“Object-Centric Process Mining: Data Extraction, Process Discovery, and Conformance Checking”

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

M.Sc.
Alessandro Berti
aus Soave, Italien

Berichter:
Universitätsprofessor h. c. Dr. h. c. Dr. ir. Wil van der Aalst
Universitätsprofessor Marco Montali, PhD

Tag der mündlichen Prüfung: 02.07.2025

Abstract (English)

"*Chaos was the law of nature; Order was the dream of man.*"
Henry Adams

This thesis explores the domain of Object-Centric Process Mining (OCPM), a novel approach within the field of process mining that addresses the limitations of traditional methods by focusing on the interactions among multiple objects involved in business processes. Unlike conventional process mining, which primarily considers sequences of events per case, OCPM considers the complex relationships and interactions between different types of objects (e.g., orders, invoices, deliveries) that participate in processes. This research aims to extend the theoretical foundations of OCPM, develop methodologies for effectively mining Object-Centric Event Logs (OCELs), and apply these methodologies to real-world data to gain insights into complex process behaviors.

We start by establishing a comprehensive understanding of the current state of process mining, identifying gaps and challenges in analyzing modern, complex business processes that involve multiple interacting entities. We then propose a framework for OCPM that includes the extraction, modeling, and analysis of Object-Centric Event Logs (OCELs). Key to this framework is the development of techniques for discovering object-centric process models that accurately represent the behavior of multiple interacting objects in business processes.

We detail the approach for extracting OCELs from relational databases, addressing challenges such as data granularity and schema complexity. We also describe techniques for preprocessing data, ensuring the quality and relevance of the event logs for process discovery and analysis.

Through empirical studies, we apply the OCPM framework to several case studies, demonstrating its effectiveness in uncovering insights into process behaviors that are not observable with traditional process mining techniques. These case studies span various processes, showcasing the versatility and applicability of OCPM in analyzing complex, multi-entity business processes.

The thesis concludes with a discussion of the implications of our findings for the field of process mining, highlighting the potential of OCPM to advance the understanding of complex processes in various domains. It also identifies open research questions and directions for future work, emphasizing the need for further development of methodologies, tools, and techniques to fully realize the potential of OCPM.

In sum, this thesis contributes to the advancement of process mining by proposing and validating an object-centric approach that enables the analysis of complex interactions among multiple objects in business processes, offering new perspectives and tools for researchers and practitioners in the field.

Zusammenfassung (Deutsch)

Diese Arbeit untersucht das Gebiet des objektzentrierten Process Mining (OCPM), eines neuartigen Ansatzes im Bereich des Process Mining, der die Einschränkungen herkömmlicher Methoden überwindet, indem er sich auf die Interaktionen mehrerer an Geschäftsprozessen beteiligter Objekte konzentriert. Im Gegensatz zum konventionellen Process Mining, das hauptsächlich Ereignissequenzen pro Fall betrachtet, berücksichtigt das OCPM komplexe Beziehungen und Interaktionen zwischen verschiedenen Arten von Objekten (z. B. Bestellungen, Rechnungen, Lieferungen), die an den Prozessen beteiligt sind. Ziel dieser Forschung ist es, die theoretischen Grundlagen des OCPM zu erweitern, Methoden zur effektiven Analyse objektzentrierter Ereignisprotokolle (OCELs) zu entwickeln und diese Methoden auf reale Daten anzuwenden, um Erkenntnisse über komplexe Prozessverhalten zu gewinnen.

Zunächst wird ein umfassendes Verständnis des aktuellen Stands im Process Mining geschaffen, indem Lücken und Herausforderungen bei der Analyse moderner, komplexer Geschäftsprozesse mit mehreren interagierenden Einheiten identifiziert werden. Anschließend wird ein Rahmenwerk für OCPM vorgestellt, das die Extraktion, Modellierung und Analyse objektzentrierter Ereignisprotokolle (OCELs) umfasst. Entscheidend für dieses Rahmenwerk ist die Entwicklung von Techniken zur Entdeckung objektzentrierter Prozessmodelle, die das Verhalten mehrerer interagierender Objekte in Geschäftsprozessen akkurat abbilden.

Es werden detailliert Verfahren zur Extraktion von OCELs aus relationalen Datenbanken beschrieben, wobei Herausforderungen wie Datengranularität und Schemakomplexität behandelt werden. Darüber hinaus werden Techniken zur Vorverarbeitung der Daten erläutert, um die Qualität und Relevanz der Ereignisprotokolle für die Prozessentdeckung und -analyse sicherzustellen.

Durch empirische Untersuchungen wird das OCPM-Rahmenwerk in mehreren Fallstudien angewendet und dessen Wirksamkeit demonstriert, insbesondere bei der Aufdeckung von Einsichten in Prozessverhalten, die mit traditionellen Process-Mining-Methoden nicht erkennbar sind. Diese Fallstudien umfassen verschiedene Prozesse und verdeutlichen die Vielseitigkeit und Anwendbarkeit von OCPM bei der Analyse komplexer, mehrgliedriger Geschäftsprozesse.

Die Arbeit schließt mit einer Diskussion der Implikationen der Ergebnisse für das Gebiet des Process Mining und hebt das Potenzial des OCPM hervor, das Verständnis komplexer Prozesse in unterschiedlichen Domänen zu fördern. Zudem werden offene Forschungsfragen und zukünftige Arbeitsrichtungen aufgezeigt, wobei insbesondere auf die Notwendigkeit hingewiesen wird, Methoden, Werkzeuge und Techniken weiterzuentwickeln, um das Potenzial des OCPM vollständig auszuschöpfen.

Zusammenfassend leistet diese Arbeit einen Beitrag zur Weiterentwicklung des Process Mining, indem sie einen objektzentrierten Ansatz vorschlägt und validiert, der die Analyse komplexer Interaktionen mehrerer Objekte in Geschäftsprozessen ermöglicht und neue Perspektiven sowie Werkzeuge für Forscher und Praktiker auf diesem Gebiet bereitstellt.

Contents

Chapter 1

Introduction

"*Knowledge is the treasure, but judgment is the treasurer, of a wise man.*"
William Penn

1.1 Introduction

Organizations rely on business processes to function. These processes are supported by systems that collect data about what happens in the organization, storing this information in databases. Process mining provides a set of methods used to analyze this data to understand and improve how an organization works. There are different methods within process mining, including discovering how processes happen (process discovery), checking if processes are happening as they should (conformance checking), and improving process models based on real data.

Traditionally, process mining focuses on specific business objects, like documents related to sales or purchases, and examines their lifecycle from start to finish. This requires choosing a particular way to view and organize the data, known as a case notion. However, it is also important to look at how these business objects interact with each other. For example, a purchase might be made in response to a sale. This interaction is crucial but often overlooked in traditional process mining methods.

Object-Centric Process Mining (OCPM) was developed to solve this problem. Unlike traditional methods, it looks at all business objects and their interactions, providing a fuller picture of what is happening in an organization. This approach tries to overcome the main challenge with older methods: not considering the complex web of interactions between different business objects.

1.2 Problem Statement

When using traditional process mining, several challenges arise.

One major issue is the need to create separate event logs for different business processes. This process of Extracting, Transforming, and Loading (ETL) data is both time-consuming and complex. Moreover, traditional event logs treat processes as independent entities, ignoring the important ways they interact.

This isolated view leads to inaccuracies (as described in [115]):

- *Deficiency*: Events not linked to the chosen case notion are left out. This means some activities that could be important for understanding the whole process are not included in the analysis.
- *Convergence*: Events related to several objects of the same object type are replicated in different cases.
- *Divergence*: Events in a case having the same activity may be related to different objects, leading to misleading causalities.

These problems show the limits of traditional process mining. They point to a need for new approaches that can handle the complexity of how processes and business objects actually interact in organizations.

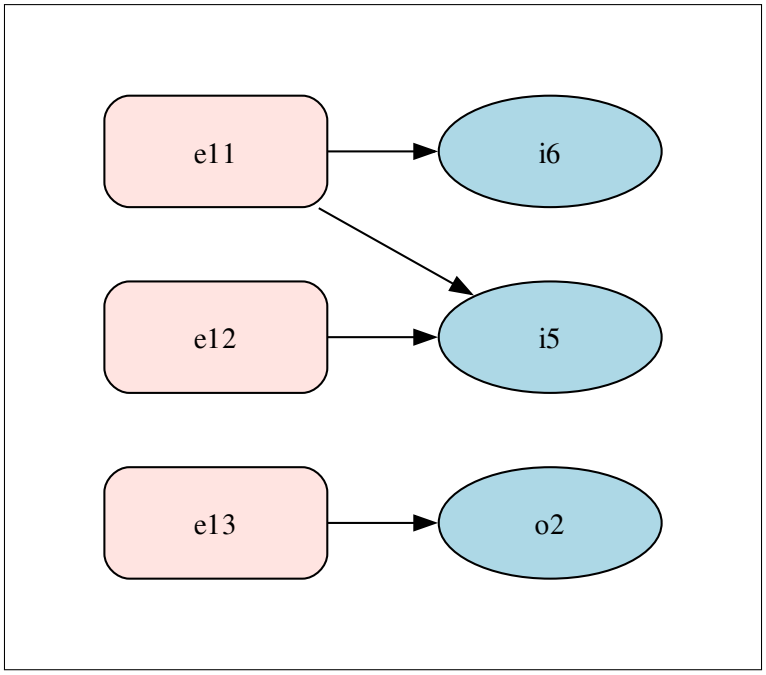


Figure 1.1: Diagram showing the interconnection at the instance level (events to objects) in an OCEL. Events are pink boxes, while objects are blue ellipses.

1.3 OCPM

Object-Centric Process Mining (OCPM) offers a new way to look at an organization's processes. It uses a model that includes all business objects and the events that happen to them. This approach allows us to see how these objects and events are connected. We can look at these connections in two ways: by looking at specific instances (which specific events are connected to which objects, see Figure 1.1) and by looking at types (how types of events are related to types of objects, see Figure 1.2).

An important part of this approach is the Object-Centric Event Log (OCEL). An OCEL is a view that focuses on a part of the organization's processes, selecting only the types of events and objects required for analysis. Using OCELs has several advantages:

- It reduces the need to repeatedly extract separate logs for different processes. This single source of information is not tied to any specific system, making it easier to work with.
- It helps us see and understand how different types of objects interact with each other. This is important because many issues in organizations occur at the points where processes and objects meet.
- It avoids the problems that come with the traditional focus on single cases. This means we can get a clearer picture of what is happening without the issues of missing or duplicated events.

By focusing on both the individual objects and their interactions, OCPM gives us a more complete understanding of an organization's processes. This can lead to better decisions on how to improve these processes.

1.4 Objectives of the Thesis

The main objectives of this research are multi-faceted and aim to significantly advance the field of OCPM:

O1 *Address Data Extraction and Preprocessing Challenges*: A significant part of this research will be devoted to tackling issues related to data extraction and preprocessing for OCELs. This encompasses the creation of effective methodologies for data extraction, table selection, formulating blueprints, and handling OCELs.

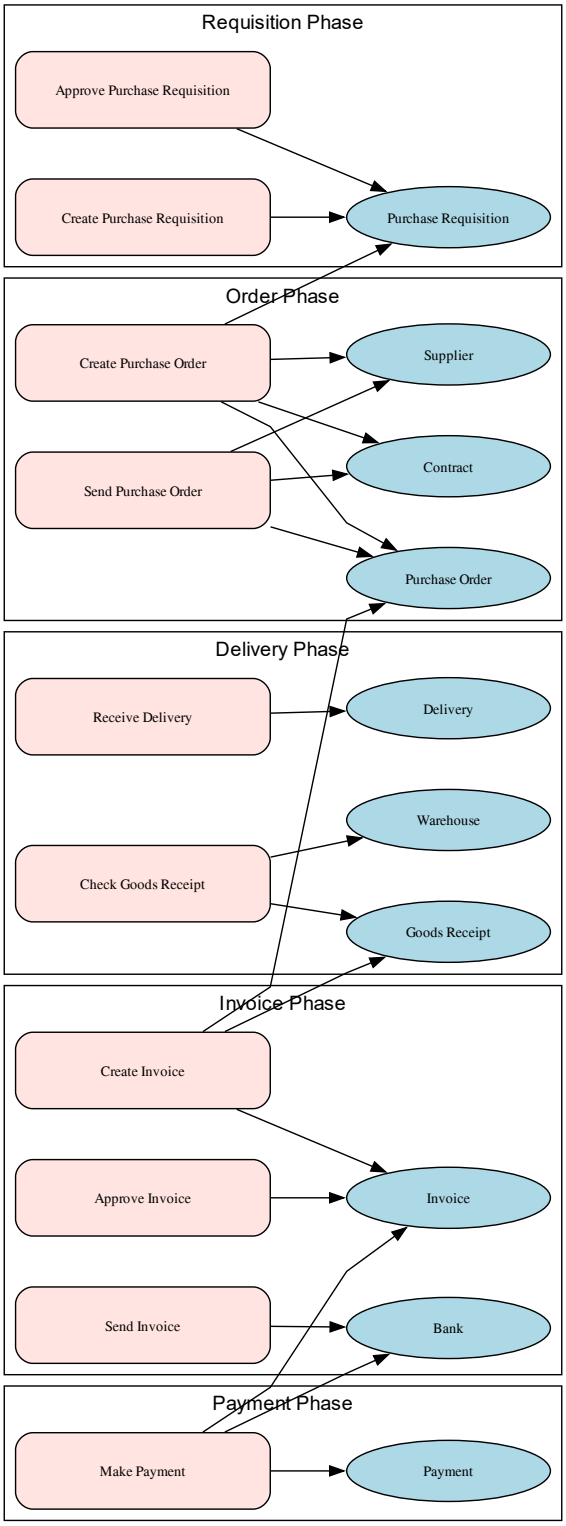


Figure 1.2: Diagram showing the interconnection at the type level (event types to object types) in an OCEL.

O2 *Improve Process Discovery and Conformance Checking Methods*: One of the central goals of this study is to devise novel and effective algorithms for Object-Centric Process Discovery (OCPD)

and Object-Centric Conformance Checking (OCCC). The thesis aims to develop new methods, ranging from discovering Object-Centric DFGs (OCDFGs) to anomaly detection in object-centric process models.

O3 *Implement Real-world Case Studies and Applications*: By employing the developed methods on real-world scenarios like SAP ECC ERP [19, 14], insights will be gained on the practicality and efficacy of the proposed techniques. These insights will contribute towards the generalization of findings and, in turn, advance the field of OCPM.

O4 *Develop and Provide Effective Tools for OCPM*: A critical ambition of this research is the development and provision of innovative software tools tailored for OCPM, focusing on the PM4Py library and the OC-PM tool [25].

1.5 Research Questions

This thesis tackles critical inquiries to establish the field of OCPM. These fundamental research questions aim to decode complex issues, paving the way for novel research paths. The main research questions this thesis seeks to answer are:

RQ1 How can we define OCELs to better represent complex, intertwined process instances?

RQ2 What kind of data extraction and preprocessing techniques are needed to construct these OCELs from raw data?

RQ3 Can we formulate novel methods for process discovery that accommodate the complexity of OCELs?

RQ4 How can we develop approaches for conformance checking in an object-centric setting?

RQ5 How can feature extraction and anomaly detection techniques be adapted or developed for OCELs?

RQ6 How can we extend traditional event log definitions to accommodate the complexities inherent in OCELs, such as Object-to-Object relationships, and Event-to-Object qualifications?

RQ7 Can we create tools and libraries specifically designed for OCPM? What should they look like, and how should they be optimized for performance and scalability?

RQ8 How can we meaningfully compare the performance and functionality of different OCPM tools?

The presented research questions form the core of this thesis, underlining the crucial challenges in OCPM. Addressing these will significantly advance the field. As we progress through this work, these research questions serve as our roadmap, guiding us towards our scientific goals

1.6 Scope and Limitations

This study focuses primarily on the development, implementation, and evaluation of OCPM techniques. While we take a broad approach in analyzing the current landscape and potential of this field, our research specifically targets certain key aspects.

In terms of scope, we have concentrated on *processes supported by ERP systems* [19, 14] as examples for our methodologies. We believe these provide a representative and insightful look into the potential of OCPM, given the inherent complexity and object-centric nature of such processes. However, it is important to note that the methods proposed in this thesis could be adapted to other types of processes where the interaction between multiple objects plays a critical role.

The methodologies developed and applied in this research have been aimed at extracting data, discovering processes, and checking conformance from object-centric perspectives. While this allows us to discuss specific issues and opportunities within these areas, there are additional aspects of OCPM, such as *process enhancement or prediction*, which fall outside the scope of our current study.

As for limitations, the findings and results presented in this research are, to some extent, contingent upon the data used for the study. While every effort was made to ensure that the data sets used were robust and reliable, inherent limitations tied to the availability and quality of data exist in any empirical research.

Lastly, the case studies presented in this work were selected to highlight the applicability of OCPM techniques. However, as with any case study, *the results may not be directly transferable to other settings* or circumstances. These limitations should be considered when interpreting and applying the results of our research.

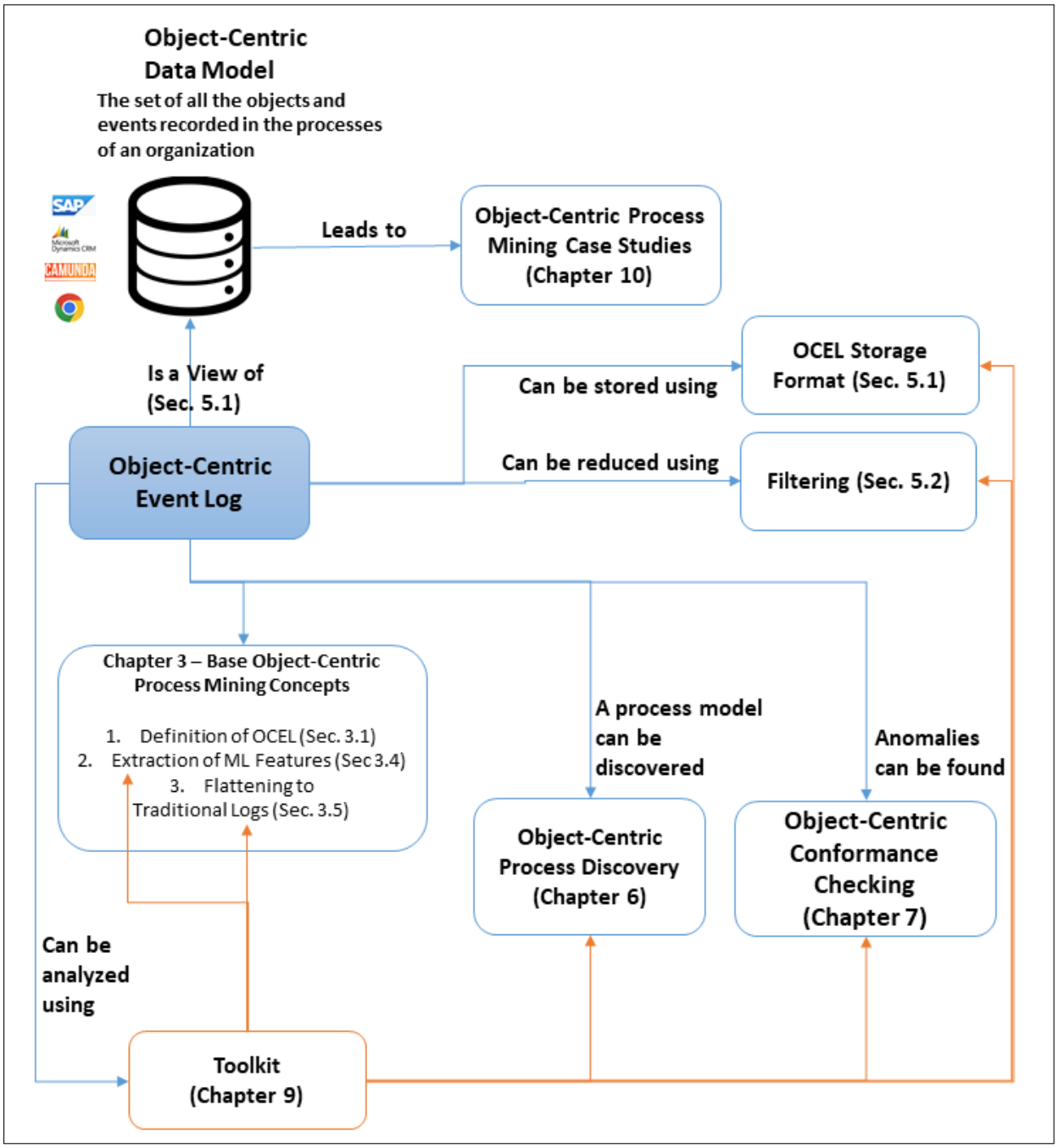


Figure 1.3: Structure of the thesis.

1.7 Structure of the Thesis

The structure of this thesis (illustrated in Figure 1.3) is designed to provide a comprehensive exploration of OCPM, its advantages over traditional methods, and its potential applications. Following the introduction, which sets the stage for the ensuing research and provides the necessary context, we dive deeper into the subject matter.

In Chapter 2, we establish a foundation for understanding process mining by exploring its fundamental

concepts. We introduce formal definitions, exploring traditional event logs and Directly-Follows Graphs, then we shift our focus to process mining techniques and tools, explaining how these have evolved and set the stage for OCPM.

Chapter 3 is dedicated to laying down the foundational concepts of OCPM. It begins by explaining basic object-centric concepts, supported by running examples taken from ERP processes. We then propose a formal definition of OCELs. In addition, we discuss feature extraction on OCELs, and the process of flattening these logs to traditional event logs.

In Chapter 4, we examine existing approaches in OCPM. We discuss various process models in current use, and conduct a literature review on OCPM techniques. This sets the stage for the development of our own methodology.

Chapters 5, 6, and 7 are dedicated to methodological developments in data extraction and preprocessing, OCPD, and OCCC respectively. In each chapter, we propose novel methods, develop methodological frameworks, and evaluate these methods both quantitatively and qualitatively.

In Chapter 8, we showcase real-world applications and case studies of OCPM. We highlight the insights obtained, process improvement actions taken, and lessons learned from each case study. This includes examples from different real-world scenarios, which help to demonstrate the versatility and utility of OCPM.

Chapter 9 provides an overview of the tools available for OCPM. It presents an in-depth comparison of the tools and offers guidelines for selecting and using the appropriate tool for scientific research or practical tasks.

Finally, Chapter 10 concludes the thesis. It summarizes the main contributions and findings of the research, discusses the implications of the research for the field of process mining, and identifies open research questions and directions for future work. This final chapter serves to wrap up the thesis and points the way forward for future research in the field.

Chapter 2

Background

"*An understanding of the natural world and what is in it is a source of not only a great curiosity but great fulfillment.*"
David Attenborough

Introduction

This chapter lays the foundation for core concepts and techniques used throughout this thesis, providing a necessary foundation for understanding the subsequent exploration of Object-Centric Process Mining (OCPM). We begin by examining traditional process mining building blocks, starting with event logs as the fundamental data source for process analysis. We then delve into the representation and analysis of process behavior using Directly-Follows Graphs (DFGs) and Petri nets, highlighting their roles in process discovery and modeling. Conformance checking, a critical aspect of process mining, is introduced through the lens of Token-Based Replay (TBR), a technique used to assess the alignment between observed behavior and process models. Finally, we explore feature extraction and anomaly detection techniques, essential for uncovering patterns and outliers within event data, paving the way for a deeper understanding of process behavior and performance. These foundational concepts, rooted in traditional process mining, will serve as a springboard for our later investigation into the object-centric perspective.

Universes

We used the following universes throughout the thesis:

- U_{Σ} is the universe of strings.
- U_{qual} is the universe of qualifiers.
- U_{etype} is the universe of activities.
- $U_{time} \subseteq \mathbb{R}^{\geq 0}$ is the universe of timestamps.
- U_{attr} is the universe of attribute names.
- U_{val} is the universe of attribute values.
- U_{dom} is the universe of attribute domains.
- U_{table} is the universe of database tables.
- $U_{transact}$ is the universe of database transactions.
- U_{omap} is the universe of Event-to-Object relationships.
- U_{case} is the universe of cases (for the traditional event logs).
- U_{ev} is the universe of events (identifiers).
- U_{obj} is the universe of objects (identifiers).
- U_{otype} is the universe of object types (identifiers).
- U_{TL} is the universe of traditional event logs.
- U_{OL} is the universe of OCELs.
- U_{TM} is the universe of traditional process models.
- U_{OM} is the universe of object-centric process models.
- U_{APN} is the universe of accepting Petri nets.
- U_{DFG} is the universe of Directly-Follows Graphs.

2.1 Mathematical Preliminaries

This section introduces fundamental mathematical concepts and notations essential for understanding the framework used throughout this document. We cover sets, numbers, tuples, multisets, functions, sequences, graphs, orderings, and other related topics.

2.1.1 Sets and Numbers

A *set* is a collection of distinct objects, called *elements*. Sets are denoted by uppercase letters such as A, B, S, and their elements are listed within curly braces $\{\}$. For example, $A = \{a, b, c\}$ denotes a set containing elements a, b, c.

We use the following standard sets of numbers:

- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$: the set of *natural numbers*, including zero.
- $\mathbb{R}$: the set of all *real numbers*.
- $\mathbb{R}^{\geq 0} = \{x \in \mathbb{R} \mid x \geq 0\}$: the set of all *non-negative real numbers*.

The *power set* of a set A, denoted by $\mathcal{P}(A)$, is the set of all subsets of A:

$$\mathcal{P}(A) = \{B \mid B \subseteq A\}.$$

The *Cartesian product* of sets A and B, denoted $A \times B$, is the set of all ordered pairs where the first element is from A and the second from B:

$$A \times B = \{(a, b) \mid a \in A,\, b \in B\}.$$

This concept generalizes to more than two sets: the Cartesian product $A_1 \times A_2 \times \cdots \times A_n$ is the set of all n-tuples $(a_1, a_2, \ldots, a_n)$, where $a_i \in A_i$ for each i.

2.1.2 Tuples

An *n-tuple* is an ordered collection of n elements. Tuples are denoted by parentheses ( ), and the elements are ordered and can be of different types. For example, (a, b, c) is a 3-tuple containing elements a, b, c.

Tuples are of fixed length and are often used to represent structured data with potentially heterogeneous elements, unlike sequences (defined below), which can vary in length and typically contain elements of the same type.

2.1.3 Multisets (Bags)

A *multiset*, also known as a *bag*, is a generalization of a set that allows multiple instances of its elements. Multisets are denoted by square brackets [], and elements are listed with their multiplicities if necessary. For example, the multiset $[a, a, b]$ contains two instances of a and one instance of b.

Operations on multisets are defined as follows:

- *Addition*: The sum of two multisets M_1 and M_2 is a multiset containing the sum of multiplicities for each element:

  $$M_1 + M_2 = [e^{k_1+k_2} \mid \forall e,\, k_1 = \text{count}(e, M_1),\, k_2 = \text{count}(e, M_2)].$$

  For example, $[a, b] + [b, c] = [a, b^2, c]$.

- *Subtraction*: The difference $M_1 - M_2$ is a multiset where the multiplicity of each element is the multiplicity in M_1 minus the multiplicity in M_2, but not less than zero:

  $$M_1 - M_2 = [e^{\ max\ (0, k_1-k_2)} \mid \forall e,\, k_1 = \text{count}(e, M_1),\, k_2 = \text{count}(e, M_2)].$$

  For example, $[a, b^2, c] - [b, c] = [a, b]$.

- *Inclusion*: We say $M_1 \leq M_2$ if for every element e, the multiplicity of e in M_1 is less than or equal to its multiplicity in M_2:

  $$M_1 \leq M_2 \Leftrightarrow \forall e,\, \text{count}(e, M_1) \leq \text{count}(e, M_2).$$

  For example, $[a, b] \leq [a, b^2, c]$ because the counts of a and b in $[a, b]$ do not exceed their counts in $[a, b^2, c]$.

Here, $\text{count}(e, M)$ denotes the multiplicity of element e in multiset M.

2.1.4 Functions and Relations

A *relation* R between sets A and B is a subset of the Cartesian product $A \times B$; that is, $R \subseteq A \times B$. An element $(a, b) \in R$ indicates that a is related to b.

A *function* f from A to B is a relation that assigns each element $a \in A$ to exactly one element $b \in B$. We denote this as $f : A \to B$. The *domain* of f, denoted $\mathrm{dom}(f)$, is the set of all $a \in A$ for which $f(a)$ is defined. The *range* or *image* of f, denoted $\mathrm{rng}(f)$, is the set of all $f(a)$ for $a \in \mathrm{dom}(f)$.

The notation $f : A \not\to B$ indicates that f is a partial function, defined on a subset of A.

The *restriction* of a function f to a subset $S \subseteq \mathrm{dom}(f)$, denoted $f|_S$, is the function defined by:

$$f|_S(a) = f(a), \quad \forall a \in S.$$

For example, if $f : \mathbb{R} \to \mathbb{R}$ is defined by $f(x) = x^2$, then the restriction $f|_{[0,\infty)}$ limits f to non-negative inputs.

2.1.5 Sequences and Strings

A *sequence* is an ordered list of elements where repetition is allowed, and order matters. Sequences are denoted by angle brackets $\langle\,\rangle$. For example, $s = \langle a, b, a\rangle$ is a sequence of length 3.

Notation and Indexing. The elements of a sequence s can be accessed by their index:

$$s = \langle s(1), s(2), \ldots, s(n)\rangle,$$

where $s(i)$ denotes the i-th element, and $n = |s|$ is the *length* of the sequence.

Sequence Operations.

- *Concatenation*: The concatenation of two sequences s and t, denoted $s@t$, is a sequence formed by appending t to s:
$$s@t = \langle s(1), s(2), \ldots, s(|s|), t(1), t(2), \ldots, t(|t|)\rangle.$$
- *Subsequences*: A *subsequence* of s is a sequence obtained by deleting zero or more elements from s without changing the order of the remaining elements. The set of all subsequences of s is denoted $\mathcal{S}(s)$.

Set of Sequences. Given a set E, the set of all finite sequences over E is denoted E^*, and the set of all infinite sequences over E is denoted E^ω.

Ordering in Sequences. For elements in a sequence s, we use $s(i) < s(j)$ to denote the order induced by their positions in the sequence when $i < j$.

2.1.6 Graph Theory Concepts

A *graph* G is a pair (V, E), where V is a set of vertices (also called *nodes*), and E is a set of edges connecting pairs of vertices. In a *directed graph*, edges are ordered pairs $(v, w) \in V \times V$, indicating an edge from vertex v (the *source*) to vertex w (the *target*).

Edge Weights and Multigraphs. In a *weighted graph*, edges have associated weights, represented by a function $w : E \to \mathbb{R}^+$. A *multigraph* allows multiple edges between the same pair of nodes. In such graphs, edges can be represented as elements of a multiset.

Start and End Nodes. A *start node* is a node with no incoming edges, and an *end node* is a node with no outgoing edges.

Bipartite Graphs. A *bipartite graph* is a graph whose vertices can be partitioned into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to one in V_2; that is, $E \subseteq V_1 \times V_2$.

2.1.7 Orderings

An *order* is a binary relation that arranges elements in a sequence. A *total order* is a binary relation $\leq$ over a set S that is reflexive, antisymmetric, transitive, and total (i.e., any two elements are comparable):

- *Reflexive*: $a \leq a$ for all $a \in S$.
- *Antisymmetric*: If $a \leq b$ and $b \leq a$, then $a = b$.
- *Transitive*: If $a \leq b$ and $b \leq c$, then $a \leq c$.
- *Totality*: For all $a, b \in S$, either $a \leq b$ or $b \leq a$.

Lexicographical Order. The *lexicographical order* $\leq_{\text{lex}}$ is an order on sequences based on the natural order of their elements. For sequences $s = \langle s(1), s(2), \ldots \rangle$ and $t = \langle t(1), t(2), \ldots \rangle$, $s <_{\text{lex}} t$ if there exists k such that $s(i) = t(i)$ for all $i < k$ and $s(k) < t(k)$.

2.1.8 Aggregation Functions

An *aggregation function* agg combines multiple values into a single value. Common aggregation functions include:

- *Sum* (sum):
$$\text{sum}(x_1, x_2, \ldots, x_n) = x_1 + x_2 + \cdots + x_n.$$
- *Average* (avg):
$$\text{avg}(x_1, x_2, \ldots, x_n) = \frac{1}{n}(x_1 + x_2 + \cdots + x_n).$$
- *Minimum* (min):
$$\min(x_1, x_2, \ldots, x_n) = \text{smallest of } x_i.$$

Aggregation functions are used to summarize data, such as computing the average duration of events in a log.

2.1.9 Other Notations

We use standard mathematical operators:

- *Addition* ($+$): The addition operator, used for numbers, sequences, and multisets.
- *Subtraction* ($-$): The subtraction operator, extended to multisets as defined above.
- *Inclusion* ($\leq$): Used for multisets to denote that one multiset is included in another based on element multiplicities.

Absolute Value and Length. The notation $|x|$ denotes the absolute value of a number or the length of a sequence. For a sequence s, $|s|$ is the number of elements in s.

Restriction of a Function. Given a function $f : A \to B$ and a subset $S \subseteq A$, the *restriction* $f|_S$ is the function f limited to inputs from S:

$$f|_S : S \to B, \quad f|_S(a) = f(a), \forall a \in S.$$

For example, in the context of event logs, $\text{evtype}|_{E'}$ denotes the function evtype restricted to the set E'.

Indexed Sequences and Events. In sequences of events, $\text{events}(c)(i)$ denotes the i-th event in the sequence of events associated with case c. The notation $|\text{events}(c)|$ represents the number of events in the sequence for case c.

String Concatenation. The concatenation of strings or sequences is denoted by the operator @. For sequences s and t, $s@t$ represents the sequence formed by appending t to s.

These mathematical preliminaries provide the foundational concepts necessary to understand the formal definitions and operations used in this document. Familiarity with these concepts will facilitate comprehension of the subsequent sections.

2.2 Traditional Event Logs

Traditional event logs play an instrumental role in the field of process mining, a discipline that intersects data mining and process modeling and analysis. These logs are utilized as input data for process mining techniques, enabling the discovery, monitoring, and improvement of real-life processes based on data.

An event log is essentially a record of sequences of events that have occurred within a system, with each event representing a specific instance of an activity. Traditional event logs originate from a multitude of systems, such as transactional systems, workflow management systems, or any information system that records operational data. They are characterized by a linear progression of events, primarily focusing on the sequence in which activities occur for each case, also known as a process instance.

In Definition 1, a traditional event log is defined as a tuple, which effectively encapsulates key elements and relationships of event data. The set $A \subseteq U_{etype}$ represents a collection of *activities*. An activity can be thought of as a well-defined piece of work or operation in a process, such as "invoice creation" or "perform payment." $E \subseteq U_{ev}$ signifies a set of *events*. Each event, as mentioned earlier, corresponds to the execution of an activity within a specific case. $C \subseteq U_{case}$ denotes a set of *cases*. A case refers to a process instance in which the activities are executed. It might represent a customer order, a patient treatment, or a loan application, for example. The mapping functions evtype : $E \to A$, time : $E \to \mathbb{R}^{\geq 0}$, and case : $E \to C$ associate an event with its corresponding activity, timestamp, and case respectively. This information is vital for analyzing the order and timing of activities for each process instance.

Definition 1 (Traditional Event Log)**.** *A traditional event log is a tuple* $TL = (A, E, C, evtype, time, case, \leq)$ *in which:*

- $A \subseteq U_{etype}$ *is a set of activities.*
- $E \subseteq U_{ev}$ *is a set of events.*
- $C \subseteq U_{case}$ *is a set of cases.*
- $evtype : E \to A$ *associates an activity to each event.*
- $time : E \to \mathbb{R}^{\geq 0}$ *associates a timestamp to each event.*
- $case : E \to C$ *associates a case to each event.*
- $\leq$ *is a total ordering on* E*, usually based on timestamps.*

We define $events : C \to E^*$ *as the function that maps each case identifier* c *to the sequence of events for that case, ordered by* $\leq$*:*

$$events(c) = \langle e_1, e_2, \ldots, e_n \rangle,$$

where $e_i \in E$*,* $case(e_i) = c$*, and* $e_i < e_{i+1}$ *for all* i*.*

For $c \in C$ *with* $events(c) \neq \langle\rangle$*, we define:*

$$start(c) = events(c)(1), \quad end(c) = events(c)(|events(c)|).$$

A traditional event log provides a structured way to capture and represent granular activity data within a system, facilitating the examination and enhancement of processes through techniques like process mining. The total order $\leq$ could be defined on the timestamp (as first sorting criteria) and the lexicographic order of the event identifiers (as second sorting criteria).

In the field of process mining, several different file formats are used to store event logs. Here are some of the most commonly used ones:

- *MXML* (Mining XML): The MXML format [125] is an older format specifically designed for storing process mining event logs in XML. Each event in an MXML file has a number of associated attributes, such as the case id, timestamp, and the name of the executed activity. This format has been largely superseded by XES, but it is still used in some older datasets and tools.

Table 2.1: Example traditional event log represented as a table.

Case ID	Activity	Timestamp	Resource
3	register request	2010-12-30 13:32:00+00:00	Pete
3	examine casually	2010-12-30 14:06:00+00:00	Mike
3	check ticket	2010-12-30 15:34:00+00:00	Ellen
3	decide	2011-01-06 08:18:00+00:00	Sara
3	reinitiate request	2011-01-06 11:18:00+00:00	Sara
3	examine thoroughly	2011-01-06 12:06:00+00:00	Sean
3	check ticket	2011-01-08 10:43:00+00:00	Pete
3	decide	2011-01-09 08:55:00+00:00	Sara
3	pay compensation	2011-01-15 09:45:00+00:00	Ellen
2	register request	2010-12-30 10:32:00+00:00	Mike
2	check ticket	2010-12-30 11:12:00+00:00	Mike
2	examine casually	2010-12-30 13:16:00+00:00	Sean
2	decide	2011-01-05 10:22:00+00:00	Sara
2	pay compensation	2011-01-08 11:05:00+00:00	Ellen
1	register request	2010-12-30 10:02:00+00:00	Pete
1	examine thoroughly	2010-12-31 09:06:00+00:00	Sue
1	check ticket	2011-01-05 14:12:00+00:00	Mike
1	decide	2011-01-06 10:18:00+00:00	Sara
1	reject request	2011-01-07 13:24:00+00:00	Pete
6	register request	2011-01-06 14:02:00+00:00	Mike
6	examine casually	2011-01-06 15:06:00+00:00	Ellen
6	check ticket	2011-01-07 15:22:00+00:00	Mike
6	decide	2011-01-07 15:52:00+00:00	Sara
6	pay compensation	2011-01-16 10:47:00+00:00	Mike
5	register request	2011-01-06 08:02:00+00:00	Ellen
5	examine casually	2011-01-07 09:16:00+00:00	Mike
5	check ticket	2011-01-08 10:22:00+00:00	Pete
5	decide	2011-01-10 12:28:00+00:00	Sara
5	reinitiate request	2011-01-11 15:18:00+00:00	Sara
5	check ticket	2011-01-14 13:33:00+00:00	Ellen
5	examine casually	2011-01-16 14:50:00+00:00	Mike
5	decide	2011-01-19 10:18:00+00:00	Sara
5	reinitiate request	2011-01-20 11:48:00+00:00	Sara
5	examine casually	2011-01-21 08:06:00+00:00	Sue
5	check ticket	2011-01-21 10:34:00+00:00	Pete
5	decide	2011-01-23 12:12:00+00:00	Sara
5	reject request	2011-01-24 13:56:00+00:00	Mike
4	register request	2011-01-06 14:02:00+00:00	Pete
4	check ticket	2011-01-07 11:06:00+00:00	Mike
4	examine thoroughly	2011-01-08 13:43:00+00:00	Sean
4	decide	2011-01-09 11:02:00+00:00	Sara
4	reject request	2011-01-12 14:44:00+00:00	Ellen

- *XES* (eXtensible Event Stream): XES [3] is the current standard format for process mining event logs, defined by the IEEE Task Force on Process Mining. It is a highly extensible XML-based format that can store complex event data. In addition to basic information about each event (like the case id, timestamp, and the name of the executed activity), XES can also store additional attributes for each event, case, or the entire log. This makes XES suitable for representing event data from complex real-world processes.
- *CSV* (Comma-Separated Values): While not specifically designed for process mining, CSV files are

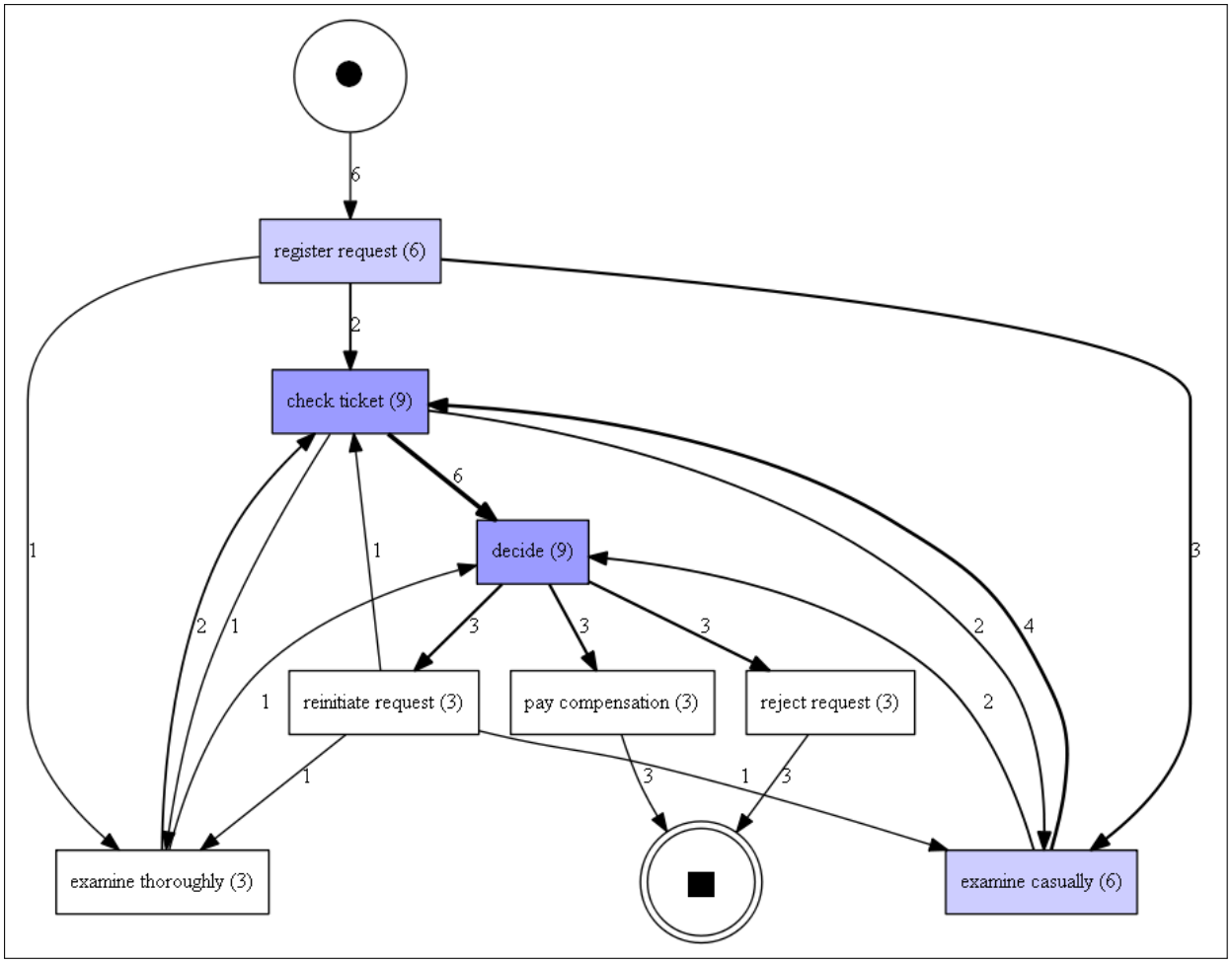


Figure 2.1: Directly-Follows Graph (DFG) computed on an example log.

a popular format for storing event logs due to their simplicity and wide compatibility. A typical process mining CSV file will have a row for each event, with columns for at least the case id, the timestamp, and the activity name, plus any additional attributes. However, compared to XES, CSV files are less structured and less flexible, making them more suited to simpler datasets.

2.3 Directly-Follows Graphs

A *Directly-Follows Graph* (DFG) [114] is an important tool that provides a visualization of the sequence of activities in a process based on an event log. It plays a critical role in the discovery phase of process mining. A Directly-Follows Graph is constructed from the event log by creating a node for each unique activity and an edge between two nodes if the activity associated with the first node is immediately followed by the activity associated with the second node in any of the traces in the event log. The edge may be annotated with the frequency of this occurrence to provide more insights.

The importance of DFGs lies in their simplicity and power to provide a quick understanding of the process flow. Here are some of their main uses:

- *Process Discovery*: DFGs are a fundamental starting point for process discovery. They show the connections between activities and thus provide an intuitive picture of the process flow.
- *Bottleneck Analysis*: by observing the frequencies on the edges, one can identify which transitions are most common and potentially pinpoint bottlenecks or hotspots in the process.
- *Conformance Checking*: DFGs can also aid in conformance checking. By comparing the DFG from the event log with the expected relationships, deviations can be spotted.

Definition 2 (Directly-Follows Graph (DFG)). *A* Directly-Follows Graph *is a tuple* $DFG = (A, F, \pi_{measn}, \pi_{mease})$ *where:*

- $A \subseteq U_{etype}$ *is a set of activities.*
- $\triangleright$ *is the start node of the graph,* $\square$ *is the end node of the graph.*
- $F \subseteq (\{\triangleright\} \cup A) \times (A \cup \{\square\})$ *is the set of edges.*
- $\pi_{measn} : A \not\to \mathbb{R}^{\geq 0}$ *is a measure on the nodes.*
- $\pi_{mease} : F \not\to \mathbb{R}^{\geq 0}$ *is a measure on the edges.*

We can discover a Directly-Follows Graph from a traditional event log. This is introduced in Definition 3.

Definition 3 (Discovery of a Directly-Follows Graph (Frequency)). *Given a traditional event log* $TL = (A, E, C, evtype, time, case, \leq)$*, we can discover a Directly-Follows Graph* $DFG = (A, F, \pi_{measn}, \pi_{mease})$ *where:*

- A *is the set of activities.*
- *The set of edges is defined as follows:*

$$F = \{(\triangleright, evtype(start(c))), (evtype(end(c)), \square) \mid c \in C\} \cup$$
$$\{(evtype(events(c)(i)), evtype(events(c)(i+1))) \mid c \in C \ \wedge \ 1 \leq i < |events(c)|\}$$

- *For* $a \in A$, $\pi_{measn}(a) = |\{e \in E \mid evtype(e) = a\}|$.
- *For* $(\triangleright, a) \in F$, $\pi_{mease}((\triangleright, a)) = |\{c \in C \mid evtype(start(c)) = a\}|$
- *For* $(a, \square) \in F$, $\pi_{mease}((a, \square)) = |\{c \in C \mid evtype(end(c)) = a\}|$
- *For* $(a, b) \in F$ *with* $a, b \in A$,

$$\pi_{mease}((a, b)) = |\{(evtype(events(c)(i)), evtype(events(c)(i+1))) \mid$$
$$c \in C \ \wedge \ 1 \leq i < |events(c)| \ \wedge \ evtype(events(c)(i)) = a \ \wedge \ evtype(events(c)(i+1)) = b\}|$$

2.4 Petri Nets

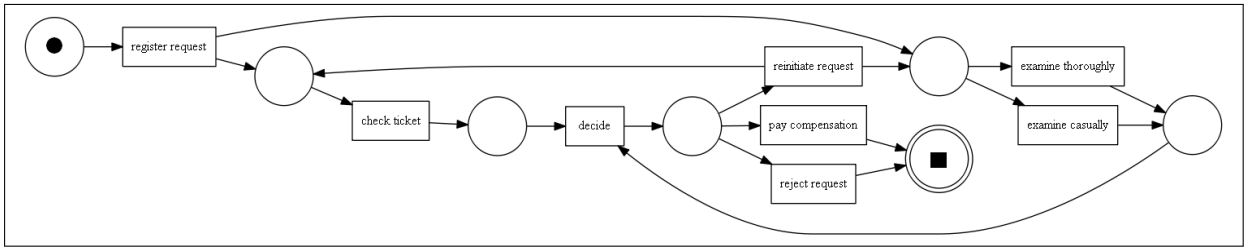


Figure 2.2: Accepting Petri net discovered using the Alpha Miner algorithm [122] on an example log.

Petri nets, as introduced in Definition 4, are a mathematical modeling language. Named after Carl Adam Petri who introduced them in his 1962 doctoral thesis [99], they have found widespread use in the modeling and analysis of systems that are concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. This comes from their graphical yet formal nature, which captures the essence of these systems by allowing them to represent states (through *places*), state changes (through *transitions*), and the influence of states on state changes (through *arcs*).

In the realm of process mining, the utility of Petri nets is especially pronounced [124]. The goal here is to extract process models from event logs, and the semantics of the models often require the notion of concurrency, choice, and synchronization, all of which can be naturally expressed using Petri nets.

Moreover, labeled Petri nets introduce the concept of invisible transitions, represented by the symbol τ. These transitions are crucial for capturing the notion of unobservable or insignificant actions within a system, often internal events that have no direct bearing on the observable behavior of the system but are nonetheless crucial for maintaining the integrity of the system's operational logic. In process mining, invisible transitions help in modeling complex behavior without the need for additional event data.

Definition 4 (Labeled Petri Net)**.** *A labeled Petri net is defined as a tuple $PN = (P, T, F, \lambda)$ where:*

- *P is a set of places.*
- *T is a set of transitions (with the assumption that $P \cap T = \emptyset$).*
- *$F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a function associating a place and a transition, or a transition and a place, to a natural number indicating the weight of the arc.*
- *$\lambda : T \to U_{etype} \cup \{\tau\}$ associates every transition with either an activity or the symbol τ (denoting an invisible transition).*

A *marking*, as per Definition 5, is a fundamental concept in the study of Petri nets, providing the dynamic aspect of the system being modelled. For a labeled Petri net (P, T, F, λ), a marking $M : P \to \mathbb{N}$ assigns a natural number to each place in the set of places P. These numbers represent tokens in the corresponding places.

In the Petri net model, tokens can be thought of as representing resources or instances of certain conditions. A transition may fire when its input places contain a certain number of tokens, as defined by the flow function F. Upon firing, the transition consumes tokens from its input places and produces tokens in its output places. Hence, a marking in a Petri net captures the state of the system at a certain point in time.

Definition 5 (Marking)**.** *Given a labeled Petri net (P, T, F, λ), we define a marking as a function $M : P \to \mathbb{N}$.*

An *accepting* labeled Petri net, as per Definition 6, is a natural extension of a labeled Petri net, introducing the notion of an *initial marking* M_I and a *final marking* M_F. In essence, an accepting labeled Petri net $(P, T, F, \lambda, M_I, M_F)$ forms a directed bipartite graph with two distinguished states: the initial state and the final state, captured by M_I and M_F, respectively.

The initial and final markings represent the beginning and end states of the system. The initial marking, $M_I : P \to \mathbb{N}$, captures the starting condition of the system, i.e., the distribution of tokens across the places at the start of the process. Conversely, the final marking, $M_F : P \to \mathbb{N}$, represents a desirable end state for the process, i.e., the distribution of tokens across places when the system reaches a condition considered as completion.

Definition 6 (Accepting Labeled Petri Net)**.** *An accepting labeled Petri net is defined as a tuple $APN = (P, T, F, \lambda, M_I, M_F)$, where (P, T, F, λ) is a labeled Petri net and $M_I : P \to \mathbb{N}$ and $M_F : P \to \mathbb{N}$ are the initial and final marking.*

In the context of process mining, an accepting labeled Petri net provides a comprehensive model for complete execution paths of the system. It not only characterizes the possible behaviors of the system (through transitions and their labeling), but also defines a "valid" process instance as one that can transition from the initial marking to the final marking. This validity condition is crucial when mining event logs, as it allows the identification of regular behavior and the detection of anomalies.

