializing attributes as objects.** For example, assume an event  $e$  observes object  $[Order\ 17]$  which is related to a materialized attributed object  $[Order\ 17+price]$ . Should  $e$  also have an **observes** relation to  $[Order\ 17+price]$  or is this **observes** relation implicit? Likewise, if  $e$  observes, both,  $[Order\ 17]$  and  $[Order\ 17+price]$ , do the qualifiers of both relations have to be identical or can they differ, e.g., **CREATE**  $[Order\ 17]$  and **MODIFY**  $[Order\ 17+price]$ . Extracting OCED from source systems may result in such cases.

**Assigning timestamps to other concepts.** OCEL 2.0 proposes to extend the OCED-MM Core Model to let an **object attribute value**  $v$  have a **timestamp**  $t$ . While this allows for an efficient representation (and extraction) of a series of changes to an **object attribute**  $a$ , it introduces a new, implicit representation of an **event**  $e$  that **observes** an **object attribute value**  $v$  for attribute  $a$  at time  $t$ .

Consumers of OCED with such an extension must be aware of the semantics of this extension and must agree on whether the implicit representation of an event is to be converted into an explicit representation of the event and which conventions to follow. For instance, the explicit representation of the event may require to materialize an object attribute as an artificial object itself (with all associated considerations discussed above). Further, as illustrated also in Fig. 7, timestamps of **object attribute values** may coincide with timestamps of events that **observe** the owning **object** suggesting event correlation, e.g., that the object attribute value  $[/]$  has been set by the *PO created* event that has been observed at the same time. OCED analysis techniques have to be aware of and identify such correlations, and more involved consolidations of the implicitly and explicitly represented events are required to obtain an unambiguous representation.

## 5 Initial Implementations and Lessons Learned

Following the “Call for Reference Implementations” for the OCED Meta-Model proposal, four independent implementations were presented and discussed at the OCED Symposium of the 5th International Conference on Process Mining (Rome, 2023) while a fifth implementation was recently published.

### 5.1 Konekti

Konekti is a commercial data transformation tool for process mining. It reduces the required effort for process mining, allowing practitioners to focus on higher-value tasks. Its primary focus is on document-based information systems, such

as ERP, CRM, and WMS platforms. Through our industry experience, we have identified that the most efficient method for generating case-centric event logs from document-based systems is by first constructing an object-centric data model. This model can serve as a staging layer from which to create case-centric logs or be used for object-centric analysis. Hence, Konekti supports both object-centric and case-centric analysis.

**Key Features of Konekti.** Konekti offers several key features that distinguish it from other process mining tools:

- **Guided object-centric meta-modeling** ensures that all users follow a consistent meta-model, promoting collaboration and knowledge transfer. Konekti also suggests next steps to streamline workflow.
- **Built-in data quality checks** accelerate the data validation process by automatically detecting potential issues.
- **Conversion to case-centric logs** enables users to convert the object-centric data model into case-centric logs for use in conventional process mining analysis tools.
- **Automated script generation** reduces the effort and skill required. Users follow a low-code workflow to construct the data model, after which Konekti generates an exportable PostgreSQL or Spark script.
- **Interactive object-centric data model visualization** improves comprehension of the data model and fosters collaboration among users.

**OCED-MM Implementation in Konekti.** To expedite the production of case-centric event logs, certain elements from the OCED-MM model are omitted in Konekti’s implementation:

1. **Object-relation types:** Konekti does not implement predefined relation types (such as `CHILD_OF` or `PARENT_OF`) as outlined in Section 3.1. Instead, relationships are inferred implicitly through their cardinality (1:1, 1:n, n:m), which is computed by Konekti.
2. **Event-to-object relation qualifiers:** Konekti omits the predefined qualifiers (`CREATE`, `MODIFY`, `DELETE`) described in Section 3.1 and 4.2. Instead, event qualifiers are embedded in the activity name (e.g., “create,” “modify,” or “delete”), and Konekti computes the cardinality of event-to-object relations (1:1, 1:n, n:m).

As a minor extension, event types in Konekti can have zero events instantiated, deviating from the event definition in Section 3.1.

**Lessons Learned.** Our experience of implementing Konekti has led to several insights regarding object-centric data modeling. Below, we list our two key insights:

1. **Relating event types to a single object type minimizes complexity.** Ambiguity often arises regarding which objects should be related to an event. Although Konekti allows linking an event to multiple object types, we recommend associating each event type with a single object type. For example, in this example introduced in Section 4.3:

“Event #e17 [Clear Invoice] observes object Payment P1 and objects [Invoice I1] and [Invoice I2], where payment P1 is used to clear invoices I1 and I2.”

We advise to initially link Event #e17 only to Payment P1 and model the relationship between Payment P1 and the two invoices (I1 and I2) through an object relation. If required for object-centric analysis, users can later add additional event-object relations. This way of working reduces the model complexity.

2. **Ambiguity in the definition of objects.** In Section 3.1, an object is defined as a uniquely identifiable entity observed in the world. However, this definition can be blurred due to several factors:
  - (a) **Objects as attributes:** Not every identifiable entity needs to be modeled as an object. For instance, while “Customer” could be an object, it might also be modeled as attributes (e.g., CustomerId, CustomerName) of the “Order” object. The decision on whether to treat an entity as an object or an attribute is context-dependent and determined by the data modeler.
  - (b) **Hierarchical structures in objects:** The level of granularity for modeling objects can vary. A user could model one object “Financial document” or decide to split them into multiple objects (e.g., “Invoices”, “Credit memos”, etc.). The best choice depends on the context, making it difficult to establish a universal best practice.
  - (c) **Attributes as proxy objects:** In some cases, attributes can be modeled as objects. For example, while “Order Status” might seem better suited as an attribute, it can be modeled as a separate object, aligning with systems like SAP ECC, where status information is stored in separate tables (e.g., table VBUK for sales document statuses).
  - (d) **Relations as proxy objects:** Konekti expects relationships between objects to be modeled through object tables (using primary key-foreign key combinations). If relations are stored in separate tables, this relation needs to be modeled as an object (e.g., “Sales Document Flow”), even though such tables are not real-world entities.
  - (e) **Grouping activities via proxy objects:** As described in Section 4.6, proxy objects can be used to group activities logically within the data model.

The inherent ambiguity in defining objects contributes to variability in how process data models are constructed, potentially complicating the understanding, modeling, and validation processes.

