OCEL: A Standard for Object-Centric Event Logs

Conference Paper · July 2021

DOI: 10.1007/978-3-030-85082-1_16

4 authors, including:

Gyunam Park
RWTH Aachen University
12 PUBLICATIONS 62 CITATIONS
SEE PROFILE

Wil Van der Aalst
RWTH Aachen University
1,357 PUBLICATIONS 81,806 CITATIONS
SEE PROFILE

OCEL: A Standard for Object-Centric Event Logs

Anahita Farhang Ghahfarokhi[1], Gyunam Park[1], Alessandro Berti[1,2], and Wil M.P. van der Aalst[1,2]

[1] Process and Data Science Chair, RWTH Aachen University, Aachen, Germany
[2] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany
{farhang,gnpark,a.berti,wdaalst}@pads.rwth-aachen.de

Abstract. [14]The application of process mining techniques to real-life information systems is often challenging. Considering a Purchase to Pay (P2P) process, several case notions such as order and item are involved, interacting with each other. Therefore, creating an event log where events need to relate to a single case (i.e., process instance) leads to convergence (i.e., the duplication of an event related to different cases) and divergence (i.e., the inability to separate events within the same case) problems. To avoid such problems, object-centric event logs have been proposed, where each event can be related to different objects. These can be exploited by a new set of process mining techniques. This paper describes OCEL (Object-Centric Event Log), a generic and scalable format for the storage of object-centric event logs. The implementation of the format can use either JSON or XML, and tool support is provided.

Keywords: Object-Centric Event Logs · Object-Centric Process Mining

1 Introduction

Process mining is a field of data science bridging the gap between model-based analysis and data-oriented analysis. Process mining techniques include process discovery techniques to discover process models describing the event log, conformance checking algorithms that compare process models with event logs, and model enhancement methods that enrich the process model with some information inferred from the event log [12].

Event logs are the starting points to apply process mining techniques. Several approaches have been proposed towards having a standard for event logs that are summarized in Figure 1. The most successful one is XES, which has been accepted as the IEEE standard in 2014 for the storage of (traditional) event logs [13]. Numerous event logs have been recorded using the XES, and several approaches have focused on extraction of complex data to obtain XES event logs [3,2,11]. However, although the XES standard can capture the behavior of the events with a single case notion, in real processes such as O2C (Order to Cash) supported by ERP systems, multiple case notions are involved, and XES cannot record that.

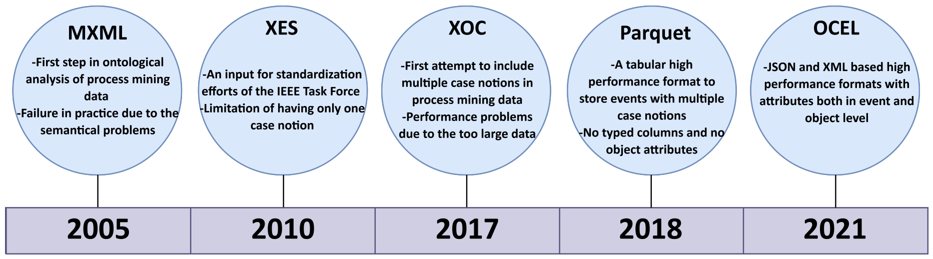
MXML
-First step in ontological analysis of process mining data
-Failure in practice due to the semantical problems
XES
-An input for standardization efforts of the IEEE Task Force
-Limitation of having only one case notion
XOC
-First attempt to include multiple case notions in process mining data
-Performance problems due to the too large data
Parquet
-A tabular high performance format to store events with multiple case notions
-No typed columns and no object attributes
OCEL
-JSON and XML based high performance formats with attributes both in event and object level
2005
2010
2017
2018
2021

Fig. 1. A timeline demonstrating the proposed standards to store event data.