Moreover, an accepting labeled Petri net inherently provides a notion of "process completion", which is essential for many real-life systems. By defining a final marking, the model can represent processes that have a clearly identifiable end state. This could be crucial in certain applications such as workflow management or business process management where understanding the end-to-end process is essential for efficiency and optimization.

The *execution semantics* of a Petri net describe how a system modeled by the Petri net evolves over time. The execution semantics are fundamentally driven by the enabling and firing of transitions, which capture the dynamic behaviors of the system.

Definition 7 (Execution Semantics)**.** *Given a labeled Petri net $PN = (P, T, F, \lambda)$, we say that a transition $t \in T$ is enabled in the marking $M : P \to \mathbb{N}$ if $M(p) \geq F(p, t)$ for all $p \in P$. If t is enabled in M, the marking $M'(p) = M(p) - F(p, t) + F(t, p)$ for all $p \in P$ is obtained firing t.*

A transition $t \in T$ in a Petri net $PN = (P, T, F, \lambda)$ is said to be enabled in a given marking $M : P \to \mathbb{N}$ if and only if each place $p \in P$ contains at least as many tokens as the weight of the arc from p to t, formally written as $M(p) \geq F(p, t)$ for all $p \in P$. This intuitively signifies that all the preconditions (or inputs) for a transition to take place are satisfied in the current state of the system.

If a transition is enabled, it may fire, thereby leading to a change in the system state or marking. The firing of an enabled transition t produces a new marking $M'(p) = M(p) - F(p, t) + F(t, p)\ \forall p \in P$, essentially updating the token distribution across the places. Upon firing, the transition consumes tokens from its input places and produces tokens in its output places according to the flow function F.

Definition 8 (Firing Sequence)**.** *A sequence of transitions $\sigma = \langle t_1, t_2, \dots t_n \rangle$ where $t_i \in T$ is said to be a firing sequence from marking M to M' in a labeled Petri net $PN = (P, T, F, \lambda)$ if and only if there exists a sequence of markings $\langle M_0, M_1, \dots, M_n \rangle$ such that $M_0 = M, M_n = M'$ and for each $1 \leq i \leq n$, transition t_i is enabled in M_{i-1} and M_i is obtained by firing t_i in M_{i-1}.*

For the definition of a *language* of a Petri net, we first need to establish the concept of *trace* in Definition 9. A trace is a sequence of labels corresponding to a firing sequence (see Definition 8). We can then define the language of a labeled Petri net (in Definition 10) as the set of all possible traces.

Definition 9 (Trace)**.** *Given a labeled Petri net $PN = (P, T, F, \lambda)$ and a firing sequence $\sigma = \langle t_1, t_2, \dots t_n \rangle$, the trace of σ is the sequence of labels $\langle \lambda(t_1), \lambda(t_2), \dots \lambda(t_n) \rangle$, excluding any instances where $\lambda(t_i) = \tau$.*

Definition 10 (Language)**.** *The language of an accepting labeled Petri net $APN = (P, T, F, \lambda, M_I, M_F)$, denoted by $L(APN)$, is the set of all traces of firing sequences from M_I to M_F.*

The language is particularly relevant in the context of accepting Petri nets, because it is the set of traces that start at the initial marking and end at the final marking.

2.5 Conformance Checking using TBR

In the field of process mining, one of the key tasks is *conformance checking*, which refers to the comparison of the discovered process model (often in the form of a Petri net) with the actual observed behavior as recorded in event logs. The goal is to determine whether the discovered model accurately represents the real-world process and, if deviations are found, where and how these discrepancies occur.

Replay methods are fundamental to conformance checking as they simulate the execution of the actual recorded process instances (traces) on the discovered model. By "replaying" the events of a trace on the model, one can observe if and where the model fails to accurately capture the actual process behavior. Any deviations can then be analyzed to gain insights into the reasons behind these discrepancies, providing valuable feedback for model improvement.

Among the various replay techniques, Token-Based Replay (TBR) [109, 23, 24] is a popular method owing to its intuitive approach and clear visual feedback.

TBR operates by simulating the execution of recorded process instances (traces) against the model. This simulation starts with setting the model to its initial state or marking. As each event in the trace is encountered, the corresponding transition within the Petri net is activated, or "fired", thereby moving tokens according to the transition rules: *consuming* tokens from input places and *producing* tokens at output places. The process continues for each event in the trace, effectively "playing the token game". If at any point a transition cannot be fired due to a lack of tokens in its input places, missing tokens are artificially inserted to enable the transition, signifying a discrepancy between the model and the actual event log. Finally, the replay concludes by evaluating tokens that remain within the model's places after all events have been processed. These *remaining* tokens, along with any instances of missing tokens inserted, provide insights into the model's conformance and highlight areas for potential improvement. Examples executions of TBR are contained in Figure 2.3 and Figure 2.4.

In the context of TBR, four concepts have been introduced:

- *Produced tokens*: Tokens that are generated at the output places of a transition when it is fired. In a perfect replay, these tokens will match the subsequent events in the trace.
- *Consumed tokens*: Tokens that are removed from the input places of a transition when it is fired. Ideally, these tokens were produced by previous events in the trace.
- *Missing tokens*: Tokens that are needed to enable a transition (i.e., they should be in its input places), but are not there during the replay.
- *Remaining tokens*: Tokens that are left in the places of the model after the replay of a trace is completed.

Definition 11 introduces the concept of Token-Based Replay (TBR) in the context of process mining, extending the basic method by keeping track of the usage of transitions.

In addition to keeping count of the produced, consumed, missing, and remaining tokens in each place, this function associates each transition in the Petri net with a set of corresponding events from the process log. This association is captured by the $usage : T \to \mathcal{P}(E)$ function. For each transition

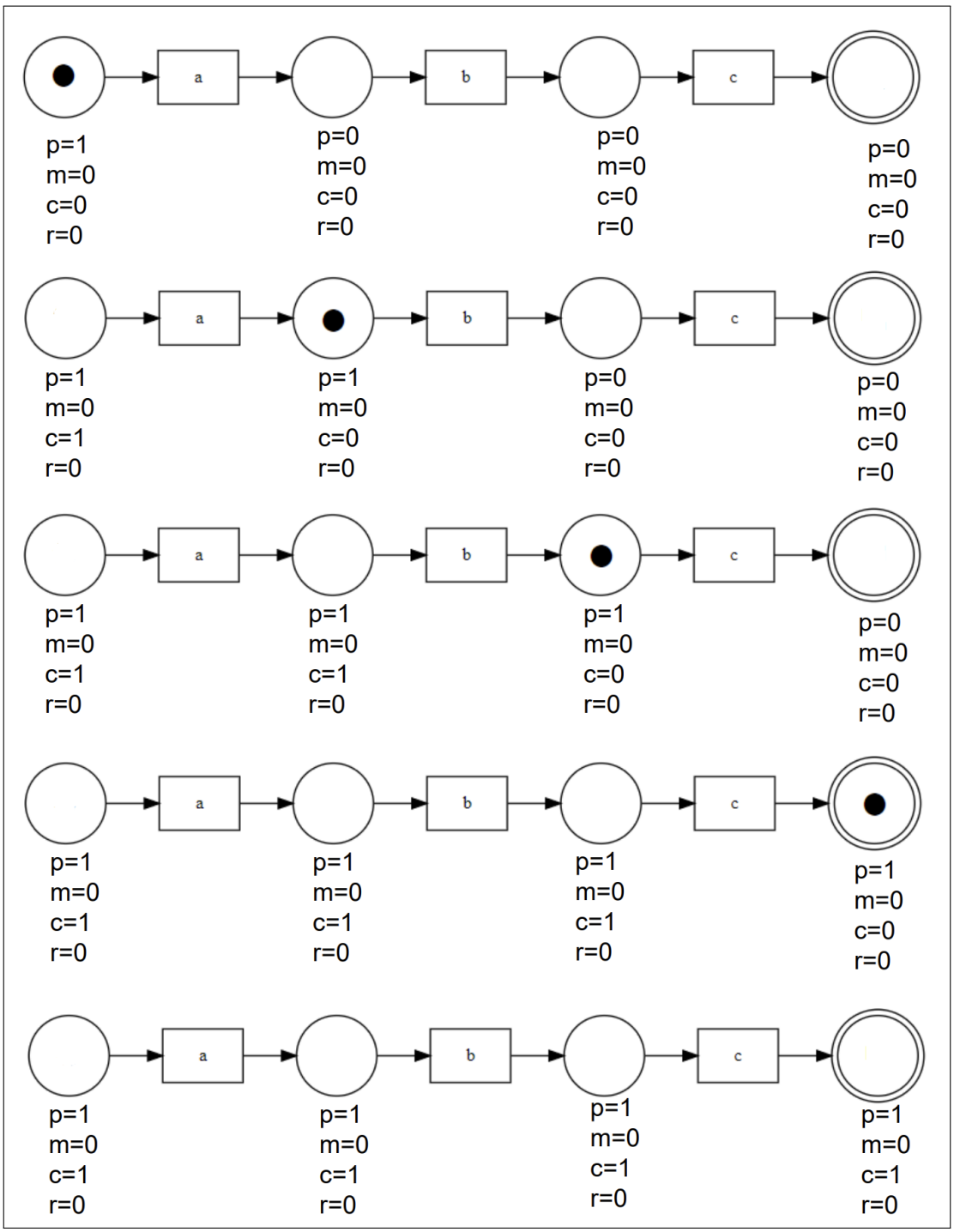


Figure 2.3: Example of TBR between the trace $\langle a, b, c \rangle$ and a sequential model fitting the trace. At the beginning, 1 token is produced in the initial marking. The execution of a consumes and produces 1 token. The execution of b consumes and produces 1 token. The execution of c consumes and produces 1 token. Eventually, the token in the final marking is consumed by the TBR algorithm. Therefore, we have $c = p = 4$ and $m = r = 0$, and a fitness value of 1.0.

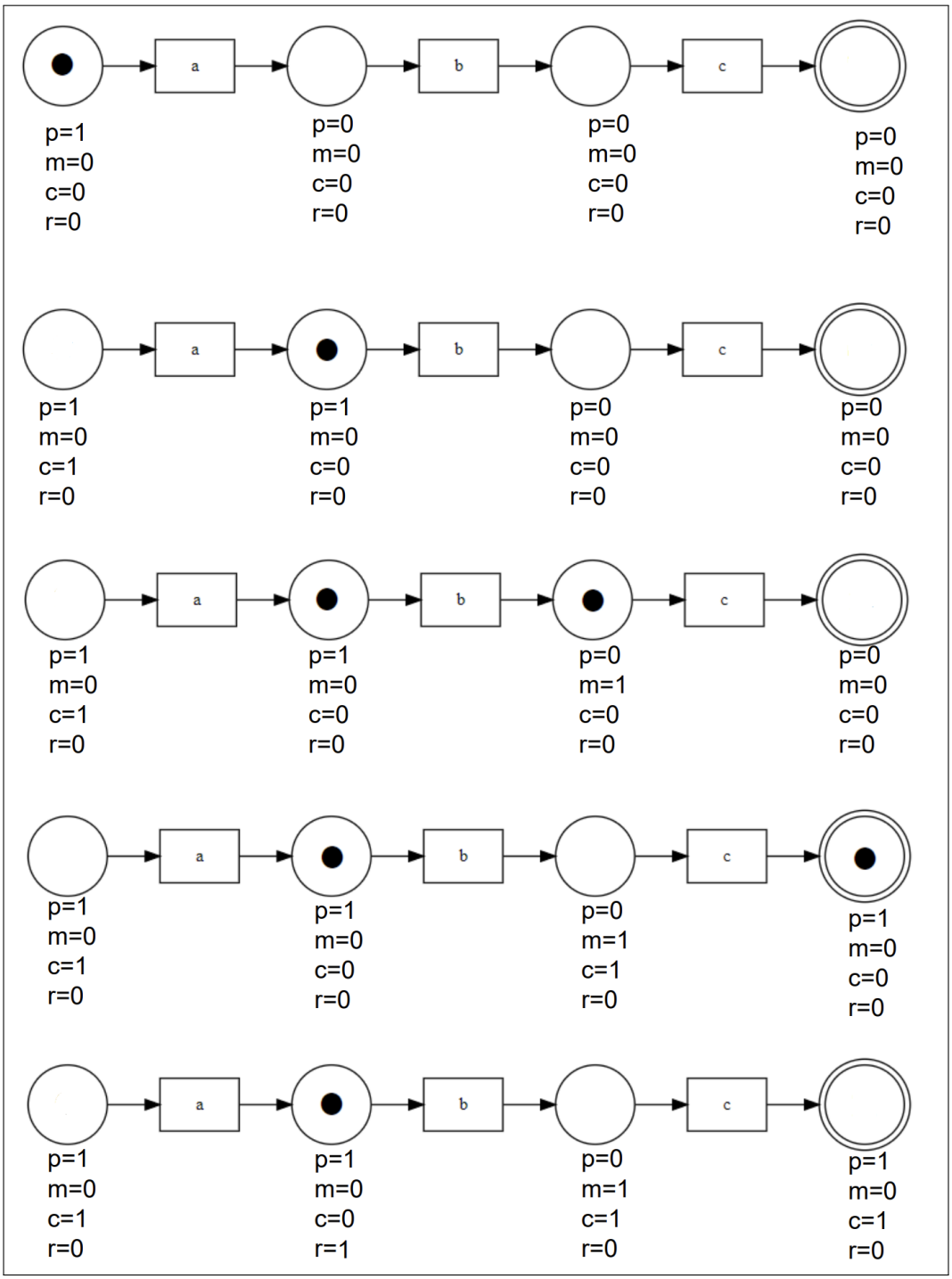


Figure 2.4: Example of TBR between the trace $\langle a, c \rangle$ and a sequential model which does not fit the trace. At the beginning, 1 token is produced in the initial marking. The execution of a consumes and produces 1 token. Then, c cannot be executed because one required token is missing from its source place. The TBR algorithm places the missing token before c. Then, c consumes and produces 1 token, and the token in the final marking is consumed by the TBR algorithm. However, there is one token remaining after a. Therefore, we have $c = p = 3$ and $m = r = 1$, and a fitness value of $\frac{2}{3}$.

$t \in T$, *usage* provides the set of events that led to the firing of the transition during the replay. Moreover, *time_to_cons* measures the time between the corresponding events and the minimum time in which the transition could fire.

Definition 11 (TBR). *Let $TL = (A, E, C, evtype, time, case, \leq)$ be a traditional event log, and $APN = (P, T, F, \lambda, M_I, M_F)$. We define as TBR a function such that, for $c \in C$, $TBR(c, APN) = (PROD_{c,APN}, CONS_{c,APN}, MISS_{c,APN}, REMA_{c,APN}, usage_{c,APN}, time\_to\_cons_{c,APN})$ where:*

- $PROD_{c,APN} : P \to \mathbb{N}$ *is the number of produced tokens.*
- $CONS_{c,APN} : P \to \mathbb{N}$ *is the number of consumed tokens.*
- $MISS_{c,APN} : P \to \mathbb{N}$ *is the number of missing tokens.*
- $REMA_{c,APN} : P \to \mathbb{N}$ *is the number of remaining tokens.*
- $usage_{c,APN} : T \to \mathcal{P}(E)$ *associates every transition to a set of corresponding events..*
- $time\_to\_cons_{c,APN} : T \to \mathcal{B}(\mathbb{R}^{\geq 0})$ *associates every transition to a set of times (each of them is the difference between the corresponding event and the minimum time in which the transition could fire).*

Furthermore, we define $TBR_{log}(TL, APN) = \{TBR(c, APN) \mid c \in C\}$.

Definition 12 introduces the concepts of *case fitness* and *log fitness* [24], which are fundamental metrics in process mining for quantifying the degree of conformance of a discovered process model to a given event log.

The *case fitness* metric is computed for each case (or process instance) in the event log. It measures how well the Petri net model can replay the individual process instance based on the results of the TBR. The formula takes into account both missing tokens, which represent the number of tokens that were needed but not available to fire the transitions in the model, and remaining tokens, which represent the number of tokens left over after the replay of a trace. Each of these quantities is normalized by the total number of tokens consumed or produced, respectively. The case fitness is the average of these two normalized quantities, subtracted from one and then halved, yielding a number between 0 and 1. A case fitness of 1 implies a perfect match between the model's execution and the actual process instance, while a lower value indicates discrepancies.

The *log fitness* metric is an extension of the case fitness to the entire event log. It provides a holistic view of the model's conformance to the entire log, not just individual instances. To calculate the log fitness, we sum the number of missing and remaining tokens for all cases in the log, normalize these by the total number of consumed and produced tokens across all cases, and average these two quantities in a similar fashion to the case fitness calculation. The log fitness also ranges between 0 and 1, with 1 indicating a perfect fit of the model to the entire log.

Definition 12 (Case and Log Fitness). *Let $TL = (A, E, C, evtype, time, case, \leq)$ be a traditional event log, $c \in C$, and $APN = (P, T, F, \lambda, M_I, M_F)$. and*

$$TBR(c, APN) = (PROD_{c,APN}, CONS_{c,APN}, MISS_{c,APN}, REMA_{c,APN}, usage_{c,APN}, time\_to\_cons_{c,APN})$$

a TBR function. We define:

- Case Fitness *for $c \in C$*

$$fitness_{c,TBR} = \frac{1}{2}(1 - \frac{\sum_{p \in P} MISS_{c,APN}(p)}{\sum_{p \in P} CONS_{c,APN}(p)}) + \frac{1}{2}(1 - \frac{\sum_{p \in P} REMA_{c,APN}(p)}{\sum_{p \in P} PROD_{c,APN}(p)})$$

- Log Fitness

$$fitness_{L,TBR} = \frac{1}{2}(1 - \frac{\sum_{c \in C}\sum_{p \in P} MISS_{c,APN}(p)}{\sum_{c \in C}\sum_{p \in P} CONS_{c,APN}(p)}) + \frac{1}{2}(1 - \frac{\sum_{c \in C}\sum_{p \in P} REMA_{c,APN}(p)}{\sum_{c \in C}\sum_{p \in P} PROD_{c,APN}(p)})$$

TBR faces challenges such as duplicate transitions, invisible transitions, and the token explosion problem [24]. *Duplicate transitions* create ambiguity in aligning events to model transitions due to identical labels. *Invisible transitions*, which don't correspond to log events, complicate determining their activation order during replay and obscure model clarity. The *token explosion problem* occurs when the model inaccurately reflects the log, leading to excessive token insertions to facilitate transition execution. This issue increases replay complexity and computational demands, hindering efficient conformance checking.

2.6 Feature Extraction and Anomaly Detection

This section introduces key tasks in event log analysis: feature extraction, dimensionality reduction, feature selection, and anomaly detection, essential for deriving insights from event data.

2.6.1 Feature Extraction

Feature extraction from event logs is crucial in business process analysis. It transforms raw logs into structured data for anomaly detection, process discovery, and performance mining.

An event log typically contains events linked to cases, activities, timestamps, and possibly other attributes, forming the basis for feature extraction.

Features derived from these events offer insights into process behaviors. Activity sequences per case highlight workflow patterns and deviations. Meanwhile, timing data can inform on process timing dynamics, such as activity duration or case completion times, shedding light on execution speed and efficiency.

2.6.2 Dimensionality Reduction

Dimensionality reduction converts data from high to lower dimensions, retaining essential information. It is used for data compression, visualization, noise reduction, and improving algorithm performance. High-dimensional data, facing the "curse of dimensionality", complicates analysis and increases computational demands. Dimensionality reduction aims to preserve data's core structure by eliminating extraneous features.

Benefits include:

- *Noise Reduction*: Removes irrelevant features, reducing noise.
- *Simplified Visualization*: Facilitates data understanding when reduced to 2 or 3 dimensions.
- *Enhanced Algorithm Performance*: Enables algorithms to run more efficiently with fewer dimensions.
- *Prevention of Overfitting*: Lowers the risk of overfitting in machine learning models by reducing feature space.

Key techniques:

- *Principal Component Analysis (PCA)*[1]: Statistically transforms correlated variables into uncorrelated principal components, focusing on variance maximization. PCA helps in data exploration and predictive modeling by identifying important variables from a large dataset, enhancing data visualization. However, principal components may be less interpretable due to their composition from multiple original features.
- *FastMap* [66]: Projects high-dimensional data into a lower dimension while preserving non-linear relationships, without needing a full distance matrix, making it efficient for large datasets. FastMap identifies extreme data pairs, projects data onto lines defined by these pairs, and iterates for the desired dimensionality. It is particularly useful for data with non-linear relationships and tasks like visualization and clustering, despite assuming the Triangle Inequality for data distances.

2.6.3 Feature Selection

Feature selection is the process of choosing the most relevant features for predictive modeling, crucial in handling high-dimensional data. It enhances model simplicity, efficiency, accuracy, and mitigates overfitting by discarding irrelevant or redundant predictors.

Two primary approaches are[2]:

- *Target-independent methods*: Assess features' relevance based on their properties, like variance or correlation among features, without considering the target variable. These methods suit unsupervised learning and preliminary dimensionality reduction, though they might overlook features important in conjunction with others or the target variable.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`
[2] `https://scikit-learn.org/stable/modules/feature_selection.html`

- *Target-dependent methods*: Evaluate features based on their relationship with the target variable, better uncovering predictive power and relevant interactions. These methods require labeled data and are typically more computationally intensive but effective in revealing the most informative features for supervised learning tasks.

Common techniques include:

- *Variance Thresholding*: Eliminates features with low variance, under the assumption that they contribute little to prediction.
- *Mutual Information*: Quantifies the dependency between each feature and the target variable, with higher scores indicating more relevance.
- *Recursive Feature Elimination (RFE)*: Iteratively removes least important features based on model performance.
- *Principal Component Analysis (PCA)*: Reduces dimensionality by transforming features into principal components based on variance, suited for exploratory analysis.
- *L1-based Feature Selection (Lasso)*: Uses L1 regularization to penalize the absolute size of coefficients, driving some to zero and thus selecting relevant features.
- *Tree-based Feature Selection*: Employs tree models to rank features by their importance in improving model predictions.

2.6.4 Anomaly Detection

Anomaly detection, or outlier detection, identifies unusual data patterns, crucial for fraud detection, health monitoring, and ecosystem disturbance analysis. This process is vital across various domains, including credit card fraud detection and cybersecurity, where anomalies indicate significant information like fraudulent activity or system faults.

Key methods include:

- *Isolation Forests* [88]: Utilizes Decision Trees to isolate anomalies in high-dimensional data efficiently, based on the principle that anomalies are few and distinct, making them easier to separate with fewer conditions. It measures normality through the path length in a forest of random trees, where shorter paths indicate potential anomalies.
- *Local Outlier Factor (LOF)*[3]: A density-based technique comparing the local density of a point to its neighbors' densities to identify anomalies. It is effective in detecting anomalies that are not apparent in the global distribution. LOF scores near 1 suggest normality, while significantly higher values indicate anomalies.

Anomaly detection helps identify process inefficiencies or non-standard behaviors for improvement. It is different from conformance checking, which compares actual processes against predefined models to find discrepancies. While conformance checking requires a standard model for comparison, anomaly detection focuses on spotting deviations within the data itself, making both approaches complementary in process analysis.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html`

Conclusion

This chapter has established the essential background in traditional process mining, covering key concepts and techniques that will inform our subsequent exploration of the object-centric paradigm. We have examined event logs, the raw material for process analysis, and explored how DFGs and Petri nets provide powerful means for representing and analyzing process behavior. Understanding the principles of conformance checking, particularly through TBR, is crucial for evaluating the fidelity of process models against real-world execution. Finally, we have touched upon feature extraction and anomaly detection, highlighting their role in uncovering hidden patterns and deviations within process data. These traditional techniques, while powerful, often fall short when applied to complex, interconnected processes involving multiple interacting objects. This limitation motivates the shift towards an object-centric perspective, which will be the central focus of the subsequent chapters. The concepts introduced here, particularly regarding event logs, process models, and conformance checking, will be revisited and extended within the object-centric framework, enabling a more comprehensive understanding of complex real-world processes.

Chapter 3

OCPM Foundations

Introduction

This chapter introduces the core concepts and formal foundations of Object-Centric Process Mining (OCPM) by building upon contributions developed by the author of this thesis, which revolve around the standardization and feature extraction of Object-Centric Event Logs (OCELs), as well as scalable data representations. These contributions include:

- The standardization of OCELs and their underlying model, supported by the OCEL standard [53], and the recent OCEL 2.0 resources [67]. These works provide a formal and extensible basis for representing multi-object data, ensuring compatibility and scalability of object-centric logs.
- The development of techniques for extracting meaningful features from object-centric event logs [13], enabling deeper analyses of object behavior and relationships.

By building on the standardized representation of OCELs and leveraging established feature extraction methods, this chapter illustrates how object-centric data can be systematically understood, manipulated, and analyzed. We also discuss how these OCELs can be flattened into conventional event logs, thus bridging the object-centric perspective with traditional process mining tools, while highlighting the inherent limitations of the single-case viewpoint. Through this synthesis of standards, methodologies, and techniques, the chapter equips the reader with a robust understanding of the underlying principles of OCPM, setting the stage for subsequent explorations into more advanced methods and applications.

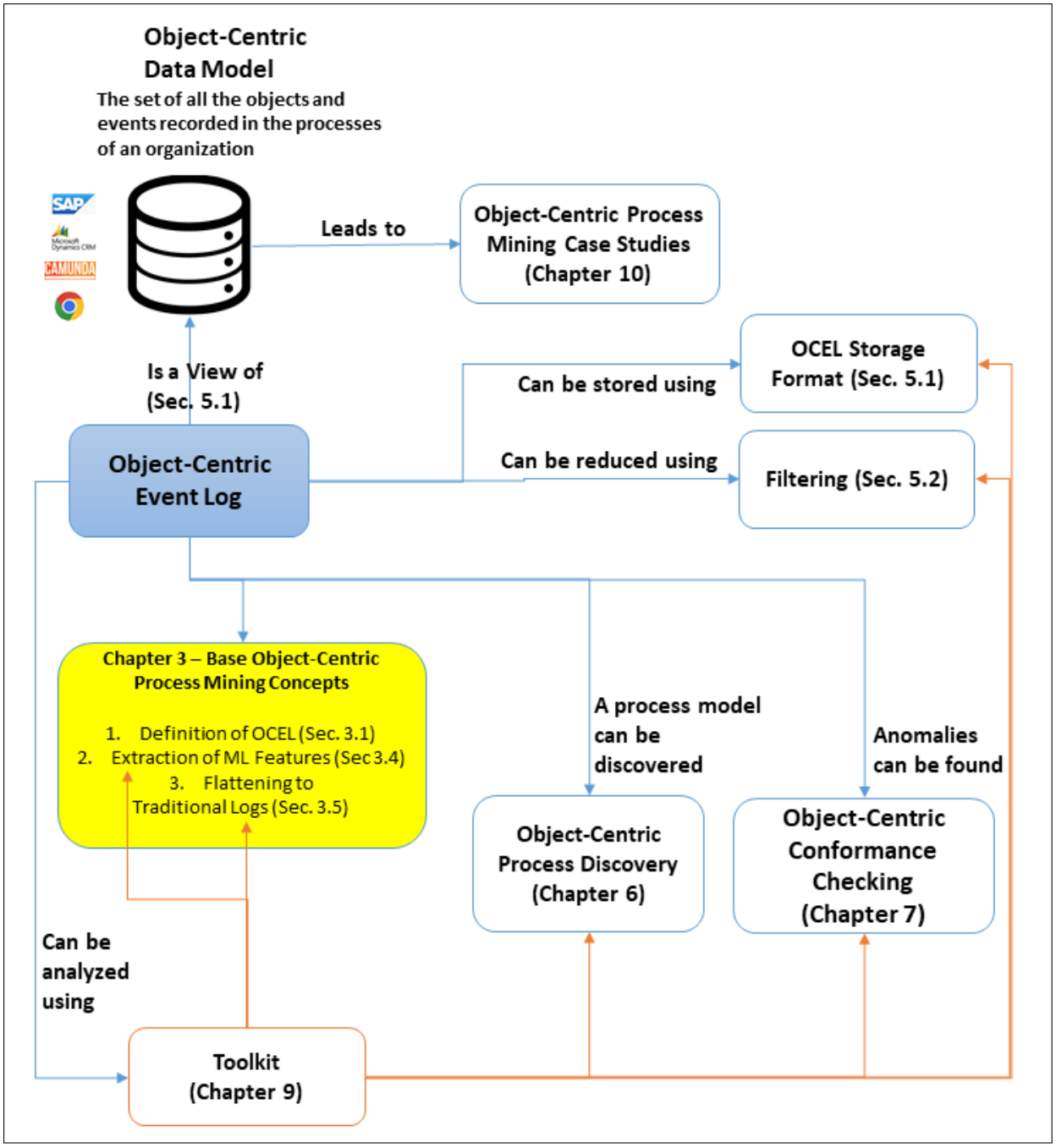


Figure 3.1: Highlight of the problems tackled in this specific chapter.

3.1 Basic Object-Centric Concepts

In the context of process mining, an OCEL presents a paradigm that extends beyond the traditional process perspective, incorporating multi-dimensional analysis that caters to complex business processes. This model shifts the focus from a purely case-centric approach, emphasizing the role of entities, termed "objects", which interact and evolve throughout the process execution. An OCEL, therefore, contains a collection of events, where each event corresponds to a change in state or activity associated with one or more objects. Each object, categorized by its "object type", embodies a particular role or entity within the process, such as a purchase order or a supplier in a procurement process. These objects interact with each other through events, thus defining a rich set of interrelations that depict the process's dynamics. The examination of these objects and their interactions via the event log offers insights into the complexity and nuances of business processes, thereby enabling enhanced process analysis, optimization, and control. The basic concepts related to OCELs are the following:

- *Events*: The most basic unit of an OCEL is the event. These represent the various actions or activities that occur within a system or process, such as approving an order, shipping an item, or making a payment. Every event is unique and corresponds to a specific action at a specific point in time. These events serve as the bedrock of process analysis, providing detailed insights into what actions are taking place, when they occur, and how they contribute to the overall flow of the process.

- *Objects*: In an OCEL, objects represent the entities that the events act upon. These might be physical items like products in a supply chain, or abstract entities like orders, invoices, or contracts in a procurement process. Objects are integral as they help establish the context of events, making it possible to understand not just what actions are taking place, but also what they are affecting or influencing.

- *Event Types (Activities)*: Events are often categorized into different types based on their nature or function. For example, in a procurement process, you might have event types such as "Order Created", "Order Approved", "Invoice Sent", and so on. Each type of event represents a specific kind of action that can take place in the process. Event types allow us to group similar events together and analyze patterns or trends within these groups.

- *Object Types*: Just as events can be categorized into types, so too can objects. Object types might include categories like "Product", "Order", "Invoice", "Supplier", and so on. By grouping similar objects into types, it is possible to draw insights about how different categories of objects are acted upon within the process. This can be invaluable for identifying areas for improvement or optimization.

- *Event-to-Object Relationships (E2O)*: At the heart of an OCEL lies the relationships between events and objects. Every event is typically connected to one or more objects through various relationships such as "affects", "modifies", or "initiates". These qualifiers illustrate how the actions in a process are impacting the entities within that process. By studying these Event-to-Object relationships, it is possible to trace the lifecycle of an object through a process, understand the sequence of events that an object undergoes, and identify dependencies or bottlenecks within the process.

- *Object-to-Object Relationships (O2O)*: Additionally, OCELs often contain relationships between objects themselves, which can be crucial for understanding the interconnectedness of the system. For instance, these relationships might be "associated with", "part of" or "dependent on" one another. For example, a "Product" might be "part of" a specific "Order" and an "Invoice" might be "dependent on" the approval of an "Order". Analyzing these Object-to-Object relationships helps in understanding the complex web of interactions within a process, offering deeper insights into the structural dependencies that affect process outcomes.

Events and objects in OCELs often carry various attributes that provide additional details about them. Attributes of events might include timestamps, locations, and involved personnel, each offering more context about when, where, and by whom an event was carried out. Similarly, objects carry attributes such as status, quantity, or value, which describe their properties at any given moment.

In the context of an object-centric process, it is important to note that objects have evolving attribute values. This evolution reflects changes over time, driven by the events they encounter. For instance, the status of an order object might change from "pending" to "approved" following an "Order Approved"

event. This capability to track changes in object attributes allows for dynamic monitoring and analysis of processes. It helps identify trends, predict outcomes, and improve decision-making processes by providing a more comprehensive, real-time picture of the object's lifecycle within the business environment. Such detailed and evolving information is essential for optimizing operations, enhancing the accuracy of analytics, and ensuring effective process management.

In the realm of OCELs, the term "lifecycle" refers to the chronological sequence of events that an object undergoes during its existence within a process. From creation to completion or disposal, the lifecycle of an object encapsulates its entire journey and interactions within the system. This concept is fundamental in providing a comprehensive understanding of how entities (objects) progress and are manipulated through various stages in a given process. Moreover, it aids in deciphering the Event-to-Object relationships, as each event typically corresponds to a particular stage in the object's lifecycle. Evaluating an object's lifecycle offers profound insights into the system's operational efficiency, potential bottlenecks, process compliance, and overall performance. This evaluation also facilitates a deeper comprehension of dependencies between different object types, paving the way for effective process optimization and control.

3.2 Running Examples - Enterprise Resource Planning (ERP) processes

We introduce two running examples for the Purchase-to-Pay (P2P) and Order-to-Cash (O2C) process, which will be helpful to present the Event-to-Object and Object-to-Object relationships, and different Object-to-Object interactions which could be inferred from Event-to-Object relationships. The reader's understanding will be helped by visualizing the relationships on top of these popular and well-known ERP processes.

3.2.1 Purchase-to-Pay (P2P)

The Purchase-to-Pay (P2P) process is a fundamental business operation describing the full cycle of events from the initial stage of creating a purchase requisition, to the final stage of making a payment. The process commences with the need for a good or service, after which a purchase requisition is created and approved. This leads to the creation and sending of a purchase order to the supplier, bound by the terms of a contract.

The supplier then fulfills the order, resulting in the receipt of delivery at the designated warehouse. This triggers the check of goods received against the purchase order. Upon validation, an invoice is created and approved, subsequently sent to the bank for processing. Finally, the payment is made, marking the completion of the P2P process.

Each step in the P2P process is associated with different object types including Purchase Requisition, Purchase Order, Delivery, Goods Receipt, Invoice, Payment, Bank, Supplier, Contract, and Warehouse. These objects help in establishing a structured and traceable process that aids in financial planning, inventory management, and supplier relationship management.

The P2P process involves several key object types:

- *Purchase Requisition*: A formal request to start the P2P process, needing approval to proceed.
- *Purchase Order*: A document detailing the goods or services ordered.
- *Delivery*: The act of supplying the ordered goods or services.
- *Goods Receipt*: The physical receipt of goods, checked for accuracy.
- *Invoice*: A bill for the delivered goods or services.
- *Payment*: Completes the process by paying the invoice.
- *Bank*: Manages the financial transaction.
- *Supplier*: Provides the goods or services.
- *Contract*: The agreement detailing the transaction terms.
- *Warehouse*: The location for receiving, storing, and checking goods.

Table 3.1: Activities of the P2P Process and their descriptions

Activity	**Description**
Create Purchase Requisition	This is the first step in the P2P process, where an internal need for goods or services is formally documented. A purchase requisition typically includes information about the requested items, quantities, and a suggested supplier.
Approve Purchase Requisition	Once the purchase requisition is created, it must be approved by an authorized person or department. This ensures that the request is necessary, within budget, and aligned with business objectives.
Create Purchase Order	After the approval of the purchase requisition, a purchase order is created. This is a formal document sent to the supplier detailing the items to be purchased, quantities, prices, delivery date, and other terms.
Send Purchase Order	This activity involves sending the created purchase order to the supplier. Once the supplier accepts the purchase order, it becomes a legal contract between the buyer and supplier.
Receive Delivery	This activity occurs when the goods or services ordered are delivered by the supplier. It may involve confirming the arrival of the goods at the warehouse.
Check Goods Receipt	After the goods are received, they are checked against the purchase order to ensure the correct items and quantities have been delivered. Any discrepancies are addressed at this stage.
Create Invoice	Once the goods have been received and checked, the supplier will issue an invoice. This document provides a detailed account of the goods or services provided and the amount due for payment.
Approve Invoice	The invoice is checked against the goods receipt and the original purchase order. If everything matches, the invoice is approved for payment.
Send Invoice	The invoice is sent to the financial institution or relevant department for processing. It includes details of the payment due to the supplier.
Make Payment	The final activity in the P2P process is making payment to the supplier. This is done after the invoice is approved and typically involves transferring funds from the company's bank to the supplier's bank.

The main activities of the P2P process are described in Table 3.1.

In the Purchase-to-Pay (P2P) process, traditional process mining approaches can face significant limitations due to the deficiency, convergence, and divergence problems. The deficiency problem arises when interactions between multiple objects, such as purchase orders, goods receipts, invoices, and payments, cannot be adequately captured by a single case identifier, leading to an incomplete representation of the process. The convergence problem refers to situations where multiple purchase orders are fulfilled with a single delivery or invoice, which can be challenging to depict accurately using a single case identifier. In contrast, the divergence problem occurs when a single purchase order results in multiple deliveries, again complicating the accurate portrayal of the process. These issues highlight the limitations of traditional case-centric process mining approaches in dealing with complex business processes and underscore the need for more sophisticated, object-centric techniques.

3.2.2 Order-to-Cash (O2C)

The Order-to-Cash (O2C) process is a series of business operations revolving around the execution of customer orders, spanning from their inception to the realization of revenue. O2C is vital for businesses dealing with physical or digital goods and services sold directly to customers, as it governs crucial aspects of customer service, logistics, and financial management.

The O2C process typically starts with a customer placing an order for a product or service. The organization then checks the customer's creditworthiness and confirms the order. The inventory is checked to ensure that the ordered items are available for dispatch. If the items are available, they are issued from the inventory, packed, and dispatched for delivery.

Alongside these steps, a delivery note is generated, signifying that the order is on its way to the customer. Once the goods are dispatched, an invoice is created and sent to the customer, detailing the cost of the purchased items. After receiving the invoice, the customer remits payment to the company.

All activities are carefully logged and tracked in the company's Customer Relationship Management (CRM) system to maintain comprehensive records of customer interactions and transactions, which can then be leveraged for strategic planning and decision-making.

The O2C process plays a pivotal role in maintaining customer satisfaction, managing inventory effectively, ensuring timely delivery of goods and services, and most importantly, securing the company's cash flow. It is a multi-disciplinary process that necessitates coordination across several departments within a company, including sales, finance, operations, logistics, and customer service.

The O2C process encompasses several primary object types:

- *Customer Order*: Details the customer's order.
- *Customer*: The entity placing the order.
- *Credit Check*: Assesses customer's credit before order approval.
- *Sales Contract*: The sale terms between company and customer.
- *Inventory*: The stock of goods, checked for order fulfillment.
- *Goods Issue*: Records goods taken from inventory for an order.
- *Delivery Note*: Document details for goods dispatched.
- *Transportation*: Details about goods dispatch.
- *Invoice*: Bills the customer for the purchase.
- *Bank*: Processes payments from customers.
- *Payment*: Records received payments.
- *Warehouse*: Stores and prepares goods for orders.
- *CRM*: Tracks customer interactions and orders.

The main activities of the O2C process are contained in Table 3.2.

In the Order-to-Cash (O2C) process, traditional process mining approaches may stumble upon the deficiency, convergence, and divergence problems. The deficiency problem in this context is notable when dealing with interconnected events, such as customer orders, inventory checks, goods issues, and payments, which might not be holistically encapsulated using a singular case identifier. Convergence becomes an issue when multiple orders are served by a single inventory check or goods dispatch, challenging the model's precision when only a single case identifier is used. Divergence, on the other hand, arises when a single customer order triggers multiple inventory checks or results in several goods issues, leading to complexities in process representation. These issues underline the insufficiency of conventional case-centric process mining techniques in capturing the intricacies of the O2C process, advocating for the adoption of more advanced, object-centric methodologies.

3.2.3 Small Example

In this section, we present a concise example of an OCEL, with the goal of illustrating how these logs can effectively capture complex, multi-object process instances. For this, we have opted for a tabular representation that offers a clear overview of how events, their corresponding activities, and involved objects interrelate over time. This representation, though seemingly simple, brings forward the intricacies of multi-object interplay in process execution, and is particularly suited for visualizing smaller logs. Nevertheless, it is important to note that as the complexity and size of logs increase, tabular representation becomes less feasible, prompting the need for more sophisticated visualization or storage solutions.

Table 3.2: Activities of the O2C Process and Their Descriptions

Activity	**Description**
Create Customer Order	The initiation of the O2C process where a customer formulates an order specifying desired products or services. The system captures all relevant details such as items, quantities, and customer information necessary for subsequent steps.
Check Credit	This step assesses the creditworthiness of the customer to ensure they are eligible for the transaction. It involves reviewing credit scores and payment histories to mitigate financial risks before proceeding with the order.
Confirm Order	This activity signifies the approval of the customer order following a successful credit check. It sets in motion the fulfillment operations and communicates order confirmation to the customer.
Check Inventory	Prior to fulfilling an order, this step verifies the availability of required items within the inventory. It ensures that all ordered products are in stock and allocates resources for upcoming dispatch.
Issue Goods	This activity marks the retrieval of goods from inventory specifically for the order in question. It involves updating inventory records to reflect reduced stock levels.
Pack Goods	This step prepares the issued goods for shipment. It includes securely packaging the products to ensure they remain undamaged during transportation.
Dispatch Goods	The physical movement of packed goods from the warehouse towards the customer's designated location. This step also triggers updates in the system to track the shipment's progress.
Generate Delivery Note	Alongside dispatching goods, a delivery note is produced. This document accompanies the shipment, detailing the contents and providing proof of dispatch.
Create Invoice	Post-dispatch, an invoice is generated reflecting the total cost associated with the purchased goods. This document is prepared to request payment from the customer.
Send Invoice	The completed invoice is sent to the customer, typically via email or postal service. This action formally requests the payment due for the goods delivered.
Receive Payment	The final step in the O2C process where payment is collected from the customer. The payment is processed and recorded, officially concluding the transaction.

Table 3.3 provides a log excerpt from a Purchase-to-Pay process. Each row corresponds to an event, associated with a specific timestamp and an activity, as well as the various objects implicated in the activity. The objects are categorized per their types - *Purchase Requisition*, *Purchase Order*, *Quality Checks*, *Invoices*, and *Payments* - and are presented as lists under the respective columns. Blank cells denote the absence of involvement of a certain object type in the corresponding event. This table underscores the inherent richness and complexity of OCELs, serving as a springboard for our subsequent discussions on mining techniques that can harness this richness for more nuanced process insights.

3.3 Definition of OCEL

3.3.1 Basic Definitions

At the heart of OCPM is the Object-Centric Event Log (OCEL), which records events and their associations with various objects. The OCEL serves as a rich source of information, enabling the discovery of intricate process patterns and object interactions that are not easily discernible in traditional event logs. By formalizing the definition of OCEL, we establish a common language and framework that facilitate the analysis and modeling of object-centric processes.

Table 3.3: Example OCEL of a procurement process represented as a table.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	['pr1']				
e2	Close Purchase Requisition	2021-03-20 14:00	['pr1']				
e3	Create Purchase Requisition	2021-03-21 09:30	['pr2']				
e4	Create Purchase Order	2021-03-22 14:59	['pr2']	['po1']			
e5	Invoice Receipt	2021-03-25 11:00		['po1']		['r1']	
e6	Perform Payment	2021-03-30 11:58				['r1']	['p1']
e7	Create Purchase Requisition	2021-04-01 09:15	['pr3']				
e8	PR Formal Approval	2021-04-01 10:15	['pr3']				
e9	Create Purchase Order	2021-04-02 17:00	['pr3']	['po2']			
e10	Change Purchase Requisition	2021-04-03 10:00	['pr3']				
e11	Invoice Receipt	2021-04-05 15:00		['po2']		['r2']	
e12	Perform Payment	2021-04-15 09:27				['r2']	['p2']
e13	Create Purchase Order	2021-04-17 14:29		['po3']			
e14	Invoice Receipt	2021-04-28 10:00		['po3']		['r3']	
e15	Perform Payment	2021-04-30 15:00				['r3']	['p3']
e16	Invoice Receipt	2021-05-28 10:01		['po3']		['r4']	
e17	Perform Payment	2021-05-30 15:17				['r4']	['p4']
e18	Invoice Receipt	2021-06-28 10:01		['po3']		['r5']	
e19	Perform Payment	2021-06-30 15:29				['r5']	['p5']
e20	Create Purchase Requisition	2021-07-01 11:15	['pr4']				
e21	Create Purchase Order	2021-07-02 09:38	['pr4']	['po4']			
e22	Invoice Receipt	2021-07-09 16:00		['po4']		['r6']	
e23	Quality Check	2021-07-11 10:30		['po4']	['qc1']		
e24	Perform Payment	2022-05-15 09:00				['r6']	['p6']
e25	Invoice Receipt	2022-05-20 12:00				['r7']	
e26	Create Purchase Order	2022-05-20 15:00		['po5']		['r7']	
e27	Create Purchase Order	2022-06-01 09:17		['po6']			
e28	Create Purchase Order	2022-06-02 11:48		['po7']			
e29	Create Invoice	2022-06-05 09:00		['po6','po7']		['r8']	

Definition 13 (OCEL). *An OCEL is a tuple* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *with:*

- $A \subseteq U_{etype}$ *is a set of event types (activities).*
- $OT \subseteq U_{otype}$ *is a set of object types.*
- $E \subseteq U_{ev}$ *is the set of events.*
- $O \subseteq U_{obj}$ *is the set of objects.*
- $evtype : E \to A$ *assigns types to events.*
- $time : E \to U_{time}$ *assigns timestamps to events.*
- $EA \subseteq U_{attr}$ *is the set of event attributes.*
- $eatype : EA \to A$ *assigns event types to event attributes.*
- $eaval : (E \times EA) \not\to U_{val}$ *assigns values to event attributes (not all the attributes are mapped for each event).*
- $objtype : O \to OT$ *assigns types to objects.*
- $OA \subseteq U_{attr}$ *is the set of object attributes.*
- $oatype : OA \to OT$ *assigns object types to object attributes.*
- $oaval : (O \times OA \times U_{time}) \not\to U_{val}$ *assigns values to object attributes.*
- $E2O \subseteq E \times U_{qual} \times O$ *are the qualified event-to-object relationships.*
- $O2O \subseteq O \times U_{qual} \times O$ *are the qualified object-to-object relationships.*
- $\pi_{omap} : E \to \mathcal{P}(O)$ *is such that for any* $e \in E$, $\pi_{omap}(e) = \{o \in O \mid \exists_{q \in U_{qual}}(e, q, o) \in E2O\}$.
- $\leq$ *defines a total order on the events (which could be based on the timestamp and the lexicographic order).*

such that:

- $dom(eaval) \subseteq \{(e, ea) \in E \times EA \mid evtype(e) = eatype(ea)\}$ *to ensure that only existing event attributes can have values.*
- $dom(oaval) \subseteq \{(o, oa, t) \in O \times OA \times U_{time} \mid objtype(o) = oatype(oa)\}$ *to ensure that only existing object attributes can have values.*

Definition 13 is equivalent to the one proposed in the OCEL 2.0 standard [16]. Diagrams showcasing the connections between event types (or activities) and object types serve a crucial role in understanding the intricate relationships within a process. These graphical representations, although seemingly simple, can effectively demonstrate the involvement of different object types in various activities. These diagrams offer a bird's-eye view of the entire process, indicating what types of objects are being acted upon by which activities. This information, at a high level, simplifies the complexity of the process, enabling an easy understanding of the various interplays at work. Moreover, these diagrams can also be used for data validation purposes. If an activity is associated with an object type that it shouldn't be, it could indicate a problem in the data or the process itself. Figure 3.2 shows the main event types (activities) and object types of a P2P process, along with their interconnection. For the O2C process, Figure 3.3 shows the main event types (activities) and object types, along with their interconnection.

Attributes: Every event and object in an OCEL can be understood better when additional attributes are associated with them. These attributes provide more specific, personalized information about each event or object, going beyond the standard details like event/object type and timestamp. Consider, for instance, a process event such as "Approve Invoice" in a Purchase-to-Pay (P2P) scenario. We can associate additional attributes to this event, such as "Approver Role", "Approval Time", "Department", or "Approval Type". These added attributes enable us to perform more detailed analyses, such as examining approval times across different departments, investigating approval types, or exploring how the role of the approver influences the approval process. Similarly, when considering objects, attributes can provide more comprehensive insights. For instance, in an Order-to-Cash (O2C) process, a "Customer Order" object could have attributes like "Order Value", "Order Quantity", "Order Priority", or "Customer Region". These attributes could be instrumental in profiling orders, understanding customer behavior, assessing regional sales patterns, or evaluating operational efficiency concerning order priorities.

Qualified Event-to-Object Relationships: In an OCEL, events are associated with multiple objects, and the nature of these relationships is often more complex than can be captured by a simple binary association. To better convey this complexity, each Event-to-Object relationship can be assigned a qualifier. A qualifier describes the role played by an object in an event, adding additional context to the relationship and making the process behavior more explicit.

Consider the Purchase-to-Pay (P2P) process as an example (Figure 3.4). In this process, events related to procurement, goods receipt, invoicing, and payment are interconnected through various objects such as purchase requisitions, purchase orders, deliveries, invoices, and payments. These relationships can be enriched using qualifiers. For instance, in the "Create Purchase Requisition" event, the qualifier "Creates" describes the role of the purchase requisition object. Similarly, the "Send Purchase Order" event can have multiple qualifiers. It "Sends" the purchase order and also "Sends to" the supplier, establishing the destination of the purchase order. The qualifier "Under" attached to the relationship with the contract object suggests that the purchase order is sent according to a certain contract. In the "Receive Delivery" event, the "Fulfills" qualifier shows the connection of the delivery to the completion of a purchase order. Meanwhile, "Stores in" indicates that the delivery is stored in a warehouse. Such qualifiers provide a richer understanding of the process, revealing how different events use or affect different objects in a specific context. In conclusion, the incorporation of qualifiers in Event-to-Object relationships provides a more detailed and contextually relevant representation of OCELs. Figure 3.6 shows some qualified Object-to-Object relationships that exist in a P2P process.