**Conclusion.** In short, Konekti streamlines the process of generating object-centric and case-centric event logs using an OCED-MM implementation. While

challenges remain in consistently relating events to objects and defining objects clearly, our experience with Konekti demonstrates that the model is effective and practical in real-world applications.

## 5.2 OpenOCED

*OpenOCED* is an open source reference implementation using a Model Driven Engineering (MDE) perspective of the *OCED-MM Core Model* of Sect. 3. The MDE approach and first Java library implementation of the OCED base proposal from March 2023 by the IEEE Task Force on Process Mining was presented in [8]. To represent OCED models compliant with the Ecore meta-model the XML Metadata Interchange (XMI) standard (<https://www.omg.org/spec/XMI/>) format is used, which can be transformed to JSON and XML OCEL 2.0 files, as well as CSV files. The *OCED-MM Full Model* of App. A is also defined as an extension of the base meta-model using the EMF standard extension mechanism.

**Features.** The OpenOCED reference implementation provides a Python and a Java library that includes:

- Ecore-based definition of the OCED meta-model.
- Import/export from/to XMI files of OCED model instances.
- Import/export from/to CSV files of OCED model instances.
- OCED model transformation to OCEL 2.0 (JSON and XML format).
- OCEL 2.0 (JSON and XML format) model transformation to OCED.

**OCED-MM implementation in OpenOCED.** The OpenOCED meta-model directly represents the original object-centric concepts and expresses relations through qualified associations represented as pivot elements (e.g., event-object, object-object) to store its qualifier. The implementation does not enforce predefined qualifiers for the relations and allow navigation from the object-object relation to its source and target objects and from the object to the relations in which the object participates. Input files can be in XMI/CSV/OCEL 2.0 JSON and XML formats, the Java and Python code automatically generated from the meta-model allows in-memory manipulation of OCED models, which in turn can be output in XMI/CSV/OCEL 2.0 JSON and XML formats. Initial steps have also been taken to extend the interoperability with other OCED implementations, e.g., input/output to OCED-PG.

OpenOCED libraries can be integrated into several Java or Python software to serve as an exchange format for existing Object-Centric Process Mining (OCPM) [1] techniques and tools to support the manipulation of OCED models and/or to implement new algorithms for the OCPM perspectives. Also, with the MDE meta-model extension mechanism, the OCED-MM Core Model and Full Model can be extended to support other approaches directly, for which the associated code can be automatically generated.

**Lessons Learned.** As lessons learned, a key aspect to consider when manipulating OCED models is information loss when extracting data and representing relationships between elements or exchanging different representations/implementations based on mappings. Also, to better exploit object relationships, e.g., in the context of data refinement (e.g., domain-specific filters applied to the model) and conformance checking (e.g., adding domain rules), the provision or extension of tools is needed. Current and future work includes applying OpenOCED to case studies with actual data to further deal with these key aspects, and extend the capabilities of the libraries to support more functionalities.

**Resources.** Open OCED code and libraries for Python and Java are available at (<https://open-coal.pages.fing.edu.uy/oced/>), as well as the meta-model (Ecore, PyEcore) definition and examples in .XMI, CSV, and OCEL 2.0 JSON and XML format from (<https://ocel-standard.org/event-logs/overview/>).

### 5.3 Event Knowledge Graphs (OCED-PG)

*OCED-PG* [18] is an open-source library that uses Labeled Property Graphs (LPGs) to represent and store OCED in a (Neo4j) graph database. OCED-PG specifically addresses the use case of transforming (arbitrary) source data into an OCED store without an intermediate OCED-specific data exchange format, and querying OCED using mature graph query languages, enabling object-centric process mining using graph databases.

**Features.** OCED-PG formalizes the OCED meta-model using *PG-schema*, the schema definition language for property graphs. Thus, the PG-schema formalization of OCED defines the concepts for storing and querying OCED in a graph database, enabling the following features:

- **Refining the OCED data schema into a domain-specific OCED data schema**, which allows to explicitly document the object types, relation types, and event types present in the data.
- **Pattern-based transformation rules**, which allow describing a mapping from raw source data (in tabular format) into the domain-specific OCED data schema, the so-called *semantic header*.
- **A fully automated pipeline** for loading and transforming raw source data into OCED.
- **Querying OCED using domain-specific concepts** by using the graph database’s native query language over a domain-specific data schema.

**OCED-MM implementation in OCED-PG.** OCED-PG implements the *OCED-MM Core Model* of Sect. 3 and the *OCED-MM Full Model* of App. A.

The OCED core model interpretation is inherent in the model of *Event Knowledge Graphs* (EKG) [11] on which OCED-PG is based. Refining the EKG concepts allows to express OCED-MM Core Model extensions, which are currently implemented for the OCED-MM Full Model. The current semantic header allows creation of OCED-MM Full Model instances with some limitations: (1) All `observes` relations from one event receive the same qualifier (i.e., cannot distinguish different qualifiers originating from the same event). (2) Generating `observes` relations to `object attribute values` or `object relations` uses relation inference from existing relations (via the `object` associated to the attribute or relation). While more involved, the latter does ensure semantic consistency between relations, see Sect. 4.8.

**Lessons Learned.** OCED-PG is based on lessons learned in a series of industrial and academic case studies for extracting and modeling object-centric event data using graph databases and subsequent process mining analyses.

1. **Domain-specific schemata:** Industrial use cases benefited from the development of a domain-specific data schema in which the OCED core concepts of events, objects, and their relations are refined into domain concepts (e.g., distinguishing object types, relation types, and groups of event types). [9,10,7,16,15]
2. **Structuring Schema-less Source Data:** Raw source data is often schema-less, or available schema information is not aware of events. Extracting OCED requires to first bring structure to the source data by specifying which attributes constitute information for which OCED concept, e.g., a record with a timestamp and an activity allows extracting an event.
3. **Disentangling Events and Objects:** Source records are not inherently events or objects but may contain attributes describing multiple concepts (events, different objects and attributes). Extraction of OCED requires methods to disentangle them.
4. **Some relationships are implicit:** Relationships can be established based on the co-occurrence of two elements (events and/or objects) within a record, using provenance as a basis (see Sect. 4.3). Additionally, relationships can be inferred from existing relationships. For example, if a member has borrowed a book listed in a library catalog, it implies that the member is a member of the library.
5. **Object and relation snapshots:** In industrial configuration management, source data includes snapshots of data objects and relations (describing the creation and evolution of configuration management data structures), requiring a “snapshot” interpretation of objects and relations (see Sect. 4.1) and clear qualifier semantics (see Sect. 4.2)<sup>21</sup>. The results suggest it is beneficial to generalize timestamped attribute values of Sect. 4.7 to entire “snapshots”, i.e., time-stamping objects and relations representing their state. [15]