In extracting an event log from some information system, challenges such as convergence (an event is related to multiple cases) and divergence (repeated execution of the activities with the same case notion) may occur. These problems will affect the results of process mining techniques such as process discovery and lead to not real results. It is not possible to avoid completely from these problems; however, it is worthwhile to consider them in process mining analysis. To describe these challenges, consider the example shown in Figure 2 where we have two object types, i.e., *order* and *item*, and three activities, i.e., *place order*, *check item*, and *pack item*. To apply process mining techniques, we have two possible case notions (i.e, *order* and *item*). For example, here, there are 100 *orders* and 1000 *items*. Each *item* refers to one *order* and *place order* is executed per *order*. So it is executed 100 times.

– *Convergence*: Suppose that we want to use *item* as the case notion and *place order* as the activity. The number of *items* is 1000; therefore, in applying traditional process mining techniques, we need 1000 times *place order* instead of 100. This is related to convergence, where an event includes multiple cases.
– *Divergence*: If we use *order* as the case notion and consider *pack item* as an activity, for the same case notion we will have many *pack item* events. Each *check item* should be followed by the *pack item*. However, we cannot distinguish between the different *items* within an *order*, and random arrangement may exist between these two activities. This is called divergence challenge, where the order between two activities is lost.

Object-centric event logs have been proposed to address the above challenges in the extraction of process-related information from information systems. Most of the studies in object-centric process mining is focused on process discovery such as *artifact-centric modeling* [4,9]. Some studies have been done on the extraction of object-centric event logs from information systems [10,1,5,6]. This also includes contributions related to the storage format. For example, in [6]

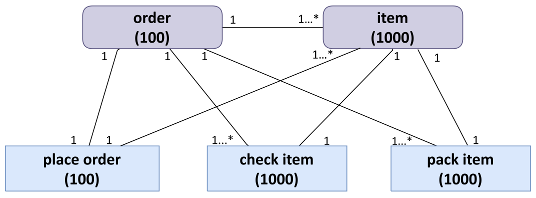

Fig. 2. A fragment of the relationships between case notions (i.e., *order* and *item*) and activities (i.e., *place order*, *check item*), and *pack item*.

Table 1. Informal representation of the events of an OCEL.

id	activity	timestamp	order	item	package	customer	resource	price
e_1	*place order*	*2020-07-09 08:20:01.527+01:00*	$\{ o_1 \}$	$\{ i_1, i_2, i_3 \}$	$\emptyset$	$\{ c_1 \}$	*Alessandro*	*200.0*
e_2	*confirm order*	*2020-07-10 09:23:01.527+01:00*	$\{ o_1 \}$	$\emptyset$	$\emptyset$	$\emptyset$	*Anahita*	*302.0*
e_3	*check availability*	*2020-07-10 17:10:08.527+01:00*	$\{ o_1 \}$	$\{ i_1 \}$	$\emptyset$	$\emptyset$	*Gyunam*	*125.0*
...	...	...	...	...	...	...	...	...

Table 2. Informal representation of the objects of an OCEL.

id	type	product	color	age	job
i_1	*item*	*iPod*	*silver*		
c_1	*customer*			*young*	*teacher*
...	...	...	...	...	...

authors propose a meta-model, called OpenSLEX, that integrates process and data perspectives. It can generate different views from the database flexibly. New storage formats such as XOC have been proposed as for object-centric event logs [8,7]. An XOC log contains a list of events. Each event is characterized by some attributes, by a list of related objects, and by the state of the object model. However, this format suffers from performance issues since the size of the event log scales badly with the number of events and objects related to the process. Moreover, the attributes of an object are replicated with all the events that are related to such an object.

In this paper, we provide a new format (OCEL) to store object-centric event logs to address these limitations, focusing on its specification, serialization in JSON and XML formats, and tool supports. In `http://ocel-standard.org/`, the formal definition of OCEL standard is available along with the detailed conceptualization and specification.

The rest of the paper is organized as follows. Section 2 specifies the OCEL format. Section 3 proposes an implementation, tool support, and a set of event logs for the OCEL format. Finally, Section 4 concludes the paper.