The O2C process traces the path from receiving a customer's order to the final payment collection. This journey involves multiple stages, each entailing various events and objects. The use of qualifiers (Figure 3.5) provides context, explaining the role of each object in relation to the event. For instance, the "Create Customer Order" event involves the "Customer Order" object with the qualifier "Creates", indicating the initiation of an order. The "Places from" qualifier associated with the "Customer" object signals that the order originates from a specific customer. Similarly, the qualifier "Records in" connected to the "CRM" (Customer Relationship Management system) object represents the recording of the order in the CRM. As the O2C process progresses, the "Check Credit" event uses qualifiers like "Performs" for "Credit Check" and "Checks for" for "Customer" to signify the credit assessment activity for the customer. The "Confirm Order" event "Confirms" the "Customer Order" and does so "Under" the

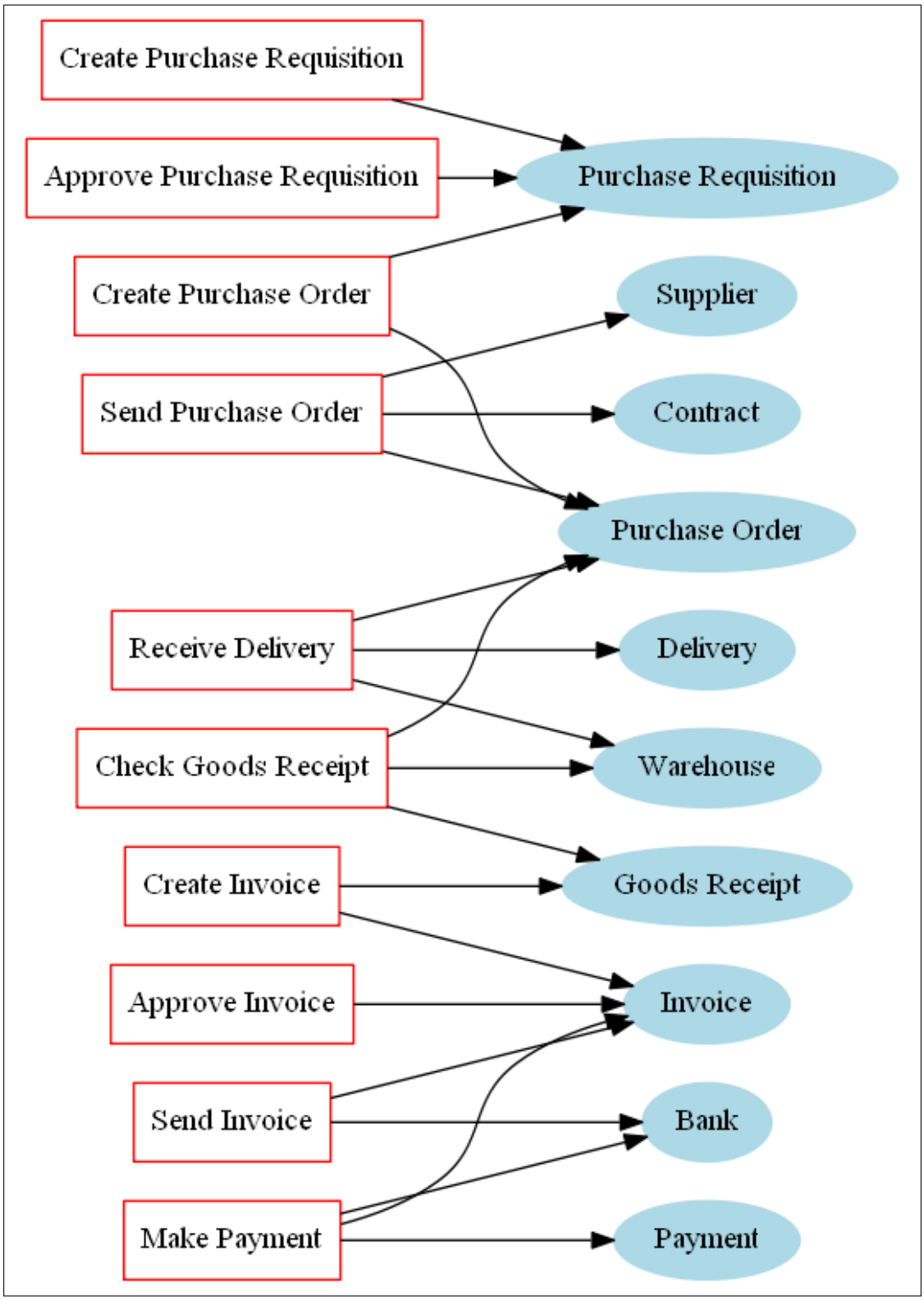


Figure 3.2: Main event and object types of a P2P process, along with their interconnection.

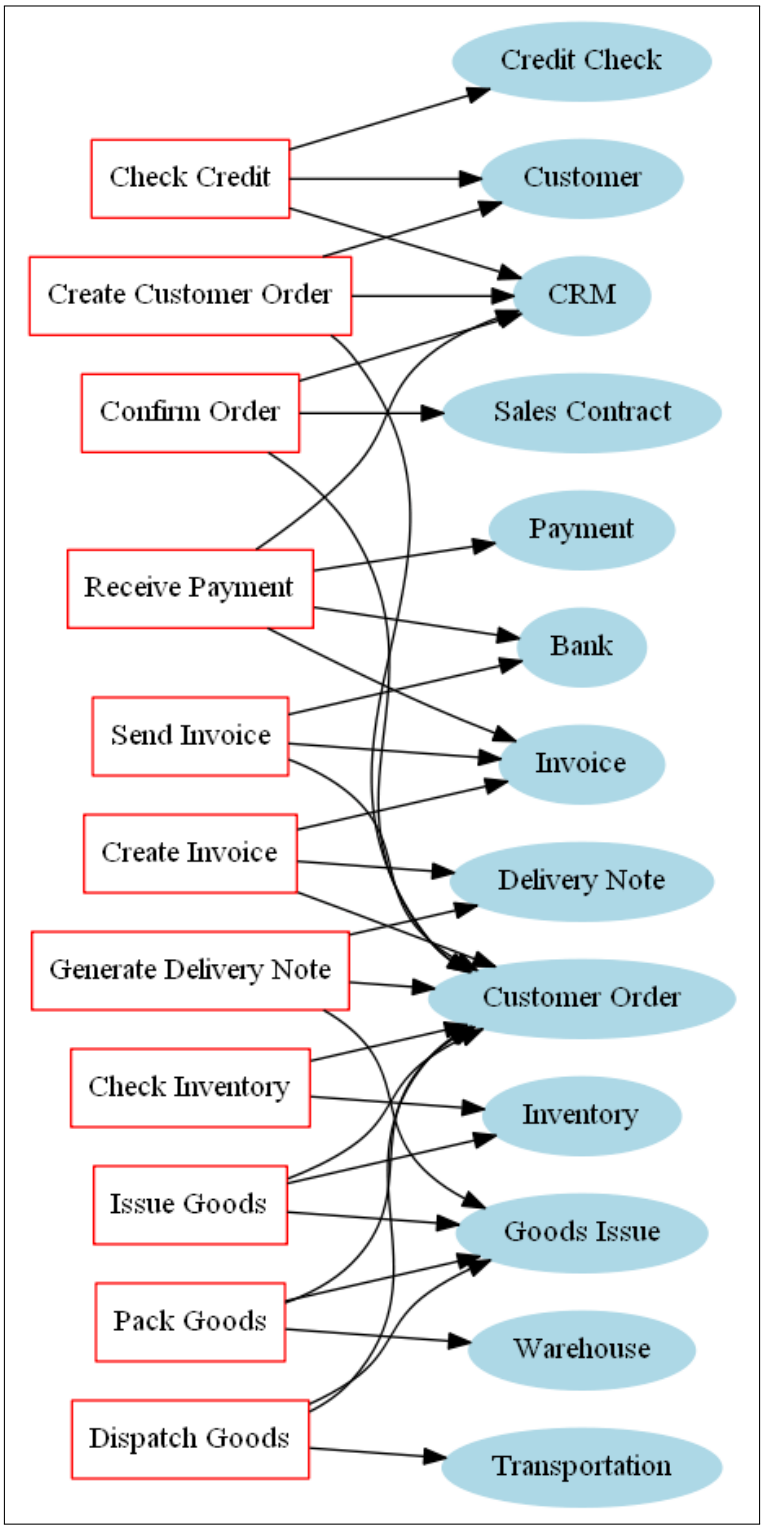


Figure 3.3: Main event and object types of an O2C process, along with their interconnection.

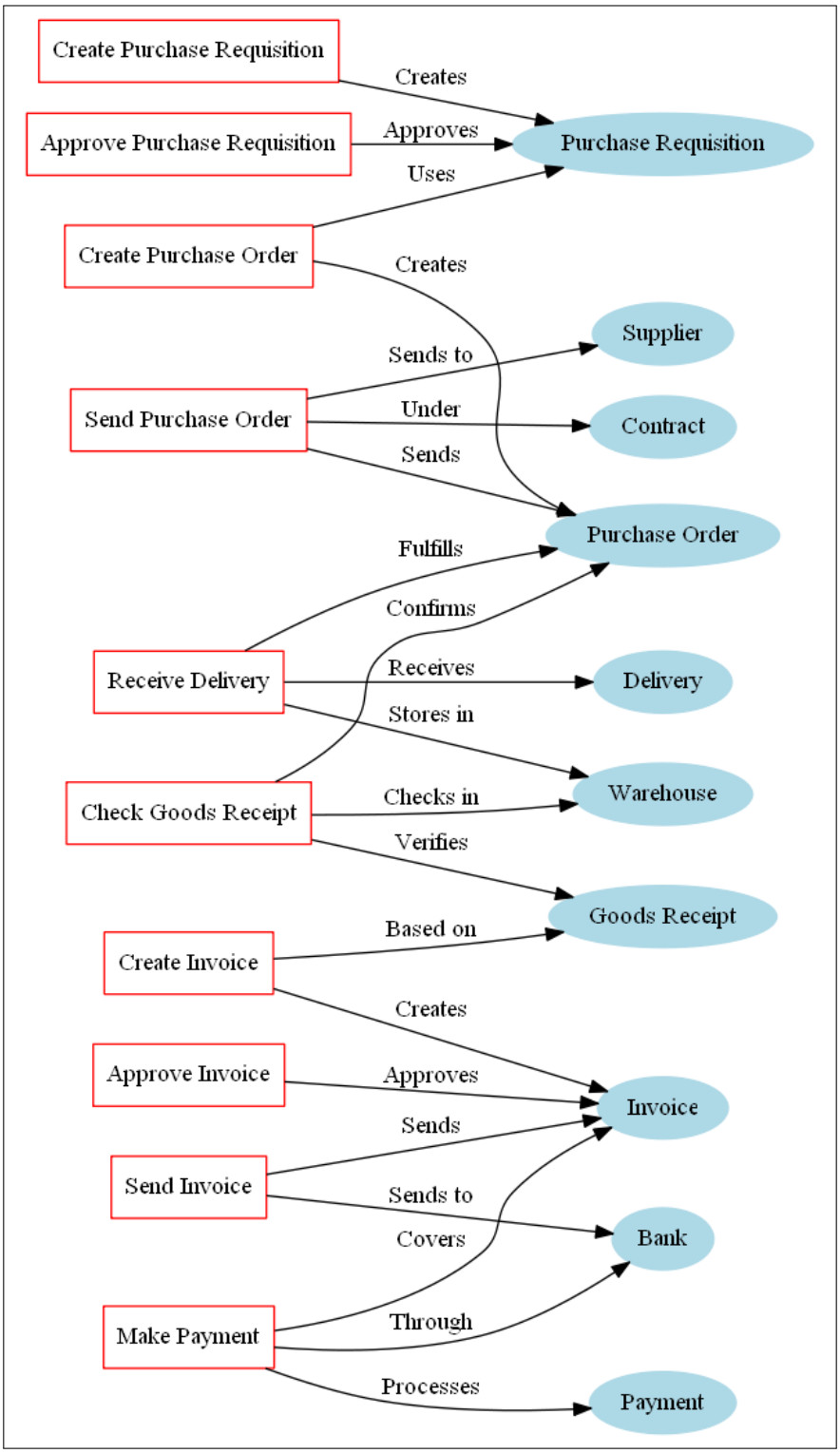


Figure 3.4: Possible qualifiers associated to the Event-to-Object relationships of a P2P process.

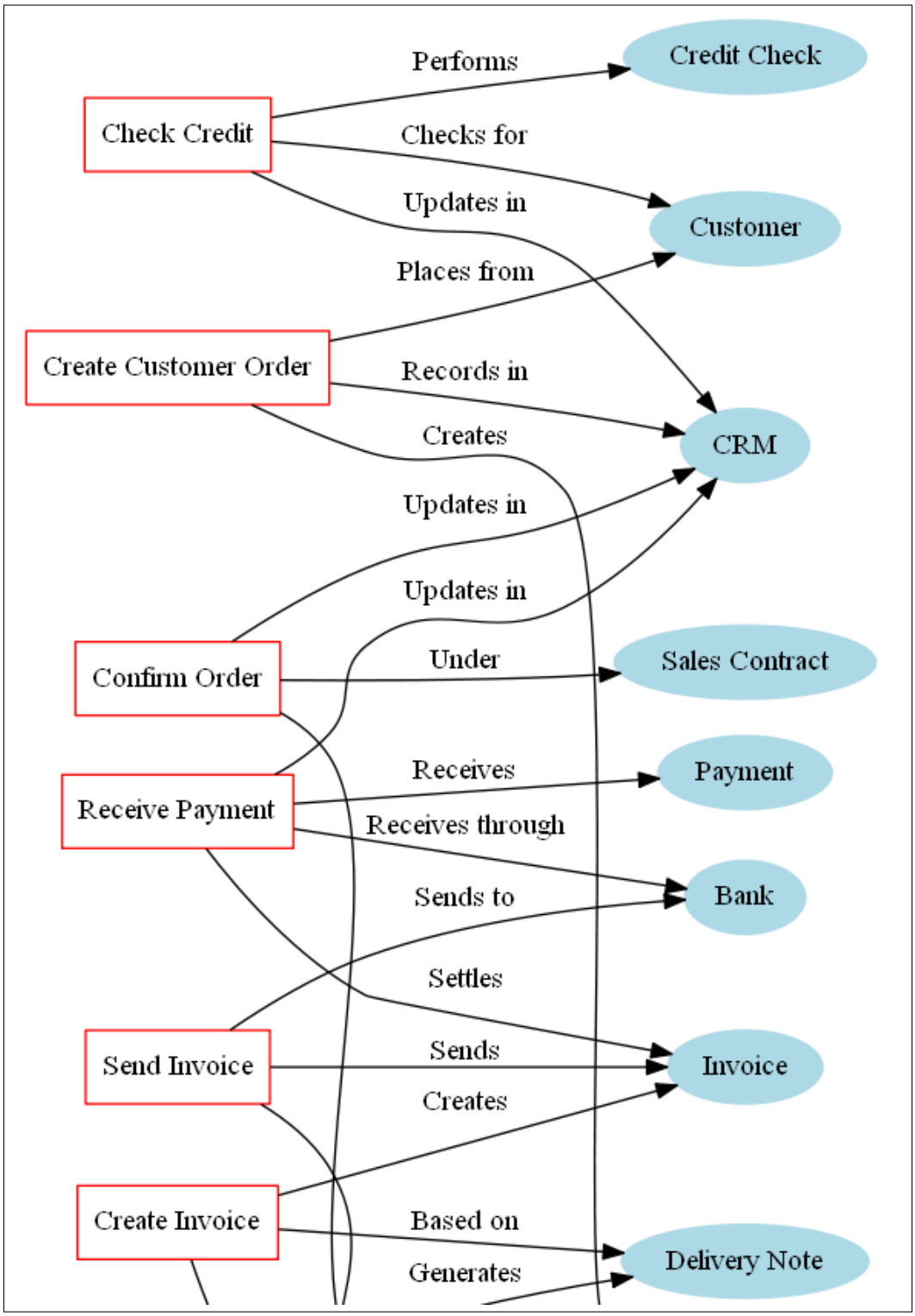


Figure 3.5: Possible qualifiers associated to the Event-to-Object relationships of an O2C process.

"Sales Contract", providing a legal basis for the transaction. In the delivery stage, the event "Issue Goods" "Processes" the "Goods Issue", "Reduces" the "Inventory", and "Fulfills" the "Customer Order". The "Pack Goods" event "Packs" the "Goods Issue" and does it "In" the "Warehouse". It also "Packs for" the "Customer Order", thereby providing a clear view of the goods' path from the warehouse to the customer. The later events of "Create Invoice", "Send Invoice", and "Receive Payment" involve qualifiers like "Creates", "Sends", "Sends to", "Receives", and "Settles" to establish the relationships with the objects like "Invoice", "Bank", and "Payment". These qualifiers demonstrate the flow of actions from invoice creation to its delivery to the bank, and finally, the receipt of payment.

Qualified Object-to-Object Relationships: In process analysis, objects often interact or have relationships with other objects, creating an intricate web of connections that influence how the process unfolds. Object-to-Object relationships add a layer of interconnectedness between the objects involved in the process. It pairs objects that have some form of relationship, highlighting the network of interactions that often drive the progression of the process. Nevertheless, simply knowing that two objects are connected isn't always sufficient. In real-world scenarios, the nature of the relationship can have a profound impact on the process. For instance, the relationship between a purchase order and an invoice is fundamentally different from the relationship between a purchase order and a delivery note, although all these objects are part of the same process. Therefore, we add a qualifier to each pair of related objects, effectively "typing" the relationship to provide additional context and meaning. This way, we can distinguish between different types of relationships and better understand their influence on the process.

Figure 3.7 shows some qualified Object-to-Object relationships that exist in an O2C process.

We also introduce the concept of an Object Graph Enrichment, as presented in Definition 14, to define custom relationships upon an OCEL. Introducing the concept of a custom Object Graph Enrichment in the context of an OCEL is highly beneficial, even when predefined Object-to-Object (O2O) relationships already exist. The key reason for this capability is the dynamic nature of real-world processes, where not all relevant or useful relationships are defined at the time of log creation. Custom Object Graph Enrichments allow for flexibility in analyzing these relationships beyond the static definitions typically captured in an OCEL.

Definition 14 (Object Graph Enrichment)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, an Object Graph Enrichment is a tuple* $GE = (L, \Lambda)$*, where* $\Lambda \subseteq O \times O$*.*

Consider a scenario in a manufacturing process where an OCEL tracks events and objects such as parts, machines, and assemblies. Predefined O2O relationships might detail which parts are used in which machines, or which parts are included in specific assemblies. However, during a detailed analysis, it might become necessary to explore relationships that are not predefined in the log. Suppose an analysis needs to determine which parts and assemblies were in proximity to a particular machine during a specific time window when an unexpected machine downtime occurred. A custom Object Graph Enrichment can dynamically link these parts and assemblies based on the temporal data and proximity rather than traditional fixed relationships, helping to pinpoint potential causes related to specific batches of parts or assemblies. This type of analysis requires creating temporary, context-specific relationships that are not originally stored in the OCEL because they depend on the specific parameters of the analysis, such as the time window of interest or specific events like machine downtime. Storing all possible combinations of relationships in the original event log would be impractical and inefficient due to the vast amount of data and the variability of analysis needs.

Definition 15 (Object Interaction Enrichment)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we define the* Object Interaction Enrichment *as* $GE_{IG} = (L, \Lambda_{IG})$ *where* $\Lambda_{IG} = \{(o_1, o_2) \in O \times O \mid \exists_{e \in E} o_1, o_2 \in \pi_{omap}(e)\}$

The Object Interaction Enrichment (which is an Object Graph Enrichment) introduced in Definition 15 plays a crucial role in understanding and visualizing the interactions between objects within an OCEL. As per Definition 15, this graph is formulated by considering each object as a node, and creating an edge between two nodes if there exists at least one event in the log that references both of these objects. By examining this graph, one can infer patterns of object interactions, dependencies, and correlations that could provide some insights.

Objects' Lifecycle

In process analysis, one of the central concepts is the "lifecycle of an object". This concept essentially refers to the series of events or activities an object undergoes from its creation until its conclusion within

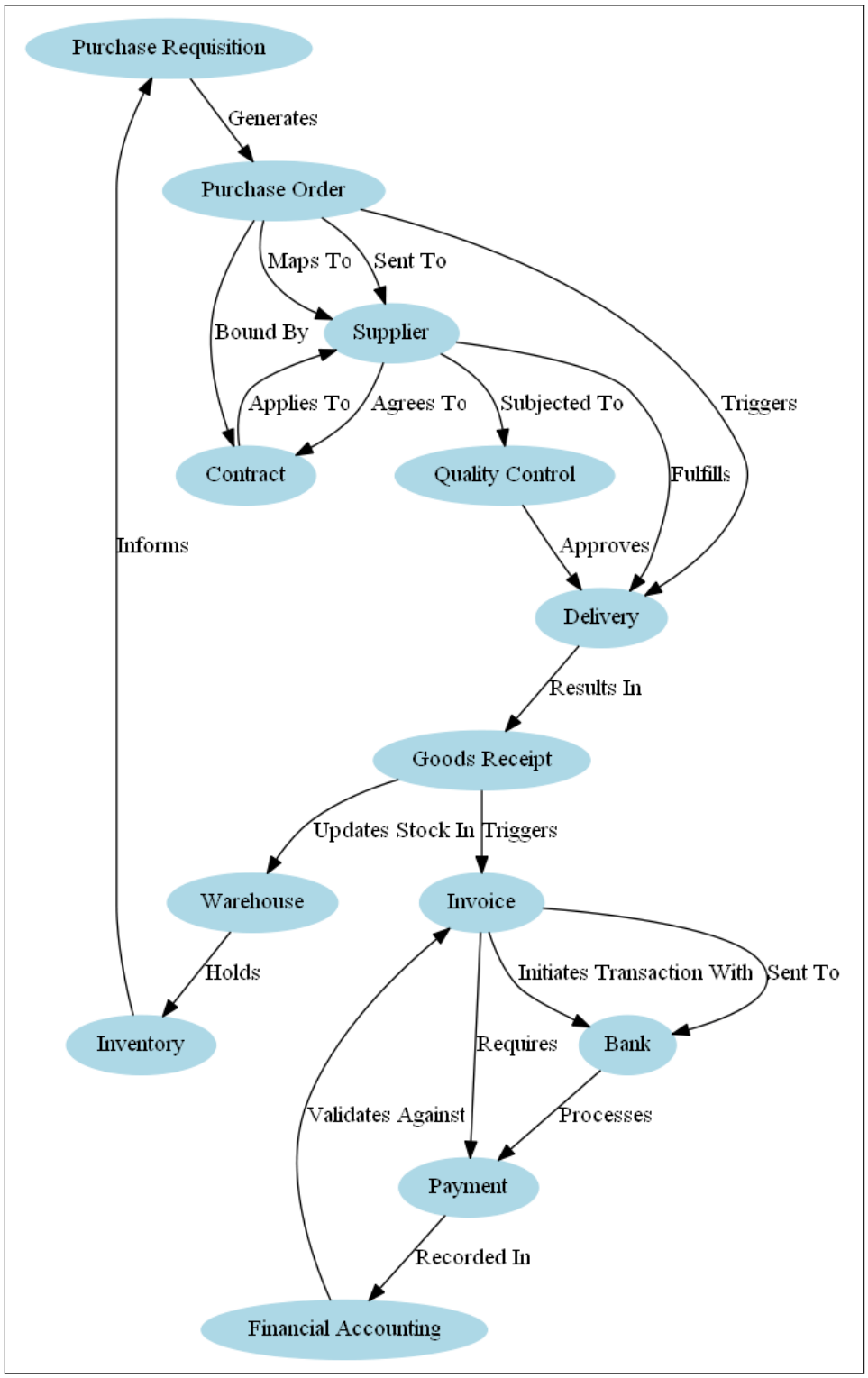


Figure 3.6: Qualified Object-to-Object relationships in a P2P process.

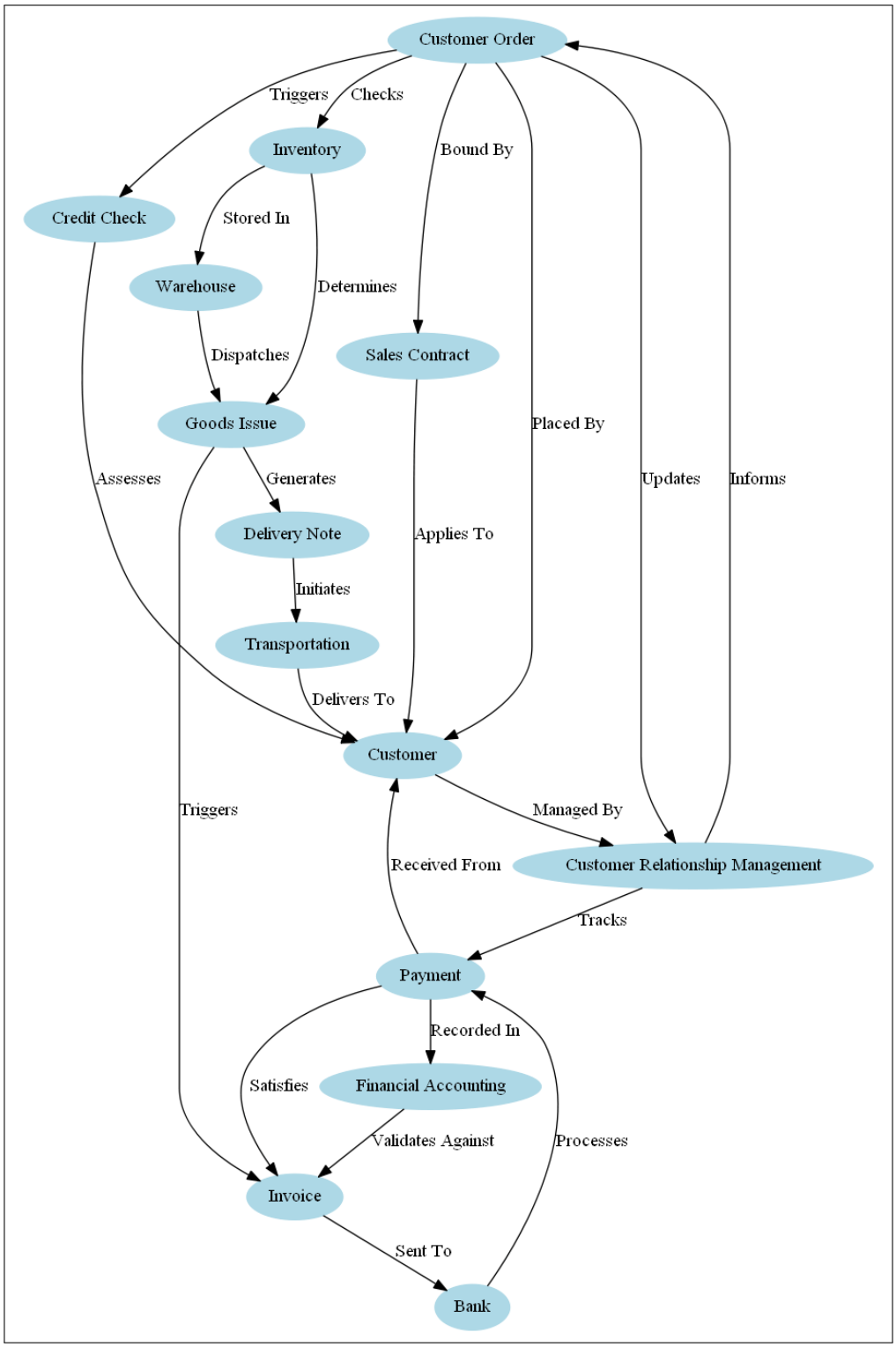


Figure 3.7: Qualified Object-to-Object relationships in an O2C process.

a process. It is like tracing the journey of an object through the process map, identifying each step or activity it participates in along the way. This concept is defined formally in Definition 16. Here, an OCEL (as defined earlier) is the context for an object and its associated lifecycle. The lifecycle of a specific object, denoted as $lif(o)$, is the ordered sequence of events that the object is related to. In other words, these are all the events where this object plays a role, arranged chronologically. This definition also introduces the "start" and "end" events of an object's lifecycle. The start event is the very first event in which the object participates, marking the object's entrance into the process. Conversely, the end event is the last event that involves the object, signifying the conclusion of its journey in the process.

Definition 16 (Lifecycle of an Object)**.** *Let* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *be an OCEL. Given an object* $o \in O$*, we define* $lif(o) = \langle e_1, \ldots, e_n \rangle$ *such that:*

- $\{e_1, \ldots, e_n\} = \{e \in E \mid o \in \pi_{omap}(e)\}$
- $e_i < e_{i+1} \ \forall \ i \in \{1, \ldots, n-1\}$

We define the start event $start(o) = lif(o)(1)$ *and the end event* $end(o) = lif(o)(|lif(o)|)$*.*

The lifecycle of an object is crucial in process analysis as it allows us to trace the journey of an object through the process. This can help identify bottlenecks, inefficiencies, or irregularities that could be targeted for process improvement. Definition 17 introduces the lifecycle of a set of objects.

Definition 17 (Lifecycle of a Set of Objects)**.** *Let* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *be an OCEL. Given a set of objects* $O' \subseteq O$*, we define* $lif_S(O') = \langle e_1, \ldots, e_n \rangle$ *such that:*

- $\{e_1, \ldots, e_n\} = \{e \in E \mid \exists_{o \in O'} o \in \pi_{omap}(e)\}$
- $e_i < e_{i+1} \ \forall \ i \in \{1, \ldots, n-1\}$

3.4 Ambiguity in Relationships and Ordering of Objects in OCELs

A key challenge in the realm of object-centric process mining is the ambiguity surrounding the representation of relationships among objects and the corresponding order in which activities occur. Unlike traditional case-centric event logs, where each event is intrinsically associated with a single case notion, an OCEL admits scenarios where multiple objects co-occur within a single event. As a result, determining how objects relate to each other and in what order their associated activities unfold is not always straightforward.

In particular, two different perspectives can be adopted to capture these relationships:

- *Event-to-Object (E2O) Perspective:* In this view, each event is regarded as the source of relational information. If an event e references multiple objects, o_1 and o_2, it implicitly establishes a relationship between these objects within the context of that event. One could say that the event *binds* these objects together at that specific timestamp, thus providing a notion of ordering derived directly from the event timeline. For example, if an event *Receive Delivery* involves both a *Purchase Order* object and a *Warehouse* object, we may infer that the warehouse came into play for that particular order at the time of delivery. The ordering relationship between these objects emerges from their co-occurrence in the event, and the temporal ordering of events themselves.
- *Object-to-Object (O2O) Perspective:* Alternatively, relationships can be defined directly at the object level, without relying solely on events. In this approach, a link between o_1 and o_2 is maintained independently from the events that first introduced them. This O2O relationship might be interpreted as a static or persistent link between objects, indicating, for instance, that a particular *Invoice* object is always associated with a specific *Purchase Order*, regardless of the specific events that manipulate them. While this perspective captures ongoing dependencies or associations between objects, it does not inherently provide a temporal ordering derived from their co-occurrence in events.

Example Illustrating E2O vs. O2O Ambiguities

Consider a simplified procurement scenario where we track two objects: a *Purchase Order (PO)* and a *Supplier* object.

Event-to-Object (E2O) Definition: Suppose there is an event *PO Creation* that references both a *PO* object and a *Supplier* object. From an E2O perspective, this event reveals that these two objects are related at the moment of PO creation. The temporal ordering is straightforward: the *Supplier* is associated with the *PO* from the instant the *PO Creation* event occurs. Should a subsequent event *PO Approval* also reference the same *PO*, we can see how the *PO* moves through the process timeline, with the *Supplier* implicitly linked to each of these events involving the *PO*. The event order directly informs how and when these objects interact.

Object-to-Object (O2O) Definition: Now imagine that, apart from the events, we have a static relationship stating that the *Supplier* is *always* associated with this particular *PO* (as part of a contract or master data linkage). The O2O definition does not tell us at what specific point in time the *Supplier* became relevant. It merely informs us that the *Supplier* is connected to the *PO*, with no intrinsic event-derived ordering. We lose the direct temporal reference that would otherwise be provided by the event log. Without additional rules or assumptions, we do not know if the *Supplier* preceded or followed any other object in the *PO*'s lifecycle; we only know there is a static, persistent relationship.

This example highlights why relationship ambiguity emerges in object-centric event logs: the process of disentangling the timing, ordering, and conditions under which relationships form or influence the lifecycle of objects is non-trivial. Depending on whether one adopts an E2O or O2O perspective, the ordering of objects and the inferred flow of the process can differ substantially.

Analysts must explicitly define the rules and assumptions governing how object relationships are established and interpreted. This may involve filtering and refining these relationships, as well as integrating additional domain knowledge, in order to unambiguously reconstruct the desired object interaction patterns and temporal orderings from the data.

3.5 Feature Extraction on OCELs

In the pursuit of deeper analysis and understanding of OCELs, we now introduce the concept of feature extraction. Feature extraction is a fundamental process in the application of machine learning techniques, as it allows us to represent our data in a form that these techniques can leverage. By mapping each object in our log to a set of features, we can capture key characteristics and behaviors in a succinct and structured manner.

Our first definition (Definition 18) concerns the Object-Based Feature Map. This is a function that assigns to each object a set of numerical values, each representing a specific feature. The set of features, which we denote by Σ, is a chosen collection of characteristics we consider relevant or interesting for our analysis.

Definition 18 (Object-Based Feature Map)**.** *Given a set of objects $O \subseteq U_{obj}$ and a set of features $\Sigma \subseteq U_{\Sigma}$, an Object-Based Feature Map is a function $f : O \to (\Sigma \to \mathbb{R})$.*

To give an idea of the kind of features we can extract, we introduced in Definition 18 a basic feature map. In this map, each object is associated with a variety of features that encapsulate different aspects of its lifecycle within the process. For instance, *numrelevs* refers to the total number of events the object is involved in, capturing the extent of an object's involvement in the process. The *throughput* feature represents the duration of the object's lifecycle, giving an indication of how long the object remains active in the process. The *wip*, or work-in-progress, feature quantifies the number of other objects that have overlapping lifecycles with the given object, reflecting the degree of concurrency in the process. Finally, for each activity in the process, we define a feature that counts the number of times the object is involved in that specific activity.

Definition 19 (Basic Feature Map)**.** *Let $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ be an OCEL.*
Given and $\Sigma = \{\text{“numrelevs”}, \text{“throughput”}, \text{“wip”}\} \cup \{\text{“\#”@}a \mid a \in A\}$ (where @ is the concatenation of strings) we define the basic feature map $f_{bas} : O \to (\Sigma \to \mathbb{R})$ in which:

- *for $o \in O$, $f_{bas}(o)(\text{“numrelevs”}) = |lif(o)|$.*
- *for $o \in O$, $f_{bas}(o)(\text{“throughput”}) = time(end(o)) - time(start(o))$.*
- *for $o \in O$, $f_{bas}(o)(\text{“wip”}) = |\{o' \in O \mid [time(start(o)), time(end(o))] \cap [time(start(o')), time(end(o'))] \neq \emptyset\}|$.*
- *for $o \in O$ and $a \in A$, we define $f_{bas}(o)(\text{“\#”@}a) = |\{e \in lif(o) \mid evtype(e) = a\}|$.*

Table 3.3 shows a basic feature map computed on top of the OCEL represented in Table 3.3. From here, we can notice for example that $r6$ (of type invoice) is the object with the longest lifecycle, while $po3$ is the object with the highest work-in-progress.

Table 3.4: Basic feature map on top of the OCEL represented in Table 3.3.

ID	numrelevs	throughput	wip	#Change PR	#Close PR	#Cr.Inv.	#Cr.PO	#Cr.PR	#Q.Checks	#Inv.Receipt	#PR Appr.	#Payment
pr1	2	12600.0	1	0	1	0	0	1	0	0	0	0
pr2	2	106140.0	2	0	0	0	1	1	0	0	0	0
pr3	4	175500.0	2	1	0	0	1	1	0	0	1	0
pr4	2	80580.0	2	0	0	0	1	1	0	0	0	0
po1	2	244860.0	3	0	0	0	1	0	0	1	0	0
po2	2	252000.0	3	0	0	0	1	0	0	1	0	0
po3	4	6204720.0	6	0	0	0	1	0	0	3	0	0
po4	3	780720.0	4	0	0	0	1	0	1	1	0	0
po5	1	0.0	2	0	0	0	1	0	0	0	0	0
po6	2	344580.0	3	0	0	1	1	0	0	0	0	0
po7	2	249120.0	3	0	0	1	1	0	0	0	0	0
r1	2	435480.0	3	0	0	0	0	0	0	1	0	1
r2	2	844020.0	3	0	0	0	0	0	0	1	0	1
r3	2	190800.0	3	0	0	0	0	0	0	1	0	1
r4	2	191760.0	3	0	0	0	0	0	0	1	0	1
r5	2	192480.0	3	0	0	0	0	0	0	1	0	1
r6	2	26758800.0	4	0	0	0	0	0	0	1	0	1
r7	2	10800.0	2	0	0	0	1	0	0	1	0	0
r8	1	0.0	3	0	0	1	0	0	0	0	0	0
gi1	1	0.0	3	0	0	0	0	0	1	0	0	0
p1	1	0.0	2	0	0	0	0	0	0	0	0	1
p2	1	0.0	2	0	0	0	0	0	0	0	0	1
p3	1	0.0	3	0	0	0	0	0	0	0	0	1
p4	1	0.0	3	0	0	0	0	0	0	0	0	1
p5	1	0.0	2	0	0	0	0	0	0	0	0	1
p6	1	0.0	2	0	0	0	0	0	0	0	0	1

Building upon these basic features, we further explore the possibility of extracting graph-based features (Definition 20). Given an Object Graph Enrichment that depicts the connections between objects, we can define additional features that encapsulate an object's position and role within the network of interactions. For instance, *outdegree* captures the number of outgoing connections an object has in the graph, a reflection of how many other objects it directly interacts with. We can also consider the same notion in a typed manner: for each type of object, we can count how many outgoing connections an object has to objects of that type.

Definition 20 (Graph-Based Feature Map)**.** *Let* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *be an OCEL and* $GE = (L, \Lambda)$ *an Object Graph Enrichment. Given* $\Sigma = \{\text{“outdegree”}\} \cup \{\text{“outdegreeot”@}ot \mid ot \in OT\}$ *(where @ is the concatenation of strings), we define the feature map* $f_{GE} : O \to (\Sigma \to \mathbb{R})$ *in which:*

- *for* $o \in O$, $f_{GE}(o)(\text{“outdegree”}) = |\{(o_1, o_2) \in \Lambda \mid o_1 = o\}|$
- *for* $o \in O$ *and* $ot \in OT$, $f_{GE}(o)(\text{“outdegreeot”@}ot) = |\{(o_1, o_2) \in \Lambda \mid o_1 = o \ \wedge \ objtype(o_2) = ot\}|$.

Table 3.5 shows a graph-based feature map computed on top of the object-centric represented in Table 3.3. From here, we can notice for example that $pr1$ has null outdegree (because the purchase requisition is not approved and therefore no subsequent order is created).

Table 3.5: Graph-based feature map based on the object-centric in Table 3.3 and its its Object Interaction Enrichment.

ID	outdegree	outdegree Goods Issues	outdegree Invoices	outdegree Payments	outdegree Purch.Orders	outdegree Purch.Req.
pr1	0	0	0	0	0	0
pr2	1	0	0	0	1	0
pr3	1	0	0	0	1	0
pr4	1	0	0	0	1	0
po1	2	0	1	0	0	1
po2	2	0	1	0	0	1
po3	3	0	3	0	0	0
po4	3	1	1	0	0	1
po5	1	0	1	0	0	0
po6	2	0	1	0	1	0
po7	2	0	1	0	1	0
r1	2	0	0	1	1	0
r2	2	0	0	1	1	0
r3	2	0	0	1	1	0
r4	2	0	0	1	1	0
r5	2	0	0	1	1	0
r6	2	0	0	1	1	0
r7	1	0	0	0	1	0
r8	2	0	0	0	2	0
gi1	1	0	0	0	1	0
p1	1	0	1	0	0	0
p2	1	0	1	0	0	0
p3	1	0	1	0	0	0
p4	1	0	1	0	0	0
p5	1	0	1	0	0	0
p6	1	0	1	0	0	0

In our ongoing endeavor to extract rich and meaningful features from OCELs, we introduce the concept of feature propagation. Recognizing that objects within a process are not isolated entities, but rather part of a broader network of interconnections and dependencies, feature propagation aims to capture not just the direct characteristics of an object, but also the influence of its surrounding context.

Feature propagation (introduced in Definition 21) operates on the principle of aggregating information across related objects in the Object Graph Enrichment. For a given object, we consider not just its own features, but also those of the objects it is connected to in the graph. We then aggregate these features into a single value, using a chosen aggregation function. This function could be a simple average, a sum, a maximum or minimum value, or any other operation that condenses a set of numbers into a single representative value. The result is a propagated feature map, which captures both the original features of each object and the aggregated features from its connections in the graph.

Definition 21 (Feature Propagation)**.** *Let* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *be an OCEL,* $GE = (L, \Lambda)$ *be an Object Graph Enrichment and* $f : O \to (\Sigma \to \mathbb{R})$ *be an Object-Based Feature Map. Given an aggregation function* $agg : \mathcal{B}(\mathbb{R}) \to \mathbb{R}$*, we define* $f_{agg,G} : O \to (\Sigma \to \mathbb{R})$ *such that for* $s \in \Sigma$ *and* $o \in O$, $f_{agg,G}(o)(s) = agg(\{f(o')(s) \mid o = o' \ \vee \ (o, o') \in \Lambda\})$.

We introduce here some potential use cases for feature propagation within the P2P process:

- Consider the case where the feature of interest is *throughput*, which measures the time between the start and end of an object's lifecycle. Traditionally, we might consider this feature on an object-by-object basis, treating each object as an isolated entity. However, this view neglects the interconnected nature of the P2P process. By propagating the throughput feature across related objects in the Object Graph Enrichment, we begin to capture a broader picture of transaction duration in the process as a whole. With feature propagation, we effectively compute an aggregated measure of throughput for each object, taking into account not just the object's own throughput, but also those of its associated objects. In doing so, we might unveil systemic issues impacting transaction duration, such as consistent delays associated with certain object types or specific stages of the process. This propagation of the throughput feature hence facilitates a process-wide perspective on transaction speed, promoting a more comprehensive understanding of process efficiency.

- For the *wip* feature, which quantifies the number of objects concurrently active with a given object. In the traditional view, this feature provides an indication of process concurrency at the individual object level. However, with feature propagation, we aggregate the "wip" features of connected objects, thereby yielding a more encompassing view of concurrency in the process. This propagated *wip* feature represents the collective concurrency across an object and its linked peers in the graph. High values of the propagated *wip* feature might suggest areas of the process characterized by high concurrency and potential bottlenecks. Such insights could inform resource allocation decisions, thereby contributing to process optimization efforts.

- The activity-related features, represented by the $\#@a$ set of features in the basic feature map. These features account for the frequency of particular activities associated with an object. Propagating these features across the Object Graph Enrichment can unveil trends in activity execution across connected objects. For instance, if a propagated activity-related feature is high for a specific activity across a group of interconnected objects, this might signal the overall prominence of that activity within that segment of the P2P process. Such insights can be instrumental in understanding process dynamics, informing decision-making, and guiding interventions aimed at process improvement.

In conclusion, feature propagation in OCELs offers a powerful means of extracting richer and more meaningful features from the data. By extending our focus beyond individual objects and considering the interconnectedness of objects within a process, we can gain deeper insights into the underlying process dynamics.

3.6 Flattening to Traditional Event Logs

In the realm of process mining, the conventional approach to event logging involves a single case notion, where each event is intrinsically tied to a single case. However, this traditional method often falls short in the face of more complex processes, particularly those where different objects interact in a non-trivial manner. This is the case, for instance, in many real-world settings where a process involves interactions between multiple entities, each with their own lifecycle and relationships with other entities. The object-centric approach to event logging was developed to address this very complexity.

However, there are scenarios where we may need to distill this complexity into a simpler, more traditional form. This is where the concept of "flattening" comes into play. The goal of flattening is to convert the rich, multi-faceted information encapsulated in an OCEL into a traditional event log, with each event tied to a single case notion.

In this section, we will explore two types of flattening strategies: Basic Flattening and Object-Graph-Based Flattening.

In the case of Basic Flattening, we pick an object type of interest and each instance of that object type is transformed into a separate case in the traditional event log. This method aligns the lifecycle of each object instance of the selected type (Definition 16) with a single case, effectively simplifying the information structure from the original object-centric log.

Table 3.6: Lifecycle of the objects (Basic Flattening) of the OCEL presented in Table 3.3.

Object ID	Events in Lifecycle	Activities in Lifecycle
Purch.Req.		
pr1	⟨ e1, e2 ⟩	⟨ Create Purchase Requisition, Close Purchase Requisition ⟩
pr2	⟨ e3, e4 ⟩	⟨ Create Purchase Requisition, Create Purchase Order ⟩
pr3	⟨ e7, e8, e9, e10 ⟩	⟨ Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition ⟩
pr4	⟨ e20, e21 ⟩	⟨ Create Purchase Requisition, Create Purchase Order ⟩
Purch.Ord.		
po1	⟨ e4, e5 ⟩	⟨ Create Purchase Order, Invoice Receipt ⟩
po2	⟨ e9, e11 ⟩	⟨ Create Purchase Order, Invoice Receipt ⟩
po3	⟨ e13, e14, e16, e18 ⟩	⟨ Create Purchase Order, Invoice Receipt, Invoice Receipt,Invoice Receipt ⟩
po4	⟨ e21, e22, e23 ⟩	⟨ Create Purchase Order, Invoice Receipt, Quality Check ⟩
po5	⟨ e26 ⟩	⟨ Create Purchase Order ⟩
po6	⟨ e27, e29 ⟩	⟨ Create Purchase Order, Create Invoice ⟩
po7	⟨ e28, e29 ⟩	⟨ Create Purchase Order, Create Invoice ⟩
Invoices		
r1	⟨ e5, e6 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r2	⟨ e11, e12 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r3	⟨ e14, e15 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r4	⟨ e16, e17 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r5	⟨ e18, e19 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r6	⟨ e22, e24 ⟩	⟨ Invoice Receipt, Perform Payment ⟩
r7	⟨ e25, e26 ⟩	⟨ Invoice Receipt, Create Purchase Order ⟩
r8	⟨ e29 ⟩	⟨ Create Invoice ⟩
Quality Checks		
qc1	⟨ e23 ⟩	⟨ Quality Check ⟩
Payments		
p1	⟨ e6 ⟩	⟨ Perform Payment ⟩
p2	⟨ e12 ⟩	⟨ Perform Payment ⟩
p3	⟨ e15 ⟩	⟨ Perform Payment ⟩
p4	⟨ e17 ⟩	⟨ Perform Payment ⟩
p5	⟨ e19 ⟩	⟨ Perform Payment ⟩
p6	⟨ e24 ⟩	⟨ Perform Payment ⟩

Definition 22 (Basic Flattening)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, and an object type* $ot \in OT$*, we define the flattened event log* $flat(L, ot) = (A, E', C, evtype', time', case, \leq')$ *in which:*

- $C = \{o \in O \mid objtype(o) = ot\}$.
- $E' = \{(e, o) \in E \times O \mid e \in E \ \wedge \ o \in \pi_{omap}(e) \ \wedge \ objtype(o) = ot\}$.
- $case((e, o)) = o$ *for* $(e, o) \in E'$.
- $evtype'((e, o)) = evtype(e)$ *for* $(e, o) \in E'$.
- $time'((e, o)) = time(e)$ *for* $(e, o) \in E'$.

- $\leq' = \{((e, o), (e', o')) \in E' \times E' \mid e < e' \vee (e' = e \wedge o \leq_{lex} o')\}$.

Table 3.6 shows the results of Basic Flattening (i.e., the lifecycle of the objects) for the different object types of the log in Table 3.3.

Table 3.7: Object-Graph-Based Flattening of the OCEL presented in Table 3.3.

Object ID	Events in Lifecycle	Activities in Lifecycle
Purch.Req.		
pr1	⟨ e1, e2 ⟩	⟨ Create Purchase Requisition, Close Purchase Requisition ⟩
pr2	⟨ e3, e4, e5, e6 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Perform Payment ⟩
pr3	⟨ e7, e8, e9, e10, e11, e12 ⟩	⟨ Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition, Invoice Receipt, Perform Payment ⟩
pr4	⟨ e20, e21, e22, e23, e24 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Quality Check, Perform Payment ⟩
Purch.Ord.		
po1	⟨ e3, e4, e5, e6 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Perform Payment ⟩
po2	⟨ e7, e8, e9, e10, e11, e12 ⟩	⟨ Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition, Invoice Receipt, Perform Payment ⟩
po3	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
po4	⟨ e20, e21, e22, e23, e24 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Quality Check, Perform Payment ⟩
po5	⟨ e25, e26 ⟩	⟨ Invoice Receipt, Create Purchase Order ⟩
po6	⟨ e27, e28, e29 ⟩	⟨ Create Purchase Order, Create Purchase Order, Create Invoice ⟩
po7	⟨ e28, e28, e29 ⟩	⟨ Create Purchase Order, Create Purchase Order, Create Invoice ⟩
Invoices		
r1	⟨ e3, e4, e5, e6 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Perform Payment ⟩
r2	⟨ e7, e8, e9, e10, e11, e12 ⟩	⟨ Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition, Invoice Receipt, Perform Payment ⟩
r3	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
r4	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
r5	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
r6	⟨ e20, e21, e22, e23, e24 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Quality Check, Perform Payment ⟩
r7	⟨ e25, e26 ⟩	⟨ Invoice Receipt, Create Purchase Order ⟩
r8	⟨ e27, e28, e29 ⟩	⟨ Create Purchase Order, Create Purchase Order, Create Invoice ⟩
Quality Checks		
qc1	⟨ e23 ⟩	⟨ Quality Check ⟩
Payments		
p1	⟨ e3, e4, e5, e6 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Perform Payment ⟩
p2	⟨ e7, e8, e9, e10, e11, e12 ⟩	⟨ Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition, Invoice Receipt, Perform Payment ⟩
p3	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
p4	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
p5	⟨ e13, e14, e15, e16, e17, e18, e19 ⟩	⟨ Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment ⟩
p6	⟨ e20, e21, e22, e23, e24 ⟩	⟨ Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Quality Check, Perform Payment ⟩

On the other hand, the Object-Graph-Based Flattening approach takes a more nuanced perspective. Instead of focusing on a single object type, this method takes into account the intricate interconnections among different objects. In this case, a "case" may encompass the lifecycles of multiple interrelated objects, thus providing a more comprehensive picture of the complex interplays in the original process.

In the following, we conveniently introduce the concept of Eventually Follows Object Graph Enrichment, which represents an important extension to traditional Object Graph Enrichments. It takes into account not just immediate connections between objects, but also the potential relationships that manifest over time and multiple interactions. These graphs aim to encapsulate these dynamics by exploring how objects eventually relate to each other, based on the Object Graph Enrichment's interconnections. Three types of Eventually Follows Object Graph Enrichments are presented in in Definition 23: the Forward, the Backward, and the Bidirectional:

- The Forward Graph Enrichment, denoted as $GE_{\Longrightarrow}$, focuses on the downstream relations from a given object. It tracks all objects that can be reached from a starting object via a sequence of interactions, effectively capturing all potential "future" relations for each object.
- The Backward Graph Enrichment, represented by $GE_{\Longleftarrow}$, explores upstream relations to a given object. It maps all objects that could have interacted and led to the existence of the current object through a series of steps. This offers a valuable perspective on the "past" relations of an object.
- The Bidirectional Graph Enrichment, $GE_{\Longleftrightarrow}$, combines both forward and backward directions. It signifies objects that can be reached from a given object, either directly or indirectly, through a series of interactions, regardless of the direction of those interactions. This encapsulates the entirety of an object's possible interactions within the process, giving us the most comprehensive view of the system's dynamics.

By considering an Eventually Follows Object Graph Enrichment, we can capture more in-depth relational intricacies that exist in an OCEL. This ultimately enables us to create more accurate and nuanced mappings when performing the flattening process, particularly for Object-Graph-Based Flattening.

Definition 23 (Eventually Follows Object Graph Enrichments)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, and an Object Graph Enrichment* $GE = (L, \Lambda)$*, we define:*

- $GE_{\Longrightarrow} = (L, \Lambda_{\Longrightarrow})$*, where*

$$\Lambda_{\Longrightarrow} = \{(o, o') \in O \times O \mid (o, o') \in \Lambda \ \vee \ \exists_{o_1, o_2, \ldots, o_n \in O} (o, o_1), (o_1, o_2), \ldots, (o_n, o') \in \Lambda\}$$

- $GE_{\Longleftarrow} = (L, \Lambda_{\Longleftarrow})$*, where*

$$\Lambda_{\Longleftarrow} = \{(o, o') \in O \times O \mid (o', o) \in \Lambda \ \vee \ \exists_{o_1, o_2, \ldots, o_n \in O} (o', o_1), (o_1, o_2), \ldots, (o_n, o) \in \Lambda\}$$

- $GE_{\Longleftrightarrow} = (L, \Lambda_{\Longleftrightarrow})$*, where* $\Lambda_{\Longleftrightarrow} = \Lambda_{\Longrightarrow} \cup \Lambda_{\Longleftarrow}$.

Object-Graph-Based Flattening involves the flattening of an OCEL by considering an Object Graph Enrichment and a specific object type from the log. The main steps involve defining a set of case identifiers as objects of the chosen type, creating new events that take into account the interactions between the objects, and finally associating these new events with the original objects they correspond to. The main benefit of this technique is its capacity to provide a more nuanced and holistic perspective of the process by considering the interconnectedness of objects within the system.

Definition 24 (Object-Graph-Based Flattening)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, an Object Graph Enrichment* $GE = (L, \Lambda)$*, and an object type* $ot \in OT$*, we define the flattened event log* $ccflat(L) = (A, E', C, evtype', time', case, \leq')$ *in which:*

- $C = \{o \in O \mid objtype(o) = ot\}$.
- $E' = \{(e, o) \in E \times O \mid objtype(o) = ot \ \wedge \ \exists_{\bar{o} \in O} \ \bar{o} \in \pi_{omap}(e) \ \wedge \ (\bar{o} = o \vee (o, \bar{o}) \in \Lambda)\}$.
- $case((e, o)) = o$ *for* $(e, o) \in E'$.
- $evtype'((e, o)) = evtype(e)$ *for* $(e, o) \in E'$.