<sup>21</sup> <https://www.linkedin.com/pulse/boost-your-knowledge-graph-events-gain-untapped-martijn-dullaart-wunse/>



6. **Materializing implicit structures:** Querying and analyzing OCED benefits from explicitly materializing concepts and relations that otherwise are encoded implicitly in event or object attributes. For instance, object states (i.e., “snapshots”, see Sect. 4.1) [7,9], context knowledge such as connection and layout of objects describing components supporting a physical process in manufacturing or logistics [9,10,7,16], or temporal and causal relations [11,13]. While these should not be stored in OCED for transport, it requires agreement on the implicit semantics of event, objects, and relations (see Sect. 4.2) for reliably constructing them.
7. **Circular processes:** Use (or inference) of a consistent object identifier enables OCED to describe and track objects in circular processes where objects repeatedly return to the process over extended periods of time. [9]

**Resources.** OCED-PG is implemented as part of the open-source process mining library PromG (<https://github.com/promg-dev>) which provides functionality to develop process mining analyses over OCED in graph databases [19]. Schemas and transformation rules using OCED-PG are available for five public real-life datasets and one educational example [17]. Further, a JSON-based OCED export from Konekti (Sect. 5.1) can be imported directly into OCED-PG as an EKG without any data transformation, see <https://github.com/PromG-dev/promg-konekti>.

#### 5.4 Object-Centric Event Log (OCEL 2.0)

OCEL, which stands for *Object-Centric Event Log*, serves as the exchange format for Object-Centric Event Data (OCED) and is the foundation for a range of Object-Centric Process Mining (OCPM) techniques and over ten process mining tools and libraries [2,5,14]. OCEL 2.0 was released in 2023, extending OCEL 1.0 (released in 2020). Appendix B discusses the OCEL 2.0 meta-model (see Figure 10).

**OCEL 2.0 Storage Formats.** There are different storage formats for OCEL 2.0: XML, JSON, and SQL. The detailed specifications for these formats can be found on the OCEL website: <https://www.ocel-standard.org/>. Also, several event logs are provided in all three formats. There are also publicly available tools to check the validity of an OCEL 2.0 dataset, e.g., an XML Schema, a JSON schema, and an SQL validator.

**Lessons Learned.** The most important lesson learned is to avoid adding concepts for which there are no analysis techniques or that allow for multiple ways of representing the same information. For example, there is already a trade-off between representing event-to-object relations (e.g., **observes**) and object-to-object relations. An order may have multiple items (object-to-object), and an event of type “place order” may include the order and its items (event-to-object). Further extending the meta-model will lead to even more trade-offs, e.g.,

the same information can be represented in many different, possibly redundant, ways. We first need guidelines for using the existing concepts before adding new ones.

In the experiments and case studies with OCEL and dozens of process mining implementations using the Process Intelligence Graph (PIG) of Celonis (which uses a meta-model similar to OCEL 2.0 to store events and objects), we noted that object-to-object relations are mostly used for filtering and querying, and event-to-object relations are mostly used for process discovery and conformance checking. Due to the similarity, it is possible to load OCEL 2.0 into the Celonis ecosystem without any problems.

These experiences suggest that, at this stage, there is no need for additional concepts. XES also suffered from the problem that few of the extensions were actually being used. Therefore, it is much more important to support existing concepts well and develop powerful process mining techniques (process discovery, conformance checking, predictive analytics) using both event-to-object and object-to-object relations. Concepts should only be added if there are mature analysis techniques exploiting these.

### Process Mining Tools and Libraries Supporting OCEL 2.0.

- OCEL 2.0 *validators* using XML Schema, JSON schema, and an SQL checker, accessible via <https://www.ocel-standard.org/>.
- The web-based event log inspector for OCEL *Ocelot*, accessible via <https://ocelot.pm/>.
- The *OCPM* (Object-Centric Process Mining) tool, accessible via <https://www.ocpm.info/>. *OCPM* supports object-centric process discovery, object-centric conformance checking, and object-centric machine learning (using DFGs and Petri nets).
- The *Process Mining for Javascript* (PM4JS) implementation (see <https://www.pm4js.org/>).
- The *OCELStandard* plugin for the *ProM* framework (<https://promtools.org/>).
- The *OCPA library* supporting object-centric process discovery, object-centric conformance checking, object-centric process enhancement, and object-centric process monitoring, accessible via <https://ocpa.readthedocs.io/>.
- The *Object-Centric Process Insights* (OC $\pi$ ) tool using OCPA. OC $\pi$  supports object-centric process discovery and filtering, and provides elaborate support for object-centric variants. Download from <https://ocpi.ai/>.
- Connectors for Celonis, SAP, and Oracle (see <https://www.ocel-standard.org/>).
- The *Object-Centric Process Querying* (OCPQ) tool to query OCEL 2.0 datasets (supporting the JSON, XML, and SQL formats), see <https://ocpq.aarkue.eu>.
- *Process Mining for Python* (PM4Py) fully supports OCEL 2.0, including object-centric process discovery and object-centric conformance checking (using Petri nets, DFGs, and object graphs), accessible via <https://pr>

`ocessintelligence.solutions/pm4py` and <https://github.com/pm4py/pm4py-core>.

The list of tools and libraries illustrates the interoperability achieved by adopting OCEL 2.0.

## 5.5 Stack't

Stack't is an open-source data transformation tool designed to support data preparation for object-centric process mining in a dynamic context, i.e., allowing for continuously adding new data while accommodating changes in the process data architecture such as new object types, event types, or attributes [6]. The tool's development is driven by practical data-engineering considerations, drawing from industry experience of continuously extracting data from source systems, such as ERP and MES databases, for various analytical purposes.

**Features.** Similar to Konekti, described in Section 5.1, we opt for a staggered approach by first mapping the process data to an intermediate data store (hub) and providing extractors to generate object-centric event logs in various formats from this. Below is an overview of the current core capabilities of the Stack't tool.