2 Specification

This section introduces the specification of OCEL based on its formal definition introduced in `http://ocel-standard.org/`. Based on the definition of OCEL, an event log consists of information about events and objects involved in the events. An event contains an identifier, an activity, a timestamp, some additional attributes, and is related to some objects. The events are ordered based on the timestamp. The objects can have properties themselves. A sample of the informal representation of OCEL is shown in Tables 1 and 2. Here, we introduce the meta-model for the specification of OCEL.

The meta-model for the specification of OCEL is shown in Figure 3 as a UML class diagram. The log, event, and object classes define the high-level structure of logs. The description for each class is as follows:

– **Log:** The log class contains sets of events and objects. A log contains global elements such as global log, global event, and global object elements. First, a global log element contains the version, attribute names, and object types that compose the log. Second, a global event element specifies some default values for the elements of events. Finally, a global object element specifies some default values for the elements of objects.

- **Event:** An event represents an execution record of an underlying business process. It associates multiple elements (e.g., an identifier, an activity, a timestamp, and relevant objects) and possibly optional features (e.g., event attributes). For example, each row in Table 1 shows an event.
- **Object:** An object indicates the information of an object instance in the business process. It contains required (e.g., type) and optional (e.g., *color* and *size*) elements. For example, each row in Table 2 shows an object.

The element class specifies the elements of the high-level classes. It is composed of a key and value(s). The key is string-based, whereas the value may be of type *string*, *timestamp*, *integer*, *float*, and *boolean*.

An element can be nested, i.e., a parent element can contain child elements. Among nested elements we have *lists* and *maps*.

We can state some advantages in comparison to the existing formats for the storage of object-centric event logs:

- In comparison to tabular formats (such as CSVs), the information is strongly typed.
- Support to lists and maps elements, in comparison with existing formats (XOC, tabular formats, OpenSLEX) that do not properly support them.
- Decoupling of the attributes at the object level from the event level. This helps to avoid replication of the same information, in comparison to the XOC format.

In the next section, we will propose the serialization of OCEL based on the definition, which is fully described in the website of OCEL and the tool support.

3 Serialization and Tool Support

The specification has been serialized in both the JSON and XML formats. An example of JSON format is shown in Listing 1.1. This example shows a part of

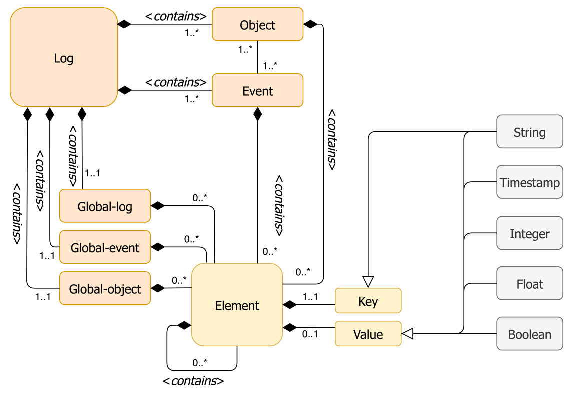

Fig. 3. The UML class diagram for the complete meta-model for the OCEL format.

the data, describing event *place order* and object i_1. "__INVALID__" denotes the default values to be used when activity and type information is missing in event and object, respectively. Furthermore, we provide tool support for the OCEL, for both serializations, for the Python[3] and Java programming language[4]. In our tool, the user can import and export OCELs in both serializations and validate the JSON/XML log files according to the serialization schema. Some other features, such as flattening OCELs into traditional event logs, are also possible [1]. To show more examples of OCEL, we have provided some non-trivial OCELs related to the SAP IDES system in both formats.

The current implementations load the entire object-centric event log in memory. This could represent a problem when managing big logs. However, the JSON implementation of the standard could be transferred to a document database such as MongoDB or ElasticSearch in a straightforward way.