Table 3.8: Basic variants computed on the OCEL in Table 3.3.

Variant	Objects
Create Purchase Requisition, Close Purchase Requisition	⟨ pr1 ⟩
Create Purchase Requisition, Create Purchase Order	⟨ pr2, pr4 ⟩
Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition	⟨ pr3 ⟩
Create Purchase Order, Invoice Receipt	⟨ po1, po2 ⟩
Create Purchase Order, Invoice Receipt, Invoice Receipt,Invoice Receipt	⟨ po3 ⟩
Create Purchase Order, Invoice Receipt, Quality Check	⟨ po4 ⟩
Create Purchase Order	⟨ po5 ⟩
Create Purchase Order, Create Invoice	⟨ po6, po7 ⟩
Invoice Receipt, Perform Payment	⟨ r1, r2, r3, r4, r5, r6 ⟩
Invoice Receipt, Create Purchase Order	⟨ r7 ⟩
Create Invoice	⟨ r8 ⟩
Quality Check	⟨ qc1 ⟩
Perform Payment	⟨ p1, p2, p3, p4, p5, p6 ⟩

- $time'((e,o)) = time(e)$ *for* $(e,o) \in E'$.
- $\leq' = \{((e,o),(e',o')) \in E' \times E' \mid e < e' \vee (e' = e \wedge o \leq_{lex} o')\}$.

In Table 3.7 the Object-Graph-Based Flattening (built on the Bidirectional Graph Enrichment) is computed on top of the OCEL presented in Table 3.3.

We can also introduce the concepts of traces and variants in the object-centric setting. We can understand traces and variants from two perspectives: a basic perspective and an Object-Graph-Based perspective.

A "trace" (Definition 25) is essentially a sequence of activities related to a specific object. Now, if we adopt a basic viewpoint, a trace of an object is nothing more than the series of activities in the object's lifecycle. In other words, we're simply following the chain of actions that the object has undergone in chronological order. This is what we call a "basic trace". However, if we shift our lens to an Object-Graph-Based viewpoint, things get a bit more nuanced. In this case, we're not only looking at the lifecycle of the individual object, but we're also considering the interconnections with other objects. That means we're tracing the activities not only for the object in question but also for the objects that it is linked with. This broader perspective is termed as an "Object-Graph-Based trace".

Definition 25 (Traces (Basic and Object-Graph-Based))**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA,$ *evtype*, *time*, *objtype*, *eatype*, *eaval*, *oaval*, *E2O*, *O2O*, $\pi_{omap}, \leq)$*, we define the functions:*

- Basic Trace $trace_{basic} : O \to A^*$*,* $trace_{basic}(o) = \langle evtype(e_1), \ldots, evtype(e_n)\rangle$ *where* $\langle e_1, \ldots, e_n\rangle =$ *lif*(o) *is the sequence of activities of the lifecycle of an object.*
- Object-Graph-Based Trace *given an Object Graph Enrichment* $GE = (L, \Lambda)$*, we define an intermediate function* $g : O \to \mathcal{P}(O)$*,* $g(o) = \{o' \in O \mid (o,o') \in \Lambda \ \vee \ o = o'\}$*. Then, we can define* $trace_G : O \to A^*$*,* $trace_G(o) = \langle evtype(e_1), \ldots, evtype(e_n)\rangle$ *where* $\langle e_1, \ldots, e_n\rangle = lif_S(g(o))$ *is the sequence of activities of the lifecycle of the interconnected objects.*

Similarly, a "variant" (Definition 26) can be seen as a set of objects that share the same series of activities, or the same trace. Using the basic concept of a variant, we simply group together the objects that have identical sequences of activities, i.e., they have the same basic trace. When we apply the Object-Graph-Based concept, we consider the interconnected objects and group together those that share the same series of activities, including the activities of their interconnected objects. This gives us the "Object-Graph-Based variant". These perspectives provide us with different ways of looking at the same process. The basic view allows us to examine the process from the lens of individual objects, while the Object-Graph-Based view provides a more holistic perspective by considering the relationships between objects.

Table 3.9: Object-Graph-Based variants computed on the OCEL in Table 3.3. The relationships are built on the Bidirectional Graph Enrichment.

Variant	**Objects**
Create Purchase Requisition, Close Purchase Requisition	⟨ pr1 ⟩
Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Perform Payment	⟨ pr2, po1, r1 ⟩
Create Purchase Requisition, PR Formal Approval, Create Purchase Order, Change Purchase Requisition, Invoice Receipt, Perform Payment	⟨ pr3, po2, r2 ⟩
Create Purchase Requisition, Create Purchase Order, Invoice Receipt, Quality Check, Perform Payment	⟨ pr4, po4, r6, p6 ⟩
Create Purchase Order, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment, Invoice Receipt, Perform Payment	⟨ po3, r3, r4, r5, p3, p4, p5 ⟩
Invoice Receipt, Create Purchase Order	⟨ po5, r7 ⟩
Create Purchase Order, Create Purchase Order, Create Invoice	⟨ po6, po7, r8 ⟩

Definition 26 (Variants (Basic and Object-Graph-Based))**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we define the functions:*

- Basic Variants*: We define* $variant_{basic} : A^* \to \mathcal{P}(O)$ *such that for all* $\sigma \in A^*$ *and* $o \in O$*,*

$$o \in variant_{basic}(\sigma) \Leftrightarrow trace_{basic}(o) = \sigma$$

- Object-Graph-Based Variants*: Given an Object Graph Enrichment* $GE = (L, \Lambda)$*, we define* $variant_G : A^* \to \mathcal{P}(O)$ *such that for all* $\sigma \in A^*$ *and* $o \in O$*,*

$$o \in variant_G(\sigma) \Leftrightarrow trace_G(o) = \sigma$$

Table 3.8 introduces the "basic variants" computed from the OCEL in the earlier provided table. The term "variant", in this context, represents a unique sequence of activities. For example, the sequence "Create Purchase Requisition, Close Purchase Requisition" forms a variant. Now, each variant is linked to a set of objects. These objects are the ones that share the same sequence of activities - i.e., they have the same basic trace. Thus, the table gives us an organized view of the different sequences of activities in the log and which objects go through each sequence.

Looking at Table 3.9, we find the "Object-Graph-Based variants". Unlike the basic variants, these variants take into account not just the individual objects but also their interrelationships. This is done by building on an "eventually follows graph", which captures the sequence of activities across related objects. This graph is, in turn, derived from the Object Interaction Enrichment, thereby ensuring that the interconnected nature of objects is taken into consideration. The Object-Graph-Based variants table (Table 3.9) provides a more nuanced understanding of the process flow. Each variant here represents a sequence of activities considering interconnected objects. Therefore, compared to the basic variant, each sequence may encompass activities of multiple objects that are related to each other. Accordingly, the objects associated with each variant are those that share this sequence, either directly or through their interactions with other objects.

To sum up, the basic variants table provides a view of the process from the perspective of individual objects. In contrast, the Object-Graph-Based variants table presents a broader perspective that accounts for the relationships between objects and the collective sequence of activities across these interconnected objects. This distinction can be crucial when analyzing complex processes, where understanding inter-object interactions can provide valuable insights.

3.7 Assessment

In this section, we introduce the computational intricacies of various operations on OCELs in this section. Our exploration begins with a thorough assessment of the Basic Flattening execution time (Section 3.7.2), where we evaluate its performance under varied scenarios, distinguished by parameters such as the number of events and object types, among others. This is followed by a deep dive into the Object-Graph-Based Flattening execution time (Section 3.7.3). Here, our aim is to understand the influence of several event log parameters on the execution time of flattening operations targeting a specific object type.

Shifting our focus to feature extraction, we assess the performance of the feature table's extraction (Section 3.7.4), a crucial step for in-depth analysis of event logs. Through this, we aim to comprehend how the extraction performance behaves in correlation with various parameters of the event logs.

Lastly, Section 3.7.5 offers a quantitative appraisal on the quality of features obtained from our approach. This is accomplished by contrasting different feature extraction settings to ascertain which ones yield the most accurate predictions for lifecycle durations.

The experiments were executed on a notebook with an I7-7500U CPU, 16 GB DDR4 RAM, and the PM4Py 2.7.5.1 library (introduced in Chapter 9). Still, we are confident that the findings are generic and are also applicable to other configurations/tools.

3.7.1 Event Log Simulation Methodology

In order to comprehensively assess the diverse techniques proposed in this paper, we employed a simple event log simulation methodology. This approach allowed us to generate an assortment of scenarios and control variables, which facilitated the examination of the impacts of individual parameters on the overall results. Herein, we outline the primary steps of this methodology:

1. We initially generate X unique activity labels, providing a diverse pool of potential tasks to be performed within the process.
2. Following this, we generate Y distinct object types. These serve to add complexity and variation to the event log, mirroring the multi-dimensional nature of real-world processes.
3. We then generate N event identifiers, each of which is linked to a specific activity label. This forms the basis for tracking the progression and execution of individual activities.
4. Subsequently, we generate M object identifiers, each tied to a specific object type. These identifiers help map the interaction of various entities within the process.
5. For each event identifier, we sample a number from a random exponential distribution with a mean value of μ. This number is then rounded up to the nearest integer, and the resulting value is used to associate the event identifier with an equivalent number of object identifiers.

Through this approach, we can construct event logs where all variables are kept constant except one. By altering this single variable, we can directly evaluate its influence on the outcomes, providing a clear understanding of how individual parameters impact the performance and efficacy of the proposed techniques. This comprehensive and systematic assessment aids in refining the techniques to handle various scenarios and optimizes their performance for a wide array of potential use cases.

3.7.2 Assessment of the Basic Flattening Execution Time

In this section, we present our evaluation of the performance of the Basic Flattening operation. As detailed in Table 3.10, we assessed the operation's execution time across different scenarios, each distinguished by the number of events, objects, object types, and activities, as well as the number of Event-to-Object relationships.

Table 3.10 consolidates the results of multiple experiments, each varying a specific parameter while holding others constant. For example, in the first section, we adjusted the number of events while keeping the number of objects, object types, and activities constant, and so forth for each subsequent section.

Following the tabular presentation, we provide a visual depiction of the same data in Figure 3.8. Each of the four subplots focuses on a different parameter, showing the trend of execution time as this parameter increases.

Our analysis reveals that the execution time of the flattening operation scales linearly with the number of events, number of objects, and object types. The influence of other parameters such as the number of activities appears to be less pronounced.

Table 3.10: Execution times for the Basic Flattening operation.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	78721	50	50	1.2132048
100000	10000	158648	50	50	2.4023522
200000	10000	316088	50	50	4.5969565
250000	10000	395001	50	50	4.9876833
300000	10000	473952	50	50	6.4090705
Varying the number of objects					
10000	50000	16064	50	50	0.5457078
10000	100000	15784	50	50	0.6847662
10000	200000	15833	50	50	0.9564766
10000	250000	15708	50	50	1.0942039
10000	300000	15842	50	50	1.2447173
Varying the number of object types					
10000	10000	15554	50	50	0.4452643
10000	10000	15713	100	50	0.8407514
10000	10000	15743	200	50	1.7031903
10000	10000	15750	300	50	2.5509293
Varying the number of activities					
10000	10000	15903	50	50	0.4388332
10000	10000	15734	50	100	0.4443444
10000	10000	15751	50	200	0.4388262
10000	10000	15542	50	300	0.4472764
Varying the number of related objects					
10000	10000	16036	50	50	0.4418169
10000	10000	25444	50	50	0.4783739
10000	10000	34974	50	50	0.5006970
10000	10000	45231	50	50	0.5261483
10000	10000	54990	50	50	0.5525253

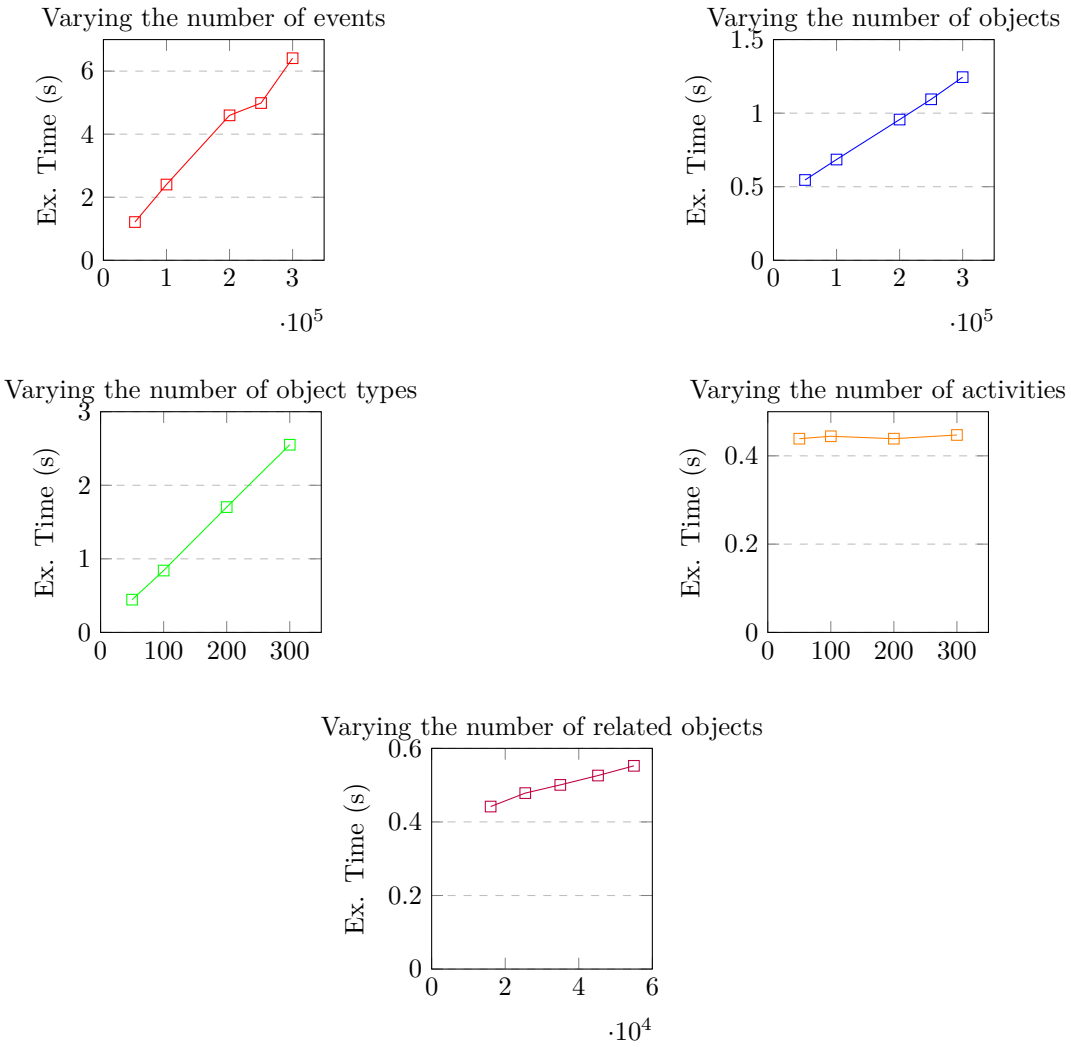


Figure 3.8: Execution times for the Basic Flattening operation.

3.7.3 Assessment of the Object-Graph-Based Flattening Execution Time

In order to gauge the computational efficiency of Object-Graph-Based Flattening on OCELs, we conducted a series of experiments. The objective was to observe how the execution time behaves with the variation in several parameters of the event logs: the number of events, objects, object types, activities, and the quantity of Event-to-Object relationships. It is essential to note that in all experiments, the flattening was consistently performed on one object type from the event log.

The results of these experiments are systematically tabulated in Table 3.11 and graphically illustrated in Figure 3.9.

A few observations emerge from the collected data:

- *Linear Growth with Events and Objects*: The execution time for the flattening operation demonstrates a linear growth trend as the number of events and objects in the log increases. This is an anticipated behavior, suggesting that each additional event or object adds a roughly consistent amount of processing time.
- *Stability with Increasing Object Types*: An intriguing insight from the experiments is the stable execution time in the face of increasing object types. Considering that the flattening operation is carried out on a singular object type, it makes intuitive sense. The operation seems to be largely unaffected by the sheer variety of object types, underscoring its efficiency in scenarios where the diversity of object types is high but the focus is only on one specific type.

In conclusion, the experiments shed light on the scalability and specificity of the Object-Graph-Based Flattening operation in handling OCELs. Such insights can inform practitioners about the potential performance bottlenecks and efficiencies when dealing with large-scale, diverse logs.

Table 3.11: Execution times for the Object-Graph-Based Flattening.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79154	50	50	1.2958393
100000	10000	158368	50	50	2.6163894
200000	10000	316452	50	50	5.3483154
250000	10000	394555	50	50	6.5118422
300000	10000	475095	50	50	7.7274229
Varying the number of objects					
10000	50000	15698	50	50	21.8129181
10000	100000	15777	50	50	33.2267917
10000	200000	15864	50	50	49.4606195
10000	250000	16041	50	50	60.6836148
10000	300000	15956	50	50	68.8758236
Varying the number of object types					
10000	10000	15884	50	50	2.4691261
10000	10000	15896	100	50	2.7061117
10000	10000	15766	200	50	2.4879450
10000	10000	15841	300	50	2.5904176
Varying the number of activities					
10000	10000	15689	50	50	2.7355063
10000	10000	15684	50	100	2.5174117
10000	10000	15918	50	200	2.5646962
10000	10000	15929	50	300	2.3879817
Varying the number of related objects					
10000	10000	15961	50	50	2.6064566
10000	10000	25554	50	50	0.4457783
10000	10000	34693	50	50	0.4486246
10000	10000	45058	50	50	0.5824439
10000	10000	54504	50	50	0.7669498

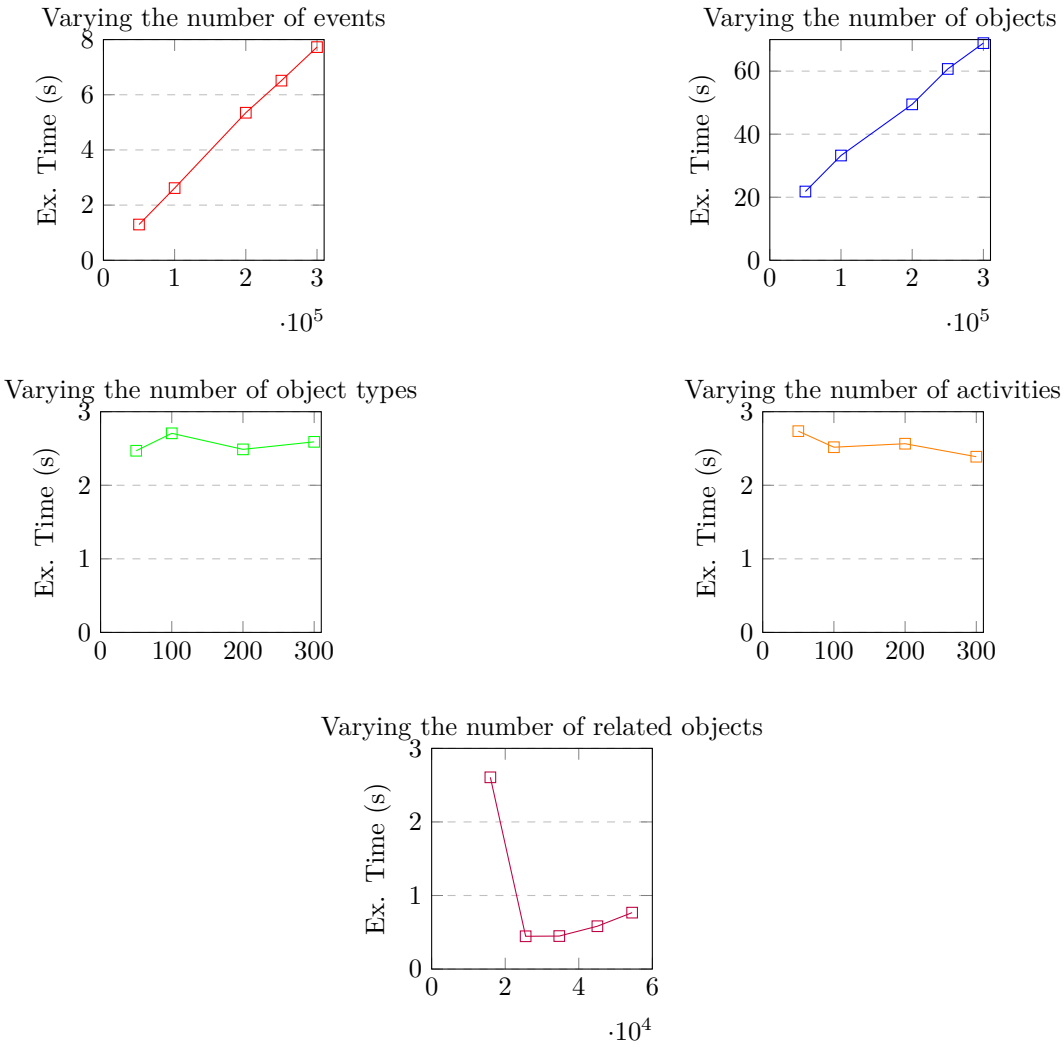


Figure 3.9: Execution times for the Object-Graph-Based Flattening.

3.7.4 Performance of the Feature Table's Extraction

The extraction of feature tables from OCELs stands as a crucial step towards deeper analysis and insights. As such, a systematic assessment was undertaken to understand the performance characteristics of this extraction process.

Our primary findings indicate a linear relationship between the execution time and each of the involved parameters. Specifically, the execution time was observed to grow linearly with the number of events, the number of objects, the diversity of object types, the variety of activities, and the intricacies of Event-to-Object relationships. This linear behavior underscores the predictable scalability of the extraction process, providing valuable insights for future implementations and adaptations.

For a comprehensive understanding:

- The execution times for the extraction of a feature table from an OCEL are tabulated in Table 3.12.
- To provide a visual perspective on the same, the corresponding graph detailing these execution times can be referenced in Figure 3.10.

Table 3.12: Execution times for the extraction of a feature table starting from an OCEL.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	78716	50	50	10.5661117
100000	10000	158041	50	50	20.4898767
200000	10000	316681	50	50	40.2521503
250000	10000	395430	50	50	53.7625236
300000	10000	474689	50	50	59.9981032
Varying the number of objects					
10000	50000	15869	50	50	6.8344423
10000	100000	15931	50	50	12.5959856
10000	200000	15900	50	50	20.8117765
10000	250000	15828	50	50	25.8729881
10000	300000	15801	50	50	29.5575412
Varying the number of object types					
10000	10000	15837	50	50	2.9373541
10000	10000	15722	100	50	3.2501874
10000	10000	15892	200	50	3.8688095
10000	10000	15599	300	50	4.3733918
Varying the number of activities					
10000	10000	15756	50	50	3.0117373
10000	10000	15898	50	100	3.3211060
10000	10000	15764	50	200	4.0964246
10000	10000	15859	50	300	4.8267142
Varying the number of related objects					
10000	10000	15828	50	50	2.9480021
10000	10000	25144	50	50	3.3877771
10000	10000	34789	50	50	4.1231465
10000	10000	45735	50	50	5.2536902
10000	10000	54223	50	50	6.9844339

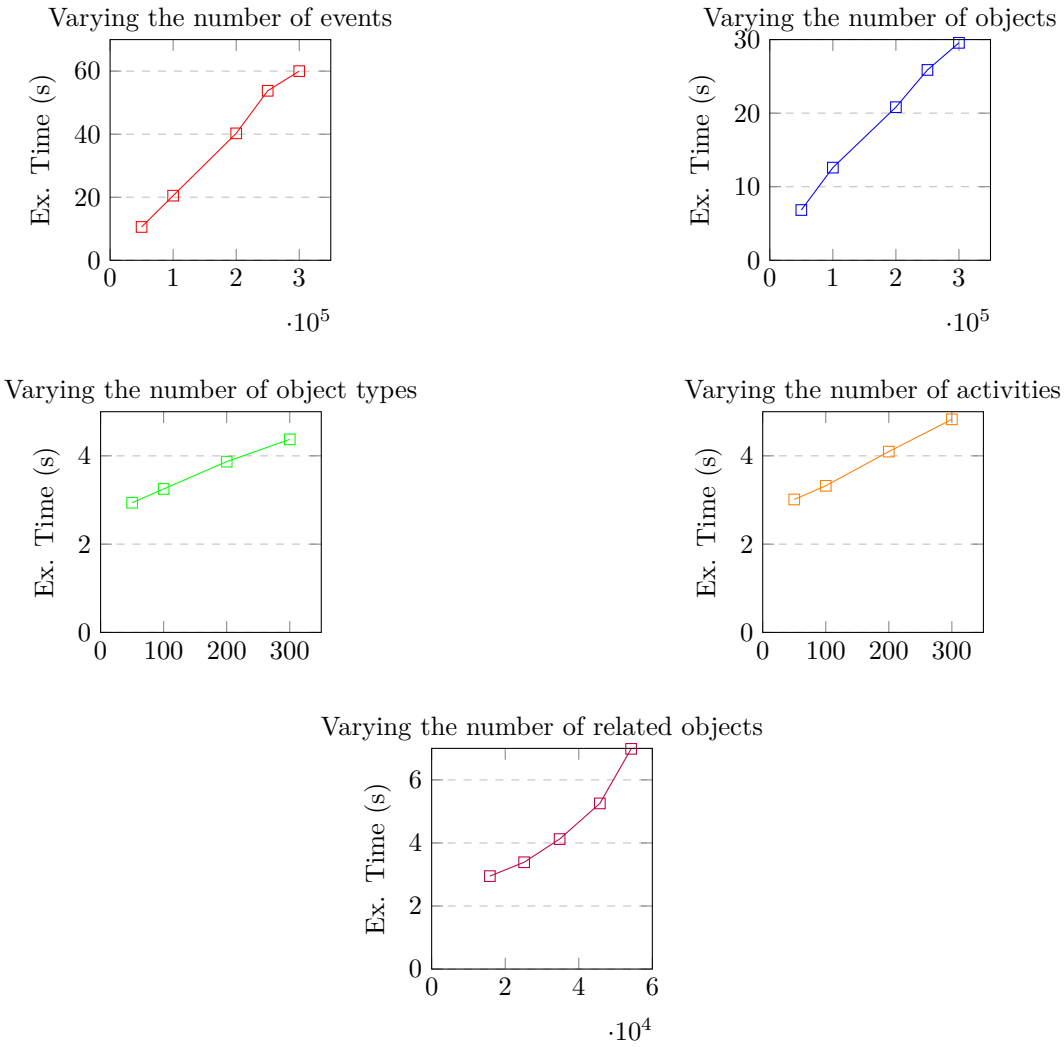


Figure 3.10: Execution times for the extraction of a feature table starting from an OCEL.

3.7.5 Predictive Analytics Using Features Extracted from OCELs

In this subsection, we propose a quantitative assessment of the features extracted from our approach on top of more complex publicly available event logs. Indeed, we compare four different settings for feature extraction:

S1 (Baseline) only features related to the lifecycle of an object are considered. This is equivalent to performing feature extraction on top of traditional event logs.

S2 Features related to the lifecycle of an object and the Object Interaction Enrichment are considered. This is equivalent to the setting described in [52].

S3 Features related to the lifecycle of an object and the object interaction and creation graph are considered.

S4 Features related to the lifecycle of an object and the object interaction, creation, and continuation graphs are considered.

Table 3.13: Event logs used in the quantitative assessment.

Event log	**Cons. Obj. Types**	**N. Events**	**N. Objects**	**N. Activities**
Order Management log	orders; items; packages	22367	11484	11
SAP ERP IDES instance - O2C log	all	98350	107767	23
SAP ERP IDES instance - P2P log	EBELN_EBELP; EBELN; BELNR_EBELN_EBELP; BELNR	20554	62353	13
Recruiting Process	applicants; applications; offers	6607	1339	12

Inputs: the logs that are used in the quantitative assessment are available at the address `https://www.ocel-standard.org/` and their features are reported in Table 3.13. For some of the event logs, we filtered out the object types as specified in Table 3.13. This was done because objects of some object types were connected to a significant number of events, inducing a connection between most of the events and objects of the event log, therefore making graph-based feature extraction less effective. As an example, many orders contained the same products, so the *products* object type was filtered out from the "Order management log".
Technique-Dependent Considerations: we assessed the quality of features based on the quality of machine learning models that can be learned from the data. In particular, we want to predict the lifecycle duration starting from four different scenarios, **S1-t**, **S2-t**, **S3-t** and **S4-t**, which are equivalent to the aforementioned scenarios with the exclusion of the lifecycle duration feature. For this, we use an extra trees regressor trained on 80% of the objects to predict the lifecycle duration of the remaining objects and compare the prediction with the actual value.

Table 3.14: Prediction error (MAPE) of the lifecycle duration in the four experimental scenarios S1-t, S2-t, S3-t, and S4-t (lower is better).

Event Log	**S1-t**	**S2-t**	**S3-t**	**S4-t**
Order Management log	**54%**	55%	55%	55%
SAP ERP IDES instance - O2C log	53%	36%	28%	**27%**
SAP ERP IDES instance - P2P log	147%	134%	132%	**129%**
Recruiting Process	45%	45%	44%	**43%**

Table 3.15: Prediction error (RMSPE) of the lifecycle duration in the four experimental scenarios S1-t, S2-t, S3-t, and S4-t (lower is better).

Event Log	**S1-t**	**S2-t**	**S3-t**	**S4-t**
Order Management log	**0.14D**	0.15D	0.15D	0.15D
SAP ERP IDES instance - O2C log	3.63D	3.14D	2.85D	**2.79D**
SAP ERP IDES instance - P2P log	0.14D	0.12D	**0.11D**	0.13D
Recruiting Process	0.82D	0.82D	0.82D	0.82D

The effectiveness of the prediction was measured using two standard metrics (Mean Absolute Percentage Error (MAPE) and Root of Mean Squared Percentage Error (RMSPE)), briefly reported below. Here, A_i is the actual value (the effective throughput time) and F_i is the predicted completion time.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{A_i - F_i}{A_i}|$$

$$RMSPE = \frac{\sqrt{\sum_{i=1}^{n}(A_i - F_i)^2}}{n}$$

The results are reported in Table 3.14 (for the MAPE metric) and Table 3.15 (for the RMSPE metric). Lower values indicate better predictions, and, generally, the addition of the Object Interaction Enrichment and the Object Creation Enrichment contributes to improving the results of the prediction. Considering the MAPE metric, the best predictive results are obtained in the scenario **S4-t** for three of the four logs.

Therefore, considering several different graph-based features contributed to a better prediction in comparison to considering only features related to the lifecycle of an object (**S1-t**, which is equivalent to what is done in [38]) or just the Object Interaction Enrichment (**S2-t**, which is equivalent to what is done in [52]).

Conclusion

This chapter has laid the essential groundwork for Object-Centric Process Mining, introducing the fundamental concepts and formal definitions that underpin this emerging field. We have established the core object-centric notions, illustrating their practical relevance through examples drawn from familiar ERP processes. The formal definition of OCELs provides the necessary structure for capturing the rich interactions and attribute dynamics within object-centric processes. We have explored how feature extraction techniques can be applied to OCELs to create a more informative representation for analysis. Finally, by examining the process of flattening OCELs, we have highlighted the limitations of traditional event logs and motivated the need for specialized object-centric techniques. The concepts and definitions presented in this chapter form the foundation upon which the rest of the thesis is built. The subsequent chapters will delve into specific methodologies for data extraction, process discovery, and conformance checking in object-centric settings, building on the foundational understanding established here. The formal framework for OCELs and the feature extraction techniques introduced will be instrumental in developing and evaluating these advanced methodologies.

Chapter 4

Existing Approaches in OCPM

"*The advancement and diffusion of knowledge is the only guardian of true liberty.*"
James Madison

Introduction

This chapter provides a comprehensive overview of the existing landscape of OCPM, building on the foundational insights obtained from [17]. We begin by defining the scope of our literature review, ensuring a focused and systematic analysis of relevant research following the categorization strategies presented in the referenced contribution. By leveraging the structured approach and the taxonomies detailed in [17], we delineate key concepts and clarify core terminologies within OCPM.

We then categorize existing works according to their focus: from data extraction and storage solutions to the preprocessing strategies that facilitate smooth transitions into process discovery, conformance checking, and performance analysis. Throughout this chapter, the identified classification and criteria extracted from [17] guide the detailed examination of current methods, including their strengths, weaknesses, scalability, and applicability to real-world scenarios. Trends and future directions are also discussed based on the temporal evolution and insights derived from the referenced systematic literature review.

Ultimately, this chapter lays out a structured and in-depth understanding of the current state-of-the-art in OCPM, setting the stage for the novel contributions presented in subsequent chapters.

4.1 Scope of the Literature Review

In doing this literature review, we want to include all relevant scientific papers on OCPM.

Table 4.1 illustrates the process of refinement of the search query. The *scopus* search engine [58] is considered in this case. We started with some well-known concepts (process mining, object-centric, artifact-centric) and added some extra keywords:

- Synonyms of the artifact-centric and object-centric concepts (including object-aware, multi-case, multi-instance, multiple cases, multiple instances, and multiple process instances) have been tried. However, the only ones that resulted relevant to refine the search query and find additional results are *object-aware* and *multiple entities*.
- The usage of *graph databases* [43] is a natural choice for storing and querying object-centric event data, given their ability to store event data with lots of interconnections (Event-to-Event, Event-to-Object, Object-to-Object). We tried different synonyms for the concept of graph database (e.g., event graph, knowledge graph, property graph) but no additional results were found in comparison to the original query, hence we do not include these synonyms in the search query.
- OCPM techniques have been defined to overcome the necessity to choose a case notion on the data and the limitations coming with it. Therefore, an OCPM analysis is done on all the possible object types/case notions. When OCPM techniques were not still developed, the choice of a *case notion* was necessary. Hence, including all the techniques helping to choose the case notion is of historical interest for this review.

Table 4.1: Refinement process of the search query executed on scopus [58].

Search Query	**Number of results**
TITLE-ABS-KEY("process mining") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric")	73
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric")	82
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric" OR "object-aware")	89
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric" OR "object-aware" OR "multiple entities")	94
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric" OR "object-aware" OR "multiple entities" OR "graph database")	114
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric" OR "object-aware" OR "multiple entities" OR "graph database" OR "case notion")	119
TITLE-ABS-KEY("process mining" OR "process discovery" OR "conformance checking") AND TITLE-ABS-KEY("object-centric" OR "artifact-centric" OR "object-aware" OR "multiple entities" OR "graph database" OR "case notion" OR "merging event logs") OR (TITLE-ABS-KEY("object-aware") AND TITLE-ABS-KEY("petri net"))	127

- Also, *merging techniques* have been used to collate event data coming from different systems/organizations. Including the log merging techniques in this review helps to understand how multiple sources of event data were treated when no object-centric technique was available.
- As a reference, we include some *object-aware extensions of the Petri net concept*, on top of which discovery/conformance checking techniques will be proposed in the near future.

The final query is reported below:

```
[frame=single]
(TITLE-ABS-KEY("process mining"  OR "process discovery"
OR "conformance checking")
AND TITLE-ABS-KEY("object-centric" OR "artifact-centric"
OR "object-aware" OR "multiple entities"
OR "graph database" OR "case notion" OR "merging event logs"))
OR (TITLE-ABS-KEY("object-aware")
AND TITLE-ABS-KEY("petri net"))
```

The search query produced some unrelated results, so we propose acceptance criteria to filter the results according to the goals of the review:

1. The reference should be properly inserted and mentioned in *scopus*. Moreover, for the duplicated entries, only one entry is kept. This reduces the number of results from **127** to **96**. An example of an entry excluded with this criteria is *28th International Conference on Cooperative Information Systems, CoopIS 2022* because it (wrongly) reports the name of a conference as a reference to a specific paper.

2. The event data or process model used in the paper should allow for the interaction of an event with several objects of different types, the interaction between cases of different event logs, the choice of a case notion between many available possibilities, or being about advanced object-aware processes on top of which process discovery or conformance checking techniques are currently in the work. This reduces the number of results from **96** to **71**. Examples of entries excluded with this criteria are *Uncertain Case Identifiers in Process Mining: A User Study of the Event-Case Correlation Problem on Click Data*[1] (because the event correlation is done on data without any case identifier) and *Graph-Based Token Replay for Online Conformance Checking*[2] (because a traditional TBR technique is applied).

3. The paper should be published between January 2010 and June 2023 (this did not reduce the number of results, but makes the review reproducible).

This allows us to obtain the related works included in this review (Table 4.2).

4.2 Conceptualization

We could identify some categories for the different types of OCELs formats proposed in the literature:

- **E2O**: the events are related to different objects of different object types.
- **O2O**: Object-to-Object relationships (for example, parent-children relationships) are recorded.
- **E2E**: event-to-event relationships (for example, connecting events with different levels of granularity) are recorded.

The original OCEL 1.0 format `https://www.ocel-standard.org` belongs to the category **E2O**, but not to **O2O** or **E2E**. The commercial vendor Celonis proposed the usage of multi-event logs, which are collections of event logs and relationships between the cases/objects of such logs. Therefore, multi-event logs belong to the category **O2O**. Moreover, always Celonis proposed the signal link functionality exploiting event-to-event relationships (**E2E**) to connect event data.

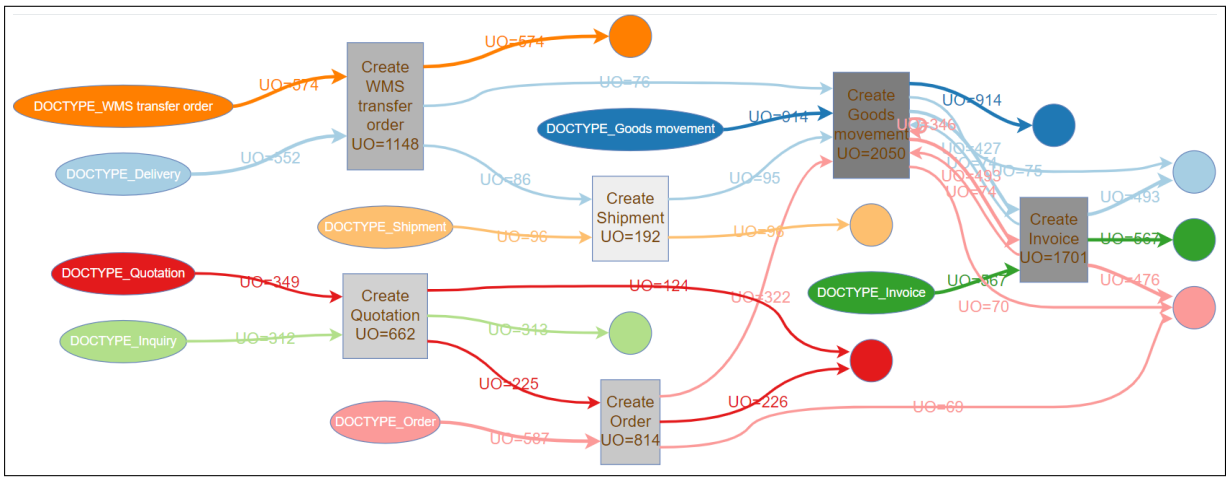


Figure 4.1: OCDFG [25]. This process model belongs to the **ET2ET** (activities are connected by arcs) and **ET2OT** (arcs are colored according to the object type) categories.

A *flattening* operation converts an OCEL to a traditional event log after the choice of a case notion. The choice of a case notion leads to the three problems of *deficiency*, *convergence* and *divergence* [115], which can be summarized as follows:

[1] Pegoraro, Marco, et al. "Uncertain case identifiers in process mining: A user study of the event-case correlation problem on click data." Enterprise, Business-Process and Information Systems Modeling: 23rd International Conference, BPMDS 2022 and 27th International Conference, EMMSAD 2022, Held at CAiSE 2022, Leuven, Belgium, June 6–7, 2022, Proceedings. Cham: Springer International Publishing, 2022.

[2] Waspada, Indra, et al. "Graph-Based Token Replay for Online Conformance Checking." IEEE Access 10 (2022): 102737-102752.

- A *deficiency* problem occurs when some events are not included in any case (for example, events of a purchase requisition that is not transformed into a purchase order are not included if the order is chosen as a case notion).
- A *convergence* problem occurs when the same event needs to be replicated among different cases (for example, the events related to an invoice with multiple orders, when the order is chosen as case notion).
- A *divergence* problem occurs when several concurrent instances of the same activity appear in the same case (for example, the events related to different invoices of the same order).

OCPM techniques exploit OCELs and *object-centric process models*. Object-centric process models can be abstracted as multigraphs in which the set of nodes are the activities or the object types involved in the business process, and logical connections of different types connect the nodes. Discovery techniques for object-centric process models have been proposed. Among object-centric process models, we identify the following categories (all at the model level):

- **ET2ET**: *the model allows for logical connections between event types/activities.* These connections can either be direct (e.g., two activities are connected in a Directly-Follows Graph, see Figure 4.1) or indirect (e.g., there is a path in the Petri net between two visible transitions in an OCPN).
- **ET2OT**: *the model allows for logical connections between event types/activities and object types/-classes.* In most situations, these connections report conditions on the start/end/synchronization between object types, or cardinality constraints (see Figure 4.10). In the literature, the term *artifact-centric* is commonly used to refer to process models (BAUML, Guard-Stage-Milestone, Proclets, Object-aware) belonging to the **ET2ET** and **OT2OT** categories.
- **OT2OT**: *the model allows for logical connections between object types/classes.* In most situations, these connections are reported as arcs (between event and object types) or as arc types (examples: activity-to-activity arcs, see Figure 4.1).

4.3 Categories of Analysis

In this section, we categorize the results obtained from the search query into different categories. Table 4.2 includes a synthetic representation of such categories. In particular, for every paper we include:

- The title and the authors of the paper.
- The year and the month in which the paper was published.
- The types of OCELs used in the paper (E2O, O2O, E2E; see Section 4.2).
- The types of object-centric process models used in the paper (ET2ET, OT2OT, ET2OT; see Section 4.2).
- The different techniques adopted in the paper (described in this section).
- The dimensions valued in the paper (see Section 4.4).

4.3.1 Extraction of Object-Centric Event Data

The techniques described in this subsection allow us to connect to a relational/non-relational database and extract an OCEL.

Here, we distinguish between the definition of meta-models for the extraction (without any proposed implementation, e.g., [95, 93, 94]) or actual extraction techniques that have been used to extract an OCEL. In [80], an approach for artifact-centric process mining on top of SAP ERP is proposed, with an implementation (as a plug-in of ProM) allowing to ingest the artifact-centricity from SAP and visualize the performance of the artifacts. According to the authors, the implementation is tricky to configure and this makes the usage difficult for the final user. In [18, 134], a generic approach to extract event data from SAP ERP is proposed. This starts from the specification of a starting object type/table of SAP and its progressive extension to a bigger set of object types/tables through user interaction. Then, the tables and the relationships between them are considered to automatically extract an OCEL. This is user-friendly (no SQL query is asked to the end user), allows quick extraction of different processes in

SAP, and the tool support is publicly available. However, the paper does not properly assess the resulting OCEL. In [133], Konetki, a tool offering a structured approach to prepare event data for process mining, is presented. Konetki implements an object-centric data model from which OCELs can be extracted or different traditional event logs can be exported using different case notions.

The extraction of an event log from the Dollibar ERP system is described in [79]. This log can be used with the object-centric behavioral constraints techniques. Also, a log (always for OCBC techniques) has been extracted starting from popular social media platforms (Facebook, Linkedin, Twitter) in [75].

In [136, 137], the Virtual Knowledge Graph (VKG) approach to access data in relational databases is exploited to extract OCEL logs from databases. The VKG approach is knowledge-based (and a similar approach, OnProm, has been previously proposed to extract XES logs from databases).

4.3.2 Storage of Object-Centric Event Data

Here, we introduce the different formats that have been proposed for the storage of object-centric event data.

For OCELs satisfying **E2O**, a tabular format has been proposed in [22] and [115]. Each row corresponds to an event, the attributes at the event level are columns, and also the object types are columns. Each cell corresponding to an object type column contains the set of related objects to the event of such an object type. Eventually, the OCEL specification has been proposed in [53] with some implementations in JSON and XML. The standard allows for attributes at the event and object level; moreover, each event is related to a set of objects.

For OCELs satisfying **O2O**, the ACEL standard has been proposed in [86, 84] for the storage of artifact-centric event data from blockchain applications.

The XOC format that has been proposed in [79] to store object-centric event data satisfying **E2O** and **O2O**. This can be exploited for the discovery of object-centric behavioral constraint models. Every event that changes the status of the database model (creation, update, deletion of rows in the tables) is stored along with the entire status of the database (since the discovery of object-centric behavioral constraint models requires such information). This results in large logs even for small databases (because of the persistence of the status of the database for every event).

Some alternative storages have been adopted in the context of object-centric event data. In particular, graph databases [44, 45, 43] have been used to store and query object-centric event data. Also, document databases are used to store object-centric event data (in particular, MongoDB [12]). The Parquet columnar format has been proposed in [22] for keeping object-centric event data in-memory. As efficient compression algorithms can be used on the columns of a columnar database, the storage proves enormously competitive in comparison to OCEL for basic object-centric event data but loses the possibility to store nested attributes.

Limitations of the Current Formats: The importance of storing dynamic object attribute values has been highlighted as a necessity in two recent studies. In the context of Internet of Things (IoT), the study [28] highlights the need for accommodating rapidly changing IoT data within process mining techniques, underscoring the inability of existing formats like OCEL to track dynamic attribute values effectively. The study [57], concentrating on data-aware process mining, points out the ambiguity problems in linking attributes to events or objects in current OCEL formats, emphasizing the importance of support for dynamic object attributes.

4.3.3 Preprocessing of Object-Centric Event Data

We consider in this section only techniques that prepare/transform data for the application of other techniques (not the internal preprocessing steps of every algorithm).

In this section, we divide between merging techniques (some event logs are provided as input, and an event log is obtained as output containing the information of the different logs), the flattening/choice of a case notion (a criterion that is used to group events together) and filtering techniques (allowing to get an OCEL with a subset of the behavior of the original log).

Merging event logs: different approaches have been proposed to merge the contribution of different event logs into one. In [106], an approach to merge event logs with many-to-many relationships is proposed. A more user-guided technique is proposed in [36]. The usage of a hybrid artificial immune algorithm for event log merging is proposed in [139]. Since the merged logs may have different levels of granularity, in [107] an approach to consider such granularity is proposed. Therefore, merging allows the application of traditional process mining techniques to a set of interconnected logs. This is of particular importance for inter-organizational processes. In [116], federated process mining is proposed to create

an event log out of event data collected across organizational boundaries. Another approach for merging event logs in the context of inter-organizational process mining is proposed in [59].

Case Notion: we assume that events can be extracted from an information system with many different attributes and connections to objects, but it is unclear which are the suitable case notions in order to use the event data with traditional process mining techniques. Implementation of the OpenSLEX meta-model [40, 39] act as a bridge between the database and the eventual event logs. The contents of the database are transformed into a format that allows to easily query the data and retrieve a list of possible case notions. This can then be used to construct a traditional event log.

Filtering: the tool proposed in [25] allows some basic filtering operations (activities occurrences, number of objects per object type, sampling, activity - object type filtering). Also, the MongoDB support for OCEL [12] naturally allows for many filters on the event data. Graph databases usually have powerful query languages. In [43], the Neo4J graph database language Cypher is used to preprocess the event data before other operations, such as the discovery of a process model, are used.

Feature Extraction: in [4], the interconnections between different objects in an OCEL are used to define variants in the object-centric setting. In particular, the connected components of the Object Interaction Enrichment are used to identify clusters of related objects. These clusters are used to identify correlated events in the OCEL. The graph-related properties of these events and objects are then used to identify events/objects with similar behavior (this is eventually called variant). These properties are also exploited in [4] for feature extraction on OCELs. Different use cases (including predictive analytics) and types of feature extraction (table-based, graph-based) are proposed. In particular, the importance of the single features for the target is identified using explainable AI techniques (including the usage of SHAP values). The assessment of the paper is done starting from a traditional event log "adapted" as an OCEL. The related tool support is publicly available and can be easily installed. The paper [52] introduces feature extraction on OCELs for the goals of predictive analytics (prediction of the remaining time, activities occurrences, and customer satisfaction). It shows how considering the interaction between objects (Object Interaction Enrichment) leads to an improvement in the quality of the prediction on OCELs, compared to only considering the flattened event logs. This has been evaluated in a real-life case study. Also, [55] considers predictive monitoring starting from OCELs, focusing on the prediction of the next activity and remaining time starting from traditional and interaction-based features.

Alternative Approaches: in [108], an approach to transform traditional event logs into OCEL(s) is proposed. In [68], an approach to normalize an OCEL to a STAR and fully normalized database schema is shown. In [51], an object-centric clustering approach with the purpose of identifying alternative "behavioral patterns" is proposed, helping the user to understand different executions of the same process. Clustering is also proposed in [62], but with the different purpose of identifying object types that would produce a similar flattened traditional event log.

4.3.4 Process Discovery Techniques

Here, we divide the techniques that discover models satisfying **ET2ET+OT2OT** (but not **ET2OT**), **ET2ET+ET2OT** (but not **OT2OT**)
and **ET2ET+OT2OT+ET2OT**.

ET2ET+OT2OT: in [91], a set of discovery techniques is proposed starting from the relational database. First, the artifact schema is discovered from the relational database. Then, the data is divided between artifacts. This data is used to create event logs for the single artifacts, and to discover the lifecycle of the artifacts. An approach for artifact-centric process discovery specific to SAP ERP is proposed in [80]. Moreover, a meta-model that can be used to extract an event log from generic ERP systems is described in [95], with an application on Dynamics NAV ERP.

A discovery approach for Guard-Stage-Milestone models is proposed in [101, 103], with an initial discovery of Petri nets from the event data and its conversion to GSM models.

The discovery of composite state machines is introduced in [126], along with an implementation as a plug-in of the ProM framework and an assessment on the popular BPI Challenge 2017 dataset. A refinement is proposed in [127], allowing the enhancement of the model with performance metrics.

A user-guided approach for artifact-centric process modeling is proposed in [89] (BAPE, bottom-up artifact-centric process environments) with a process discovery integration that allows to construct the overall process model from the event data of the fragments.

The discovery of object-aware processes in the PHILharmonicFlows platform is described in [31].

ET2ET+ET2OT: the basic idea of mainstream OCPD algorithms is to flatten the OCEL into the single object types, apply a traditional process discovery algorithm for such logs and then collate the

results together. This has been done in [22, 43] (collation of Directly-Follows Graphs) and [119] (collation of Petri nets). In [25], several new metrics (number of events, unique objects, total objects) are proposed to decorate the diagrams. Also, MPM process mining proposes a model as a collation of models for the single object types [85]. An alternative approach includes the Event-to-Object and Object-to-Object relationships [11]. The tool [8] offers OCPD capabilities (OCPNs; visualization of variants).

Case studies on the discovery of OCPNs from manufacturing systems are proposed in [81, 110].

A tool for the discovery of process cubes on top of object-centric event data is proposed in [54], which uses OCDFGs and Petri nets.

ET2ET+OT2OT+ET2OT: several papers have been published on the discovery of object-centric behavioral constraints models [76] [74] [138]. Based on the constraints of the object model, an event correlation approach is also defined [77]. Also, some example applications of the discovery on top of event logs extracted from the Dollibar ERP system and from popular social media platforms [75, 37] are proposed. Object-centric behavioral constraint models are very sensitive to noise. Therefore, [138] introduces a technique to manage the noise during the discovery of such process models. Tool support is available [74], but it is not working with the current version of the ProM framework.

In [2], the care pathways for multi-morbid patients have been visualized using event graphs. These graphs include activity-to-activity, activity-to-entity, and entity-to-entity relationships.

Alternative Approaches: in [47], a process discovery approach for *event knowledge graphs* is proposed, in which the relationships between the single events and objects are represented.

4.3.5 Conformance Checking Techniques

Here, we divide between techniques performing conformance checking on models satisfying **ET2ET+OT2OT** (but not **ET2OT**) or **ET2ET+ET2OT** (but not **OT2OT**).

ET2ET+OT2OT: the main idea to perform conformance checking, proposed in [49, 48], is to split the problem into *behavioral conformance* of the single entities (this can be done using traditional conformance checking techniques such as TBR and alignments) and *interaction conformance* between different entities.