- **Continuous Ingestion:** Supports append-only incremental batch processing of process data.
- **Interactive Visuals:** Provides interactive visualizations for exploratory data exploration
- **Export Capabilities:** Allows exporting to OCEL 2.0, DOCEL, and Neo4j graph database formats.
- **Import from OCEL 2.0:** Includes functionality to import event logs formatted according to OCEL 2.0.

Stack't is a modular and flexible data stack (in our implementation using DuckDB and dbt) in a Docker container, and can be integrated into existing data architectures. More information, together with the source code, can be found in the GitHub repository <https://github.com/LienBosmans/stack-t>.

**OCED-MM Changes.** As described in [6], Stack't adopts a flexible meta-model to extend the longevity of the data store. The implementation supports the OCED-MM Core Model, but also the OCED-MM Full Model and OCEL 2.0. The flexibility manifests itself in two changes: object-to-object relationships are allowed to change over time by attaching a timestamp to the relationship qualifier, and direct relationships between events and object attribute value updates are used to store any known causal relationships between them and thus support many-to-many relationships. Storing process data that strictly adheres to the core model can be achieved by imposing restrictions, which are defined as additional data quality tests in the code:

1. The table storing event-to-object-attribute-value relationships must be empty.
2. The timestamp columns for object attribute values and object-to-object relationships must only contain NULL values.
3. Relationships and attribute values must be uniquely identifiable using their foreign keys. The existing primary key column is kept, but must be supplemented with an additional check to ensure this is the case.

### Lessons Learned.

1. **Lack of publicly available real-life source data:** The absence of publicly available source data hinders the ability to validate the correctness of a source-to-target mapping implementation and assess the effect of several decisions regarding the conversion of raw data into an event log on aspects such as performance.
2. **Scalability and Maintainability in Dynamic Environments:** Designing data storage solutions for processes in dynamic or agile environments requires extra care. For example: even minor modifications such as renaming a type or attribute, can cascade into code-level impacts if not anticipated. More of these considerations, and how they are tackled by the proposed relational schema can be found in [6].
3. **Dynamic Object-to-Object Relations:** The OCED-MM Core Model does not allow object-to-object relationships to evolve over time. Although visualizing such changes is complex, it remains essential for certain use cases. Consider an organizational chart for example: roles within a team might change. When, e.g., retroactively checking for irregularities, it would be important to know which relationship existed at certain points in time. A potential solution could be allowing dynamic relations between objects (see Sect. 4.4), but formulating guidelines on how additional restrictions such as static object-to-object relations should be handled when algorithms cannot handle dynamic ones as input.
4. **Standards and Definitions:** The lack of clear definitions for object-centric event (log) standards creates challenges, especially for those unfamiliar with the academic field. This was mainly apparent when trying to write connectors that can transform data from one format into another, as automating this requires the data to adhere to an unambiguous definition, and therefore knowing table and column names, as well as their data types, in advance (and for them to remain consistent over time).
5. **Data Anonymization:** We have tried to take into account the possible need for data anonymization with Stack't. While hiding the descriptions of types and attributes is rather straightforward, there is no current solution for masking sensitive information revealed through relationships.

### 5.6 Further Known Implementations

Further implementations of OCED are under development.

- *OCEDO* — a semantic-web-based ontology for core OCED defining the OCEDO namespace at <https://semsys.ai.wu.ac.at/ocedo/core> and algorithms for converting flat event logs into OCED via semantic technology <https://gitlab.isis.tuwien.ac.at/Ekaputra/ocedo>

## 6 Conclusion

The current consensus and state of discussion on identifying an object-centric event data format that can succeed XES can be summarized as follows.

1. **Core concepts.** We identified the core concepts needed to represent object-centric event data that emerged from the intersection of multiple prior proposals and use cases. These core concepts are described in the OCED-MM Core Model in Sect. 3.
2. **Baseline interpretation and five implementations.** We provide a baseline interpretation for these core concepts in Sect. 3.3 that is shared by all five existing OCED implementations, demonstrating basic viability of object-centric event data exchange.

Subsequently, we (1) outline concrete steps that can be undertaken with the current OCED-MM Core Model and existing implementations, (2) summarize the challenges in realizing interoperability with OCED that need to be considered, and (3) outline general considerations towards standardizing OCED.

**Reliable basis for research and development.** While the OCED-MM Core Model does not yet cover all practical requirements for data exchange in all use cases, and thus requires further clarification and (careful, limited) extension, it is adequate for the time being. Specifically, the model and the lessons learned documented in Sections 4 and 5 provide a reliable basis for the process mining community to engage with OCED and related use cases, which include:

- Creating more object-centric event data sets to serve as realistic examples. This specifically includes creating or providing raw source data sets for conversion into OCED.
- Making sure that existing and future implementations (see Sect. 5) can be connected, e.g., creating bridges and/or import/export functionality between OCEL 2.0 and OCED-PG and ensuring data imports and exports of different tools are compatible/compliant with each other.
- Creating more OCED extractors for various source systems.
- Identifying, documenting, and sharing process mining use cases that benefit from the analysis of object-centric event data.
- Researching, developing, and sharing process mining algorithms that consume object-centric event data to address OCED-specific use cases.
- Encouraging vendors to support OCED and object-centric process mining and educating students and users.

These results of these efforts will provide relevant information for a community-wide adoption of OCED and its standardization.

**Challenges in realizing interoperability with OCEDs.** The diversity of lessons learned in the five independent implementations (see Sect. 5) reveals the limitations of the OCED-MM Core Model and the different design decisions made in extending or using OCED in particular ways. Current and further developments of OCED should be aware of the ambiguities in the OCED-MM Core Model and the implications of either design decision.

1. While the description of the OCED-MM Core Model allows for a reasonably unambiguous interpretation of each individual OCED concept, the description is *not specific enough* for interpreting all *combinations of concepts* unambiguously, e.g., how are multiple observations of an object represented (see Sect. 4.1), standardizing *qualifiers* to provide specific interpretation (see Sect. 4.2), and can object attributes be used to represent relations between objects (see Sect. 4.3). This semantic ambiguity needs to be resolved to ensure that independent implementations of OCED can produce and consume object-centric event data with the same interpretation.
2. The minimal concepts in OCED-MM Core Model do not support a number of use cases (see Sect. 3.4). This requires either re-purposing existing concepts for additional use cases such as introducing artificial objects for relations, attributes, and other advanced concepts (see Sect. 4.4, 4.5, 4.6) or extending the OCED-MM with additional concepts (see Sect. 4.7). Either form of extension introduces additional ambiguity in interpreting combinations of OCED concepts that need to be resolved (see Sect. 4.8), or deliberately be left out of scope of an OCED standard and deferred to later evolution of the standard or best practices as use cases mature.
3. All mentioned ambiguities and limitations presented in this document can be resolved by taking design decisions wrt. representation and interpretation of the various OCED concepts. The existing OCED implementations presented in Sect. 5 have done so, also including first steps in achieving interoperability (though with notable limitations).