Listing 1.1. JSON-OCEL example.

```
{"ocel:global-log": {
    "ocel:version": "1.0",
    "ocel:attribute-names": [
      "resource", "price", "product", "color", "age","job"],
    "ocel:object-types": [
      "customer", "item", "order", "package"]
  },
  "ocel:global-event": {
    "ocel:activity": "__INVALID__"
  },
  "ocel:global-object": {
    "ocel:type": "__INVALID__"
  },
  "ocel:events": {
    "e1": {
      "ocel:activity": "place_order",
      "ocel:timestamp": "2020-07-09 08:20:01.527+01:00",
      "ocel:omap": ["i1", "o1", "i2", "i3", "c1"],
      "ocel:vmap": {"resource": "Alessandro", "price": 200.0}
    }  },
  "ocel:objects": {
    "i1": {
      "ocel:type": "item",
      "ocel:ovmap": {"color": silver, "product": iPad}
    },  } }
```

4 Conclusion

In this paper, we presented OCEL as a format for storing object-centric event logs, which overcomes the limitations of the previous proposals (e.g., XOC and Parquet). The format is implemented in either JSON and XML. An object is mapped to an entity of the log. So, each event log contains a list of objects, and the properties of the objects are written only once in the log (and not replicated for each event). Furthermore, tool support is provided (Python and Java).

Some challenges remain open. The format does not provide consistency checks based on advanced constraints (such as the number of objects per event). The

[3] standalone library; `https://github.com/OCEL-standard/ocel-support`

[4] ProM 6.10 nightly build; package: *OCELStandard*

main challenge lies in the adoption of object-centric process mining techniques such as object-centric process discovery. Moreover, the support for streams in the context of object-centric event logs is missing. Furthermore, case studies are needed to illustrate the usage of OCEL in real-world processes.

Acknowledgments

We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–EXC-2023 Internet of Production – 390621612.

References

1. Alessandro Berti and Wil M. P. van der Aalst. Extracting multiple viewpoint models from relational databases. In *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium*, volume 379, 2019.
2. Diego Calvanese, Tahir Emre Kalayci, Marco Montali, and Ario Santoso. Obda for log extraction in process mining. In *Reasoning Web Summer School*, 2017.
3. Diego Calvanese, Marco Montali, Alifah Syamsiyah, and Wil M. P. van der Aalst. Ontology-driven extraction of event logs from relational databases. In *International Conference on Business Process Management*, 2016.
4. David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32, 2009.
5. E González López de Murillas, GE Hoogendoorn, and Hajo A Reijers. Redo log process mining in real life: data challenges & opportunities. In *International Conference on Business Process Management*, 2017.
6. E Gonzalez Lopez de Murillas. Process mining on databases: Extracting event data from real-life data sources. 2019.
7. Guangming Li, Renata Medeiros de Carvalho, and Wil M. P. van der Aalst. Automatic discovery of object-centric behavioral constraint models. In *International Conference on Business Information Systems*, volume 288, 2017.
8. Guangming Li, Eduardo González López de Murillas, Renata Medeiros de Carvalho, and Wil M. P. van der Aalst. Extracting object-centric event logs to support process mining on databases. In *Information Systems in the Big Data Era - CAiSE Forum*, volume 317, 2018.
9. Erik HJ Nooijen, Boudewijn F van Dongen, and Dirk Fahland. Automatic discovery of data-centric and artifact-centric processes. In *International Conference on Business Process Management*. Springer, 2012.
10. Ana Pajic Simović, Sladan Babarogić, and Ognjen Pantelić. A domain-specific language for supporting event log extraction from ERP systems. In *International Conference on Computers Communications and Control*. IEEE, 2018.
11. Álvaro Valencia-Parra, Belén Ramos-Gutiérrez, Angel Jesus Varela-Vaca, María Teresa Gómez López, and Antonio Garcia Bernal. Enabling process mining in aircraft manufactures: extracting event logs and discovering processes from complex data. In *International Conference on Business Process Management*, 2019.
12. Wil M. P. van der Aalst. *Process Mining*. Springer-Verlag Berlin Heidelberg, 2016.
13. H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Xes, XESame, and ProM. In *Information Systems Evolution - CAiSE Forum*, volume 72, 2010.

14. Xinwei Zhang, Yaoci Han, Wei Xu, and Qili Wang. HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture. *Inf. Sci.*, 557:302–316, 2021.