The approach proposed in [46] is based on checking different conditions (constraints over the maximum number of related entities; conditions over the transitions in the lifecycles) starting from a BAUML model.

In [30], conformance checking over object-aware processes is done, with a formalism that allows for flexible lifecycle executions (different workflow nets are created for different conformance categories).

ET2ET+ET2OT: OCPNs have been proposed in [119], which allow for conformance checking on OCELs. On top of OCPNs, the concepts of fitness and precision have been defined [7].

In [98], *object-centric constraint graphs* are introduced with the purpose of representing constraints considering the interaction of objects. These constraints are checked on a live stream of events.

A tool support offering support for OCCC is described in [25]. This allows for the application of standard conformance checking techniques on the flattened logs (e.g., the log skeleton and the temporal profile), but also conformance rules based on the interaction between different objects in the OCEL. The proprietary tool of [85] also implements an OCCC approach, but the exact specification is unavailable.

4.3.6 Performance Analysis Techniques

Here, we collect some performance analysis techniques on top of object-centric process models.

Among models satisfying **ET2ET+OT2OT**, but not **ET2OT**, [127] allows for the decoration of composite state machine models with performance metrics (are delays in the processing of an artifact correlated with delays in another artifact?). Among models satisfying **ET2ET+ET2OT**, but not **OT2OT**, we cite the tool OC-PM [25]. Moreover, [96] introduces the enhancement of OCPNs with performance metrics.

Predictive analytics techniques are described in [52, 4, 55]. The prediction of variables such as the total throughput time is possible thanks to learning a model out of the features of the objects of the OCEL and their interactions.

4.3.7 Other Approaches

Here, we collect approaches that do not belong to any of the aforementioned categories.

In [56], an extension of colored Petri nets is proposed with two main highlights:

- The tokens are objects, and they can be created either by firing a transition (for example, an order with identifier o_{9356} is obtained by firing the *Create Order* transition), or from a catalogue (for example, the order o_{9356} includes the product *iPhone*).
- Advanced conditions allowing to fire a transition can be inserted and checked from an underlying database. As an example, the shipment of an order with a total mass of 10000 kg cannot be done on a truck t_1 with a capacity of 5000 kg, while it can be done on a truck t_2 with a capacity of 20000 kg (the capacity of the trucks t_1 and t_2 is checked on the database).
- Data can be added to the objects/tokens after firing a transition (always with the possibility to look up the underlying database).

This is more advanced than the models presented in the earlier sections of the paper, and a discovery operation would not need only to discover the control-flow perspective for every object type, but also to consider advanced interactions between the objects (e.g. $\sum_{\text{item} \in \text{order}} \text{weight}(\text{item}) \leq \text{capacity}(\text{truck})$). In [123], the concept of token is associated with a set of objects. For example, the placement of an order involves an *order* object and several *item* objects, which are logically connected, therefore it makes sense to consider them as a single token in the Petri net. The type and cardinality of the objects are taken into account when firing a transition.

Currently, no process discovery/conformance checking approach is available for either [56] (extension of colored Petri nets) and [123] (tokens associated with sets of objects). However, the information needed to discover such models is contained in the OCEL (more specifically, in the *E2E* and *O2O* relationships, and the attributes at the event/object level), therefore we expect the proposal of process discovery/conformance checking techniques in the future.

Table 4.2: Summary of the included results.

Title	Year-Month	Log Type			Model Type			Technique						Quality			
		E2O	O2O	E2E	ET2ET	OT2OT	ET2OT	Extraction	Storage	Preprocessing	Discovery	Conformance	Perf.Analysis	Scalability	Generalization	Tool Support	Case Study
A Virtual Knowledge Graph Based Approach for OCELs Extraction [137]	202310	X						X							X	X	
Process mining for artifact-centric blockchain applications [84]	202309							X	X								X
Normalizing object-centric process logs by applying database principles [68]	202305	X							X	X					X		
Object-Centric Process Predictive Analytics [52]	202303	X											X		X		
Object Type Clustering Using Markov Directly-Follow Multigraph in OCPM [62]	202212	X								X					X		
A Framework for Extracting and Encoding Features from Object-Centric Event Data [4]	202211	X								X			X		X	X	
Object-Centric Predictive Process Monitoring [55]	202211	X								X			X		X		
Enabling Multi-process Discovery on Graph Databases [43]	202210	X			X		X				X			X	X		
OPerA: Object-Centric Performance Analysis [96]	202210	X			X		X				X		X		X	X	
ocpa: A Python library for object-centric process analysis [5]	202210	X			X		X		X	X	X	X	X		X	X	
Enhancing Data-Awareness of OCELs [57]	202210	X							X	X					X		
Interactive Process Identification and Selection from SAP ERP [134]	202210	X						X								X	
Konekti: A Data Preparation Platform for Process Mining [133]	202210	X						X	X						X	X	
Clustering Analysis and Frequent Pattern Mining for Process Profile Analysis: An Exploratory Study for OCELs [51]	202210	X								X					X		
Defining Cases and Variants for Object-Centric Event Data [6]	202210	X								X					X	X	
Monitoring Constraints in Business Processes Using Object-Centric Constraint Graphs [98]	202210	X										X			X	X	
Uncovering Object-Centric Data in Classical Event Logs for the Automated Transformation from XES to OCEL [108]	202209	X							X	X					X		
Assessing the Suitability of Traditional Event Log Standards for IoT-Enhanced Event Logs [28]	202209	X							X								
OC-PM: Analyzing OCELs and Process Models [25]	202208	X			X		X				X	X	X	X	X	X	
Discovery of Object-Centric Behavioral Constraint Models With Noise [138]	202208	X	X		X	X	X				X				X	X	
Extraction of OCELs through Virtual Knowledge Graphs [136]	202208	X			X		X	X							X	X	
Extracting Artifact-Centric Event Logs From Blockchain Applications [86]	202207		X					X	X							X	
Petri net-based object-centric processes with read-only data [56]	202207														X		
OCπ: Object-Centric Process Insights [8]	202206	X			X		X			X	X				X	X	
Process Mining over Multiple Behavioral Dimensions with Event Knowledge Graphs [47]	202206	X	X	X							X				X		
Data and Process Resonance - Identifier Soundness for Models of Information Systems [123]	202206														X		
Enabling Conformance Checking for Object Lifecycle Processes [30]	202205		X		X	X						X				X	
Process Mining to Discover the Global Process from its Fragments' Executions [89]	202204		X		X	X				X	X				X		
A Methodology to Apply Process Mining in End-To-End Order Processing of Manufacturing Companies [110]	202201	X			X		X			X	X						X
Multi-Dimensional Event Data in Graph Databases [45]	202112	X	X	X	X	X	X		X	X	X			X	X	X	
An Event Data Extraction Approach from SAP ERP for Process Mining [18]	202111	X						X		X					X	X	
Associative Intelligence for OCPM with MPM [85]	202111	X			X		X				X	X		X		X	
A Scalable Database for the Storage of OCELs [12]	202111	X							X	X	X			X	X	X	
Precision and Fitness in OCPM [7]	202111	X			X		X					X			X		
A Python Tool for OCPM Comparison [54]	202111	X			X		X			X	X		X		X	X	
Discovering Care Pathways for Multi-morbid Patients Using Event Graphs [2]	202110	X	X		X	X	X				X						
Federated Process Mining: Exploiting Event Data Across Organizational Boundaries [116]	202109		X	X						X					X		
Storing and Querying Multi-dimensional Process Event Logs Using Graph Databases [44]	202109	X	X	X	X	X	X		X	X	X			X	X	X	
OCEL: A Standard for OCELs [53]	202108	X							X						X	X	
Discovery and digital model generation for manufacturing systems with assembly operations [81]	202108	X			X		X			X	X						X
Towards a Domain-Specific Modeling Language for Extracting Event Logs from ERP Systems [94]	202106		X					X									
Merging event logs for inter-organizational process mining [59]	202106		X	X						X					X	X	
Discovering OCPNs [119]	202012	X			X		X				X	X	X		X	X	
Case notion discovery and recommendation: automated event log building on databases [39]	202007							X	X	X					X	X	
Behavioural Analysis of Multi-Source Social Network Data Using Object-Centric Behavioural Constraints and Data Mining Technique [37]	202007	X	X		X	X	X	X			X						

Title	Year-Month	Log Type			Model Type			Technique						Quality			
		E2O	O2O	E2E	ET2ET	OT2OT	ET2OT	Extraction	Storage	Preprocessing	Discovery	Conformance	Perf.Analysis	Scalability	Generalization	Tool Support	Case Study
Towards the discovery of object-aware processes [31]	202002		X		X	X					X					X	
Guided Interaction Exploration and Performance Analysis in Artifact-Centric Process Models [127]	201912		X		X	X					X		X		X	X	
OCPM: Dealing with Divergence and Convergence in Event Data [115]	201909	X									X				X		
Process Mining on Event Graphs: a Framework to Extensively Support Projects [11]	201909	X	X	X						X					X		
Extracting Multiple Viewpoint Models from Relational Databases [22]	201908	X			X		X	X	X	X	X			X	X	X	
Process Mining in Social Media: Applying Object-Centric Behavioral Constraint Models [75]	201906	X	X		X	X	X	X			X						
Conformance checking in UML artifact-centric business process models [46]	201902		X		X	X			X			X					
Dealing with Artifact-Centric Systems: a Process Mining Approach [74]	201812	X	X		X	X	X	X	X		X					X	
Multi-instance Mining: Discovering Synchronisation in Artifact-Centric Processes [126]	201809		X		X	X					X				X	X	
A domain-specific language for supporting event log extraction from ERP systems [93]	201807		X					X	X								
Configurable Event Correlation for Process Discovery from Object-Centric Event Data [77]	201807	X	X		X	X	X			X					X	X	
Extracting OCELs to Support Process Mining on Databases [79]	201806	X	X		X	X	X	X	X						X		
Merging event logs: Combining granularity levels for process flow analysis [107]	201711		X	X						X					X	X	
Automatic Discovery of Object-Centric Behavioral Constraint Models [76]	201706	X	X		X	X	X	X			X				X	X	
Metamodel of the Artifact-Centric Approach to Event Log Extraction from ERP Systems [95]	201610		X					X									
Merging event logs for process mining with hybrid artificial immune algorithm [139]	201601		X	X						X					X		
Discovering Interacting Artifacts from ERP Systems [80]	201508		X		X	X		X		X	X					X	
Process Mining on Databases: Unearthing Historical Data from Redo Logs [40]	201508							X	X	X					X	X	
Merging Event Logs with Many to Many Relationships [106]	201409		X	X						X					X		
Merging event logs for process mining: A rule based merging method and rule suggestion algorithm [36]	201407		X	X						X					X	X	
Discovering Unbounded Synchronization Conditions in Artifact-Centric Process Models [102]	201308		X		X	X					X						
Artifact Lifecycle Discovery [103]	201303		X		X	X					X						
Automatic Discovery of Data-Centric and Artifact-Centric Processes [91]	201209		X		X	X		X			X						
From Petri Nets to Guard-Stage-Milestone Models [101]	201209		X		X	X					X						
Conformance Checking of Interacting Processes with Overlapping Instances [49]	201109		X		X	X						X			X		
Behavioral Conformance of Artifact-Centric Process Models [48]	201106		X		X	X						X			X		

4.4 Dimensions

In this section, we verify the dimensions of the papers under different criteria (scalability, generalization, availability of tool support, and case studies on real-life event data). We divide the analysis between techniques working with process models belonging to the **ET2ET** and the **ET2OT** categories (but not **OT2OT**), techniques working with process models belonging to the **ET2ET** and the **OT2OT** categories (but not **ET2OT**); techniques working with process models belonging to the three categories **ET2ET**, **OT2OT** and **ET2OT**.

4.4.1 Scalability

In this section, we discuss the scalability of the techniques as a dimension influencing the applicability of object-centric techniques in real-life contexts.

ET2ET+ET2OT: since mainstream OCPD techniques are based on: i) flattening the OCEL on the different object types; ii) discovering a traditional process model on the flattened log; iii) collating everything together, the discovery techniques ([22, 119, 25]) have good scalability. However, the process enhancement phase (decorating the model with frequency or performance metrics) requires a replay operation that has lower scalability (considering the metrics described in [119] and [96]).

ET2ET+OT2OT: these models can be discovered from the event data in three different phases: i) discovery of the lifecycle of the entities; ii) discovery of the interaction between different entities; iii) for some models, the discovery of the synchronization conditions between the entities. While for the first phase process discovery algorithms on traditional event logs can be used, therefore good scalability could be achieved, for the second and especially the third phase the scalability is generally problematic.

ET2ET+OT2OT+ET2OT: the discovery of the declarative constraints of object-centric behavioral constraint models [76] scales poorly. Moreover, also the event log format (XOC) proposed in [79] is of difficult usage in real-life contexts due to the persistence of the status of the database for every event.

4.4.2 Generalization

In this section, we discuss the generalization of the techniques as a dimension influencing the applicability in different real-life scenarios.

ET2ET+ET2OT: these techniques can be applied to OCELs, and some publicly available OCELs can be downloaded from `http://www.ocel-standard.org/`. Hence, most of these techniques are generic and do not depend on the target information systems.

ET2ET+OT2OT: composite state machines can be discovered starting from XES event logs using the *CSM miner* plugin [126]. Therefore, the technique can be applied to any event log. Other techniques are highly dependent on the database ([91])/information system ([80]), hence they cannot be generalized.

ET2ET+OT2OT+ET2OT: object-centric behavioral constraint techniques [76] start from XOC logs [79], hence they are generic. However, due to the problems in the scalability and the lack of publicly available XOC logs, the application of such techniques to mainstream systems is difficult. The authors describe in a scientific paper the extraction of event logs from the Dollibar ERP system [79] and from popular social media platforms [75].

4.4.3 Availability of Tool Support

The main open-source tool supports for OCPM are OC-PM [25] (offering process discovery, conformance checking, machine learning, and filtering algorithms) and OC-π [8] (offering process discovery and variants visualization capabilities). The following scientific tools are also covering OCPM analyses:

- Process cubes are described in [54] for process comparison purposes.
- A proprietary tool for OCPM is described in [85].
- A tool to extract OCELs from SAP ERP systems is proposed in [18].
- A Python library for OCPM is described in [5].

Tool support is offered for the discovery of object-centric behavioral constraint models [76] as a plugin of the ProM framework[3]. Other types of models can be discovered (for example, [80]), however, the tool support is older and difficult to launch.

[3]The plug-in is of difficult usage and the public available XOC logs are difficult to retrieve.

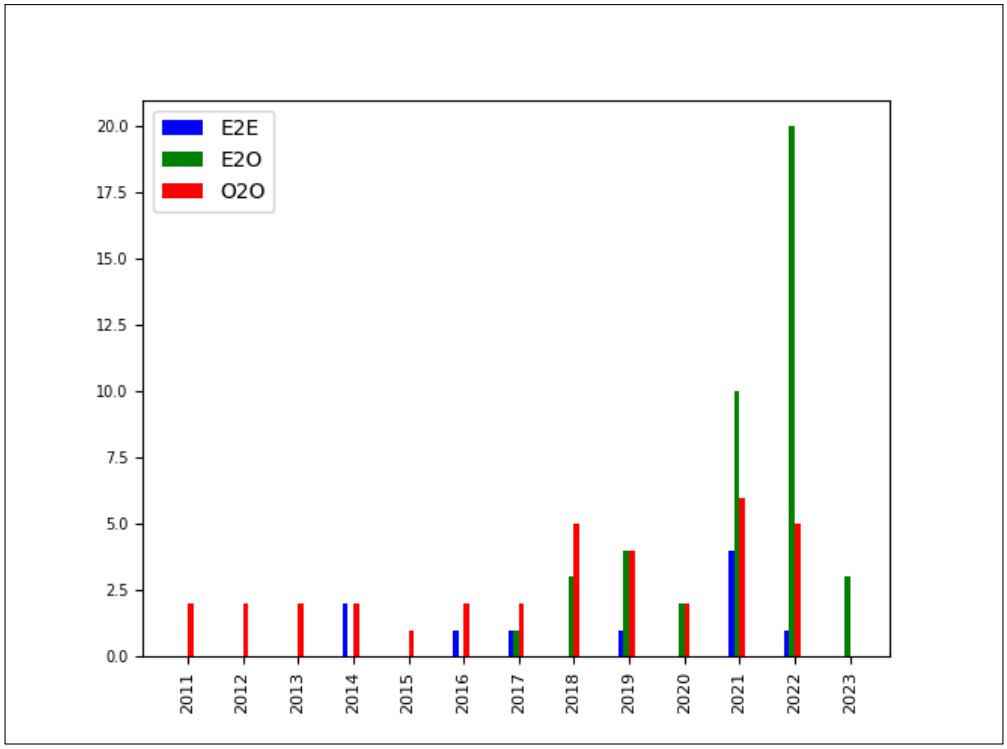


Figure 4.2: Categories of event logs used in the considered papers.

4.4.4 Case Studies using Real-Life Event Data

Here, we analyze which papers included in the review have applied their techniques to case studies on real-life information systems. These would help to prove the readiness of the discipline. This criterion excludes papers with an assessment done on a publicly available real-life event log, because the questions and the results have not been discussed with the business.

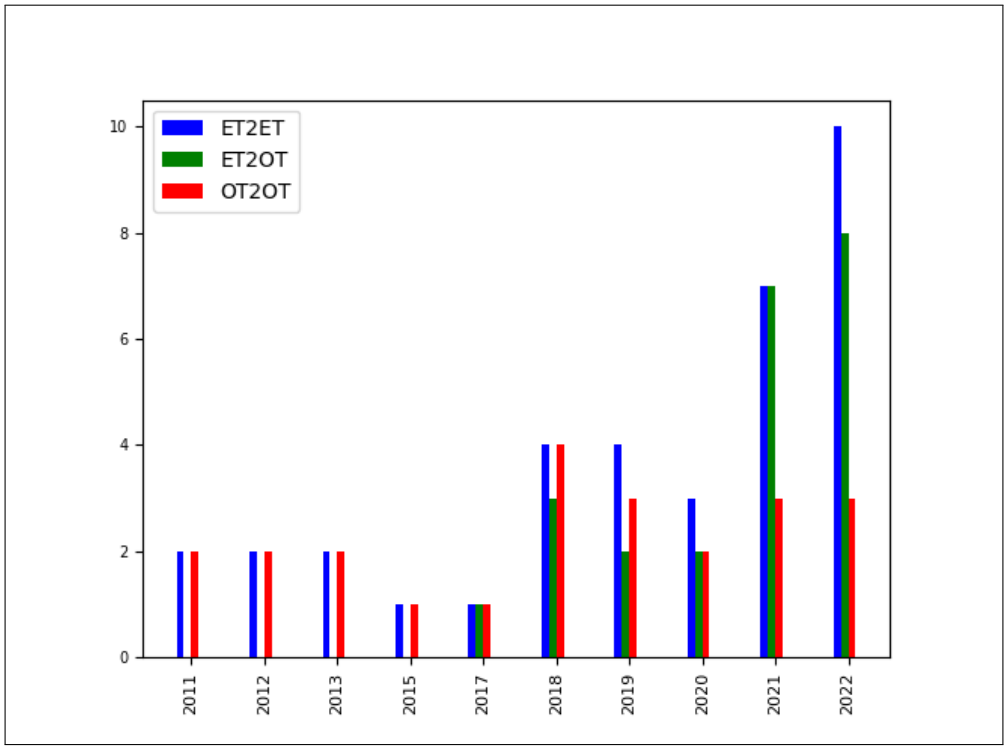


Figure 4.3: Categories of process models used in the considered papers.

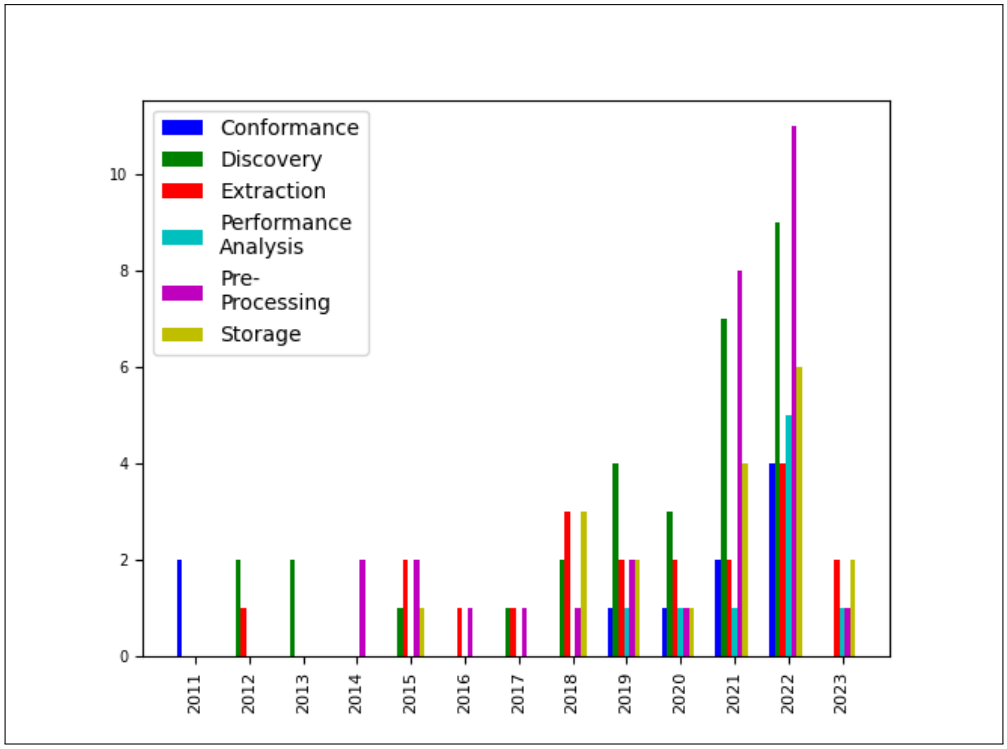


Figure 4.4: Categories of techniques used in the considered papers.

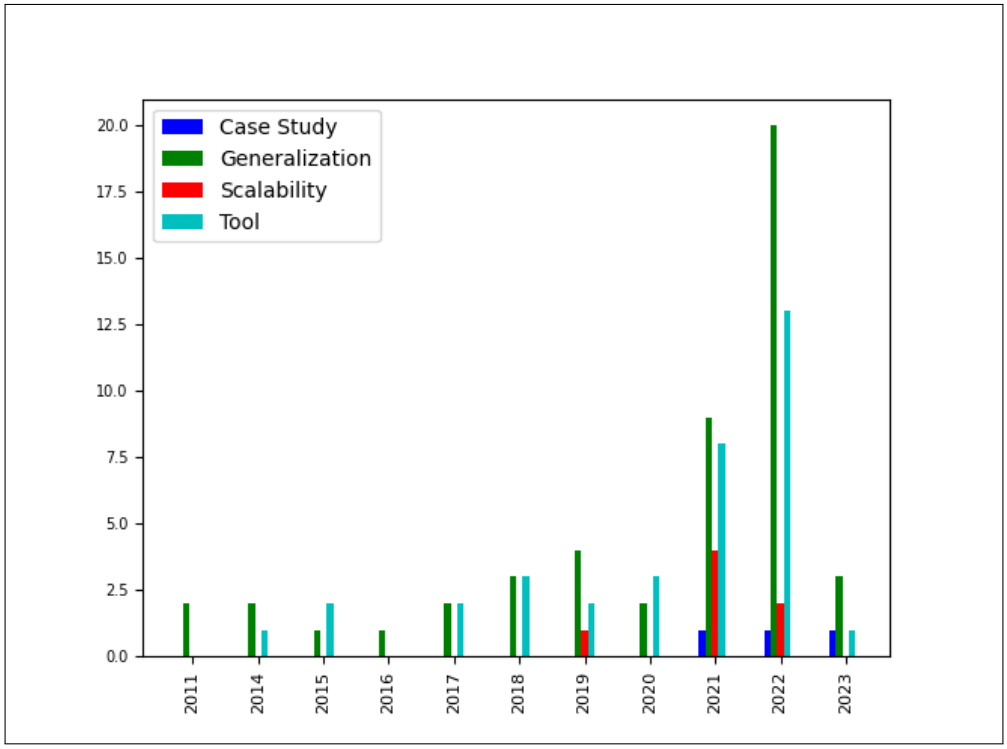


Figure 4.5: Criteria used in the considered papers.

Unfortunately, only a few case studies have been proposed. In [81], the problem of modeling a production system including different assembly stations (where every station records the event data singularly) using object-centric process models is assessed, and a flow shop including the assembly stations is successfully discovered. This allows for building a simulation model with the desired behavior. In [110], an end-to-end object-centric process is extracted with different stages of the manufacturing process (sales, manufacturing, shipping).

In [84], an application of artifact-centric process mining to public Ethereum applications is proposed.

4.5 Temporal Evolution of the Discipline

We represented in four graphs the temporal evolution of the discipline:

- In Figure 4.2, we show the usage of different categories of event logs (**E2O**, **O2O**, **E2E**) during the considered timeframe. Event logs containing Object-to-Object relationships were mostly used between 2011 and 2016, while recently the interest moved to event logs in which an event can be related to different objects of different object types.
- In Figure 4.3, we show the usage of different categories of process models (**ET2ET**, **OT2OT**, **ET2OT**) during the considered timeframe. Between 2011 and 2016, the focus was on process models describing **ET2ET** and **OT2OT** interactions, while recently the interest moved to process models describing **ET2ET** and **ET2OT** interactions.
- In Figure 4.4, we show a categorization of the techniques (extraction, storage, preprocessing, discovery, conformance, performance analysis) used in the papers during the considered timeframe. Greater focus is put in recent times on storage, preprocessing, and performance analysis.
- In Figure 4.5, we show a categorization of some criteria (scalability, generalization, tool support, case studies) used in the papers during the considered timeframe. The availability of tool support, and the interest in scalability, increased over time.

4.6 Description of Existing Process Models

Here we include the description of a subset of existing process models that have been adopted in the scientific research.

4.6.1 Artifact-Centric Models

Building upon the foundational concepts elucidated by the BALSA framework [60] and BAUML models, we explore artifact-centric business process models in this section. The BALSA framework posits four integral dimensions that should be inherent in any artifact-centric business process model, as enumerated:

- *Business Artifacts*: These are significant data crucial for the attainment of business goals. Business artifacts, their relationships, and constraints can be depicted using entity-relationship models or UML class diagrams.
- *Lifecycles*: These delineate the life journey of an artifact from its inception to its eventual disposal. Statecharts or state machine diagrams can serve as visual representations.
- *Services*: Fundamental to the business process, services evolve the process by creating, updating, and deleting artifacts. They can be represented through a range of techniques, from natural language to logic, or via operation contracts specified in OCL.
- *Associations*: These lay down the restrictions on the manner in which services can modify artifacts, imposing constraints on services. These can be depicted using procedural representations, such as a workflow or BPMN, or using declarative representations, such as condition-action rules.

This section's remaining content is dedicated to introduce several artifact-centric models.

Guard-Stage-Milestone Models

Introduced by IBM Research, the Guard-Stage-Milestone (GSM) [61] is an artifact-centric business process model. It was developed as a reaction to the shortcomings of process-centric models that are not capable of adequately handling data and complex correlations between various data objects.

The GSM model has its roots in the Business Artifact Model (BAM) [90], which was created by Nigam and Caswell. GSM extended the Business Artifact Model by adding lifecycles to the artifacts. The lifecycles are described by a series of stages, where each stage represents a significant period in the life of the artifact.

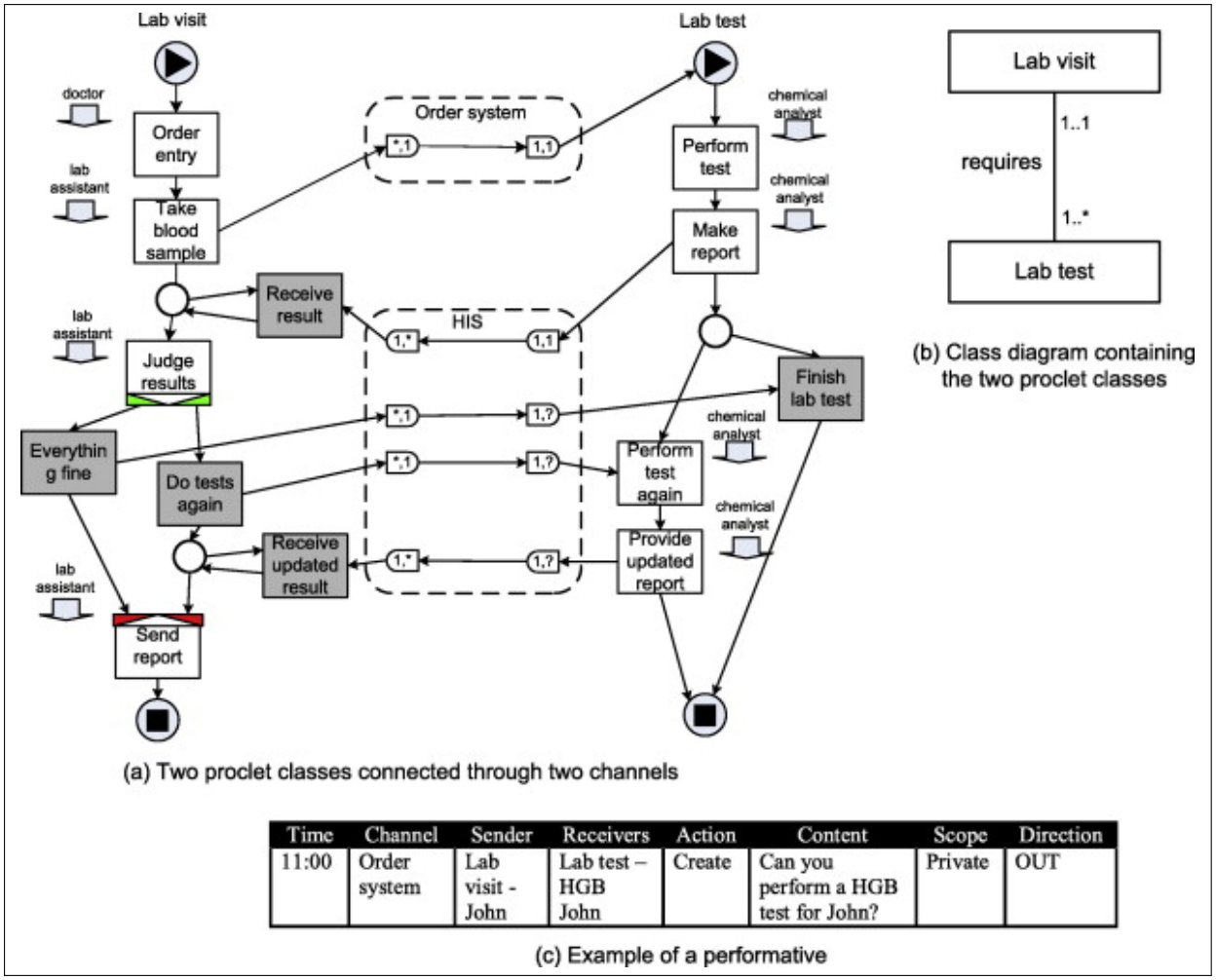
In a GSM model, an "artifact" is an abstract data construct, representing a key business entity tracked through a business process. Each artifact instance has a data schema (a set of data fields), and an associated lifecycle model which is a state-machine-like construct, built using stages, milestones, and guards:

- *Stages*: A stage represents a significant period in an artifact's lifecycle. It has a start and an end, and contains a set of milestones.
- *Milestones*: Milestones are specific, significant achievements within a stage. They indicate the progress of an artifact within a particular stage but do not influence the flow of control.
- *Guards*: Guards are conditions associated with transitions between stages. They control the progress of the artifact through the lifecycle by monitoring milestone achievements and data changes.

Proclets

In [117, 118] an innovative modeling paradigm based on Petri nets is proposed, which conceptualize complex business processes as a collection of interacting, smaller process fragments, or proclets (see Figure 4.6). This model improves the clarity and flexibility of process representation, particularly in distributed and dynamic environments.

The interaction between proclets is governed by the exchange of messages, known as *performatives*. A performative is characterized by several attributes, including the time of creation or receipt, the channel used for exchange, the sender (proclet generating the performative), and the set of receivers (proclets designated to receive the performative).



Time	Channel	Sender	Receivers	Action	Content	Scope	Direction
11:00	Order system	Lab visit - John	Lab test – HGB John	Create	Can you perform a HGB test for John?	Private	OUT

(c) Example of a performative

Figure 4.6: Example healthcare process expressed as a collection of proclets (taken from [82]).

A critical component of the proclet framework is the concept of *channels*. Channels serve as mediums to facilitate the transport of messages, or performatives, from one proclet to another. Depending on the channel's properties, different interaction types can be supported, including push/pull communication, synchronous/asynchronous interaction, and verbal/non-verbal communication. Channels also offer the flexibility to send messages to a specific proclet or a group of proclets, supporting multicast communication.

Object-Aware Processes (in PhilharmonicFlows)

The PhilharmonicFlows project, developed by a dedicated team of researchers from the University of Potsdam, Germany, is a novel initiative focusing on the development of a robust, object-aware process management system, with the aim of achieving seamless integration and flexible management of business data and processes, thereby overcoming the limitations of traditional Workflow Management Systems.

This led to the development of the concept of "Object-aware Process Management" in PhilharmonicFlows [69]. This approach aims to offer a generic component which possesses the beneficial features found in hard-coded application systems while also leveraging the advantages known from WfMS.

Several challenges were identified that, when addressed, could significantly advance process management technology. These challenges revolved around integrating views of data and processes, defining uniform granularity for process modeling considering underlying data structure, synchronizing concurrently executed process instances, enabling data-based process modeling and data-driven process execution, ensuring flexible activity granularity, and addressing user involvement in Object-aware Process Management Systems. By addressing these challenges, PhilharmonicFlows aims to provide a clear methodology for harmonized process modeling, a separation between business rules and process logic, controlled evolution of data and processes, and a knowledge-driven process execution to better assist users, among

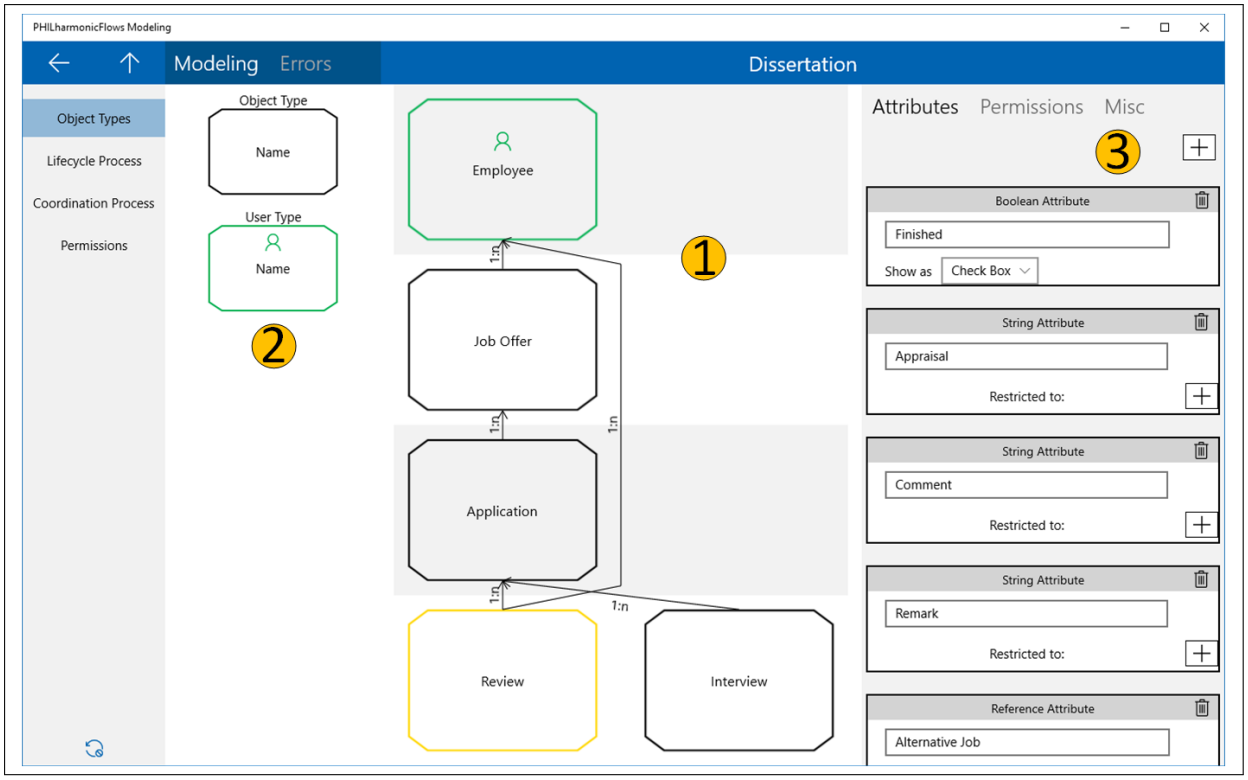


Figure 4.7: Object and relation type modeling view in PhilharmonicFlows (taken from [112]).

other benefits.

The goal of PhilharmonicFlows is to develop a comprehensive framework for an Object-aware Process Management System that addresses these challenges. This project aims to offer detailed insights into the different components of this framework and their complex interdependencies. Ultimately, PhilharmonicFlows is envisioned to make a significant contribution towards achieving more flexible process management technology, where daily work can be carried out in a more intuitive manner.

The BAUML Specification

In [34], the BALSA framework is extended through the development of the Business Artifact-centric Unified Modeling Language (BAUML) model (an example is reported in Figure 4.9). BAUML represents the four essential dimensions of an artifact-centric business process model using UML and OCL, standard languages typically used for conceptual modeling. Specifically, BAUML uses UML class diagrams for artifact, object, and relationship types; UML state machine diagrams for artifact lifecycles; UML activity diagrams for associations, and OCL operation contracts for services.

A BAUML model B is defined in [34] as a tuple (M, O, S, P), where:

- M is a UML class diagram, with some classes representing business artifacts. Each artifact is the top class of a hierarchy whose leaves are subclasses with dynamic behavior.
- O is a set of OCL constraints over M.
- S is a set of UML state transition diagrams, one per artifact in $artifacts(M)$. For each artifact, S contains a state transition diagram that encodes the allowed event-driven transitions of an artifact instance from the current state to a new subclass.
- P is a set of UML activity diagrams, such that for every transition diagram and for every event, there exists one and only one activity diagram.

In BAUML, conditions are expressed as OCL queries over the UML class diagram M. Each atomic task is associated with an operation contract, which expresses a precondition on the executability of the task, and a postcondition describing the effect of the task, both formalized in terms of OCL queries over M. Thus, tasks in this context represent services as per the BALSA framework's terminology.

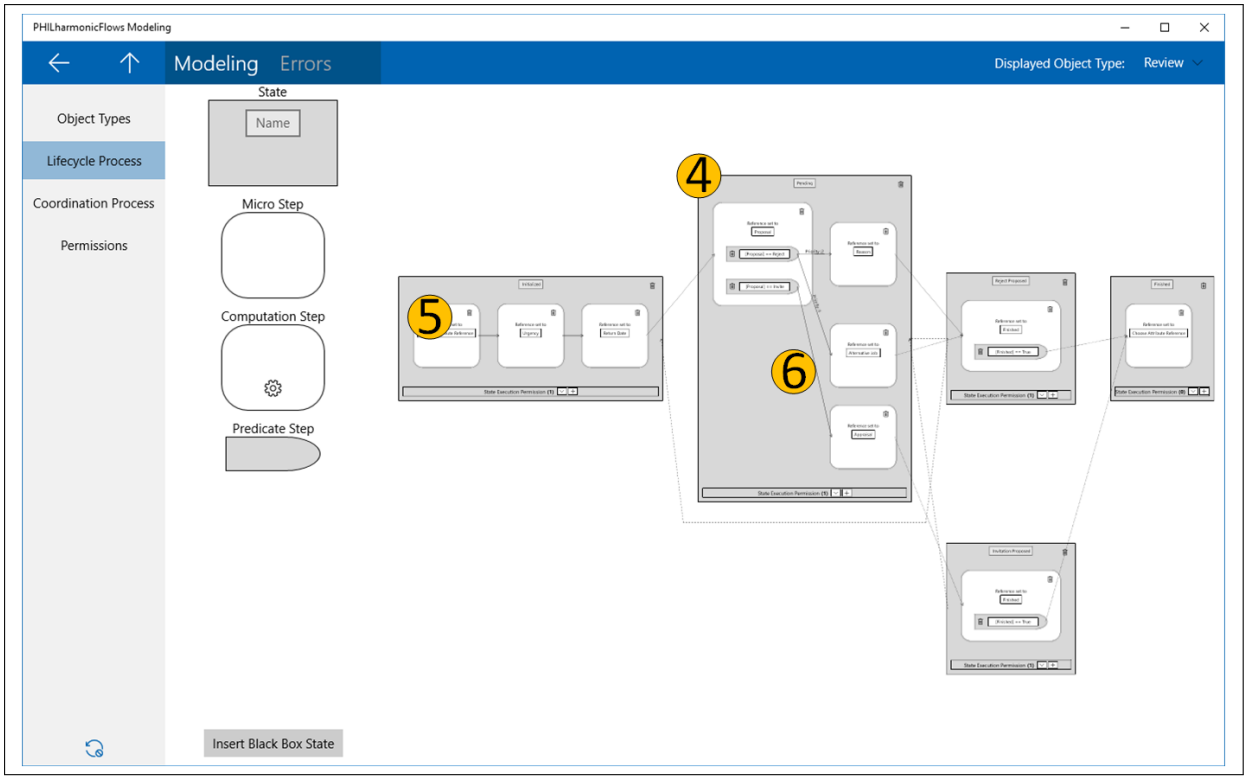


Figure 4.8: Lifecycle modeling view in PhilharmonicFlows (taken from [112]).

4.6.2 Object-Centric Behavioral Constraint Models

In [78], the novel paradigm of Object-Centric Behavioral Constraint (OCBC) models is introduced. Drawing from a diverse array of influential frameworks, the OCBC models ingeniously amalgamate concepts from declarative, constraint-based languages, such as Declare, with elements grounded in data/object modeling techniques like Entity-Relationship (ER), Unified Modeling Language (UML), or Object-Role Modeling (ORM). A distinguishing trait of OCBC models is the utilisation of cardinality constraints as a unifying mechanism. This mechanism effectively addresses the data and behavioral dependencies, as well as their intricate interplay. The primary application of OCBC models is illustrated in Figure 4.10.

Given the underlying methodologies, these models encapsulate both behavior and data aspects of business processes in a harmonised manner. Consequently, the OCBC models provide an invaluable analytical tool that reconciles the object-oriented approach to data management with the intricacies of process behavior, thereby enabling a more comprehensive and integrated analysis of business processes.

Object-Centric Behavioral Constraint (OCBC) models fundamentally rely on the input of specific logs, termed OCELs, or XOC logs in shorthand. Within each log, individual events reflect an associated object model, appearing under the "Object Model" column. This model represents the state of the process immediately subsequent to the event's execution. Besides the object model, each event aligns with a specific activity and might encompass additional attributes, for instance, the timestamp of the event's occurrence.

Despite the strengths of XOC logs in representing process behavior and data, one must acknowledge that they present scalability issues. Specifically, the requirement of storing the entire state of the model for each event can lead to substantial data storage demands, particularly for complex processes with a high number of events. This can result in both storage inefficiencies and computational difficulties when analyzing large logs.

To address these scalability issues, a subsequent version of XOC logs proposes a different approach: instead of storing the complete state for each event, the logs now keep track of the updates to the state. This approach significantly reduces the storage requirements, as only changes to the state are recorded, not the complete state at each event. This method enhances the scalability of XOC logs, making them more suitable for handling larger, more complex processes while maintaining their ability to represent both the behavioral and data perspectives of processes effectively.

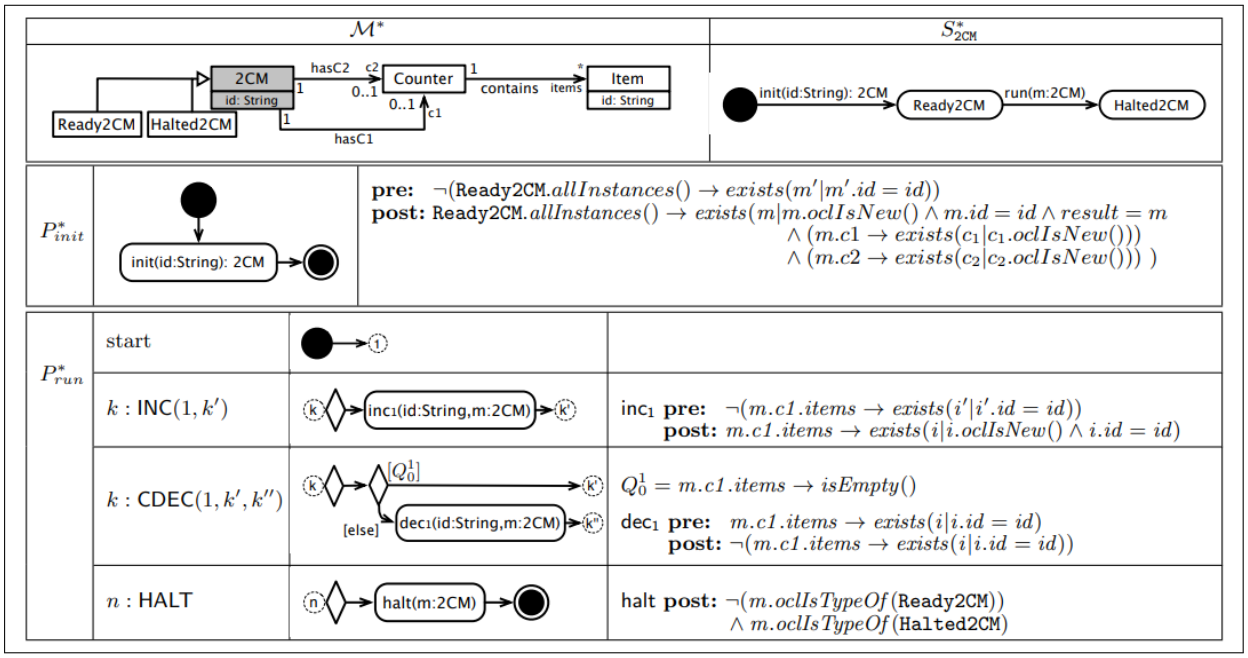


Figure 4.9: Example BAUML process model (taken from [34]).

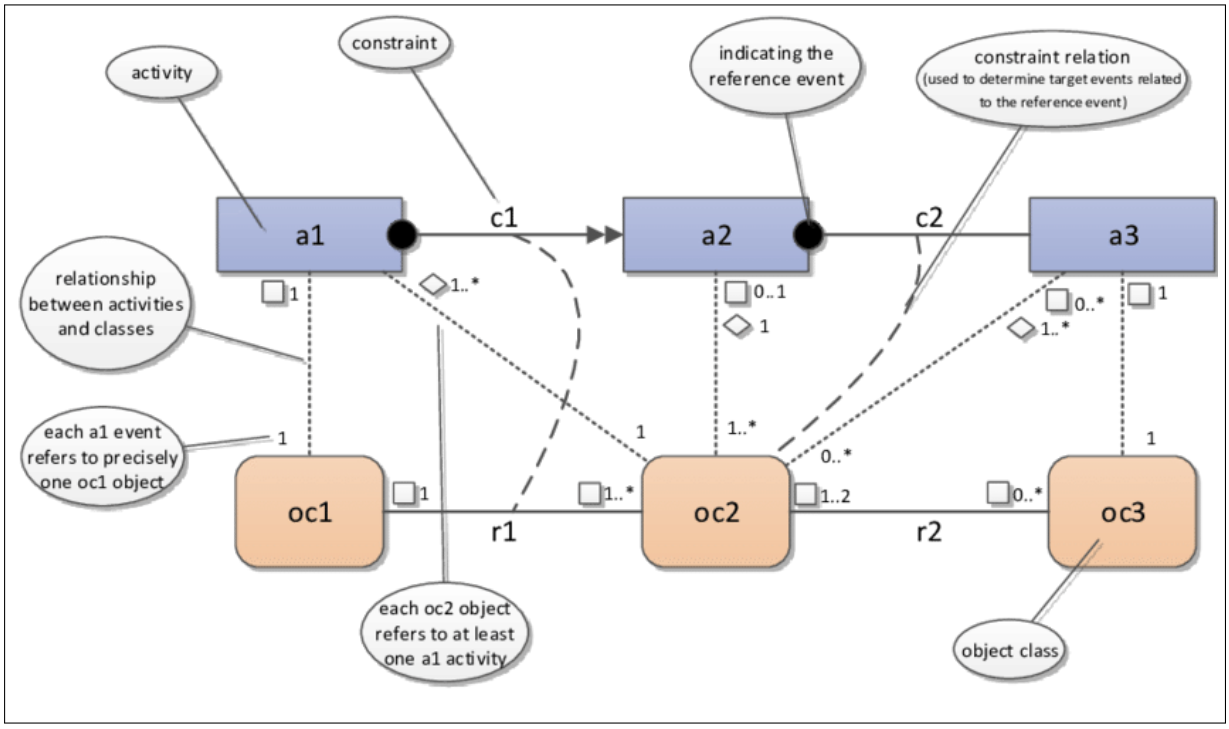


Figure 4.10: Object-centric behavioral constraints model (taken from [76]). This process model belongs to the **ET2ET** (activities are connected by arcs), **ET2OT** (activities are connected to object types), and **OT2OT** (object types are connected by arcs) categories.

Conclusion

This chapter has provided a comprehensive overview of the existing research landscape in Object-Centric Process Mining. By systematically analyzing existing approaches and methodologies, we have identified key strengths and weaknesses, common themes, and areas ripe for further exploration. The discussion of different OCEL formats, process models, and analysis techniques has highlighted the diversity of approaches within the field. The evaluation of these approaches across various dimensions, including scalability, generalization, and tool support, has provided a critical perspective on the current state-of-the-art. Furthermore, by tracing the temporal evolution of OCPM, we have identified key trends and shifts in research focus, paving the way for future research directions. While significant advancements have been made in areas like data extraction and process discovery, challenges remain, particularly in the development of scalable and robust conformance checking techniques and the application of OCPM to a wider range of real-world scenarios. These limitations and open questions motivate the research contributions presented in the following chapters, where we will introduce novel methodologies and techniques aimed at addressing these challenges and advancing the field of OCPM. Specifically, the insights gained from this literature review will inform the design and evaluation of our proposed approaches for data extraction, process discovery, and conformance checking, ensuring that our contributions build upon and extend the current state-of-the-art.

Chapter 5

Data Extraction and Preprocessing

"*Data is a precious thing and will last longer than the systems themselves.*"
Tim Berners-Lee, Inventor of the World Wide Web.

Introduction

In the realm of Object-Centric Process Mining (OCPM), the quality and structure of event data serve as the bedrock for meaningful analysis and informed decision-making. Without a robust method to extract and preprocess the data scattered across various relational databases, the potential insights remain locked behind complexity, redundancy, and noise. The methods presented in this chapter build upon previous contributions worked by the author of this thesis, including:

- The introduction of a semi-automated methodology for extracting OCELs from relational databases, specifically tailored for SAP ERP, as in [18], providing a first iterative approach for identifying tables and generating OCELs.
- The refinement and generalization of the extraction methodology, also assessing scalability and quality aspects, as in [19], thereby establishing a systematic, end-to-end solution for OCEL generation.
- The definition of the OCEL standard for storing object-centric event logs, as introduced in [53], ensuring a uniform and extensible format for representing complex, object-centric data.
- The development of a tool that supports the semi-automated extraction of OCELs from SAP ECC ERP systems, as presented in [134], enabling practitioners to interactively select processes and generate OCELs.
- The evolution of the OCEL standard into version 2.0, providing robust resources and improved specifications as described in [67], ensuring enhanced capabilities for representing dynamic attributes and object-to-object relationships.

This chapter leverages these foundational contributions to tackle the process of transforming raw relational data into enriched, well-organized OCELs. We begin by introducing a systematic approach to event data extraction, starting with process identification and table selection. By leveraging abstract models and blueprints, we establish a scalable framework for mapping domain-specific databases into standardized OCEL formats, encompassing both the original OCEL 1.0 and the enhanced OCEL 2.0 representations. This ensures that organizations can construct logs that capture the complex interplay of multiple object types—such as orders, customers, and resources—and their associated events.