**Towards a community standard.** As the current implementations of OCED differ in the scope and interpretation of OCED, there currently does not exist an *eco-system* for producing and consuming OCED.

The exhaustive exploration and discussion of OCED over the previous three years suggests that the overall space of OCED concepts, ambiguities, and design decisions for interpreting them is understood. This document can be understood as an inventory of all ambiguities and open design decisions for OCED. This suggests that the next step for realizing a community standard for OCED is to jointly review the known ambiguities design decisions and to form a community consensus of how to represent and interpret OCED concepts across a variety of use cases.

Forming consensus for representing and interpreting OCED around use cases thereby fundamentally requires considering the full life-cycle of object-centric event data. As each stage in the life-cycle has different constraints wrt. how data

can be represented and processed, OCED may have to explicitly acknowledge different levels of consistency<sup>22</sup>. For instance:

1. *Storage in source systems* (from which OCED is to be extracted) is optimized for usage, often lacking explicit, complete representations of events or objects, and lossy. Different types of events or objects may have fundamentally different storage representations.
2. *Extracting and providing OCED from source systems* requires conversion. Thereby some conversions may have prohibitive performance costs on the source system or lead to excessive data, e.g., translating an SAP ERP Change Table with 100 million records into events observing a modification of an object attribute.
3. Other conversions may depend on a record's context, e.g., require to consider multiple records and additional domain knowledge, to provide a less ambiguous interpretation of an event or an object. This could be considered as a form of *enrichment of OCED* that may depend on the *analysis objectives*.
4. Certain forms of resolving ambiguity in OCED interpretation may no longer be data (local) conversion tasks but *genuine process mining tasks* providing use case-specific interpretations of the data, e.g., mining the Create-Read-Update-Delete life-cycle of a relation between objects or the identification of higher-level tasks from lower-level events.
5. Finally, *Process Mining solutions* working with OCED also have to realize OCED-compliant data structures and data stores that provide a standardized representation and interpretation of event data for all analysis and mining algorithms. These internal data stores may have higher requirements on how explicit various concepts and relations are represented and their consistency.

The above non-exhaustive list illustrates a *spectrum of different levels of "strictness"* regarding representing and interpreting OCED that all originate from different constraints and requirements for the respective task and objective.

Subsequent steps in standardizing OCED should explicitly consider these diverse levels of requirements and invite industry practitioners, vendors, and academic experts to jointly review design decisions wrt. supporting relevant use cases along the various stages of the OCED life-cycle.

The above considerations suggest that a community-supported OCED standard may have to explicitly support a (well-defined) spectrum of representations and interpretations of OCED along the data life-cycle. One possibility is to formulate well-defined *levels* of OCED consistency and completeness. This could entail:

- Agreeing on the degree of inconsistency, i.e., kinds of ambiguity or incompleteness in representation and consistency, is allowed at a particular level

<sup>22</sup> <https://multiprocessmining.org/2022/10/26/data-storage-vs-data-semantics-for-object-centric-event-data/>

- (e.g., timestamped attribute values extracted from source system vs a minimum standard of unique representation of events and associated objects for process mining algorithms), and
- identifying core principles for increasing the level of consistency and completeness of OCED by conversion, conventions, or extensions and the associated decisions in representation and interpretation of OCED.

**Acknowledgments.** Thanks to the PADS PhDs and Postdocs that co-developed OCEL 2.0 and related tools, in particular Alessandro Berti, Istvan Koren, Niklas Adams, Nina Graves, Gyunam Park, Benedikt Knopp, Marco Pegoraro, Lukas Liß, Leah Tacke genannt Unterberg, Christopher Schwanen, Aaron Küsters, Dina Kretzschmann, and Viki Peeva.

Thanks to the team from the Inco,FING,UdelaR (Uruguay) that participated in the implementation of OpenOCED related tools: Carolina Cortés, José Pedro De León, Maximiliano Jara and Camilo López.

Thanks to the Master students and PhD students at TU Eindhoven who contributed to OCED-PG and the case studies enabling it, in particular Stefan Esser, Ava Swevels, Eva Klijn, Maren Buermann, Vi Chu, Adam Broniewski, and Kadir Marangoz, as well as Francesca Zerbato for her feedback.

The contributions by TU Eindhoven are partially supported by AutoTwin EU GA n. 101092021.

## References

1. van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) *Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings. Lecture Notes in Computer Science*, vol. 11724, pp. 3–25. Springer (2019). [https://doi.org/10.1007/978-3-030-30446-1\\_1](https://doi.org/10.1007/978-3-030-30446-1_1), [https://doi.org/10.1007/978-3-030-30446-1\\_1](https://doi.org/10.1007/978-3-030-30446-1_1)
2. van der Aalst, W.M.P.: Object-centric process mining: Unraveling the fabric of real processes. *Mathematics* **11**(12) (2023). <https://doi.org/10.3390/math11122691>, <https://www.mdpi.com/2227-7390/11/12/2691>
3. van der Aalst, W.M.P., Berti, A.: Discovering Object-Centric Petri Nets. *Fundamenta Informaticae* **175**(1-4), 1–40 (2020)
4. Acampora, G., Vitiello, A., Stefano, B.N.D., van der Aalst, W.M.P., Günther, C.W., Verbeek, H.M.W.: IEEE 1849: The XES standard: The second IEEE standard sponsored by IEEE computational intelligence society [society briefs]. *IEEE Comput. Intell. Mag.* **12**(2), 4–8 (2017). <https://doi.org/10.1109/MCI.2017.2670420>, <https://doi.org/10.1109/MCI.2017.2670420>
5. Berti, A., Koren, I., Adams, J.N., Park, G., Knopp, B., Graves, N., Rafiei, M., Liß, L., genannt Unterberg, L.T., Zhang, Y., Schwanen, C.T., Pegoraro, M., van der Aalst, W.M.P.: OCEL (object-centric event log) 2.0 specification. *CoRR abs/2403.01975* (2024). <https://doi.org/10.48550/ARXIV.2403.01975>, <https://doi.org/10.48550/arXiv.2403.01975>



6. Bosmans, L., Peeperkorn, J., Goossens, A., Lugaresi, G., Smedt, J.D., Weerdt, J.D.: Dynamic and scalable data preparation for object-centric process mining (2024), <https://arxiv.org/abs/2410.00596>
7. Broniewski, A.: Building a digital asset: An event knowledge graph approach for integrating data and persisting object-centric process mining analysis in baggage handling systems (2023)
8. Calegari, D., Delgado, A.: A model-driven engineering perspective for the object-centric event data (OCED) metamodel. In: Weerdt, J.D., Pufahl, L. (eds.) Business Process Management Workshops - BPM 2023 International Workshops, Utrecht, The Netherlands, September 11-15, 2023, Revised Selected Papers. Lecture Notes in Business Information Processing, vol. 492, pp. 508–520. Springer (2023). [https://doi.org/10.1007/978-3-031-50974-2\\_38](https://doi.org/10.1007/978-3-031-50974-2_38), [https://doi.org/10.1007/978-3-031-50974-2\\_38](https://doi.org/10.1007/978-3-031-50974-2_38)
9. Chu, V.: Using event knowledge graphs to model multi-dimensional dynamics in a baggage handling system (2022)
10. Cuprinsu, N.: Extending knowledge graphs to predict system and item behavior (2022)
11. Esser, S., Fahland, D.: Multi-dimensional event data in graph databases. *J. Data Semant.* **10**, 109–141 (2021)
12. Fahland, D.: Artifact-centric process mining. In: Sakr, S., Zomaya, A.Y. (eds.) *Encyclopedia of Big Data Technologies*. Springer (2019). [https://doi.org/10.1007/978-3-319-63962-8\\_93-1](https://doi.org/10.1007/978-3-319-63962-8_93-1), [https://doi.org/10.1007/978-3-319-63962-8\\_93-1](https://doi.org/10.1007/978-3-319-63962-8_93-1)
13. Khayatbashi, S., Hartig, O., Jalali, A.: Transforming event knowledge graph to object-centric event logs: A comparative study for multi-dimensional process analysis. In: Almeida, J.P.A., Borbinha, J., Guizzardi, G., Link, S., Zdravkovic, J. (eds.) *Conceptual Modeling - 42nd International Conference, ER 2023, Lisbon, Portugal, November 6-9, 2023, Proceedings*. Lecture Notes in Computer Science, vol. 14320, pp. 220–238. Springer (2023). [https://doi.org/10.1007/978-3-031-47262-6\\_12](https://doi.org/10.1007/978-3-031-47262-6_12), [https://doi.org/10.1007/978-3-031-47262-6\\_12](https://doi.org/10.1007/978-3-031-47262-6_12)
14. Koren, I., Adams, J.N., Berti, A., van der Aalst, W.M.P.: OCEL 2.0 resources - [www.ocel-standard.org](http://www.ocel-standard.org). In: van der Werf, J.M.E.M., Cabanillas, C., Leotta, F., Genga, L. (eds.) *Doctoral Consortium and Demo Track 2023 at the International Conference on Process Mining 2023 co-located with the 5th International Conference on Process Mining (ICPM 2023), Rome, Italy, October 27, 2023. CEUR Workshop Proceedings*, vol. 3648. CEUR-WS.org (2023), [https://ceur-ws.org/Vol-3648/paper\\_7195.pdf](https://ceur-ws.org/Vol-3648/paper_7195.pdf)
15. Marangoz, K.: Capturing multi-dimensional dynamics in a configuration management process through event knowledge graphs (2023)
16. Swevels, A., Dijkman, R.M., Fahland, D.: Inferring missing entity identifiers from context using event knowledge graphs. In: Francescomarino, C.D., Burattin, A., Janiesch, C., Sadiq, S. (eds.) *Business Process Management - 21st International Conference, BPM 2023, Utrecht, The Netherlands, September 11-15, 2023, Proceedings*. Lecture Notes in Computer Science, vol. 14159, pp. 180–197. Springer (2023). [https://doi.org/10.1007/978-3-031-41620-0\\_11](https://doi.org/10.1007/978-3-031-41620-0_11), [https://doi.org/10.1007/978-3-031-41620-0\\_11](https://doi.org/10.1007/978-3-031-41620-0_11)
17. Swevels, A., Fahland, D.: Event data and semantic header for oced-pg (Aug 2023). <https://doi.org/10.5281/zenodo.8296559>, <https://doi.org/10.5281/zenodo.8296559>

18. Swevels, A., Fahland, D., Montali, M.: Implementing object-centric event data models in event knowledge graphs. In: Smedt, J.D., Soffer, P. (eds.) *Process Mining Workshops - ICPM 2023 International Workshops*, Rome, Italy, October 23-27, 2023, Revised Selected Papers. *Lecture Notes in Business Information Processing*, vol. 503, pp. 431–443. Springer (2023). [https://doi.org/10.1007/978-3-031-56107-8\\_33](https://doi.org/10.1007/978-3-031-56107-8_33)
19. Swevels, A., Klijn, E.L., Fahland, D.: Object-centric process mining (and more) using a graph-based approach with promg. In: van der Werf, J.M.E.M., Cabanillas, C., Leotta, F., Genga, L. (eds.) *Doctoral Consortium and Demo Track 2023 at the International Conference on Process Mining 2023 co-located with the 5th International Conference on Process Mining (ICPM 2023)*, Rome, Italy, October 27, 2023. *CEUR Workshop Proceedings*, vol. 3648. CEUR-WS.org (2023), [https://ceur-ws.org/Vol-3648/paper\\_9922.pdf](https://ceur-ws.org/Vol-3648/paper_9922.pdf)
20. Verbeek, H.M.W., Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Xes, xesame, and prom 6. In: Soffer, P., Proper, E. (eds.) *Information Systems Evolution - CAiSE Forum 2010*, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers. *Lecture Notes in Business Information Processing*, vol. 72, pp. 60–75. Springer (2010). [https://doi.org/10.1007/978-3-642-17722-4\\_5](https://doi.org/10.1007/978-3-642-17722-4_5)
21. Wynn, M.T., van der Aalst, W.M.P., Verbeek, H.M.W., Stefano, B.N.D.: The IEEE XES standard for process mining: Experiences, adoption, and revision [society briefs]. *IEEE Comput. Intell. Mag.* **19**(1), 20–23 (2024). <https://doi.org/10.1109/MCI.2023.3333141>, <https://doi.org/10.1109/MCI.2023.3333141>
22. Wynn, M.T., Lebherz, J., van der Aalst, W.M.P., Accorsi, R., Ciccio, C.D., Jayarathna, L., Verbeek, H.M.W.: Rethinking the input for process mining: Insights from the XES survey and workshop. In: Munoz-Gama, J., Lu, X. (eds.) *Process Mining Workshops - ICPM 2021 International Workshops*, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers. *Lecture Notes in Business Information Processing*, vol. 433, pp. 3–16. Springer (2021). [https://doi.org/10.1007/978-3-030-98581-3\\_1](https://doi.org/10.1007/978-3-030-98581-3_1), [https://doi.org/10.1007/978-3-030-98581-3\\_1](https://doi.org/10.1007/978-3-030-98581-3_1)