Once the data is in OCEL form, we turn to the critical step of preprocessing. This chapter presents a range of filtering techniques that allow analysts to narrow their focus, removing irrelevant events and objects, and homing in on the most pertinent slices of the process. From filtering based on event timestamps or activities to focusing on specific object types or lifecycle attributes, we demonstrate how flexible and iterative data refinement leads to clearer, more tractable datasets. Maintaining consistency—ensuring no orphaned events or objects remain isolated—is emphasized as a key aspect of reliable data preparation.

To illustrate the utility of these methods, we introduce the concept of On-Time In-Full (OTIF) as a representative performance indicator that can only be meaningfully evaluated against high-quality, well-structured object-centric data. OTIF, common in supply chain and logistics domains, measures whether

deliveries meet both timing and quantity requirements. Through the lens of OTIF and other real-world applications, we show how the filtering and preprocessing strategies enable finer-grained analyses, allowing organizations to detect inefficiencies, improve their operations, and realize more value from their OCPM initiatives.

In essence, this chapter provides a roadmap from raw relational structures to refined object-centric logs—equipping practitioners and researchers with the tools needed to ensure their OCPM analyses rest on a solid foundation. With these extraction, preprocessing, and filtering techniques, practitioners can confidently derive actionable insights, benchmark performance metrics like OTIF, and pave the way for more advanced object-centric process analytics.

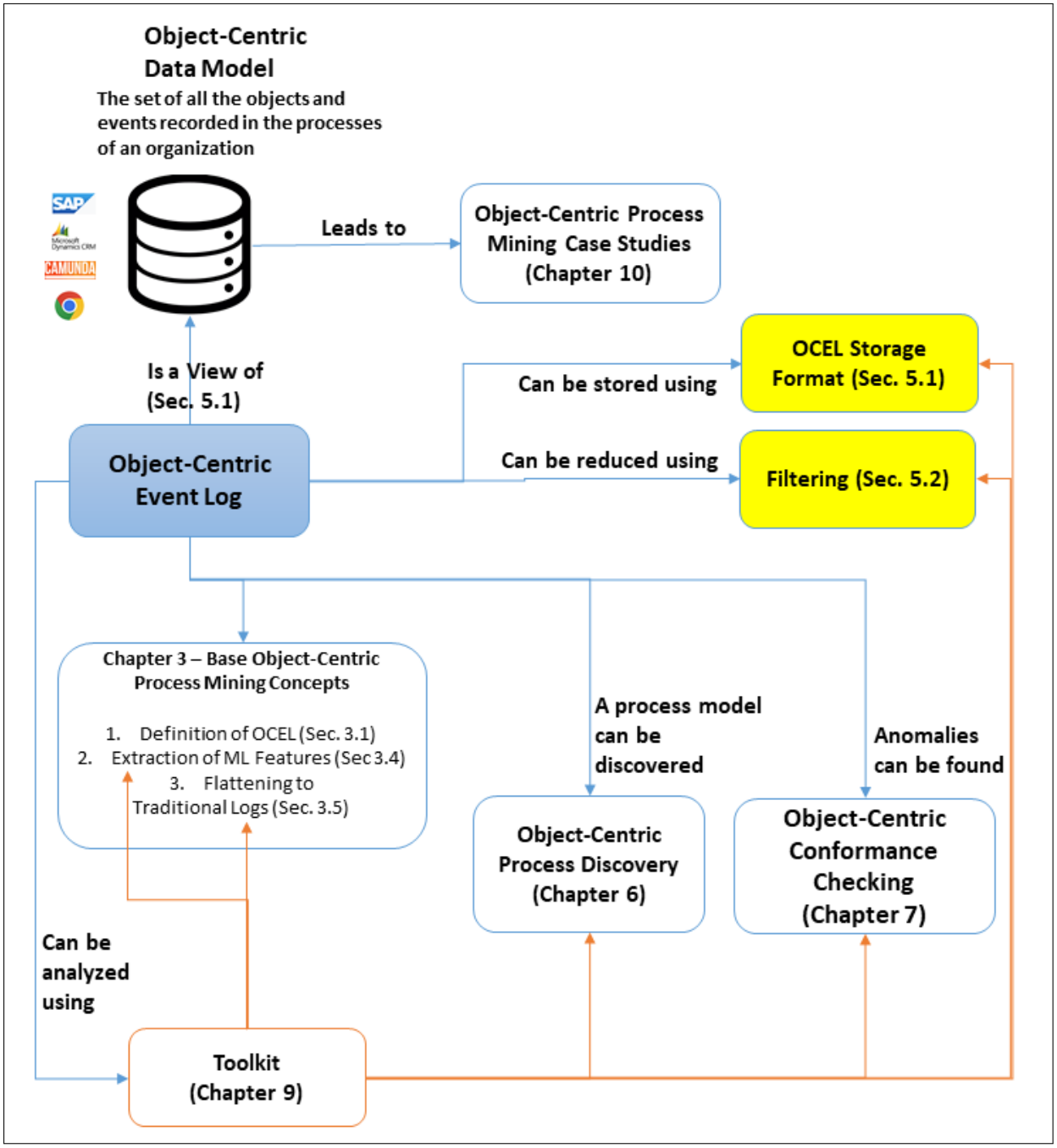


Figure 5.1: Highlight of the problems tackled in this specific chapter.

5.1 Data Extraction and Storage for OCELs

5.1.1 Data Extraction

Databases, and particularly relational databases, play an essential role in process mining. They are often the primary source of event data which can be used to reconstruct the process execution. In many organizations, business processes are supported by various IT systems like ERP, CRM, or bespoke software applications. As these systems carry out tasks, they generate data about the activities they perform. This data, which includes information about what happened, when it happened, and who or what was involved, is often stored in a database. Relational databases, with their structured tables and powerful SQL querying capabilities, are especially common in this context. The transactional nature of most business processes aligns well with the relational model, which organizes data into tables with relationships between them.

To conduct process mining, we need to extract this event data from the database. This typically involves selecting the relevant tables, fields, and records from the database, and structuring this data into an event log format that can be analyzed by process mining tools. The ability to properly extract and structure this data is a critical skill in process mining. However, extracting event data from a relational database can sometimes be complex, as the event data may be scattered across different tables or encoded in ways that are not immediately suitable for process mining. We often need to join multiple tables, filter records, and transform data to obtain a suitable event log. Once we have the event log, we can apply various process mining techniques to discover the underlying process model, check conformance with a given model, or enhance an existing model. But without the rich, detailed event data stored in databases, none of this would be possible.

The process of extracting an event log from a database can be distilled into four distinct steps: *process identification*, *table selection*, *extraction*, and *preprocessing*. Initially, the task of process identification involves pinpointing a multitude of business processes underpinned by various information systems, and then selecting a specific business process for analysis. This is followed by table selection, which focuses on picking an appropriate set of database tables for the process under scrutiny. The third stage is the extraction phase, where we retrieve event logs by formulating specific data queries and assign a case identifier to correlate related events, with the understanding that process mining typically operates under a *single case notion*, such as a patient in a healthcare process. The final step is preprocessing, which is dedicated to readying the event log for various process mining techniques.

The section ahead presents a method for extracting OCELs, offering support to process analysts at every stage of event log extraction. To aid in this, we first put forward an abstracted view of the database along with a representation of principal database-level relationships, termed as *Graph of Relationships (GoRs)*. This abstraction layer is designed to facilitate the stages of process identification and table selection. Subsequently, we present *blueprints* which lay out how the data from the selected tables correlates with the events of OCELs, thus aiding the extraction phase.

Despite the evident promise and utility of our automated approach in extracting OCELs, we recognize that it may not be suitable for all scenarios. Particularly in contexts where the database structure is highly intricate or the event log requirements are unusually specific, manual extraction might still be necessary. However, it is crucial to acknowledge the potential value of automated methods in reducing the often substantial burden of the Extraction, Transformation, and Load (ETL) process, which forms a significant chunk of time spent on every process mining project. Rather than replacing manual extraction entirely, we envision our approach serving as a helpful tool that can streamline the extraction process, handle the bulk of standard cases, and free up process analysts to focus on more complex and nuanced aspects of the process mining task. The ultimate goal is to strike a balance between automated efficiency and the bespoke precision of manual extraction, to optimize the quality and effectiveness of process mining outcomes.

5.1.2 Relational Databases

In many organizations, relational databases form the backbone of their information systems, storing everything from transactional data to user information. Consequently, they often serve as the primary source of data for process mining, where detailed event data stored in these databases is extracted, transformed, and loaded into process mining tools for analysis. Understanding the structure and operation of these databases is therefore crucial in the field of process mining.

A *relational database* organizes data into tables, where each table has a unique name and represents a specific entity or concept within a domain. Each *row* in a table corresponds to a single *record* or instance

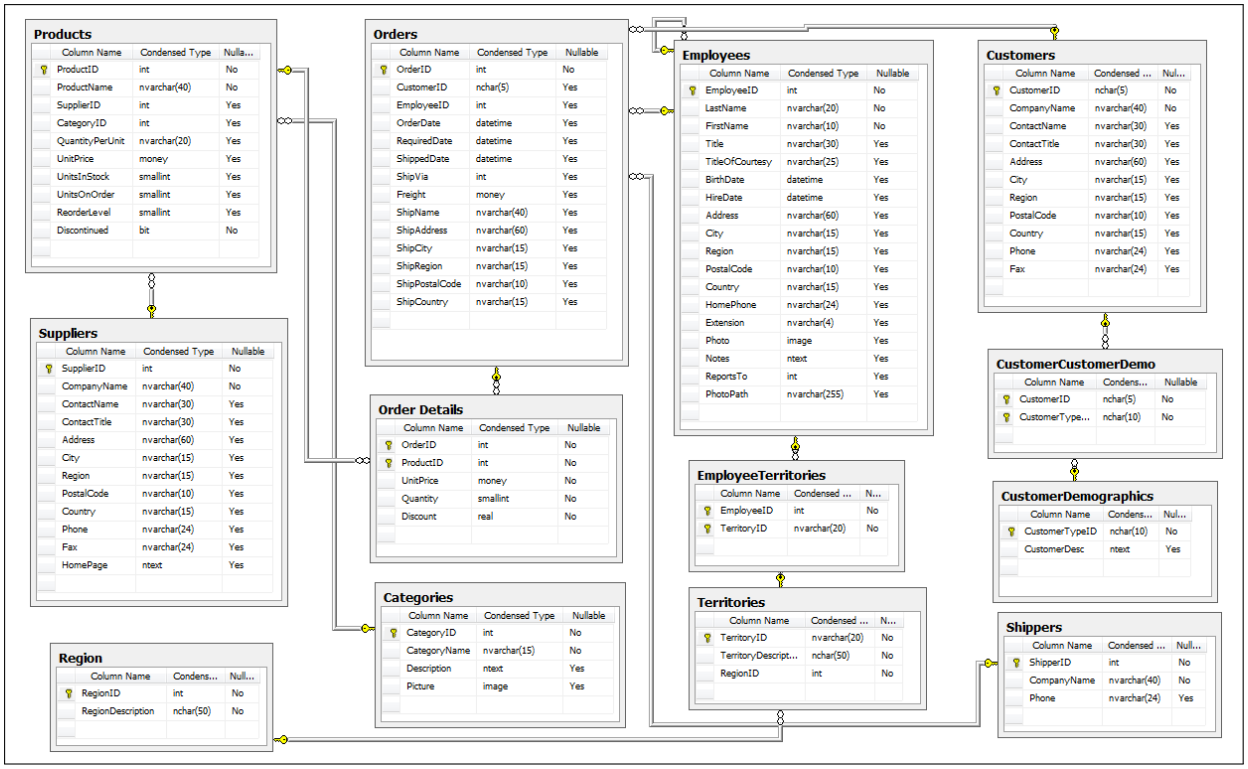


Figure 5.2: An example relational database (Microsoft Access 2000's Northwind).

of that entity, and each *column* represents an attribute or characteristic of the entity. For example, in a table representing employees, each row would hold data for an individual employee, with columns for attributes such as "employee ID", "name", "position", and so on.

The "relational" aspect comes into play when tables are linked based on shared attributes, often using *keys*. These shared attributes enable the creation of relationships between tables, ensuring data integrity and allowing for complex queries across multiple tables. For instance, an "employee" table might be related to a "department" table through a shared "department ID" field.

Users interact with relational databases using SQL (Structured Query Language), which allows them to perform complex queries, update records, and retrieve specific information efficiently. For example, one can query the database to obtain a list of all employees in a particular department, sorted by their hire date.

To generalize and abstract the concept of a database, we introduce an *abstraction of the database concept*. This abstraction captures both the relational structure—comprising tables with their attributes and relationships—and the data records with their interconnections. In this abstraction:

- *Tables* represent entities or object types.
- *Rows* or *records* within these tables correspond to *objects*—individual instances of those entities.
- *Attributes* are properties associated with each object, defined by the columns in the table.
- *Relationships* between objects are captured through shared keys, both primary and foreign.

This framework is technology-agnostic, meaning it does not depend on specific database systems like MySQL, Oracle, or MSSQL, nor on particular implementation details. For example, in some information systems like SAP ERP, schema integrity is managed at the application layer rather than within the database itself. Additionally, methods for preserving updates—such as using REDO logs, in-table versioning, or separate change tables—vary across systems. This abstraction focuses on the essential components and relationships within a database, regardless of these implementation differences.

With this understanding, we can formally define the abstracted concept of a database in Definition 27.

Definition 27 (Database Abstraction)**.** *A database is a tuple* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$ *such that:*

- $T \subseteq U_{table}$ *is a set of tables (identifiers).*
- $O \subseteq U_{obj}$ *is a set of objects (identifiers).*
- $OT \subseteq U_{otype}$ *is a set of object types.*
- $A \subseteq U_{attr}$ *is a set of attribute names.*
- $D \subseteq U_{dom}$ *is a set of attribute domains.*
- $TR \subseteq U_{transact}$ *is a set of transactions.*
- $te : T \to \mathcal{P}(O \times T \times U_{time} \times (A \nrightarrow U_{val}))$ *is the* table entries *function associating to a table name a set of table entries, each identified by an object identifier, a reference table, a (creation) timestamp, and an attribute map. We denote* $TE = O \times T \times U_{time} \times (A \nrightarrow U_{val})$ *as the set of table entries.*
- $tr : T \times T \nrightarrow \mathcal{P}(O \times O)$ *is the* tables relationships *function associating to each couple of tables a set of tuples of objects that are related. Given* $t_1, t_2 \in T$, $tr(t_1, t_2) \subseteq AE(t_1) \times AE(t_2)$, *where* $AE(t) = \{\pi_1(y) \mid y \in te(t)\}$.
- $otyp : O \to OT$ *associates to every object (identifier) the corresponding object type.*
- $attdomain : A \to D$ *associates to every attribute name the corresponding domain.*
- $transact : TE \nrightarrow TR$ *associates to some table entries a transaction.*
- $primary : T \to \mathcal{P}(A)$ *associates to every table a set of attributes that are primary keys for the given table.*
- $foreign : T \to \mathcal{P}(A)$ *associates to every table a set of attributes that are foreign keys for the given table.*

We introduce as a simple example for the previous definition an abstraction of a database with three tables, the third one hosting the changes operated to the objects of the other two tables:

- $T = \{t_1, t_2, \text{changetab}\}$
- $O = \{o_{11}, o_{12}, o_{21}, o_{22}\}$
- $OT = \{\text{ot}_1, \text{ot}_2\}$
- $A = D = \emptyset$
- $TR = \{\text{TRANSACT1}\}$
- $te(t_1) = \{\text{ent}_{11} = (o_{11}, t_1, 2007-04-05, \text{vmap}_{11}), \text{ent}_{12} = (o_{12}, t_1, 2007-04-07, \text{vmap}_{12})\}$
- $te(t_2) = \{\text{ent}_{21} = (o_{21}, t_2, 2007-04-10, \text{vmap}_{21}), \text{ent}_{22} = (o_{22}, t_2, 2007-04-11, \text{vmap}_{22})\}$
- $te(\text{changetab}) = \{\text{ent}_{31} = (o_{11}, t_1, 2007-04-15, \text{vmap}^2_{11}), \text{ent}_{32} = (o_{21}, t_2, 2007-04-20, \text{vmap}^2_{21})\}$
- $tr(t_1, t_2) = \{(o_{11}, o_{21}), (o_{12}, o_{22})\}$
- $\text{otyp}(o_{11}) = \text{otyp}(o_{12}) = \text{ot}_1$
- $\text{otyp}(o_{21}) = \text{otyp}(o_{22}) = \text{ot}_2$
- $\text{transact}((o_{11}, t_1, 2007-04-05, \text{vmap}_{11})) = \text{TRANSACT1}$

In the given abstraction illustration, we have table t_1 consisting of two entries, whose object identifiers are o_{11} and o_{12} respectively, and their object type is given by ot_1. Similarly, table t_2 holds two entries, with object identifiers o_{21} and o_{22}, and their object type is ot_2. Additionally, there exists another table, denoted as changetab, which registers changes made to entries from the previous two tables. The first logged change pertains to the entry having object identifier o_{11} from table t_1, and the second change relates to the entry marked with object identifier o_{21} from table t_2. In the context of the abstraction being proposed, object o_{11} shares a relationship with o_{21}, and in a similar manner, o_{12} has a relationship with o_{22}. We don't discuss the reasons for these relationships at this point. Furthermore, the creation of the first entry in table t_1 is associated with the transaction labeled *TRANSACT1.*

More Comprehensive DB Example

We introduce, as an additional example for Definition 27, an abstraction of a database with four tables representing a company's operations: *Employees*, *Departments*, *Projects*, and *Assignments*. This database includes various attributes, relationships, and transactions to illustrate the complexity of the abstraction.

Defining the Components:

- $T = \{Employees, Departments, Projects, Assignments\}$ is the set of **tables** (identifiers).
- $O = O_{Employees} \cup O_{Departments} \cup O_{Projects} \cup O_{Assignments}$ is the set of **objects** (identifiers), where each O_{Table} corresponds to the set of objects in that table.
- $OT = \{EmployeeType, DepartmentType, ProjectType, AssignmentType\}$ is the set of **object types**.
- A is the set of **attribute names**:

$$A = \{EmpID, EmpName, DeptID, DeptName, ProjID, ProjName, AssignedDate, Role\}.$$

- D is the set of **attribute domains**:
  - U_{Nat}: Natural numbers (for IDs).
  - U_{String}: Strings (for names, roles).
  - U_{Date}: Dates (for assignment dates).
- $TR = \{TR1, TR2, TR3\}$ is the set of **transactions**.

Attribute Domains: The function $\text{attdomain} : A \to D$ assigns domains to attributes:

$$\begin{aligned} \text{attdomain}(EmpID) &= U_{Nat}, & \text{attdomain}(EmpName) &= U_{String}, \\ \text{attdomain}(DeptID) &= U_{Nat}, & \text{attdomain}(DeptName) &= U_{String}, \\ \text{attdomain}(ProjID) &= U_{Nat}, & \text{attdomain}(ProjName) &= U_{String}, \\ \text{attdomain}(AssignedDate) &= U_{Date}, & \text{attdomain}(Role) &= U_{String}. \end{aligned}$$

Primary and Foreign Keys: The functions primary and foreign define primary and foreign keys:

- $\text{primary}(Employees) = \{EmpID\}$.
- $\text{primary}(Departments) = \{DeptID\}$.
- $\text{primary}(Projects) = \{ProjID\}$.
- $\text{primary}(Assignments) = \{EmpID, ProjID\}$.
- $\text{foreign}(Employees) = \{DeptID\}$ (references *Departments*).
- $\text{foreign}(Assignments) = \{EmpID, ProjID\}$ (references *Employees* and *Projects*).

Table Entries: The function $\text{te} : T \to \mathcal{P}(TE)$ maps each table to its set of entries. For simplicity, we define a few entries for each table:

- $\text{te}(Employees) = \{$

$$(e_1, Employees, \text{2023-01-10}, \{EmpID \mapsto 1, EmpName \mapsto \text{"}Alice\text{"}, DeptID \mapsto 10\}),$$
$$(e_2, Employees, \text{2023-02-15}, \{EmpID \mapsto 2, EmpName \mapsto \text{"}Bob\text{"}, DeptID \mapsto 20\})\}.$$

  Here, e_1 and e_2 are object identifiers in $O_{Employees}$.

- $\text{te}(Departments) = \{$

$$(d_{10}, Departments, \text{2023-01-05}, \{DeptID \mapsto 10, DeptName \mapsto \text{"}HR\text{"}\}),$$
$$(d_{20}, Departments, \text{2023-01-07}, \{DeptID \mapsto 20, DeptName \mapsto \text{"}Engineering\text{"}\})\}.$$

  Here, d_{10} and d_{20} are object identifiers in $O_{Departments}$.

- $\text{te}(\mathit{Projects}) = \{$

$$(p_{100}, \mathit{Projects}, \text{2023-03-01}, \{\mathit{ProjID} \mapsto 100, \mathit{ProjName} \mapsto \text{"}\mathit{ProjectX}\text{"}\}),$$
$$(p_{200}, \mathit{Projects}, \text{2023-04-01}, \{\mathit{ProjID} \mapsto 200, \mathit{ProjName} \mapsto \text{"}\mathit{ProjectY}\text{"}\})\}.$$

  Here, p_{100} and p_{200} are object identifiers in O_{Projects}.

- $\text{te}(\mathit{Assignments}) = \{$

  $(a_1, \mathit{Assignments}, \text{2023-03-15}, \{\mathit{EmpID} \mapsto 1, \mathit{ProjID} \mapsto 100, \mathit{AssignedDate} \mapsto \text{2023-03-15}, \mathit{Role} \mapsto \text{"}\mathit{Developer}\text{"}\}),$
  $(a_2, \mathit{Assignments}, \text{2023-04-10}, \{\mathit{EmpID} \mapsto 2, \mathit{ProjID} \mapsto 200, \mathit{AssignedDate} \mapsto \text{2023-04-10}, \mathit{Role} \mapsto \text{"}\mathit{Tester}\text{"}\})\}.$

  Here, a_1 and a_2 are object identifiers in $O_{\mathit{Assignments}}$.

Object Types: The function $\text{otyp} : O \to OT$ assigns object types:

$$\text{otyp}(e_1) = \mathit{EmployeeType}, \quad \text{otyp}(e_2) = \mathit{EmployeeType},$$
$$\text{otyp}(d_{10}) = \mathit{DepartmentType}, \quad \text{otyp}(d_{20}) = \mathit{DepartmentType},$$
$$\text{otyp}(p_{100}) = \mathit{ProjectType}, \quad \text{otyp}(p_{200}) = \mathit{ProjectType},$$
$$\text{otyp}(a_1) = \mathit{AssignmentType}, \quad \text{otyp}(a_2) = \mathit{AssignmentType}.$$

Table Relationships: The function $\text{tr} : T \times T \not\to \mathcal{P}(O \times O)$ captures relationships between tables:

- Relationship between *Employees* and *Departments* via *DeptID*:

$$\text{tr}(\mathit{Employees}, \mathit{Departments}) = \{(e_1, d_{10}),\ (e_2, d_{20})\}.$$

  This indicates that e_1 (Alice) belongs to department d_{10} (HR), and e_2 (Bob) belongs to department d_{20} (Engineering).

- Relationship between *Assignments* and *Employees* via *EmpID*:

$$\text{tr}(\mathit{Assignments}, \mathit{Employees}) = \{(a_1, e_1),\ (a_2, e_2)\}.$$

- Relationship between *Assignments* and *Projects* via *ProjID*:

$$\text{tr}(\mathit{Assignments}, \mathit{Projects}) = \{(a_1, p_{100}),\ (a_2, p_{200})\}.$$

Transactions: The function $\text{transact} : TE \not\to TR$ associates table entries with transactions:

- The creation of e_1 and d_{10} is associated with transaction *TR1*:

$$\text{transact}((e_1,\ \mathit{Employees},\ \text{2023-01-10},\ \ldots)) = \mathit{TR1},$$
$$\text{transact}((d_{10},\ \mathit{Departments},\ \text{2023-01-05},\ \ldots)) = \mathit{TR1}.$$

- The creation of a_1 is associated with transaction *TR2*:

$$\text{transact}((a_1,\ \mathit{Assignments},\ \text{2023-03-15},\ \ldots)) = \mathit{TR2}.$$

- The creation of e_2, d_{20}, p_{200}, and a_2 are associated with transaction *TR3*.

Explanation of the Example: In this abstraction, we have four tables representing different aspects of a company's operations:

1. *Employees* table:
   - Contains entries for two employees, Alice and Bob.
   - Attributes include *EmpID*, *EmpName*, and *DeptID*.
   - Each employee is associated with a department via *DeptID*.

2. *Departments* table:
    - Contains entries for two departments, HR and Engineering.
    - Attributes include *DeptID* and *DeptName*.
    - Serves as a reference table for employees' department assignments.
3. *Projects* table:
    - Contains entries for two projects, Project X and Project Y.
    - Attributes include *ProjID* and *ProjName*.
    - Projects are assigned to employees through the *Assignments* table.
4. *Assignments* table:
    - Represents the assignment of employees to projects.
    - Composite primary key consists of *EmpID* and *ProjID*.
    - Attributes include *AssignedDate* and *Role*.
    - Establishes relationships with both *Employees* and *Projects* via foreign keys.

Relationships and Integrity Constraints: The relationships between tables are enforced through primary and foreign keys, ensuring referential integrity:

- An employee must be associated with a valid department.
- An assignment must reference valid employee and project entries.
- The use of composite primary keys in the *Assignments* table prevents duplicate assignments of the same employee to the same project.

Transactions: Transactions in this example capture the operations that modify the database:

- *TR1*: Initial setup transactions, adding Alice and the HR department.
- *TR2*: Assigning Alice to Project X.
- *TR3*: Adding Bob, the Engineering department, Project Y, and assigning Bob to Project Y.

This structuring allows for tracking changes over time and associating data modifications with specific transactions, which is useful for auditing and rollback operations.

5.1.3 Table Selection (Step 1)

In this section, we propose the concept of Graphs of Relationships (GoRs). GoRs underscore the pivotal tables and object types within an information system's database, thereby enabling the selection of a set of interrelated tables that correspond to a specific process. The visual cues provided by GoRs greatly facilitate the process of identifying processes and selecting appropriate tables. As per Definition 28, a GoR is derived from an abstract representation of the database.

Definition 28 (Graph of Relationships (GoR))**.** *Given a database* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$*, the graph of relationships is defined as the tuple* $GoR(DB) = (T, A, D, OT, TR, R_{T,T}, R_{T,A}, R_{A,D}, R_{T,OT}, R_{T,TR}, R_{T,PR}, R_{T,FR})$ *where:*

- T *is a set of tables (identifiers).*
- A *is a set of attribute names.*
- D *is a set of attribute domains.*
- OT *is a set of object types.*
- TR *is a set of transactions.*
- $R_{T,T} = \{(t_1, t_2) \in dom(tr) \mid |tr(t_1, t_2)| \geq 1\}$ *is the set of connections between tables.*
- $R_{T,A} = \{(t, a) \in T \times A \mid \exists_{y \in te(t)} a \in dom(\pi_4(y))\}$ *is the set of connections between tables and attributes.*
- $R_{A,D} = \{(a, d) \in A \times D \mid attdomain(a) = d\}$ *is the set of connections between attribute names and domains.*
- $R_{T,OT} = \{(t, ot) \in T \times OT \mid \exists_{y \in te(t)} otyp(\pi_1(y)) = ot\}$ *is the set of connections between tables and object types.*
- $R_{T,TR} = \{(t, tra) \in T \times TR \mid \exists_{y \in te(t)} y \in dom(transact) \wedge transact(y) = tra\}$ *is the set of connections between tables and transactions.*
- $R_{T,PR} = \{(t, pr) \in T \times A \mid a \in primary(t)\}$ *is the set of connections between tables and primary keys.*
- $R_{T,FR} = \{(t, fr) \in T \times A \mid a \in foreign(t)\}$ *is the set of connections between tables and foreign keys.*

The GoR serves as an instrumental tool in identifying the tables that correspond to different processes within an information system. For instance, in the SAP system, tables related to the Procure-to-Pay (P2P) process are included in the GoR. To identify these interconnected tables, we can either initiate from a particular object type, such as "EINKBELEG", which is linked to a set of process-specific tables, or begin from a subset of tables, which can be expanded by tracing the paths in the GoR.

5.1.4 Formulating Blueprints (Step 2)

Subsequently, we introduce the concept of *blueprints*, which link a table to a collection of *primitive events*. Each primitive event comprises an activity, a timestamp, and a set of associated objects. This trinity succinctly describes *what* transpired, *when* it transpired, and *which* business objects were implicated in the occurrence. As an example, a blueprint can transmute a table entry (such as one stating that on 1989-02-20, a table named "BIRTHS" recorded an event with attributes BABYNAME = Alex and NURSE = Lucia) into an event (a baby was born on that specific date, involving two related entities, baby Alex and nurse Lucia). Nevertheless, a database entry may be linked with no event or more than one event. Given that a database entry always corresponds to a single object, this implies that a blueprint could potentially associate multiple events with each individual object.

Definition 29 (Blueprint)**.** *Given a database* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$*, a blueprint is a function* $bluep : T \to \mathcal{P}(U_{etype} \times U_{time} \times \mathcal{P}(O))$*, that associates to the tables a set of primitive events having an activity, a timestamp and a set of related objects.*

The data gleaned from applying the blueprint serves as the foundation for constructing an OCEL. Such a log can be examined using OCPM instruments. We unveil a *basic blueprint* that systematically correlates a primitive event to the table entries, leveraging the entry's timestamp of insertion (as prescribed by the given abstraction), the object immediately associated with the entry, and other objects connected to the said object in the database. Moreover, an activity is assigned to the table entry based on the table type. We classify some table types in Definition 30.

Definition 30 (Types of Tables)**.** *Given a database* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$*, we identify different types of tables:*

- *Change Tables* ($CT \subseteq T$)*: tables recording changes happening in other tables. If* $t \in CT$*, then* $t \notin \{\pi_2(y) \mid y \in te(t)\}$*.*
- *Transaction Tables* ($TT \subseteq T$)*: non-change tables recording different transactions executed on their entries. Moreover, every entry is associated with a transaction. If* $t \in TT$*, then*

$$|\{transact(y) \mid y \in te(t) \cap dom(transact)\}| \geq 1 \wedge \forall_{y \in te(t)} y \in dom(transact)$$

- *Object tables* ($OT \subseteq T$)*: tables not falling in any of the previous categories.*

Concerning SAP ERP, we could mention a table for every type: *CDHDR* is a generic change table; *RBKP* contains the transactions executed to verify the invoices (hence it is a transaction table), *VBAK* contains information on different sales order documents without reporting the transaction (hence it is an object table). In Definition 31, the *basic blueprint* is introduced. We see that entries of a change table are associated with the label of the table on which the change is applied; entries of a transaction table are associated with the executed transaction; entries of a creation table are associated with the label of the table. Variations of the basic blueprint are possible, for example considering the fields that were inserted/deleted/updated during the change.

Definition 31 (Basic Blueprint)**.** *Given a database* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$*, and a labeling function* $label : T \to U_\Sigma$ *(which can be the identity function), the basic blueprint is defined as* $B_{bas} : T \to \mathcal{P}(U_{etype} \times U_{time} \times \mathcal{P}(O))$ *such that for* $t \in T$*,* $B_{bas}(t) = \{(a(y), ts(y), robj(y)) \mid y \in te(t)\}$ *where:*

- $ts(y) = \pi_3(y)$
- $robj(y) = \{o \in O \mid o = \pi_1(y) \ \vee \ \exists_{(t_1,t_2) \in T \times T}(\pi_1(y), o) \in tr(t_1, t_2) \vee (o, \pi_1(y)) \in tr(t_1, t_2)\}$
- $$a(y) = \begin{cases} \text{“}Changed'' \ \oplus \ label(\pi_2(y)) & \text{if } t \in CT. \\ \text{“}Executed'' \ \oplus \ transact(y) & \text{if } t \in TT. \\ \text{“}Created'' \ \oplus \ label(t) & \text{if } t \in OT. \end{cases}$$

In the example presented previously, the basic blueprint would return the following primitive events:

- For t_1:
  - $a(\text{ent}_{11}) = \text{“}Created t1''$, $ts(\text{ent}_{11}) = 2007-04-05$, $robj(\text{ent}_{11}) = \{o_{11}, o_{21}\}$
  - $a(\text{ent}_{12}) = \text{“}Created t1''$, $ts(\text{ent}_{12}) = 2007-04-07$, $robj(\text{ent}_{12}) = \{o_{12}, o_{22}\}$
- For t_2:
  - $a(\text{ent}_{21}) = \text{“}Created t2''$, $ts(\text{ent}_{21}) = 2007-04-10$, $robj(\text{ent}_{21}) = \{o_{11}, o_{21}\}$
  - $a(\text{ent}_{22}) = \text{“}Created t2''$, $ts(\text{ent}_{22}) = 2007-04-11$, $robj(\text{ent}_{22}) = \{o_{12}, o_{22}\}$
- For changetab:
  - $a(\text{ent}_{31}) = \text{“}Changed t1''$, $ts(\text{ent}_{21}) = 2007-04-15$, $robj(\text{ent}_{21}) = \{o_{11}, o_{21}\}$
  - $a(\text{ent}_{32}) = \text{“}Changed t2''$, $ts(\text{ent}_{22}) = 2007-04-20$, $robj(\text{ent}_{22}) = \{o_{11}, o_{21}\}$

We note that the set of related objects for ent_{31} and ent_{32} obtained using the basic blueprint is suboptimal. Indeed, only o_{11} should be associated with the primitive event obtained from ent_{31}, and only o_{12} should be associated with the primitive event obtained from ent_{32}.

The basic blueprint, as previously introduced, operates on the database abstraction devoid of additional domain knowledge. However, circumstances exist where the basic blueprint falls short of capturing the interconnection between entries logged in disparate tables. A simplistic illustration is a situation where an order accompanied by three items is placed. This generates one entry in the "orders" table and three entries in the "items" table. The basic blueprint would recognize these distinct entries as four separate primitive events. If we have the capability to incorporate domain knowledge into the process, such as explicitly linking the four entries from the "orders" and "items" tables, we could correctly identify the primitive event.

We introduce the notion of domain knowledge in Definition 32. Here, a view is employed to cluster the associated entries in the database. Primitive events can be derived from these clusters, considering the activity as the amalgamation of the names of the tables of the associated entries, the timestamp as an aggregation of the timestamps of the related entries, and the set of associated objects as the union of the objects implicated in the related entries.

Definition 32 (Blueprint with Domain Knowledge)**.** *Given a database* $DB = (T, O, OT, A, D, TR, te, tr, otyp, attdomain, transact, primary, foreign)$*,* TE *as the set of table entries in* DB*, and a labeling function* $label : T \to U_\Sigma$ *(which can be the identity function), let* $VIEW(DB) \subseteq \mathcal{P}(TE)$ *be a collection of related entries specified by the domain knowledge. Let* $t \in T$ *be a table (which is arbitrarily used to represent the view). The blueprint* $B_{domk} : T \to \mathcal{P}(U_{etype} \times U_{time} \times \mathcal{P}(O))$ *is defined such that:*

- $B_{domk}(t') = \emptyset$ *if* $t' \neq t$
- $B_{domk}(t) = \{(a(v), ts(v), robj(v)) \mid v \in VIEW\}$ *where:*
  - $a(v) =$ "$Updated" \bigoplus_{y \in v} label(\pi_2(y))$ *is the concatenation of the names of the tables of the related entries.*
  - $ts(v) = min_{y \in v} \pi_3(y)$ *is the minimum of the timestamps of the related entries.*
  - $robj(v) = \{\pi_1(y) \mid y \in v\}$ *is the union of the object identifiers of the related entries.*

In the example presented previously, we could provide the following domain knowledge of the relationships between the entries:

$$\text{VIEW}(DB) = \{v1 = \{\text{ent}_{11}, \text{ent}_{21}\}, v2 = \{\text{ent}_{12}, \text{ent}_{22}\}, \\ v3 = \{\text{ent}_{31}\}, v4 = \{\text{ent}_{32}\}\}$$

In this situation, using the blueprint with domain knowledge, we obtain the following primitive events:

- $a(v1) =$ "$Updatedt1t2"$, $ts(v1) = 2007 - 04 - 05$, $robj(v1) = \{o_{11}, o_{21}\}$
- $a(v2) =$ "$Updatedt1t2"$, $ts(v2) = 2007 - 04 - 07$, $robj(v2) = \{o_{12}, o_{22}\}$
- $a(v3) =$ "$Updatedt1"$, $ts(v3) = 2007 - 04 - 15$, $robj(v3) = \{o_{11}\}$
- $a(v4) =$ "$Updatedt2"$, $ts(v4) = 2007 - 04 - 20$, $robj(v4) = \{o_{12}\}$

The resulting primitive events can be stored in an OCEL.

5.1.5 OCEL format for OCELs

Motivation

Standards for storing OCELs serve as a crucial backbone in managing and analyzing complex process data. They provide a coherent, uniform, and structured approach to representing, storing, and exchanging event data across multiple systems, platforms, and applications. The adoption of a standard has several significant benefits:

- *Interoperability*: a standard promotes seamless data interchange between diverse systems. It eliminates data silos by ensuring that event logs are represented in a universally understandable format.
- *Scalability*: a well-structured standard allows efficient handling of data, enabling it to scale with increasing complexity and volume. It ensures that the data remains manageable, reducing the overhead of dealing with unstructured or inconsistently structured logs.
- *Data Integrity and Consistency*: the standardization of event logs upholds the consistency and integrity of data across different sources. It provides a uniform structure to data, making it less prone to inconsistencies and errors, thereby improving the overall data quality.
- *Simplifies Analysis*: by adhering to a standard, the interpretation and analysis of event logs are significantly simplified. It enables the use of standard analysis tools and methods, fostering easy comparability and benchmarking of results.
- *Future-Proofing*: standards also future-proof data, ensuring that it remains accessible, reusable, and comprehensible even as technologies evolve.

The development and adoption of a standard for storing OCELs are vital for realizing the full potential of process mining and other data-driven analytics methods. It paves the way for more effective, efficient, and reliable data management and analysis strategies.

OCEL 1.0

Unlike traditional event logs that solely revolve around process instances, OCEL 1.0 goes beyond this linear perspective by introducing an object-centric model. This model embraces the complex nature of real-life processes, where a multitude of objects interact over time. It brings into focus the inherent relationships between objects, and the roles they play in the context of different events. Implemented in both JSON and XML formats, OCEL 1.0 provides the flexibility and efficiency required for diverse applications. Each event in the log is detailed with crucial information including the performed activity, timestamp, and other attributes. Additionally, each object within the log is clearly identified by its type and carries its specific attributes.

Listing 5.1 exemplifies the structure of an XML-OCEL specification. This particular configuration is marked by a primary *log* tag enclosing four crucial elements, as detailed below:

- A *global* element, attributed with *scope='log'*, is embedded to serve three primary functions:
  - The *version* element conveys the version of the OCEL standard employed.
  - The *attribute-names* key is associated with a list of attribute names used across the log's events and objects. The attribute name is listed as a string value to the *attribute-name* key.
  - The *object-type* key links to a list of the object types that pertain to the log's objects. Each object type is assigned to the *object-type* key as a string value.
- The second *global* element, characterized by *scope='event'*, details the requisite elements of events and their respective default values if not explicitly stated as event properties. This is encoded in the XML format using the attribute type as the tag, the attribute name as the key, and the default value as the value.
- Analogous to the preceding, another *global* element with *scope='object'* explicates the obligatory elements of objects and their respective default values in instances when they are not explicitly articulated as object properties.

In addition to the aforementioned, the *log* element encompasses multiple events and objects. Every event, demarcated by the *event* tag within the *events* element, and object, identified by the *object* tag within the *objects* element, possesses various properties. Each property is characterized by a tag that designates its type and a key that signifies its name. Each event is defined by an identifier (*id*, type: *string*), an activity (*activity*, type: *string*), and a timestamp (*timestamp*, type: *date*). Furthermore, it is linked to an object map (*omap*, type: *list*) and an attribute map (*vmap*, type: *list*), the latter detailing the attributes associated with the event. Every object also comprises an identifier (*id*, type: *string*) and is connected to an object type (*type*, type: *string*) as well as an attribute map (*ovmap*, type: *map*) that enlists the associated attributes. As a matter of implementation discretion, the *ocel:* prefix is deliberately omitted from the keys in the XML-OCEL format.

Listing 5.1: XML-OCEL example

```
<?xml version='1.0' encoding='UTF-8'?>
<log>
  <global scope="log">
    <string key="version" value="0.1"/>
    <list key="attribute-names">
      <string key="name" value="color"/>
      <string key="name" value="costs"/>
      <string key="name" value="customer"/>
      <string key="name" value="prepaid-amount"/>
      <string key="name" value="resource"/>
      <string key="name" value="size"/>
      <string key="name" value="total-weight"/>
      <string key="name" value="weight"/>
    </list>
    <list key="object-types">
      <string key="type" value="customer"/>
      <string key="type" value="item"/>
      <string key="type" value="order"/>
      <string key="type" value="package"/>
      <string key="type" value="product"/>
    </list>
  </global>
  <global scope="event">
    <string key="id" value="___INVALID___"/>
    <string key="activity" value="___INVALID___"/>
    <string key="timestamp" value="___INVALID___"/>
    <string key="omap" value="___INVALID___"/>
  </global>
  <global scope="object">
    <string key="id" value="___INVALID___"/>
    <string key="type" value="___INVALID___"/>
  </global>
  <events>
    <event>
      <string key="id" value="e1"/>
      <string key="activity" value="place_order"/>
      <date key="timestamp" value="2020-07-09␣08:20:01.527+01:00"/>
      <list key="omap">
        <string key="object-id" value="i1"/>
        <string key="object-id" value="o1"/>
        <string key="object-id" value="i2"/>
      </list>
      <list key="vmap">
        <string key="resource" value="Alessandro"/>
        <float key="prepaid-amount" value="200.0"/>
      </list>
    </event>
    <event>
      <string key="id" value="e2"/>
      <string key="activity" value="check_availability"/>
      <date key="timestamp" value="2020-07-09␣08:21:01.527+01:00"/>
      <list key="omap">
        <string key="object-id" value="i1"/>
      </list>
      <list key="vmap">
        <string key="resource" value="Anahita"/>
        <float key="weight" value="10.0"/>
      </list>
    </event>
    <event>
      <string key="id" value="e3"/>
      <string key="activity" value="load_package"/>
      <date key="timestamp" value="2020-07-09␣08:22:01.527+01:00"/>
      <list key="omap">
        <string key="object-id" value="r1"/>
        <string key="object-id" value="p1"/>
      </list>
      <list key="vmap">
        <string key="resource" value="Gyunam"/>
        <float key="total-weight" value="100.0"/>
      </list>
    </event>
  </events>
  <objects>
    <object>
      <string key="id" value="o1"/>
      <string key="type" value="order"/>
      <list key="ovmap">
        <string key="customer" value="Apple"/>
        <float key="costs" value="3500.0"/>
      </list>
    </object>
    <object>
      <string key="id" value="i1"/>
      <string key="type" value="item"/>
      <list key="ovmap"/>
```

```
    </object>
    <object>
      <string key="id" value="i2"/>
      <string key="type" value="item"/>
      <list key="ovmap">
        <string key="color" value="green"/>
        <string key="size" value="small"/>
      </list>
    </object>
    <object>
      <string key="id" value="p1"/>
      <string key="type" value="package"/>
      <list key="ovmap"/>
    </object>
    <object>
      <string key="id" value="r1"/>
      <string key="type" value="product"/>
      <list key="ovmap"/>
    </object>
  </objects>
</log>
```

Listing 5.2 presents a JSON-OCEL specification example. Within this structure, four crucial elements are incorporated into the log as follows:

- The property *ocel:global-log*, which provides two key aspects:
  - The *ocel:version* property indicates the version of the OCEL standard in use.
  - The *ocel:attribute-names* property corresponds to a list containing the attribute names associated with the log's events and objects.
  - The *ocel:object-types* property corresponds to a list comprising the object types related to the log's objects.
- The *ocel:global-event* property serves to specify the required elements of an event, as well as the respective values when these are not explicitly provided as event properties. These details are encoded within a dictionary.
- Analogously, the *ocel:global-object* property outlines the mandatory elements of an object and their associated values when these are not directly supplied as object properties. This information is similarly cataloged within a dictionary.

Events are housed within the *ocel:events* key. Each event is represented as a map connecting the event identifier to a dictionary. This dictionary entails the activity (*ocel:activity*), the timestamp (*ocel:timestamp*), the object map (*ocel:omap*) — a list of associated objects, and the attribute map (*ocel:vmap*) — which assigns each attribute name to an attribute value. In a similar vein, objects reside under the *ocel:objects* key. Each object is represented as a map linking the object identifier to a dictionary. This dictionary includes the object type (*ocel:type*) and the attribute map (*ocel:ovmap*), which pairs each attribute name with an attribute value.

Listing 5.2: JSON-OCEL example

```
{
  "ocel:global-log": {
    "ocel:version": "1.0",
    "ocel:attribute-names": [
      "color",
      "costs",
      "customer",
      "prepaid-amount",
      "resource",
      "size",
      "total-weight",
      "weight"
    ],
    "ocel:object-types": [
      "customer",
      "item",
      "order",
      "package",
      "product"
    ]
  },
  "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"
      ],
      "ocel:vmap": {
        "resource": "Alessandro",
        "prepaid-amount": 200.0
      }
    },
    "e2": {
      "ocel:activity": "check_availability",
      "ocel:timestamp": "2020-07-09 08:21:01.527+01:00",
      "ocel:omap": [
        "i1"
      ],
      "ocel:vmap": {
        "resource": "Anahita",
        "weight": 10.0
      }
    },
    "e3": {
      "ocel:activity": "load_package",
      "ocel:timestamp": "2020-07-09 08:22:01.527+01:00",
      "ocel:omap": [
        "r1",
        "p1"
      ],
      "ocel:vmap": {
        "resource": "Gyunam",
        "total-weight": 100.0
      }
    }
  },
  "ocel:objects": {
    "o1": {
      "ocel:type": "order",
      "ocel:ovmap": {
        "customer": "Apple",
        "costs": 3500.0
      }
    },
    "i1": {
      "ocel:type": "item",
      "ocel:ovmap": {
        "color": NaN,
        "size": NaN
      }
    },
    "i2": {
      "ocel:type": "item",
      "ocel:ovmap": {
        "color": "green",
        "size": "small"
      }
```

```
    },
    "p1": {
      "ocel:type": "package",
      "ocel:ovmap": {}
    },
    "r1": {
      "ocel:type": "product",
      "ocel:ovmap": {}
    }
  }
}
```

OCEL 1.0 was effectively integrated with MongoDB, a scalable document-oriented NoSQL database system. This was accomplished by adapting the JSON-OCEL standard, which facilitated a smooth transition to MongoDB's document data model. By utilizing MongoDB's ability to efficiently balance in-memory and on-disk computations, the scalability of the OCEL 1.0 was significantly improved. This integration expanded the ability to manage a higher volume of events and objects, without compromising on processing performance. Consequently, the implementation of OCEL 1.0 within MongoDB marked an important development for OCPM, as it facilitated the handling of larger and more complex data sets, catering to a broader range of environments.

OCEL 2.0

The initial version of the OCEL standard, OCEL 1.0, marked a significant milestone in the development of process mining techniques and tools. This approach allowed for the comprehensive and flexible storage of diverse event and object types within a single log. It innovatively facilitated each event's correlation with multiple objects of varied types, establishing a high level of complexity and detail within event logs. In addition to this, OCEL 1.0 incorporated the capability to associate numerous attribute values with each event and object. This feature notably enhanced the richness and granularity of the information captured in the logs, thus bolstering the depth of analysis possible and the insights that could be gleaned from the data. Specifications of OCEL 1.0 based on JSON and XML are offered.

Despite these substantial advancements, OCEL 1.0 had its limitations, which were perceived as an incomplete solution for OCPM. Specifically, it lacked support for capturing Object-to-Object relationships, an aspect that is critical in representing and understanding complex processes involving interactions between different objects. Furthermore, another shortfall of OCEL 1.0 was its incapacity to handle dynamic object attribute values.

OCEL 2.0 is introduced to address these limitations, in particular offering support to:

- *Object-to-Object Relationships*: the presence of qualified Object-to-Object relationships in an OCEL not only enhances the understanding of individual object lifecycles, but also unlocks a deeper comprehension of how these objects interact, depend, and influence each other within a business process. Each object in the process does not exist in isolation; instead, it is a part of a complex network of relationships, actions, and interactions that together form the fabric of the business process. By capturing these relationships, the OCEL offers an enriched perspective on the systemic interdependencies, dependencies, and associations between the objects. This level of detail can reveal pivotal insights about process performance, bottlenecks, inefficiencies, and anomalies that would otherwise remain hidden in a process-centric or non-relational view. Furthermore, Object-to-Object relationships provide a foundation for advanced analytics techniques, such as network analysis and predictive modeling, opening up new possibilities for optimizing business process performance. By storing and analyzing these relationships, we can move towards a more holistic and integrated view of the process, facilitating a deeper understanding of the process dynamics and empowering more informed decision-making.

- *Dynamic Object Attribute Values*: the conventional approach of statically assigning attribute values to objects is reimagined in favor of a more dynamic perspective that considers the temporal evolution of these attribute values. This is a significant shift, as it means that each attribute value for an object is not represented as a singular, immutable data point, but rather as a time-variant entity, capable of changing over the course of the process execution. The initial value of an attribute is stored at the onset, and any subsequent alterations to this attribute are captured as updates in the event log. This approach reflects a more realistic and comprehensive understanding of process instances, as it acknowledges that the attributes of an object are not static, but rather are subject to changes influenced by the execution of events and the progression of time. The importance of this approach is multifold. First, it provides a more accurate and complete representation of the

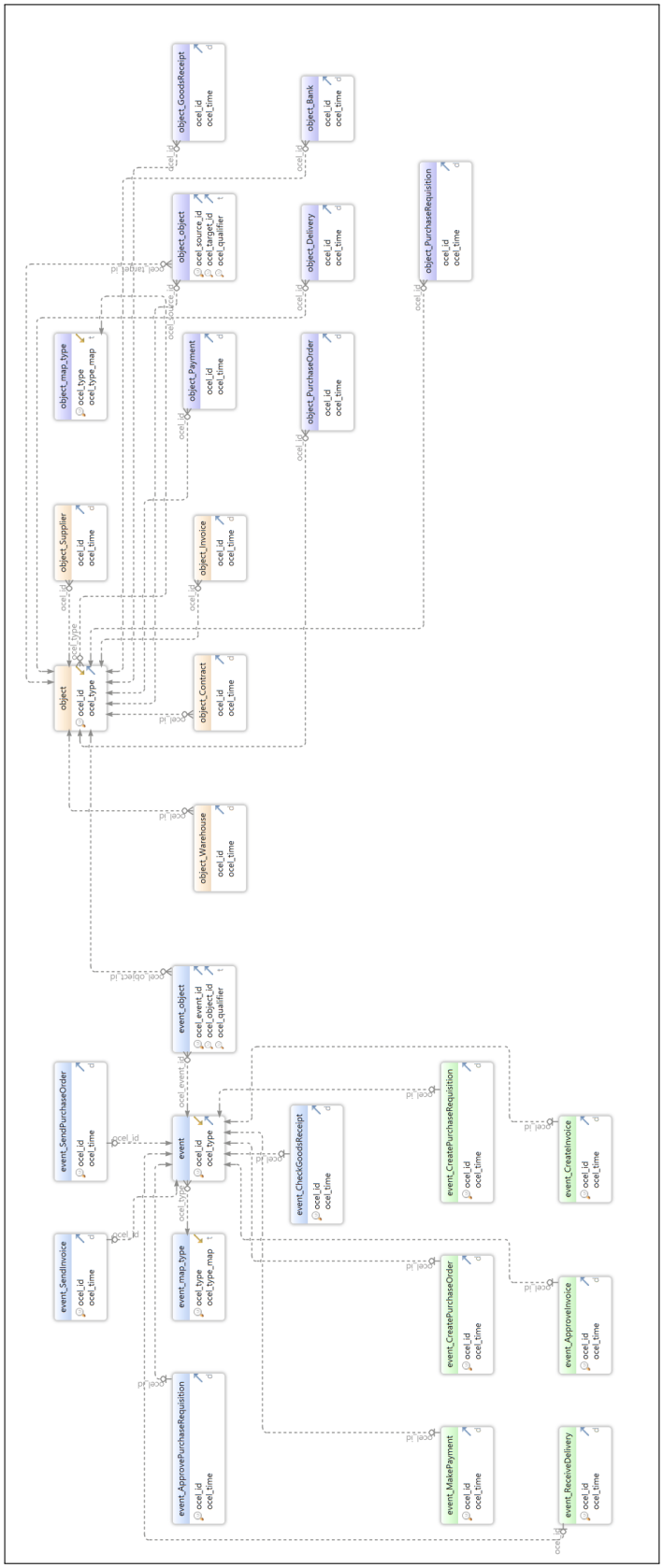


Figure 5.3: OCEL 2.0 Relational Schema for the P2P process

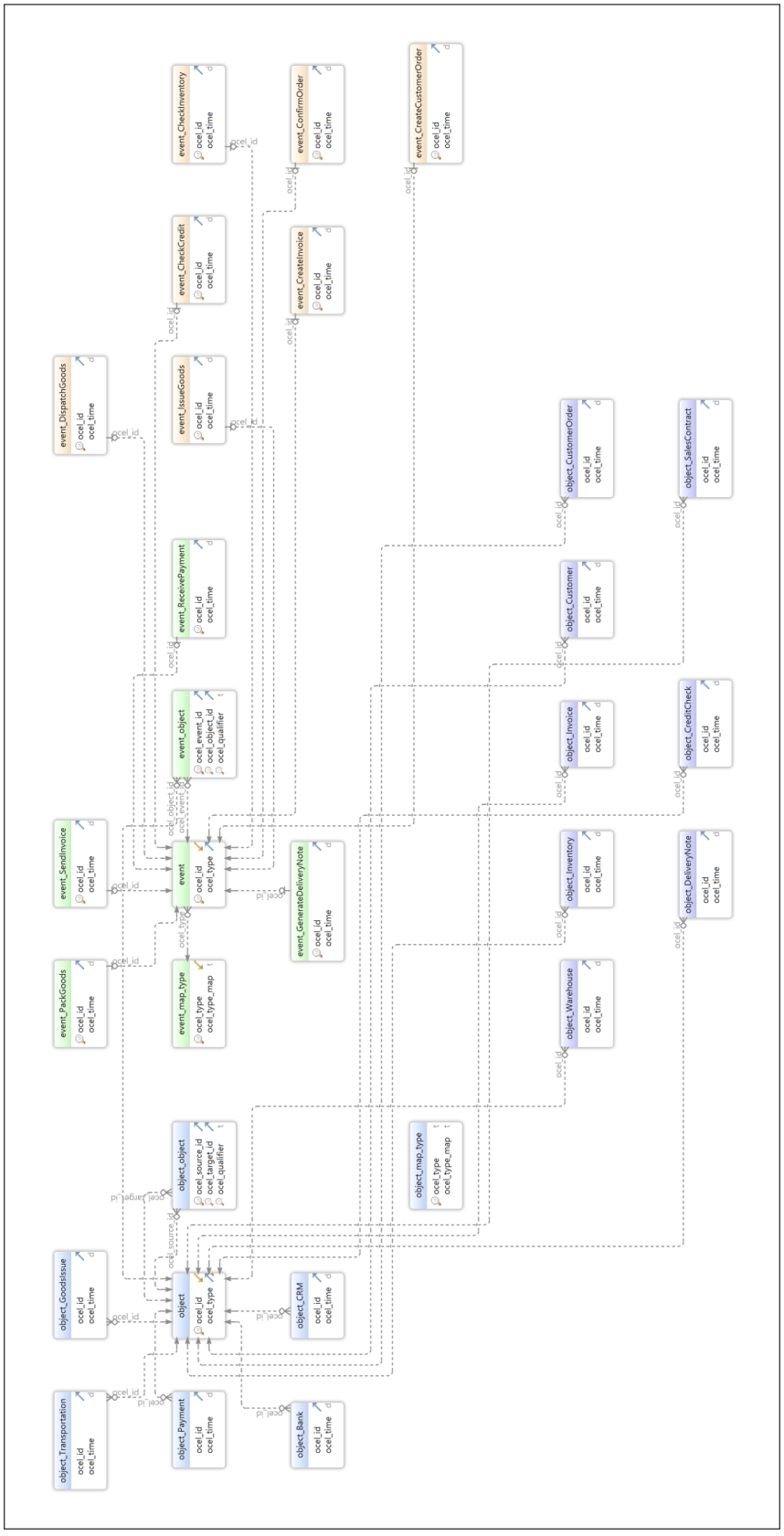


Figure 5.4: OCEL 2.0 Relational Schema for the O2C process

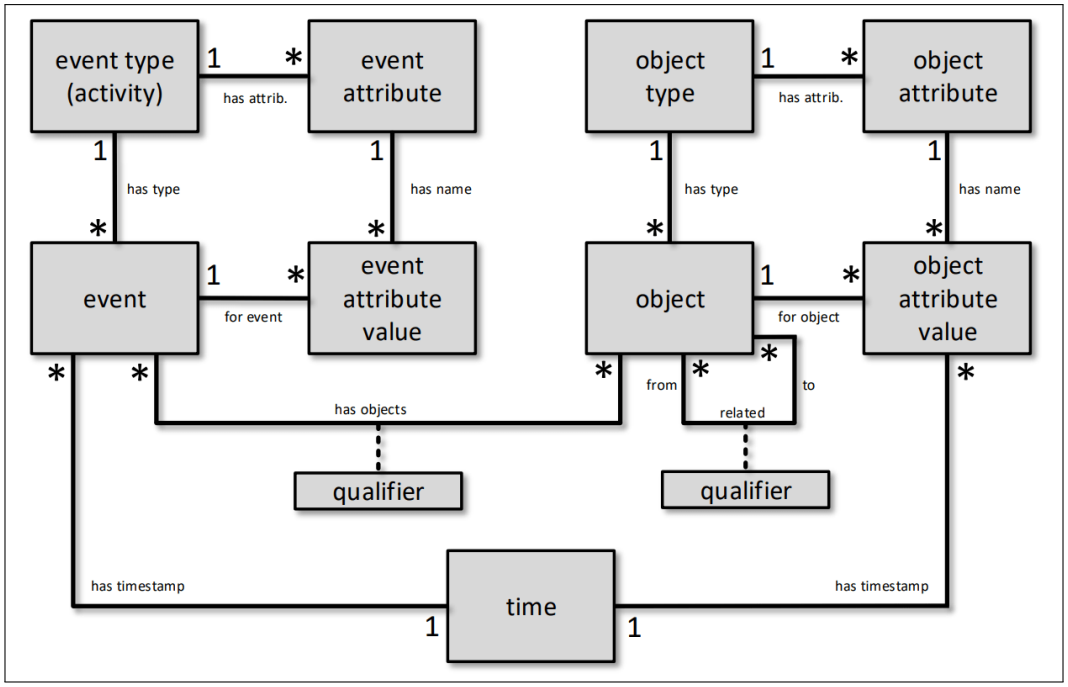


Figure 5.5: OCEL 2.0 meta-model

process, since it captures the changing nature of attribute values. Second, it allows for a more precise analysis of process behavior and performance, since the effect of attribute changes can be tracked and correlated with different outcomes. Third, it enables the discovery of patterns, trends, and anomalies related to attribute changes, which can provide valuable insights for process optimization, risk mitigation, and decision-making. Finally, it empowers more sophisticated process mining techniques that leverage the temporal dimension, such as time series analysis and predictive modeling, enriching the possibilities for process analysis and improvement. Therefore, the adoption of dynamic attribute values in OCEL 2.0 significantly enhances the richness of the event log and the depth of the analysis that can be conducted, leading to more insightful and actionable findings.

Moreover, also the specifications have been enhanced, in particular:

- *Relational Specification based on Dense Tables*: one of the salient features of OCEL 2.0 is its data structure specification, which capitalizes on the concept of dense tables. Essentially, OCEL 2.0 organizes its data into multiple tables, each corresponding to a unique event or object type. Each table stores the attributes pertinent to the associated type, ensuring that the information remains tightly clustered and contextually coherent. This approach presents several benefits which bolster the efficiency, scalability, and utility of OCEL 2.0. Firstly, this design contributes to enhanced data density, which, in turn, ensures that storage space is optimized. Instead of housing the data in a monolithic, dense table that might result in many empty or irrelevant fields for certain types, each table in this framework only contains the attributes that are relevant to the specific event or object type it represents. This approach significantly reduces data redundancy and storage overhead, improving the efficiency of data storage and retrieval operations. Secondly, this structure supports scalability. As new event or object types emerge, new tables can be easily appended to the existing schema without disrupting the current structure. This also facilitates easy updating and maintenance of the database structure as business processes evolve over time. Lastly, by organizing attributes by type in distinct tables, OCEL 2.0 promotes enhanced data accessibility and interpretability. Each table provides a focused snapshot of a particular event or object type, thereby making it easier to explore, understand, and analyze the characteristics and behavior of that type. In essence, OCEL 2.0's relational specification based on dense tables presents a technically advanced and elegant solution to managing data in process mining, marrying practical efficiencies with analytic capabilities to facilitate the extraction of comprehensive and actionable insights from event logs.
- *Improved XML Specification*: The OCEL 2.0 introduces a significant advancement in its XML specification that effectively handles complex data structures within an OCEL, ensuring an efficient,

comprehensive, and meaningful representation of data. The revised specification represents the essential information for events and objects, such as timestamp, event type, and object type, directly within the corresponding "event" and "object" tags, enhancing the readability and interpretability of the data. Furthermore, it accommodates the specification of Object-to-Object relationships and encapsulates changes to attributes at the object level. This adaptive feature is crucial in illustrating the dynamics of attributes over time, providing invaluable insights into the evolving state of objects.

The meta-model shown in Figure 5.5 introduces the main concepts of OCEL 2.0. The OCEL 2.0 schema for the P2P process is reported in Figure 5.3. The one for the O2C process is reported in Figure 5.4.

5.2 Data Preprocessing Techniques

The pre-processing stage in data analysis is a critical step that involves the transformation and restructuring of data to improve the quality and efficiency of subsequent analysis. When working with OCELs, there are several pre-processing techniques that can be leveraged, including but not limited to filtering. In the context of OCELs, filtering mechanisms allow for the reduction of noise and complexity, facilitating a more efficient and focused analysis.

Three key types of filters can be utilized in this context: event filters, object filters, and object type filters. An event filter operates by retaining a selected subset of events from the original log, effectively reducing the volume of data under consideration. Similarly, an object filter works by maintaining a specific subset of objects from the initial log. The third type, an object type filter, functions by retaining a subset of object types from the original log, which naturally induces a filter on the objects of such types.

These filtering strategies contribute significantly to the simplification of complex logs and reduction of computational load, improving both the performance and clarity of subsequent analysis tasks. Thus, their implementation serves as a crucial step in the pre-processing pipeline of OCEL analysis.

5.2.1 Filtering Events

In the realm of data preprocessing on OCELs, filtering at the event level plays a crucial role in streamlining the logs for effective analysis. In line with Definition 33, this approach involves selecting a subset of events from the total set, creating a filtered log that contains only these chosen events.

Definition 33 (Filtering at the Event Level)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *and a subset of events* $E' \subseteq E$*, we define the filtered log* $L_{E|E'} = (A, OT, E', O, EA, OA, evtype_{|E'}, time_{|E'}, objtype, eatype, eaval_{|E' \times EA}, oaval, E2O \cap (E' \times U_{qual} \times O), O2O, \pi_{omap_{|E'}}, \leq_{|E'})$.

The primary feature of this event-level filtering is its focus on reducing the overall volume of events in the log. This reduction not only applies to the total set of events but also restricts the domain of related functions, including the activity, timestamp, and object map functions. Consequently, this limitation narrows down the scope of these functions, effectively focusing their output on the selected events.

Table 5.1: Filter on timestamp (04-2021) applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Definition 34 (Different Filters at the Event Level)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we propose the following filters at the event level, requiring Definition 33.*

- Filtering on Timestamp*: given a subset of times* $T \subseteq U_{time}$*, we define*

$$E' = \{e \in E \mid time(e) \in T\}$$

- Filtering on Activities*: given a subset of activities* $A' \subseteq A$*, we define*

$$E' = \{e \in E \mid evtype(e) \in A'\}$$

- Filtering on Activity Frequency*: given a subset of natural numbers* $X \subseteq \mathbb{N}$*, we consider the filter on activities starting from the following set of activities:*

$$A' = \{a \in A \mid |\{e \in E \mid evtype(e) = a\}| \in X\}$$

- Filtering on the Overall Number of Related Objects*: given a subset of natural numbers* $X \subseteq \mathbb{N}$*, we define*

$$E' = \{e \in E \mid |\pi_{omap}(e)| \in X\}$$

.

Table 5.2: Filter on activities (*Invoice Receipt*, *Perform Payment*) applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

- Filtering on the Number of Related Objects per Object Type*: given an object type $ot \in OT$ and a subset of natural numbers $X \subseteq \mathbb{N}$, we define*

$$E' = \{e \in E \mid |\{o \in \pi_{omap}(e) \mid objtype(o) = ot\}| \in X\}$$

Each kind of filtering proposed in Definition 34 targets a specific aspect of events, which allows analysts to tailor their analysis approach according to the research questions and the nature of the event data.

- *Filtering on Timestamp*: This type of filtering is used when we are interested in events that occurred during a particular time period. It could be useful when studying seasonal trends, response times during certain hours of the day, or investigating events during a known period of disruption or change. An example is provided in Table 5.1.
- *Filtering on Activities*: This filter is applied when the focus of analysis is a subset of activities in the log. This could be beneficial in scenarios where specific activities are known to be problematic or of particular interest. For example, in a hospital setting, focusing on activities related to patient discharge might help to understand and improve the discharge process. An example is provided in Table 5.2.
- *Filtering on Activity Frequency*: This type of filter helps in investigating events that have occurred a specific number of times. It can be useful to identify outliers, for instance, activities that occur unusually often or infrequently, which might indicate bottlenecks or inefficiencies in the process. An example is provided in Table 5.3.
- *Filtering on the Overall Number of Related Objects*: This filtering method selects events based on the total number of objects associated with them. It might be valuable in settings where events with a high number of associations signify complexity or require more resources. For instance, in a manufacturing context, an event related to many parts could indicate a complex assembly process. An example is provided in Table 5.4.
- *Filtering on the Number of Related Objects per Object Type*: This filter helps in zeroing in on events that have a certain number of associated objects of a specific type. It can be especially useful when different object types have different significance or roles in the process. For example, in a customer

Table 5.3: Filter on activity frequency (≥ 7) applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

service context, events with numerous associated customer objects might be considered more critical than those with fewer customer associations. An example is provided in Table 5.5.

Random sampling of events from an OCEL can provide significant value, particularly when dealing with logs with a high degree of complexity or a substantial number of events. In certain scenarios, the sheer volume of events captured in a log might become overwhelming, leading to high computational requirements and potentially excessive detail for analysis. In such cases, random event sampling serves as an effective technique to create a representative, manageable subset of the original log. An example sampling is provided in Table 5.6.

Table 5.4: Filter on overall number of related objects (≥ 2) applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Table 5.5: Filter on overall number of related objects per type (PO ≥ 2) applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Table 5.6: Random sampling on the events applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

5.2.2 Filtering Objects

Table 5.7: Filter on lifecycle length (≥ 4) starting from the OCEL proposed in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

OCELs often contain a vast amount of data. As such, it is common to employ pre-processing techniques to improve the performance of subsequent process mining tasks and to focus on the most relevant aspects of the data. Among these techniques, filtering objects in an OCEL stands out as an effective method to pare down the data based on certain criteria.

Filtering can enhance the readability of the mined models, reduce the computation time, and allow analysts to focus on particular aspects or perspectives of the process. For instance, by filtering out objects that are not involved in the activities of interest, we can minimize noise and enhance the focus on specific behaviors. A formal definition of filtering at the object level is proposed in Definition 35.

Definition 35 (Filtering at the Object Level)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *and a subset of objects* $O' \subseteq O$*, we define the filtered log* $L_{O|O'} = (A, OT, E, O', EA, OA, evtype, time, objtype_{|O'}, eatype, eaval, oaval_{|O' \times OA \times U_{time}}, E2O \cap (E \times U_{qual} \times O'), O2O \cap (O' \times U_{qual} \times O'), \pi'_{omap}, \leq)$*. where* $\pi'_{omap}(e) = \pi_{omap}(e) \cap O'$*.*

Some filters at the object level are proposed in Definition 36.

Definition 36 (Different Filters at the Object Level)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we propose the following filters at the object level, requiring Definition 35.*

- Filtering on Lifecycle Length *given a subset of natural numbers* $X \subseteq \mathbb{N}$*, we define*

$$O' = \{o \in O \mid |lif(o)| \in X\}$$

- Filtering on Lifecycle Duration *given a subset of real numbers* $R \subseteq \mathbb{R}^{\geq 0}$*, we define*

$$O' = \{o \in O \mid (time(end(o)) - time(start(o))) \in R\}$$

- Filtering on Lifecycle Activities (Positive) *given a subset of activities* $A' \subseteq A$*, we define*

$$O' = \{o \in O \mid A' \cap \{evtype(e) \mid e \in lif(o)\} \neq \emptyset\}$$

- Filtering on Lifecycle Activities (Negative) *given a subset of activities* $A' \subseteq A$*, we define*

$$O' = \{o \in O \mid A' \cap \{evtype(e) \mid e \in lif(o)\} = \emptyset\}$$

Table 5.8: Filter on lifecycle activities (having "Create Purchase Order") starting from the OCEL proposed in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

- Filtering on Start Activities *given a subset of activities* $A' \subseteq A$*, we define*

$$O' = \{o \in O \mid evtype(start(o)) \in A'\}$$

- Filtering on End Activities *given a subset of activities* $A' \subseteq A$*, we define*

$$O' = \{o \in O \mid evtype(end(o)) \in A'\}$$

- Filtering on Variants *given a subset of activity sequences* $A^{*'} \subseteq A^*$*, and a trace function trace* : $O \to A^{*'}$*, we define*

$$O' = \{o \in O \mid trace(o) \in A^*\}$$

One way to perform this filtering is based on the lifecycle length of objects. This allows us to include or exclude objects based on the number of events they participate in. For instance, objects that have a very short or a very long lifecycle might be of special interest due to their exceptional behaviors or might be filtered out as they represent outliers. An example is provided in Table 5.7.

Filtering on lifecycle duration is another significant way of customizing the scope of data analysis in an OCEL. In a business process, objects represent cases that go through a sequence of events, and these sequences take a certain amount of time to complete. This duration, from the start to the end of an object's lifecycle, can vary greatly among different objects.

A filter based on lifecycle duration can help segregate objects whose lifecycles span a specific time range. This filtering method is particularly useful in scenarios where analysts are interested in studying patterns, trends, or anomalies associated with the time it takes to complete processes. For instance, they may choose to exclude objects with exceptionally short or long lifecycle durations, as these could be outliers that might distort the overall understanding of the process. Conversely, they may specifically focus on these objects to understand why their durations are significantly different.

Another type of filter involves focusing on the lifecycle activities of the objects. A positive filter includes objects that have participated in a specific subset of activities, highlighting their influence on these activities. Conversely, a negative filter removes objects that have participated in a certain set of activities, eliminating their influence on the process analysis. Examples are provided in Table 5.8 and Table 5.9.

Table 5.9: Filter on lifecycle activities (not having "Create Purchase Order") starting from the OCEL proposed in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Filtering can also be based on the start and end activities of the objects. This allows us to target objects that have begun or ended with certain activities, providing insight into specific process entry or exit points. Examples are provided in Table 5.10 and Table 5.11.

Random sampling of objects from an OCEL is a valuable tool, particularly when dealing with extensive datasets. Large logs can be challenging to handle due to storage and computational constraints. Moreover, thorough analysis of all objects might be time-consuming and, in many cases, unnecessary for deriving actionable insights. Random sampling allows researchers to select a representative subset of objects without any bias, giving them the opportunity to make valid inferences about the entire population. This technique can be particularly effective when the distribution of the characteristics of interest is uniform across the object population, as it preserves the statistical properties of the original log in the sampled subset. An example is provided in Table 5.12.

Table 5.10: Filter on start activities (being "Create Purchase Order") starting from the OCEL proposed in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Table 5.11: Filter on end activities (being "Perform Payment") starting from the OCEL proposed in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Table 5.12: Random sampling on the events applied on the log presented in Table 3.3.

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

Table 5.13: Filtering on object types applied on the log presented in Table 3.3 (here, we removed the *Invoices* and *Payments* object types).

Ev.ID	Activity	Timestamp	Purch.Req.	Purch.Ord.	Quality Checks	Invoices	Payments
e1	Create Purchase Requisition	2021-03-20 10:30	{'pr1'}				
e2	Close Purchase Requisition	2021-03-20 14:00	{'pr1'}				
e3	Create Purchase Requisition	2021-03-21 09:30	{'pr2'}				
e4	Create Purchase Order	2021-03-22 14:59	{'pr2'}	{'po1'}			
e5	Invoice Receipt	2021-03-25 11:00		{'po1'}		{'r1'}	
e6	Perform Payment	2021-03-30 11:58				{'r1'}	{'p1'}
e7	Create Purchase Requisition	2021-04-01 09:15	{'pr3'}				
e8	PR Formal Approval	2021-04-01 10:15	{'pr3'}				
e9	Create Purchase Order	2021-04-02 17:00	{'pr3'}	{'po2'}			
e10	Change Purchase Requisition	2021-04-03 10:00	{'pr3'}				
e11	Invoice Receipt	2021-04-05 15:00		{'po2'}		{'r2'}	
e12	Perform Payment	2021-04-15 09:27				{'r2'}	{'p2'}
e13	Create Purchase Order	2021-04-17 14:29		{'po3'}			
e14	Invoice Receipt	2021-04-28 10:00		{'po3'}		{'r3'}	
e15	Perform Payment	2021-04-30 15:00				{'r3'}	{'p3'}
e16	Invoice Receipt	2021-05-28 10:01		{'po3'}		{'r4'}	
e17	Perform Payment	2021-05-30 15:17				{'r4'}	{'p4'}
e18	Invoice Receipt	2021-06-28 10:01		{'po3'}		{'r5'}	
e19	Perform Payment	2021-06-30 15:29				{'r5'}	{'p5'}
e20	Create Purchase Requisition	2021-07-01 11:15	{'pr4'}				
e21	Create Purchase Order	2021-07-02 09:38	{'pr4'}	{'po4'}			
e22	Invoice Receipt	2021-07-09 16:00		{'po4'}		{'r6'}	
e23	Quality Check	2021-07-11 10:30		{'po4'}	{'qc1'}		
e24	Perform Payment	2022-05-15 09:00				{'r6'}	{'p6'}
e25	Invoice Receipt	2022-05-20 12:00				{'r7'}	
e26	Create Purchase Order	2022-05-20 15:00		{'po5'}		{'r7'}	
e27	Create Purchase Order	2022-06-01 09:17		{'po6'}			
e28	Create Purchase Order	2022-06-02 11:48		{'po7'}			
e29	Create Invoice	2022-06-05 09:00		{'po6','po7'}		{'r8'}	

5.2.3 Filtering Object Types

In many analyses of OCELs, there's a need to focus only on certain object types, removing the noise and redundancy presented by others. Within this section, we introduce a structured approach to achieve this granularity. By means of Definition 37, we present a methodical procedure to filter the event log based solely on a predefined collection of object types. This targeted filtering aids in refining our log to capture only the events of particular interest, providing a clearer and more concise view of the underlying processes.

Definition 37 (Filtering at the Object Type Level)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *and a subset of objects* $OT' \subseteq OT$*, let*

$$O' = \{o \in O \mid objtype(o) \in OT'\}$$

be the set of objects having type in OT'*. We define the filtered log* $L_{OT|OT'} = (A, OT', E, O', EA, OA, evtype, time, objtype_{|O'}, eatype, eaval, oaval_{|O' \times OA \times U_{time}}, E2O \cap (E \times U_{qual} \times O'), O2O \cap (O' \times U_{qual} \times O'), \pi'_{omap}, \leq)$*. where* $\pi'_{omap}(e) = \pi_{omap}(e) \cap O'$*.*

Following the filtering mechanism outlined in Definition 37, we demonstrate its practical utility using a concrete example. In Table 5.13, we apply the filter to selectively exclude the *Invoices* and *Payments* object types. This illustrates how one can strategically tailor the log to focus on specific aspects of a process, effectively omitting undesired object types and thereby streamlining the analysis.

5.2.4 Consistency of the Filtering Operation

In the realm of OCEL analysis, filtering is an essential step. While introducing numerous filters can be beneficial, it is crucial to be wary of unintended consequences. Specifically, filtering on either events or objects might unintentionally leave certain events or objects without any corresponding counterpart, rendering them as "orphans". These orphan entities could potentially skew the analysis and insights drawn from the logs. To address this, Definition 38 emphasizes the importance of ensuring the consistency of the filtering process. Through our proposed definition, we present a methodology for eliminating such orphan events and objects, ensuring a clean and consistent filtered log for further analysis.

Definition 38 (Removing Orphan Events and Objects)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we define the filtered log* $(L_{E|E'})_{O|O'}$ *where:*

- $E' = \{e \in E \mid \pi_{omap}(e) \neq \emptyset\}$
- $O' = \{o \in O \mid \exists_{e \in E} o \in \pi_{omap}(e)\}$

In light of the outlined challenges with orphaned entities, Definition 38 should invariably be employed following any filtering operation to guarantee a consistent and coherent log.

5.2.5 Influence of Implicit and Explicit Object Relationships on Filtering

While the previous sections focused on various filtering techniques for events, objects, and object types, it is crucial to recognize that the effectiveness and outcomes of these filtering steps can be significantly shaped by how object relationships are defined and leveraged. Object-centric event data inherently captures the complex interplay of multiple objects interacting over time, and this interplay can be represented using both implicit and explicit object relationships.

Implicit vs. Explicit Object Relationships

- *Implicit Relationships (Event-to-Object, E2O)*: In many OCELs, relationships between objects can be inferred from their joint participation in the same events. For instance, if an event references both a *Purchase Order* (PO) object and a *Supplier* object, these two objects are implicitly related at least for the context of that event. Such implicit relationships arise naturally whenever objects co-occur in the same event, without any additional explicit linkage.
- *Explicit Relationships (Object-to-Object, O2O)*: In some OCELs, Object-to-Object relationships are explicitly recorded. These relationships can represent stable or contractual connections, such as a *Contract* object that is explicitly linked to certain *Purchase Orders*. Unlike the ephemeral, event-level co-occurrences that define implicit relationships, explicit O2O relationships stand as enduring links between objects, independent of any single event.

Influence on Filtering Criteria and Outcomes

When applying filters at the event, object, or object-type level, the presence and nature of relationships among objects can heavily influence the selection of data:

1. *Object Retention due to Implicit Relations:* Filters based on lifecycle attributes, event frequencies, or activity types might inadvertently remove objects that appear to be disconnected if only explicit relationships are considered. However, when implicit relationships via E2O connections are taken into account, an object that might otherwise be discarded could be retained because it consistently co-occurs with other relevant objects at the event level. This richer understanding prevents the premature removal of potentially important objects, ensuring a more faithful representation of the underlying process.
2. *Pruning via Explicit Object Links:* When explicit O2O relationships are available, filtering can leverage these stable links to preserve or eliminate clusters of objects that are tightly interconnected. For example, consider a scenario where a particular object type (e.g., *Supplier*) is consistently linked to certain *Contracts* and *Purchase Orders* via explicit O2O relationships. By focusing on these stable links, filters can either target this interconnected network for deeper investigation or exclude it entirely if it does not align with the analysis objectives. This approach helps in isolating or grouping objects more systematically and can lead to more meaningful subsets of the OCEL.
3. *Refining Lifecycle-based Filters:* Lifecycle-based filters (e.g., focusing on objects with long lifecycles) may benefit from a hybrid view of relationships. Implicit relationships can highlight that certain lengthy object lifecycles are not independent but occur in tandem with other object lifecycles due to frequent joint events. Conversely, explicit relationships can confirm whether certain extended lifecycles are driven by long-standing dependencies (like a contract that continually affects multiple purchase orders).

4. *Avoiding Orphaned Entities:* Combining implicit and explicit relationships helps maintain consistency and avoids generating orphan objects or events during filtering. While relying solely on activity- or attribute-based filters may leave some objects isolated, considering both E2O and O2O relationships can ensure that the final filtered subset remains cohesive, with all objects meaningfully connected. This results in a more coherent dataset that accurately reflects the complexity and interdependencies in the original OCEL.

Practical Considerations

When designing and applying filters, consider the following:

- *Data Availability:* If O2O relationships are present in the log, explicitly harnessing them can guide more informed filtering strategies. If they are absent, relying on E2O-based implicit relationships may still yield valuable insights.
- *Process Complexity:* More complex processes often require a careful, relationship-aware filtering approach. A simplistic, relationship-agnostic filtering may strip away critical dependencies or inadvertently isolate objects.
- *Hybrid Approaches:* Combining filters—e.g., first applying a lifecycle-based filter, then refining with O2O-based criteria—can yield a balanced and targeted subset of the OCEL, suitable for advanced process mining tasks.

In summary, object relationships—both implicit and explicit—play a pivotal role in shaping the outcome of filtering steps in object-centric event logs. By acknowledging and leveraging these relationships during the filtering process, analysts can retain essential context, maintain cohesive object networks, and ultimately improve the fidelity and interpretability of subsequent analyses and results.

5.3 Example: On-Time In-Full (OTIF) in Object-Centric Processes

In this section, we illustrate how the concepts introduced in this chapter facilitate the computation of a key performance indicator (KPI) that inherently involves multiple object types. We consider *On-Time In-Full* (OTIF) as a corner example of such a KPI, where different kinds of objects (e.g., orders, shipments, deliveries) must be jointly analyzed. OTIF is widely used in supply chain and logistics to measure whether orders are delivered on schedule (on-time) and in the requested quantity (in-full).

On-Time In-Full evaluates the extent to which deliveries meet both timeliness and quantity requirements, thereby providing a comprehensive measure of delivery performance. From an object-centric perspective, computing OTIF becomes more structured and transparent, as all relevant objects and events can be properly modeled, interconnected, and filtered.

Relating OTIF to the concepts discussed in this chapter, we see that it can be computed and analyzed more effectively using an object-centric event log (OCEL). Since the chapter has focused on extracting, preprocessing, and structuring event data into OCELs, it has laid the groundwork for analyzing KPIs like OTIF. By leveraging the object-centric data model:

- We can represent each order, shipment, and product type as distinct objects.
- We can track their lifecycles through events such as order creation, picking, packing, shipping, and delivery.
- By applying the preprocessing filters introduced in this chapter, we can isolate the subset of events and objects that are relevant for assessing OTIF. For example, we may filter out irrelevant object types, focus on events within a particular timeframe, or zoom in on objects (e.g., orders) with delayed or partial deliveries.

In this sense, the techniques covered in the chapter—data extraction from relational databases, blueprint formulations to shape OCELs, and extensive filtering strategies—are all enablers that ensure we have "clean", well-structured, and process-relevant data. Without these methods, computing OTIF from raw and possibly noisy data would be far more difficult and less reliable.

Example of Using OCEL Data to Compute OTIF

Consider a scenario in which an organization uses the methodology from this chapter to extract and preprocess event data from a relational database supporting their ERP system. After selecting the relevant tables and applying blueprints, the resulting OCEL represents:

- *Order objects* (e.g., *O1*) with attributes like requested quantity, requested delivery date, and customer.
- *Shipment events* linked to these orders that detail when the goods left the warehouse.
- *Delivery confirmation events* that indicate when and how many units were actually delivered.

Once the log is structured as an OCEL, the filtering methods from this chapter can be used to isolate the exact subset of interest:

- Apply *filtering on object types* to focus on *orders* and *shipments* while removing unrelated processes.
- Use *event-level filters* to consider only deliveries that occurred within a defined time window.
- Employ *object-level filters* to retain only those orders that have a full cycle of requested, shipped, and delivered events, ensuring a coherent set of data for OTIF calculation.

With the resulting refined OCEL, each order object can be evaluated against the OTIF criteria:

- *On-Time*: Check whether the delivery confirmation event's timestamp is on or before the requested delivery date.
- *In-Full*: Verify if the delivered quantity matches the requested quantity specified in the order object's attributes.

For example, assume order *O1* requested 100 units of product *P1*, due on January 20. The shipment event recorded on January 19 contains 90 units. Although the order was delivered before the deadline (on-time), it was not delivered completely (in-full). Hence, *O1* is classified as OTIF-fail due to insufficient quantity.

In this example, the techniques introduced in the chapter—such as table-to-OCEL extraction, event/object filters, and object-centric structuring—have directly enabled a precise, context-rich computation of OTIF. Without these steps, identifying that *O1* was not fully delivered on time would be more challenging, as the data might be dispersed across multiple tables, difficult to correlate, or cluttered by irrelevant information.

In conclusion, by combining the methodologies presented in this chapter with the object-centric paradigm, we gain the capability to compute and analyze complex KPIs like OTIF that hinge on multiple types of interconnected objects. This exemplifies how the foundations laid out here support a richer, more reliable, and more insightful assessment of process performance.

5.4 Filtering Chains in Different Applications

To fully understand the utility and flexibility of the filtering techniques described in the previous sections, it is valuable to illustrate how they can be combined and adapted to form chains of preprocessing steps tailored to specific application scenarios. By presenting realistic cases, we demonstrate why it is worth introducing this methodological framework here, at this stage of the discussion: at this point, readers have a grasp of the foundational concepts and can now appreciate how these concepts translate into actionable strategies. In other words, the role of this section is to bridge the gap between the theoretical aspects of data preprocessing and their practical deployment across various domains.

Data preprocessing plays an important role in transforming raw data into meaningful, actionable insights. Through the lens of realistic situations, we introduce in this section diverse scenarios where a chain of filters can be applied, creating a comprehensive methodological framework for data preprocessing.

This framework, although not exhaustive, illustrates the vast applicability of OCEL filters across domains. By understanding the situation at hand and leveraging the appropriate filters in a logical sequence, one can extract targeted insights, driving data-informed decision-making. As industries evolve, so will the use cases, making it imperative for analysts to remain agile, continually adapting their preprocessing strategies to fit the changing landscape. Concluding this section, the reason why we introduce these examples now becomes evident: we have established a toolkit of filters and techniques, and here we show their effectiveness in real-world scenarios, thus reinforcing their practical relevance and offering a clearer pathway from abstract theory to tangible results.

Customer Journey Optimization in E-commerce

Situation: An e-commerce platform aims to understand and optimize the customer journey, from browsing products to checkout, hoping to improve user experience and sales.

Filtering Approach:

- *Filtering on Timestamp*: By focusing on peak sale periods (e.g., Black Friday, Christmas), analysts can study customer behaviors during high-traffic times.
- *Lifecycle Activities (Positive)*: Filtering to only include objects that have participated in "Add to Cart" and "Purchase" activities can highlight successful customer journeys.
- *Lifecycle Activities (Negative)*: Conversely, filtering out objects that include "Abandon Cart" can isolate problematic or interrupted customer journeys.
- *Object Filtering*: Zooming into specific object types like "Premium Customers" or "First-time Users" can provide segmented insights.

Rationale: By studying the steps leading up to a purchase and identifying potential pitfalls, the platform can streamline the process, making it more user-friendly and increasing conversion rates.

Quality Assurance in Manufacturing

Situation: A manufacturing plant wants to trace the lifecycle of faulty products to understand at which stage the most errors occur.

Filtering Approach:

- *Filtering on Activities*: Pinpoint specific activities, like "Quality Check Failed" or "Product Return".
- *Filtering on Activity Frequency*: Isolate activities that occur frequently, indicating recurrent issues.
- *Lifecycle Duration*: Identify products with unusually long production times, as these might correlate with defects.
- *Object Filtering*: Focus on specific batches or product types to trace batch-specific errors.

Rationale: By targeting the problematic stages in the manufacturing lifecycle, the plant can implement targeted interventions to enhance product quality.

Resource Optimization in Hospitals

Situation: A hospital wants to improve patient flow and resource allocation in its emergency department.

Filtering Approach:

- *Filtering on Timestamp*: Focus on high-traffic times, like weekends or flu season, to understand resource constraints.
- *Lifecycle Length*: Identify patient cases with extended wait times or prolonged treatment.
- *Start Activities and End Activities*: Trace the entry (e.g., "Triage") and exit points (e.g., "Discharge") of patients.
- *Object Filtering*: Differentiate between "Critical" and "Non-Critical" patients to ensure priority resource allocation.

Rationale: Enhancing patient flow and optimizing resource allocation can drastically improve patient care and reduce hospital crowding.

Fraud Detection in Banking

Situation: A bank is looking to enhance its fraud detection mechanisms by studying transaction patterns.
Filtering Approach:

- *Filtering on Activities*: Pinpoint suspicious activities like "Multiple Quick Transfers" or "High-Value Overseas Transaction".
- *Filtering on Activity Frequency*: Focus on accounts with a high frequency of suspicious activities in a short time frame.
- *Lifecycle Duration*: Look for transactions that are processed unusually quickly or slowly, which might indicate anomalies.
- *Object Filtering*: Focus on "High-Value Accounts" or "Newly Created Accounts" for targeted analysis.

Rationale: Identifying patterns in fraudulent activities can lead to enhanced detection algorithms and safer banking.

Marketing Campaign Analysis

Situation: A company wants to evaluate the effectiveness of its new marketing campaign.
Filtering Approach:

- *Filtering on Timestamp*: Zoom into the duration of the marketing campaign.
- *Lifecycle Activities (Positive)*: Identify customers who interacted with the campaign, signed up for offers, or made a purchase.
- *Lifecycle Length*: Track the time between campaign interaction and conversion to gauge campaign effectiveness.
- *Object Type Filtering*: Differentiate between "New Customers" and "Returning Customers" to measure campaign reach and retention.

Rationale: Evaluating campaign effectiveness can inform future marketing strategies, ensuring better reach and conversion.

5.5 Assessment

In this assessment we take a closer look at how different OCEL specifications perform during importing and exporting processes. We begin by evaluating the earliest JSON and XML versions, moving on to more recent XML and relational formats.

The experiments were executed on a notebook with an I7-7500U CPU, 16 GB DDR4 RAM, and the PM4Py 2.7.5.1 library (introduced in Chapter 9). Still, we are confident that the findings are generic and are also applicable to other configurations/tools.

5.5.1 Assessment of the OCEL 1.0 (JSON-OCEL) Importing and Exporting Performance

The OCEL 1.0 JSON-OCEL specification serves as a pivotal standard for OCELs, enabling seamless and standardized interactions across platforms and tools. An important aspect of this standard lies in its performance, particularly during the importing and exporting phases. Hence, a thorough evaluation was conducted to discern the associated execution times.

Our examination revealed a linear correlation between execution times and several parameters of the OCEL. Specifically, the execution times exhibited a linear increase with respect to the number of events, the count of objects, and the complexity of Event-to-Object relationships. This behavior is attributable to the inherent structure of the JSON-OCEL file; each event, object, and their interrelationship necessitates the creation of distinct lines within the file.

For a detailed breakdown:

- The execution times concerning the importing of JSON-OCEL can be consulted in Table 5.14. A visual representation elucidating these times can be viewed in Figure 5.6.
- Analogously, for insights into the exporting process of JSON-OCEL, readers can refer to Table 5.15 and Figure 5.7.

Table 5.14: Execution times for the JSON-OCEL importing.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79350	50	50	1.1654643
100000	10000	158069	50	50	2.3192996
200000	10000	315882	50	50	4.3986419
250000	10000	395731	50	50	5.4498097
300000	10000	475162	50	50	6.5014450
Varying the number of objects					
10000	50000	15722	50	50	0.4619411
10000	100000	15789	50	50	0.7287910
10000	200000	15747	50	50	1.2908618
10000	250000	15897	50	50	1.5373688
10000	300000	15730	50	50	1.8183474
Varying the number of object types					
10000	10000	15827	50	50	0.2503010
10000	10000	15832	100	50	0.2739639
10000	10000	15778	200	50	0.2791158
10000	10000	15702	300	50	0.2796873
Varying the number of activities					
10000	10000	15783	50	50	0.2820757
10000	10000	15908	50	100	0.3520604
10000	10000	15732	50	200	0.2799790
10000	10000	15694	50	300	0.2734964
Varying the number of related objects					
10000	10000	15830	50	50	0.2818980
10000	10000	25477	50	50	0.3318018
10000	10000	34864	50	50	0.3816009
10000	10000	45906	50	50	0.4408452
10000	10000	54890	50	50	0.4860194

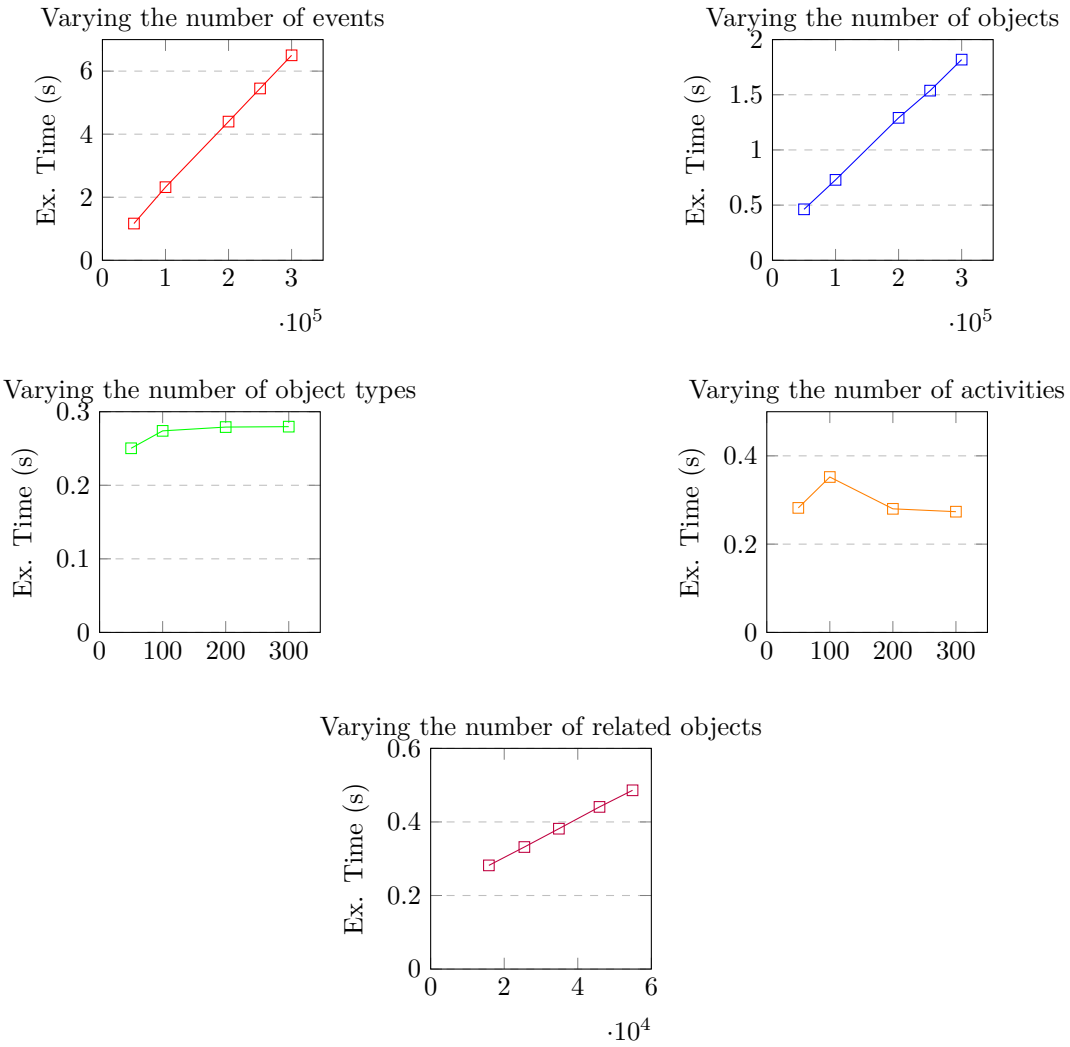


Figure 5.6: Execution times for the JSON-OCEL importing.

Table 5.15: Execution times for the JSON-OCEL exporting.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	78984	50	50	4.805901
100000	10000	158722	50	50	7.4372167
200000	10000	315860	50	50	15.2403649
250000	10000	396100	50	50	19.3114568
300000	10000	475175	50	50	25.0929785
Varying the number of objects					
10000	50000	15924	50	50	1.7018712
10000	100000	15833	50	50	2.6975883
10000	200000	15938	50	50	4.7394419
10000	250000	15785	50	50	5.7338925
10000	300000	15875	50	50	9.7941782
Varying the number of object types					
10000	10000	15972	50	50	0.8989004
10000	10000	15715	100	50	0.9429617
10000	10000	15986	200	50	0.9427095
10000	10000	15827	300	50	0.8827834
Varying the number of activities					
10000	10000	15727	50	50	0.9386734
10000	10000	15708	50	100	0.9236933
10000	10000	15758	50	200	0.8857196
10000	10000	15864	50	300	0.8808995
Varying the number of related objects					
10000	10000	15918	50	50	0.9350085
10000	10000	25615	50	50	0.9384892
10000	10000	35154	50	50	1.0180133
10000	10000	45755	50	50	1.0412745
10000	10000	55289	50	50	1.0591959

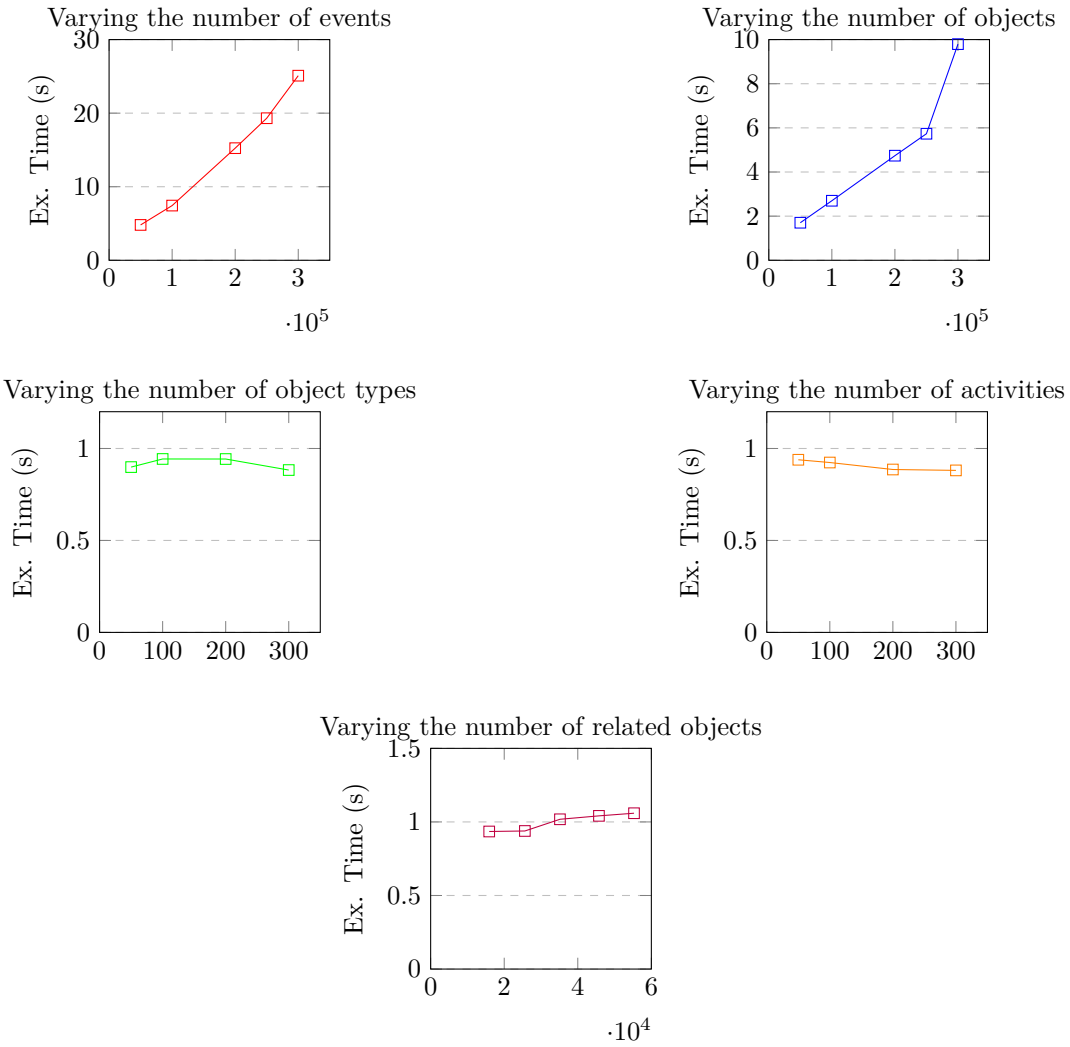


Figure 5.7: Execution times for the JSON-OCEL exporting.

5.5.2 Assessment of the OCEL 1.0 (XML-OCEL) Importing and Exporting Performance

In the domain of OCELs, the OCEL 1.0 XML-OCEL specification is a critical standard, ensuring consistent and effective interactions across diverse platforms and analytical tools. One of the primary aspects that necessitates rigorous scrutiny is the performance efficiency of this standard, especially during the pivotal stages of importing and exporting.

Upon evaluating this standard, we discovered a clear linear relationship between the execution times and several integral components of the OCEL. Specifically, the increase in execution times is directly proportional to the volume of events, the multitude of objects, and the intricacy of Event-to-Object relationships present in the log. This observed behavior is fundamentally rooted in the structural design of the XML-OCEL file. Each unique event and object, coupled with their interconnected relationships, demands the introduction of new lines in the file.

For an in-depth understanding:

- Insights into the performance metrics during the importing phase of XML-OCEL are encapsulated in Table 5.16. A complementary visual portrayal can be accessed in Figure 5.8.
- For those keen on the exporting dynamics of XML-OCEL, the relevant data is presented in Table 5.17 with an illustrative interpretation available in Figure 5.9.