## A Appendix: Original OCED-MM Base Model and OCED-MM Full Model

Before converging on the OCED-MM Core Model described in Sect. 3, the OCED working group developed and published two predecessors: OCED-MM Based Model and OCED-MM Core Model. We include these here for reference for context of the discussion of limitations and extensions of the OCED-MM Core Model.

### A.1 Original OCED-MM Base Model

The original OCED-MM Base Model developed by the OCED working group is shown in Fig. 8.

It differs from the OCED-MM Core Model described in Sect. 3 by modeling an **object relation** as a distinct identifiable entity of an **object relation**

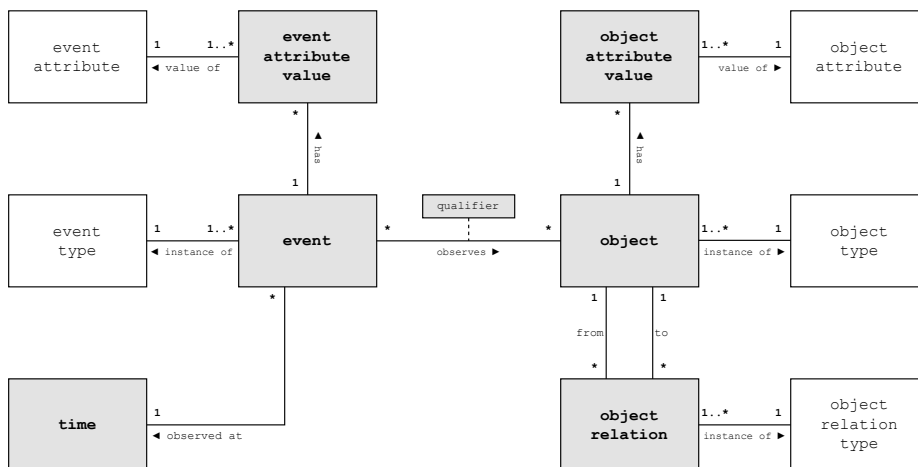


Fig. 8. Original OCED-MM Base Model (published 22nd August 2022)

type and explicit **from** and **to** relationships referring to the objects that are related to each other.

Section 4.4 discusses the difference between these two choices of modeling object relations and how the OCED-MM Core Model of Sect. 3 can be extended to express object relations as distinct entities.

## A.2 Original OCED-MM Full Model

The Full OCED-MM Model developed by the OCED working group is shown in Fig. 8.

It further extends the OCED-MM Base Model by providing additional *qualified observes* relationships from *events* to, both, *object attribute values* and *object relations*.

1. **event observes object attribute** — An **event** and an **object attribute value** can be related in a qualified (i.e., association class) manner, meaning their type of relationship is denoted. While a minimum set of qualifiers is predefined (**CREATE**, **MODIFY** and **DELETE**), additional qualifiers can be introduced as part of the data capture and used to reflect the semantics of the relationship. Each **object attribute value** can be involved in an arbitrary number of **events**, while each **event** can be related to an arbitrary number of **object attribute values**. This means, there can be events without object attribute values and vice-versa. In order to minimize the need to capture these relationships, any **object attribute value** that is not created explicitly (i.e., after it's object is in existence), is created implicitly with the **CREATE** of the **object** itself. When **objects** get deleted, all of their **object attribute values** are deleted implicitly.

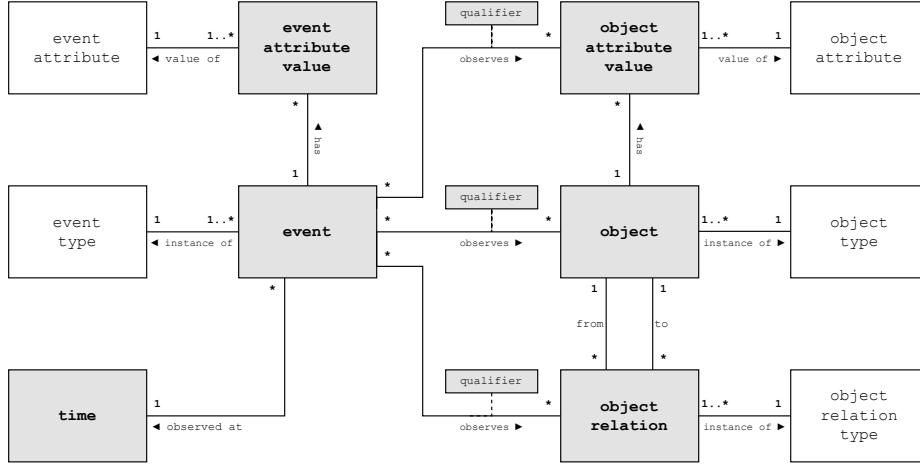


Fig. 9. Original OCED-MM Base Model (published 22nd August 2022)

2. **event observes object relation** — An event and an object relation can be related in a qualified (i.e., association class) manner, meaning their type of relationship is denoted. While a minimum set of qualifiers is predefined (CREATE, MODIFY, and DELETE), additional qualifiers can be introduced as part of the data capture and used to reflect the semantics of the relationship. Each object relation can be involved in an arbitrary number of events, while each event can be related to an arbitrary number of object relations. This means, there can be events without object relations and vice-versa. When objects get deleted, all of their object relations are deleted implicitly.

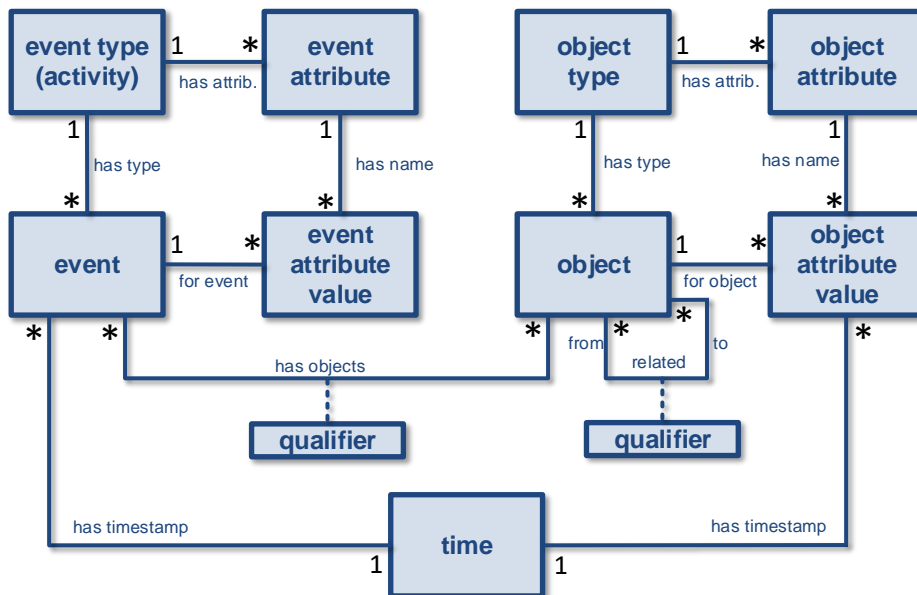
Section 4.4 discusses how the OCED-MM Core Model of Sect. 3 can be extended to also express **observes** relations to object attributes and object relations.

This extension allows an event to refer to an object, an object attribute value, and/or an object relation. But an object can also refer to the same object attribute value and the same object relation. In case a log producer uses multiple of these linkage options simultaneously, cycles may be introduced leading to inconsistent data capture. Applying implicit semantics of the qualifiers of the **observes** relation and the **object relations**, e.g. deleting a parent object, allows to prevent such inconsistency. However, this is not enforced by the meta-model itself; see Sect. 4.8.

## B Appendix: OCEL 2.0

OCEL 1.0 *Object-Centric Event Log* was released in 2020, prior to the standardization process described in Section 2. OCEL 1.0 was based on a number

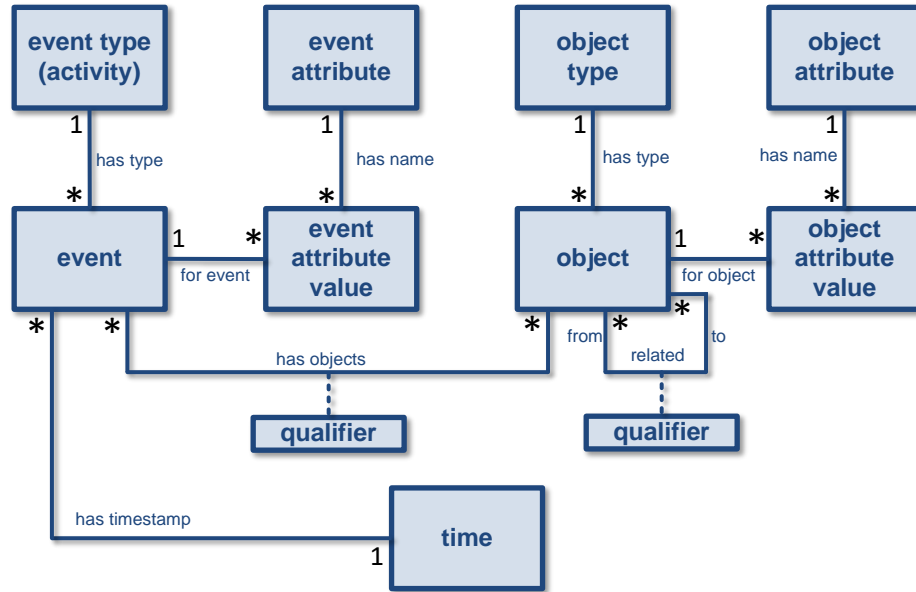
of attempts to standardize such event data in the period 2015-2019. See, for example, the eXtensible Object-Centric (XOC) event log format and a variety of artifact-centric formats (e.g., proclefs). Based on the limited adoption and support for these formats, OCEL 1.0 started deliberately simple, with a focus on object-centric discovery and conformance-checking techniques using only event-to-object (E2O) relationships [2,3]. OCEL 2.0, released in 2023, extends OCEL 1.0, leveraging experiences gathered while developing and applying these OCPM techniques [2,5,14]. In OCEL 2.0, Object-to-Object (O2O) relationships were added, next to qualifiers for both E2O and O2O relationships. Figure 10 shows the OCEL 2.0 meta-model. The design process for the original OCED-MM Base Model and OCED-MM Full Model (see Sect. 2) explored a number of design decisions corresponding to variations of the OCED 2.0 meta-model [2,5,14].



**Fig. 10.** The full OCEL 2.0 meta-model [2,5,14] extending OCEL 1.0 and the OCED-MM Base Model

Figure 10 shows the OCEL 2.0 meta-model which varies from and extends the original OCED-MM Base Model in various ways. The main differences are the ability to define the possible attributes per object and event type, and the ability to timestamp object attribute values. For an explanation of these concepts, we refer to Section 5.4 and [5,14,2].

Omitting the `has attribute` relationships between attributes and types for events and objects, and the `has timestamp` relationships from `object attribute values` to `time` results in the reduced OCEL 2.0 meta-model shown in Fig. 11



**Fig. 11.** The reduced OCEL 2.0 meta-model without timed object attributes and without the ability to specify possible attributes per object and event type

which is similar to the OCED-MM Core Model of Sect. 3 (up to naming of relationships). Note that omitting the `has attribute` relationships makes it more difficult to store the data in a relational database. When events and objects of the same type may have arbitrary attributes, it is impractical to store these in a relational database. See the OCEL 2.0 SQLite format [5] for details.