Table 5.16: Execution times for the XML-OCEL importing.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79385	50	50	2.2899837
100000	10000	158289	50	50	4.4153928
200000	10000	316563	50	50	8.6114975
250000	10000	395068	50	50	10.9812265
300000	10000	473584	50	50	12.8496765
Varying the number of objects					
10000	50000	15921	50	50	1.1052274
10000	100000	15595	50	50	3.1729632
10000	200000	15809	50	50	4.2061151
10000	250000	15813	50	50	3.9145160
10000	300000	15918	50	50	4.4373181
Varying the number of object types					
10000	10000	15957	50	50	0.5418531
10000	10000	15716	100	50	0.5438293
10000	10000	15804	200	50	0.5405987
10000	10000	15839	300	50	0.5382997
Varying the number of activities					
10000	10000	15931	50	50	0.5337336
10000	10000	15806	50	100	0.5400656
10000	10000	15896	50	200	0.5326837
10000	10000	15850	50	300	0.5358560
Varying the number of related objects					
10000	10000	15949	50	50	0.5405576
10000	10000	25367	50	50	0.6034297
10000	10000	34712	50	50	0.7415162
10000	10000	45493	50	50	0.7698205
10000	10000	55316	50	50	0.8517845

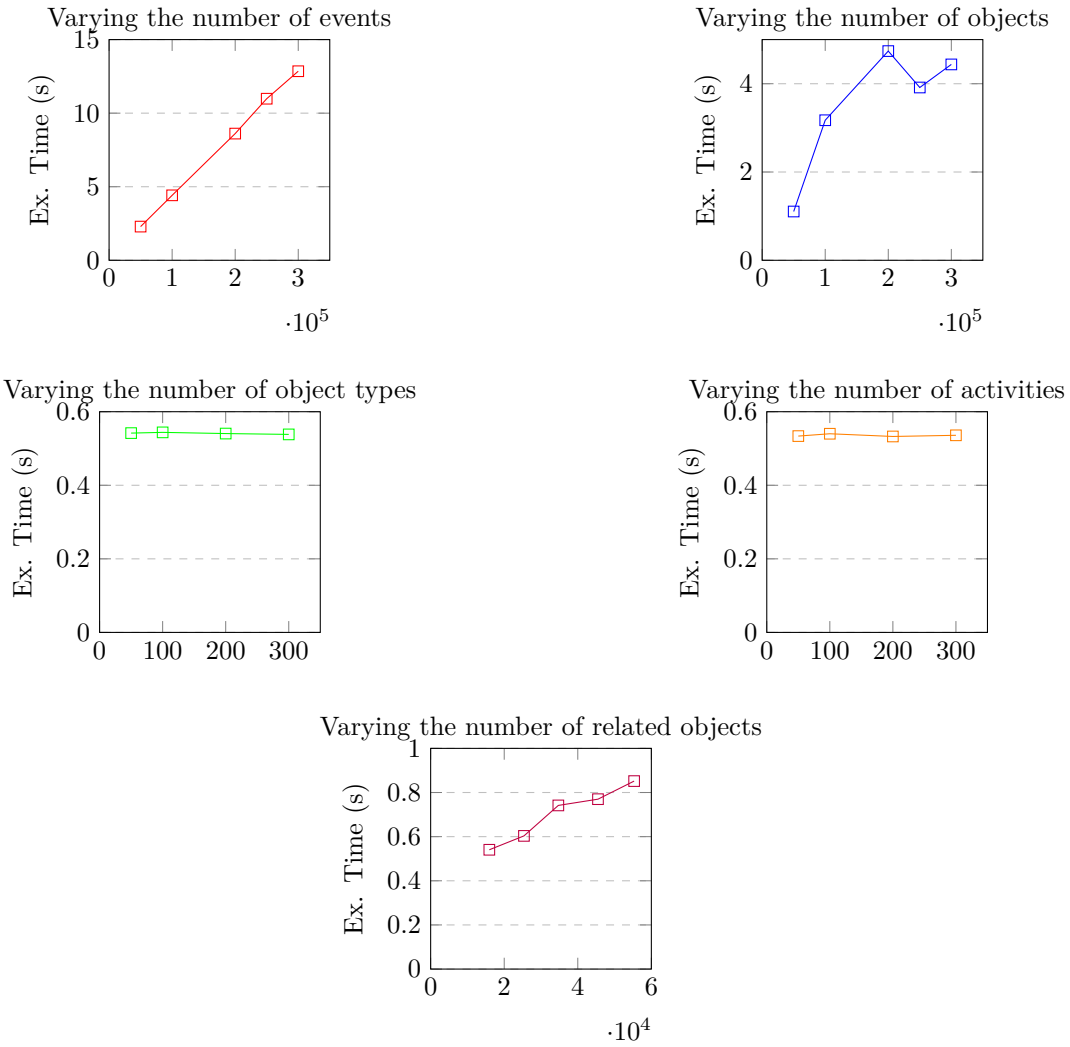


Figure 5.8: Execution times for the XML-OCEL importing.

Table 5.17: Execution times for the XML-OCEL exporting.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79012	50	50	3.9705261
100000	10000	158372	50	50	7.1479790
200000	10000	315318	50	50	14.6917108
250000	10000	395565	50	50	17.7828745
300000	10000	474097	50	50	21.5698638
Varying the number of objects					
10000	50000	15894	50	50	1.7196090
10000	100000	15948	50	50	3.0834063
10000	200000	15792	50	50	4.9735830
10000	250000	16033	50	50	5.5814352
10000	300000	15800	50	50	6.5699145
Varying the number of object types					
10000	10000	15769	50	50	0.9278010
10000	10000	15656	100	50	0.9186205
10000	10000	15931	200	50	0.8686433
10000	10000	15833	300	50	0.9073590
Varying the number of activities					
10000	10000	16006	50	50	0.9291305
10000	10000	15780	50	100	0.8990814
10000	10000	15733	50	200	0.8938648
10000	10000	15814	50	300	0.8968855
Varying the number of related objects					
10000	10000	15878	50	50	0.9714318
10000	10000	25380	50	50	0.9734456
10000	10000	35710	50	50	1.0692974
10000	10000	44836	50	50	1.1170109
10000	10000	54387	50	50	1.1800500

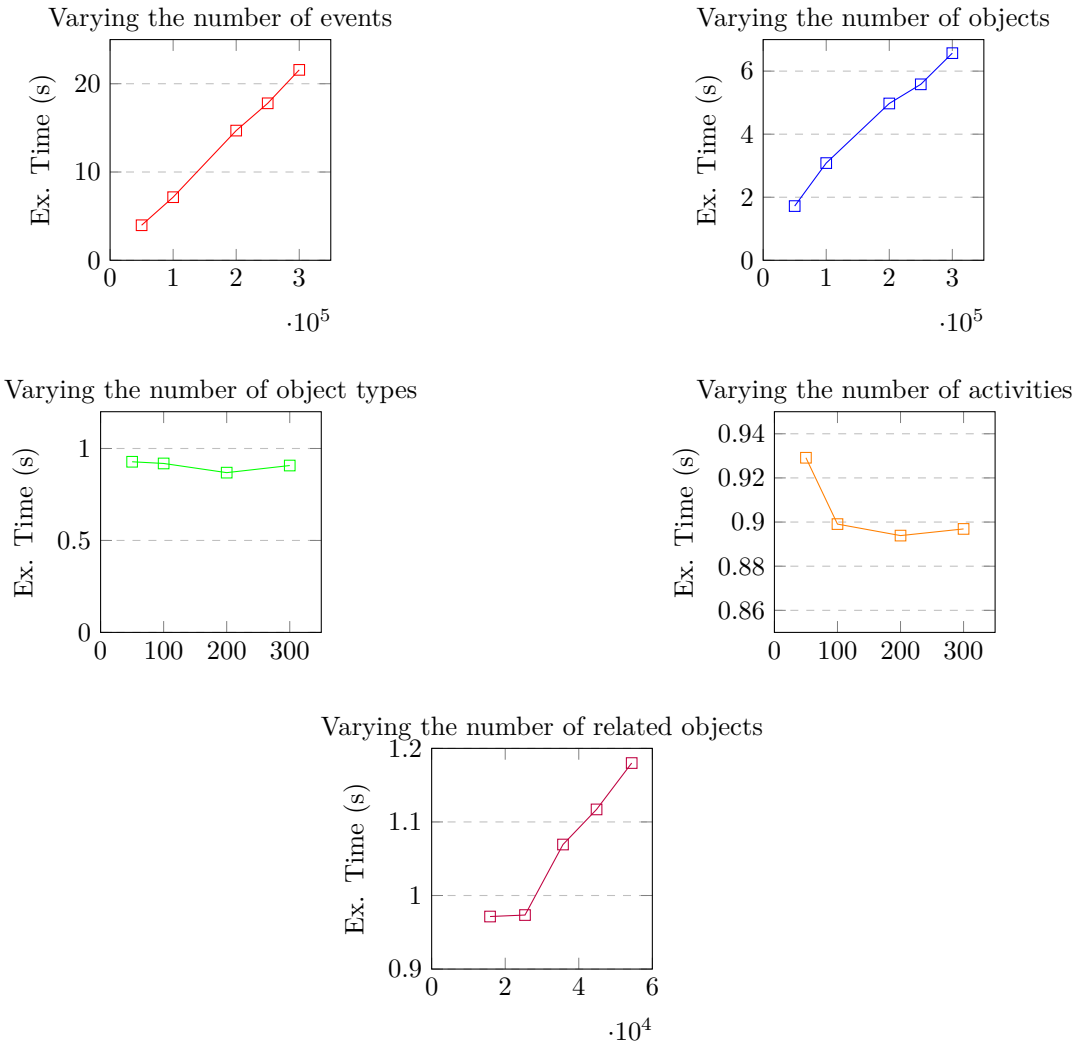


Figure 5.9: Execution times for the XML-OCEL exporting.

5.5.3 Assessment of the OCEL 2.0 (XML) Importing and Exporting Performance

Our investigation revealed a pronounced linear relationship between the execution times and integral elements of the OCEL, namely the count of events, the array of objects, and the complexity stemming from Event-to-Object relationships. The underlying rationale for this linear progression resides in the inherent architecture of the XML file associated with the OCEL 2.0 specification. The file format necessitates the generation of new lines for every event, object, and their relationships.

For a comprehensive perspective:

- Table 5.18 captures a detailed breakdown of performance nuances during the importing phase of the OCEL 2.0 (XML) standard. Complementing this, Figure 5.10 offers a graphical representation, bridging numerical data with visual comprehension.
- For insights concerning the exporting dynamics of the OCEL 2.0 (XML) specification, refer to Table 5.19. A visual representation is portrayed in Figure 5.11.

Table 5.18: Execution times for the OCEL 2.0 (XML) importing.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79435	50	50	1.7118646
100000	10000	158540	50	50	3.2552428
200000	10000	316321	50	50	6.3962816
250000	10000	395409	50	50	8.0954118
300000	10000	474810	50	50	9.5073211
Varying the number of objects					
10000	50000	15877	50	50	0.7417446
10000	100000	15684	50	50	1.1704036
10000	200000	15813	50	50	2.0019393
10000	250000	15647	50	50	2.3549721
10000	300000	15831	50	50	5.3788872
Varying the number of object types					
10000	10000	15910	50	50	0.5457578
10000	10000	15771	100	50	1.7147929
10000	10000	15735	200	50	0.4879503
10000	10000	15962	300	50	0.4647944
Varying the number of activities					
10000	10000	15734	50	50	0.5119915
10000	10000	15699	50	100	0.4340967
10000	10000	15909	50	200	0.4994736
10000	10000	15886	50	300	0.3915132
Varying the number of related objects					
10000	10000	15816	50	50	0.4394561
10000	10000	25578	50	50	0.4799230
10000	10000	35857	50	50	0.5917849
10000	10000	44929	50	50	0.6845015
10000	10000	55194	50	50	0.7571096

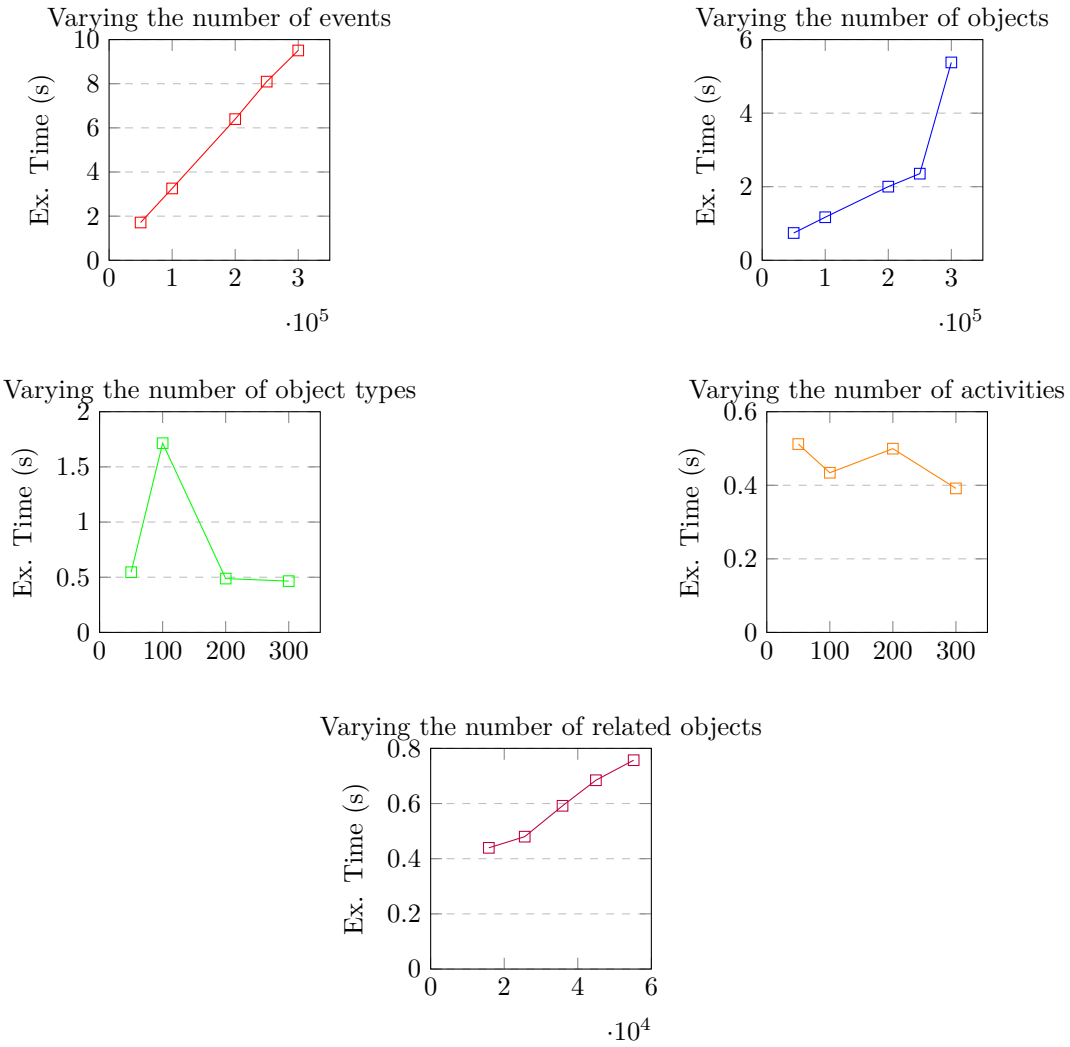


Figure 5.10: Execution times for the OCEL 2.0 (XML) importing.

Table 5.19: Execution times for the OCEL 2.0 (XML) exporting.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79028	50	50	3.9116333
100000	10000	158034	50	50	7.6043889
200000	10000	316605	50	50	15.0541424
250000	10000	396034	50	50	19.1210075
300000	10000	474523	50	50	22.2182205
Varying the number of objects					
10000	50000	15673	50	50	1.3432566
10000	100000	15919	50	50	1.8360404
10000	200000	15859	50	50	2.9361057
10000	250000	15800	50	50	3.4872515
10000	300000	15899	50	50	3.9762817
Varying the number of object types					
10000	10000	15884	50	50	0.9473345
10000	10000	15843	100	50	1.0400404
10000	10000	15817	200	50	1.0070991
10000	10000	15852	300	50	1.0771524
Varying the number of activities					
10000	10000	15803	50	50	0.9423322
10000	10000	16036	50	100	0.9442702
10000	10000	15928	50	200	1.0275111
10000	10000	15744	50	300	1.0884441
Varying the number of related objects					
10000	10000	15770	50	50	0.9317355
10000	10000	25441	50	50	0.9598051
10000	10000	35309	50	50	1.0858352
10000	10000	44893	50	50	1.0800734
10000	10000	55384	50	50	1.1682494

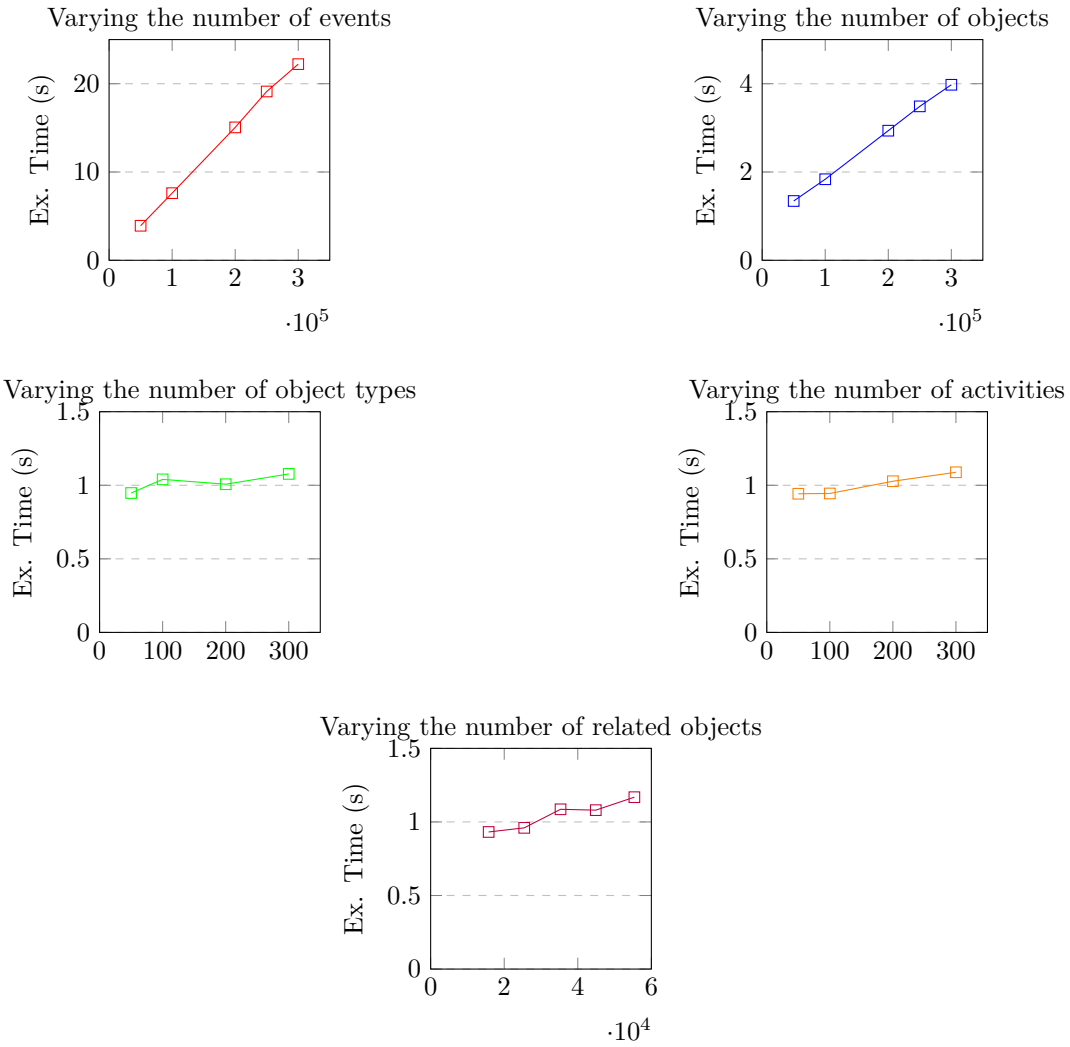


Figure 5.11: Execution times for the OCEL 2.0 (XML) exporting.

5.5.4 Assessment of the OCEL 2.0 (relational) Importing and Exporting Performance

The advent of the OCEL 2.0 SQLite implementation heralds a significant evolution in the realm of OCELs. By leveraging relational databases, this approach facilitates more efficient and structured data storage and retrieval. In this context, we embarked on a detailed exploration of its performance metrics, specifically focusing on the dynamics during the importing and exporting stages.

Regarding the *importing* phase, our analysis echoed the findings from the XML counterpart. The execution time exhibited a linear progression based on three primary elements: the volume of events, the array of objects, and the intricacies inherent to Event-to-Object relationships.

However, the *exporting* phase presented a nuanced picture. While the execution time still followed a linear trajectory, it was influenced by a broader spectrum of factors. Specifically, the number of activities and object types also played pivotal roles, in addition to events, objects, and Event-to-Object relationships. This can be attributed to the architecture of the relational implementation. Events corresponding to different activities are partitioned and stored in distinct tables, mirroring the approach taken for objects of varying object types.

For readers seeking an in-depth numerical perspective:

- Table 5.20 provides granular data on the performance metrics during the importing stage of the OCEL 2.0 (relational) format. To complement this, Figure 5.12 illustrates these findings in a graphical format.
- Insights into the exporting dynamics can be found in Table 5.21, with Figure 5.13 rendering a visual counterpart for easier comprehension.

Table 5.20: Execution times for the OCEL 2.0 (relational) importing.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79038	50	50	1.9315053
100000	10000	158146	50	50	2.8229280
200000	10000	316105	50	50	5.6736681
250000	10000	395318	50	50	7.4138748
300000	10000	474574	50	50	8.3505543
Varying the number of objects					
10000	50000	15788	50	50	0.7674552
10000	100000	15695	50	50	1.1962698
10000	200000	15781	50	50	2.1488750
10000	250000	15897	50	50	2.3957930
10000	300000	15741	50	50	2.8453596
Varying the number of object types					
10000	10000	15695	50	50	0.4353955
10000	10000	15855	100	50	0.4897248
10000	10000	15743	200	50	0.6834859
10000	10000	15760	300	50	0.6480713
Varying the number of activities					
10000	10000	15723	50	50	0.4624380
10000	10000	15858	50	100	0.4525937
10000	10000	15750	50	200	0.5684830
10000	10000	15820	50	300	0.6042211
Varying the number of related objects					
10000	10000	15938	50	50	0.4631735
10000	10000	25568	50	50	0.4693716
10000	10000	35134	50	50	0.5556717
10000	10000	44691	50	50	0.5779192
10000	10000	55974	50	50	0.6343540

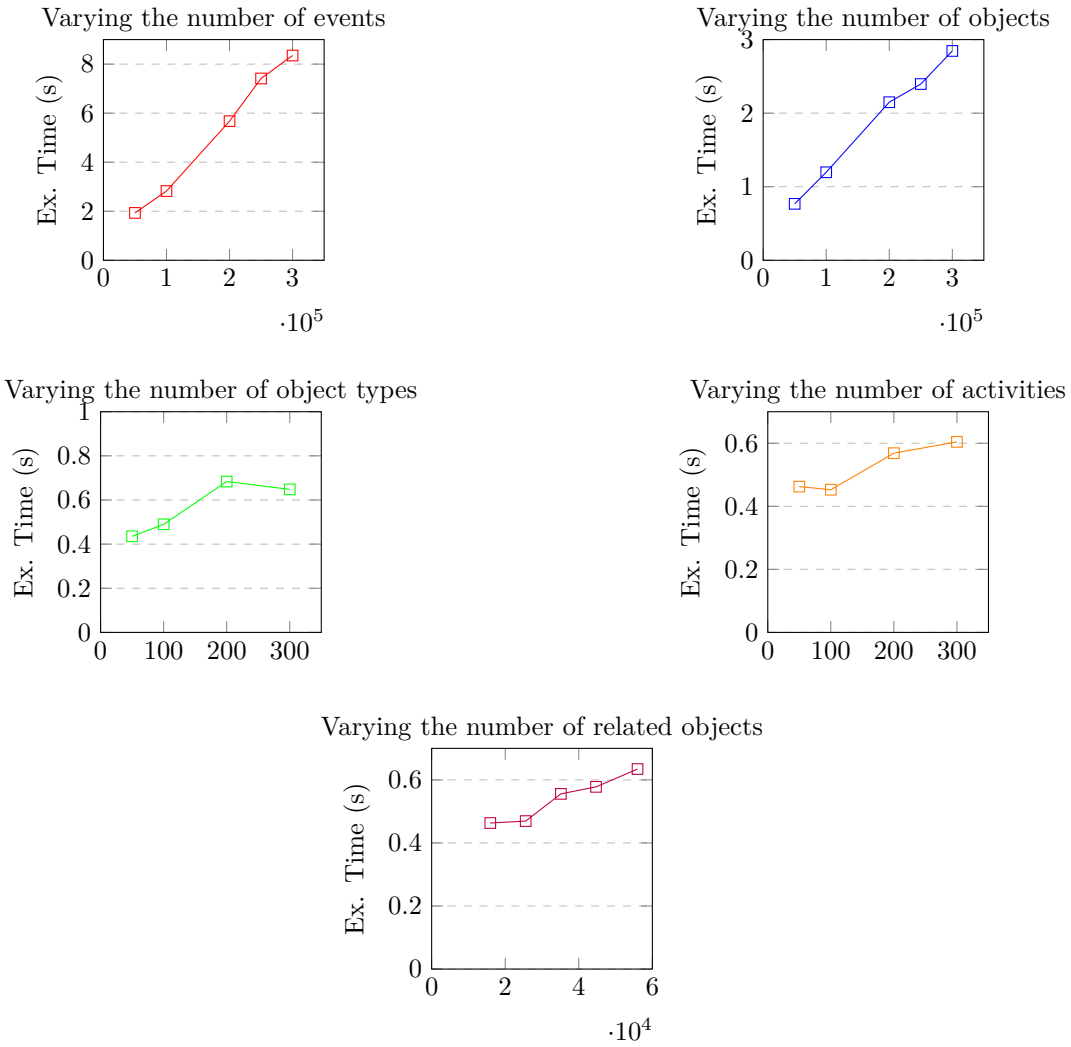


Figure 5.12: Execution times for the OCEL 2.0 (relational) importing.

Table 5.21: Execution times for the OCEL 2.0 (relational) exporting.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79023	50	50	2.8800067
100000	10000	158213	50	50	3.8179100
200000	10000	316364	50	50	5.3827405
250000	10000	394911	50	50	6.1423498
300000	10000	474662	50	50	7.1400479
Varying the number of objects					
10000	50000	15822	50	50	2.5544626
10000	100000	15843	50	50	3.4439342
10000	200000	15821	50	50	4.2837820
10000	250000	15756	50	50	4.8393653
10000	300000	15862	50	50	5.8866952
Varying the number of object types					
10000	10000	15782	50	50	2.1295528
10000	10000	15802	100	50	3.1041289
10000	10000	15934	200	50	5.0965092
10000	10000	15933	300	50	6.8869124
Varying the number of activities					
10000	10000	15906	50	50	2.4272622
10000	10000	15811	50	100	2.9434838
10000	10000	15799	50	200	5.3021660
10000	10000	15829	50	300	6.7925570
Varying the number of related objects					
10000	10000	15996	50	50	2.1074866
10000	10000	25431	50	50	2.4425101
10000	10000	35281	50	50	2.1926336
10000	10000	45689	50	50	2.6579667
10000	10000	55132	50	50	2.2449341

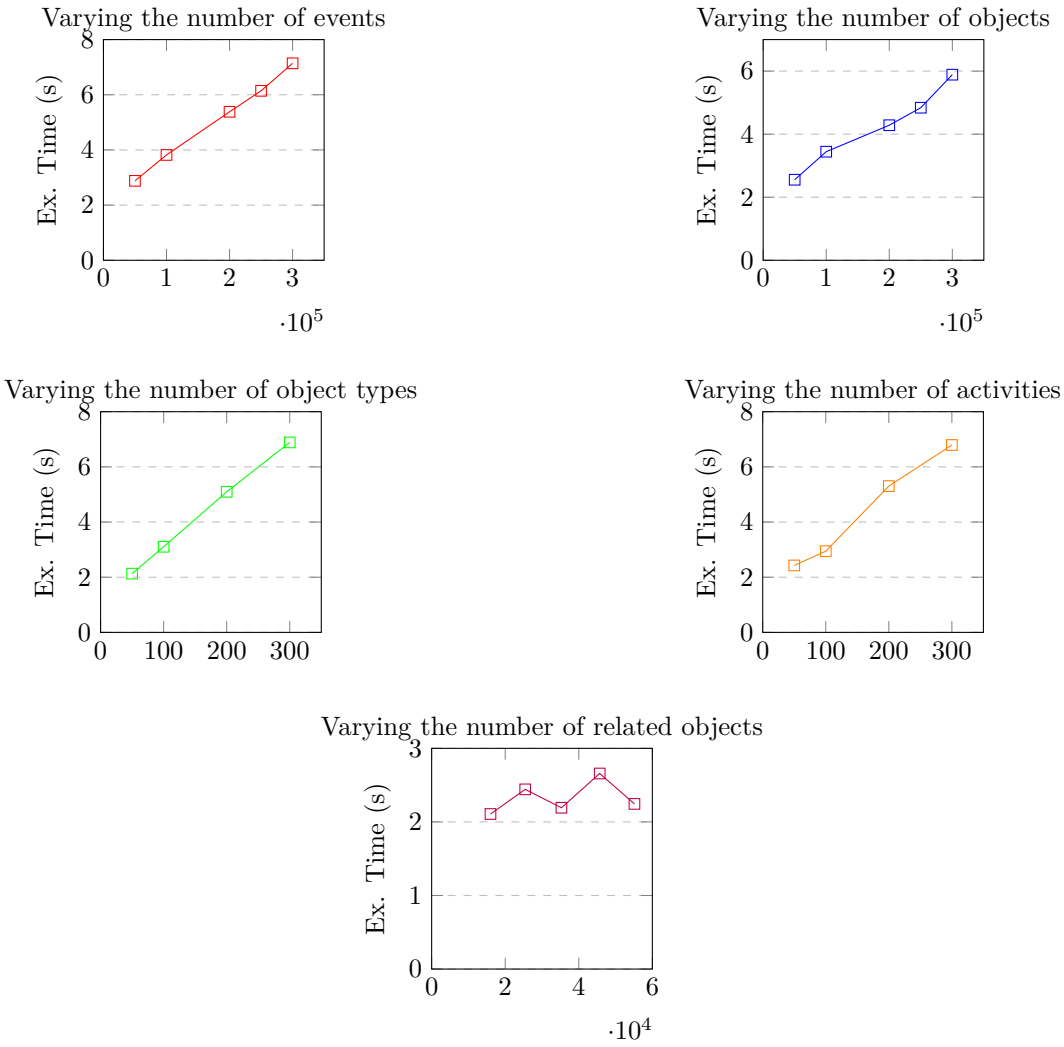


Figure 5.13: Execution times for the OCEL 2.0 (relational) exporting.

Conclusion

This chapter has addressed the critical tasks of data extraction and preprocessing for Object-Centric Process Mining. We have presented a structured methodology for extracting OCELs from relational databases, providing a step-by-step approach from table selection to event log extraction, encompassing the formulation of blueprints and the nuances of storing OCELs in various formats. Furthermore, we have explored a range of preprocessing techniques, focusing on filtering methods and their consistent application to maintain data integrity. The assessment of importing and exporting performance across different OCEL versions offers valuable insights into the practical considerations of working with OCEL data. By establishing robust data handling procedures, this chapter has laid the groundwork for the subsequent chapters on object-centric process discovery and conformance checking. The ability to extract, store, and preprocess OCEL data efficiently and effectively is crucial for the successful application of OCPM techniques. The standardized OCEL format and the preprocessing techniques presented here will enable us to conduct meaningful analyses and derive valuable insights from complex, object-centric processes in the following chapters. Furthermore, the performance evaluations conducted in this chapter provide a benchmark for future tool development and optimization in the field of OCPM.

Chapter 6

Object-Centric Process Discovery (OCPD)

"*In understanding the whole, one must seek to understand the parts; in every complex system, lies the beauty of individual components.*"
Dr. Eugene Fielding

Introduction

This chapter explores the core task of Object-Centric Process Discovery (OCPD), aiming to uncover insightful process models from OCELs that effectively capture the complex interplay of multiple objects. We begin by examining the concept of collating traditional process models, establishing a baseline for comparison with object-centric approaches. We then introduce Object-Centric Directly-Follows Graphs (OCDFGs), including their textual abstraction for analysis with Large Language Models (LLMs), as a means of visualizing the sequential relationships between activities across different object types. Subsequently, we delve into Object-Centric Petri Nets (OCPNs), providing a more formal and expressive model capable of capturing concurrency and complex control flow patterns. The chapter further explores the discovery of Object Graph Enrichments, demonstrating their application to real-world processes like Purchase-to-Pay (P2P) and Order-to-Cash (O2C). A detailed methodological framework for OCPD is presented, guiding the reader through the selection of appropriate discovery algorithms and relevant object types. Finally, a comprehensive assessment of the proposed methodology, focusing on the execution times for discovering OCDFGs, OCPNs, and Object Graph Enrichments, is provided.

The following contributions, all worked by the author of this thesis, have been integrated into the chapter:

- Concepts and techniques for extracting multi-viewpoint models from OCELs, including OCDFGs, as discussed in [22].
- A token-based replay approach that streamlines conformance checking and process enhancement, building on [24].
- Extensions of OCDFG modeling and accompanying metrics, as well as supporting tooling for OCPD, informed by [25].
- Methods for discovering OCPNs from OCELs, leveraging token-based replay for determining arc typing and evaluating model quality, grounded in [119].
- Foundational ideas on OCDFGs derived from "StarStar Models" computed from event-to-object graph enrichments, introduced in [21].
- Enhanced token-based replay strategies offering improved diagnostics, as reported in [23].
- Textual abstractions that enable LLM-driven process mining analyses, presented in [20].

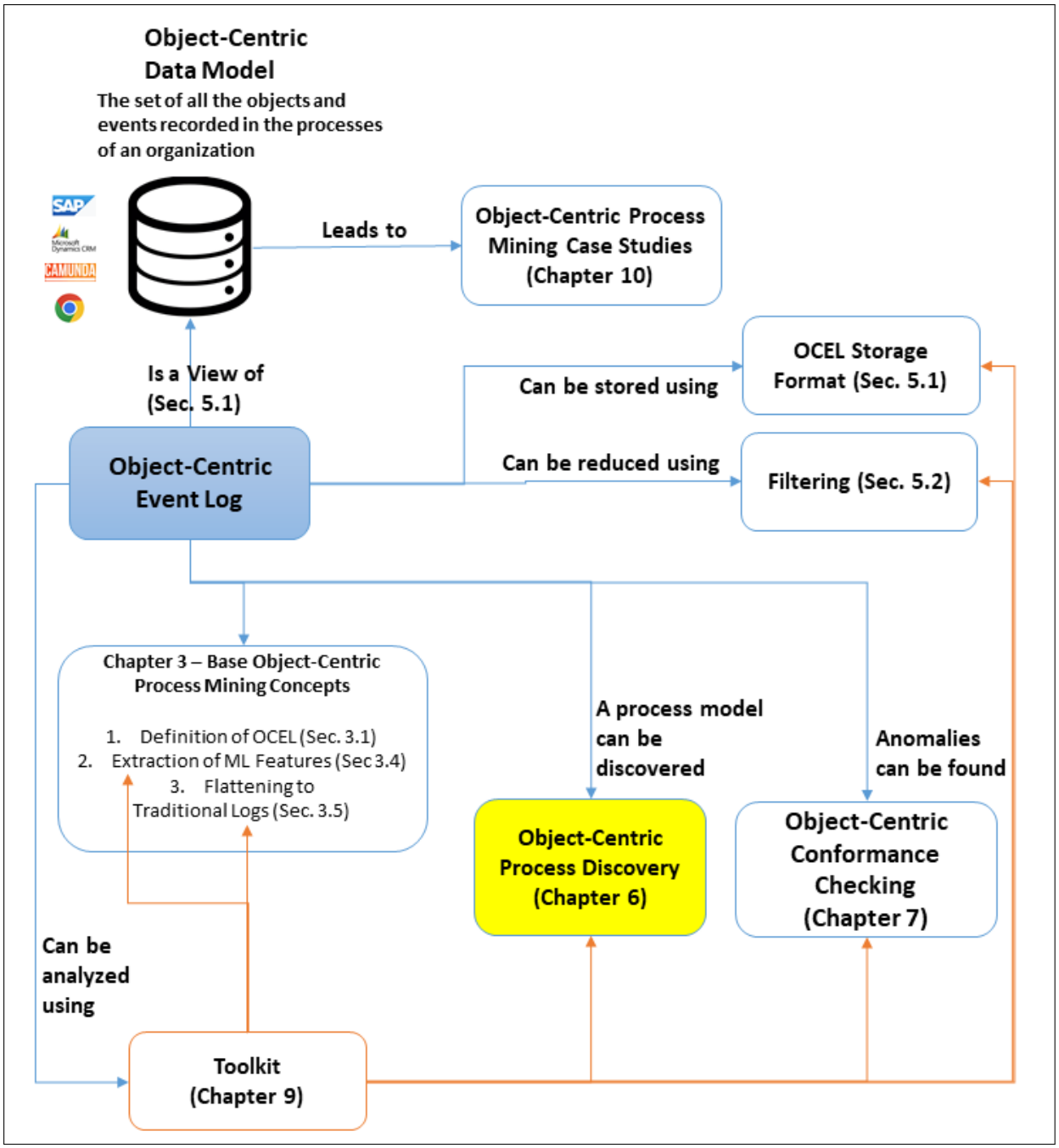


Figure 6.1: Highlight of the problems tackled in this specific chapter.

6.1 Collations of Traditional Models

Definition 39 introduces the concept of OCPD (translating an OCEL to an object-centric process model).

Definition 39 (OCPD). *An OCPD operation is a method $disc_{OC} : U_{OL} \to U_{OM}$ associating an OCEL to an object-centric process model.*

In OCPD, the simplest class of models are those that are derived by transforming the OCEL into traditional event logs and subsequently applying a conventional process discovery algorithm to it. This approach essentially converts every object type to a corresponding traditional process model, allowing for the seamless integration and reuse of existing process discovery techniques.

Definition 40 (Traditional Process Discovery). *A traditional process discovery operation is a method $disc : U_{TL} \to U_{TM}$ associating a traditional event log to a (traditional) process model.*

Various classes of process models can be effectively considered in this process. Definitions 40 and 41 introduce and elaborate on these traditional process discovery approaches and the different classifications of process models.

Definition 41 (Types of Traditional Process Discovery). *We define the following types of traditional process discovery:*

- *$disc_{DFG} : U_{TL} \to U_{DFG}$ discovers a Directly-Follows Graph from a traditional event log*

$$disc_{DFG}(TL) = (A, F, \pi_{measn}, \pi_{mease})$$

- *$disc_{APN} : U_{TL} \to U_{APN}$ discovers an accepting Petri net from a traditional event log*

$$disc_{APN}(TL) = (P, T, F, \lambda, M_I, M_F)$$

OCPD can be conceived as a comprehensive assembly of traditional process models. The different types of collations that we consider are described in Definition 42

Definition 42 (Types of Collated Object-Centric Models). *Given a set of object types $OT \subseteq U_{otype}$, we define the following types of collated object-centric models:*

- *$OT \to U_{DFG}$ is a collation of Directly-Follows Graphs.*
- *$OT \to U_{APN}$ is a collation of Petri nets.*

Definition 43 elucidates this idea, proposing it as a unification of traditional process models corresponding to different object types found within the OCEL.

Definition 43 (Discovery of Collated Object-Centric Models (Basic Flattening)). *Given an OCEL $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ and a traditional discovery operation $disc : U_{TL} \to U_{TM}$, we define $coll_L : OT \to U_{TM}$ in which for $ot \in OT$, $coll_L(ot) = disc(flat(L, ot))$.*

Definition 44 proposes the usage of Object-Graph-Based Flattening, instead of Basic Flattening. Therefore, models are discovered on flattened logs that contain the events of interconnected objects.

Definition 44 (Discovery of Collated Object-Centric Models (Object-Graph-Based Flattening)). *Given an OCEL $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ and a traditional discovery operation $disc : U_{TL} \to U_{TM}$, we define $coll_L^{OTF} : OT \to U_{TM}$ in which for $ot \in OT$, $coll_L^{OTF}(ot) = disc(ccflat(L, ot))$.*

This methodology of OCPD not only simplifies the complexity of process discovery but also proves beneficial in conformance checking. The flattened version of the OCEL enables the application of traditional conformance checking methods. Thus, the overall approach enhances both the discovery and validation processes of object-centric models, driving their effective interpretation and evaluation in complex system analysis.

6.2 Object-Centric Directly-Follows Graphs

Object-Centric Directly-Follows Graphs (OCDFGs) provide a way to visualize how activities are connected across multiple object types, capturing the idea that each object type evolves through a characteristic sequence of activities. This is formally introduced as an OCDFG in Definition 45.

Definition 45 (OCDFG). *An OCDFG is a tuple* $(A, OT, F, \pi_{measn}, \pi_{mease})$*, where*

- $A \subseteq U_{etype}$ *is a set of activities.*
- $OT \subseteq U_{otype}$ *is a set of object types.*
- *For* $ot \in OT$, $\triangleright_{ot}$ *is the start node of the object type,* $\square_{ot}$ *is the end node of the object type.*
- $F \subseteq \{(a, ot, b) \mid a, b \in A \ \wedge \ ot \in OT\} \cup \{(\triangleright_{ot}, ot, a) \mid a \in A \ \wedge \ ot \in OT\} \cup \{(a, ot, \square_{ot}) \mid a \in A \ \wedge \ ot \in OT\}$
- $\pi_{measn} : A \not\rightarrow \mathbb{R}^{\geq 0}$ *is a measure on the nodes.*
- $\pi_{mease} : F \not\rightarrow \mathbb{R}^{\geq 0}$ *is a measure on the edges.*

While a traditional Directly-Follows Graph (DFG) is typically associated with a single "case type" and models how one activity follows another in that linear perspective, an OCDFG incorporates multiple object types and their interlinked event sequences. Thus, an OCDFG allows us to see not only that an activity b may follow an activity a, but also *for which object types this ordering occurs*, and how these local sequences of events, grounded in different object types, interleave within a broader process.

6.2.1 From Data to Semantics: What Does "Follows" Mean?

When we say that one activity "follows" another activity in an OCDFG, we refer to a relation grounded in the data of an Object-Centric Event Log (OCEL). In an OCEL, events are linked to multiple objects of potentially different types (e.g., purchase orders, invoices, or items). For a given object type ot, the events that mention objects of this type can be arranged in a sequence that reflects how each object moves through various activities over time. The notion of "directly-follows" emerges from examining these sequences.

Formally, consider an object o of type ot. The *lifecycle* of o, denoted $\text{lif}(o)$, is the sequence of events related to o in chronological order. If at some point in this sequence we see an event e_1 labeled with activity a followed immediately by another event e_2 labeled with activity b (with no other event of type ot in between), then we say that "b directly follows a for object type ot." This captures a *local ordering constraint* observed in the data: whenever we look at the lifecycle of an object of type ot, if we encounter activity a, and later b appears immediately after a in that object's timeline, we record a directly-follows relationship (a, ot, b).

Thus, the meaning of the directly-follows relation is tied to how real-world objects evolve:

- Each edge (a, ot, b) in the OCDFG indicates that, for at least one object of type ot, there exists a pair of consecutive events (e_1, e_2) in that object's lifecycle where e_1's activity is a, and e_2's activity is b.
- This does not necessarily mean every occurrence of a for that object type is followed by b, but it does mean there is at least one instance of ot for which b came immediately after a with no other event of type ot interrupting the sequence.

6.2.2 Local Semantics and Partial Orders

The semantics of OCDFGs can be understood in terms of local partial orders induced by object lifecycles:

1. *Object-Centric Perspective*: For each object type ot, the events referencing objects of that type form a linear sequence (per object) that can be aggregated across all objects of type ot. The directly-follows relations extracted from these sequences reflect minimal precedence constraints: if (a, ot, b) is in the OCDFG, a must occur before b on at least one object's lifecycle of type ot, and b appears as the immediate successor of a for that object.

2. *From Local to Global Structure*: The OCDFG merges these local relations for all object types into a single graphical artifact. Edges labeled with different object types show how, within each type-specific "thread" of execution, some activities tend to follow others. Globally, these threads (one per object type) interweave to form the overall process. Although the OCDFG itself does not explicitly represent concurrency or synchronization across object types, the presence of multiple object types often implies that different portions of the process can unfold simultaneously. For example, one purchase order might be advancing from "Create Purchase Order" to "Invoice Receipt" while, independently, a related invoice is moving from "Invoice Receipt" to "Perform Payment." The OCDFG, by showing these edges per object type, reveals the multi-object interplay without forcing a single linear sequence of events.
3. *Minimal Precedence Constraints*: The directly-follows relation is a minimal and local constraint. It does not say that a must always come before b; it merely documents that there is at least one object instance for which a and b occur in immediate succession. Nor does it say anything about events of other object types happening in parallel. The OCDFG's edges are simply the recorded minimal "building blocks" of observed orderings, from which more sophisticated models can infer concurrency, choice, and synchronization patterns.

6.2.3 Assigning Meaning to Frequencies and Performance Measures

Beyond the existence of edges, the OCDFG can be annotated with frequency and timing metrics. For example, one can measure how often b follows a for a given object type, or the average time elapsed between these two activities. This introduces a richer semantic layer:

- A frequent edge (a, ot, b) suggests a strong, stable ordering pattern within the lifecycle of objects of type ot.
- A high average time between activities a and b indicates that, while b may follow a, significant delays are common.

These metrics offer a semantic interpretation of process behavior:

- High frequency edges are more likely to represent critical transitions in the object lifecycle.
- Long durations suggest potential bottlenecks or complexity in moving from one state to the next within an object's lifecycle.

In sum, the directly-follows edges do not merely say "a comes before b" in some abstract sense; they link the structural notion of succession to actual process characteristics: how consistently b succeeds a, and how much time, on average, passes between them.

6.2.4 Discovery of Object-Centric Directly-Follows Graphs

Having established the meaning and semantics of directly-follows relations in an object-centric setting, we now describe how to systematically derive an OCDFG from an Object-Centric Event Log (OCEL). The goal is to transform raw event data—where each event references one or more objects of potentially different types—into a structured model that captures how activities are chained together for each object type, and then aggregate these relationships into a comprehensive, object-centric view.

High-Level Idea. To discover an OCDFG, we iterate over each object type present in the OCEL and extract the sequences of events associated with that object type. Within each such sequence, we identify all pairs of consecutive events and derive the corresponding directly-follows relations between activities. These type-specific relations form the edges in the OCDFG, complemented by conceptual start and end nodes for each object type. Once these edges are collected from all object types, they are combined into a single model that captures the interplay of multiple object lifecycles.

Detailed Procedure. Formally, we start with an OCEL and identify its set of activities A and object types OT. For each object type $ot \in OT$, we consider all objects of that type and look at their lifecycles. By detecting consecutive activity pairs (a, b) for each ot, we introduce an edge (a, ot, b) into our graph. Additionally, we represent the initial and final activities encountered in each object's lifecycle by edges connecting to the artificial start ($\triangleright_{ot}$) and end ($\square_{ot}$) nodes for that object type. In doing so, we create a graph that shows how, for each object type, activities flow from a conceptual start node towards an end node.

Refining the Model. While the initial set of edges directly reflects the observed event sequences, they can be further enriched and filtered. We can annotate activities and edges with measures such as frequency and timing metrics. For instance, we may record how often a particular (a, ot, b) relation occurs, or the average time elapsed between a and b. Such metrics allow for a more nuanced interpretation of the process structure, highlighting the most common or time-consuming transitions.

Definitions 46 and 47 introduce *Object-Centric Directly-Follows Tuples* and their aggregations, providing the foundational elements needed to precisely identify and measure the arcs. Definition 48 then formally specifies the discovery step itself, starting from an OCEL and producing an OCDFG equipped with various optional metrics.

Definition 46 (Object-Centric Directly-Follows Tuples)**.** *Given an OCEL* $L = (A', OT', E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, and given* $ot \in OT$*, let* $O_{ot} = \{o \in O \mid objtype(o) = ot\}$*, we define the directly-follows tuples for an object type* $ot \in OT$ *of* L *as:*

$$DFT_{ot} = \bigcup_{o \in O_{ot}, lif(o) = \langle e_1, \ldots, e_n \rangle} \{(e_i, o, e_{i+1}) \mid 1 \leq i < |lif(o)|\}$$

Given two activities $a, b \in A'$*, we define*

$$DFT_{ot,a,b} = \{(e_1, o, e_2) \in DFT_{ot} \mid evtype(e_1) = a \ \wedge \ evtype(e_2) = b\}$$

Definition 47 (Aggregations of Object-Centric Directly-Follows Tuples)**.** *Given an OCEL* $L = (A', OT', E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, given* $ot \in OT$*,* $a, b \in A$*, we define*

- $EC_{ot,a,b} = \{(e_1, e_2) \mid (e_1, o, e_2) \in DFT_{ot,a,b}\}$
- $UO_{ot,a,b} = \{o \mid (e_1, o, e_2) \in DFT_{ot,a,b}\}$

The discovery of an OCDFG is presented in Definition 48. Here, the fundamental elements of the OCEL are leveraged to uncover an OCDFG. Through a series of steps involving the definition of the activities, object types, and measures on nodes and edges, a coherent and structured OCDFG can be derived.

Definition 48 (Discovery of an OCDFG)**.** *Given an OCEL* $L = (A', OT', E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we discover an OCDFG* $(A, OT, F, \pi_{measn}, \pi_{mease})$ *as follows:*

- $A = A'$.
- $OT = OT'$.
- *For* $ot \in OT$*, let* $\cap O_{ot} = \{o \in O \mid objtype(o) = ot \ \wedge \ \exists_{e \in E} o \in \pi_{omap}(e)\}$*. We define the set of edges as:*

$$F = \bigcup_{ot \in OT} \{(evtype(e_1), objtype(o), evtype(e_2)) \mid (e_1, o, e_2) \in DFT_{ot}\} \cup \bigcup_{o \in O_{ot}} \{(\triangleright_{ot}, ot, evtype(lif(o)(1))), (evtype(lif(o)(|lif(o)|)), ot, \square_{ot})\}$$

- *We can define* $\pi_{measn} : A \to \mathbb{R}^{\geq 0}$ *in three ways:*
  - *(number of events)* $\pi_{measn_E}(a) = |\{e \in E \mid evtype(e) = a\}|$
  - *(number of unique objects)* $\pi_{measn_{UO}}(a) = |\{o \in O \mid \exists_{e \in E} \ evtype(e) = a \ \wedge \ o \in \pi_{omap}(e)\}|$
  - *(number of total objects)* $\pi_{measn_{TO}}(a) = |\{(e, o) \in E \times O \mid evtype(e) = a \ \wedge \ o \in \pi_{omap}(e)\}|$
- *We can define* $\pi_{mease} : F \not\to \mathbb{R}^{\geq 0}$ *in several different ways for* $(a, ot, b) \in F$ *(where* $a, b \in A$ *and* $ot \in OT$*):*
  - *(number of event couples)* $\pi_{mease^f_{EC}}(a, ot, b) = |EC_{ot,a,b}|$
  - *(average time between events)* $\pi_{mease^p_{EC}}(a, ot, b) = avg\{time(e_2) - time(e_1) \mid (e_1, e_2) \in EC_{ot,a,b}\}$
  - *(number of unique objects)* $\pi_{mease^f_{UO}}(a, ot, b) = |UO_{ot,a,b}|$

- *(number of total objects)* $\pi_{measef_{TO}}(a, ot, b) = |DFT_{ot,a,b}|$
- *(average time in the flattened log)* $\pi_{measep_{TO}}(a, ot, b) = avg\{time(e_2) - time(e_1) \mid (e_1, o, e_2) \in DFT_{ot,a,b}\}$

Note that the cardinality functions π_{measn} and π_{mease} are partial functions because certain measures may not be defined for the start $\triangleright_{ot}$ and end $\square_{ot}$ nodes and edges; therefore, they may not have values for these elements. According to Definition 48, different metrics are introduced both for the frequency of activities and for the frequency of edges in the context of an OCDFG.

For the frequency of activities, we have:

- the *number of events* (π_{measn_E}): This metric measures the number of events for each activity, where an event is associated with the activity if its activity property (evtype) matches the activity.
- the *number of unique objects* ($\pi_{measn_{UO}}$): This measures the number of distinct objects associated with an activity. An object is associated with an activity if there exists an event where the event's activity property matches the activity, and the object is part of the object mapping of the event.
- the *number of total objects* ($\pi_{measn_{TO}}$): This calculates the total number of times an object is associated with an activity. This includes counting multiple occurrences of the same object.

For the frequency of edges, we have:

- the *number of event couples* ($\pi_{measef_{EC}}$): This counts the number of pairs of events that form an edge. An edge here is defined as a sequence of two activities associated with the same object type.
- the *number of unique objects* ($\pi_{measef_{UO}}$): This measures the number of distinct objects that are associated with an edge, where an edge is defined as before.
- the *number of total objects* ($\pi_{measef_{TO}}$): This calculates the total number of times an object is associated with an edge, including multiple occurrences of the same object.

In terms of performance metrics at the edge level, two different measurements are used:

- *average time between events* ($\pi_{measep_{EC}}$): This metric calculates the average duration between two consecutive events that form an edge.
- *average time in the flattened log* ($\pi_{measep_{TO}}$): This metric also measures the average time duration, but in the context of the flattened log, which could lead to convergence issues.

So, to summarize, Definition 48 proposes a framework for measuring both the frequency and the performance of activities and edges within an OCDFG. The metrics account for the multiple roles that objects can have in different activities and edges, which is a distinctive feature of OCELs.

The resulting OCDFG thus becomes an aggregated, object-aware map of how observed activities follow one another, offering a richer perspective than traditional DFGs and serving as a stepping stone towards more expressive and formal object-centric models.

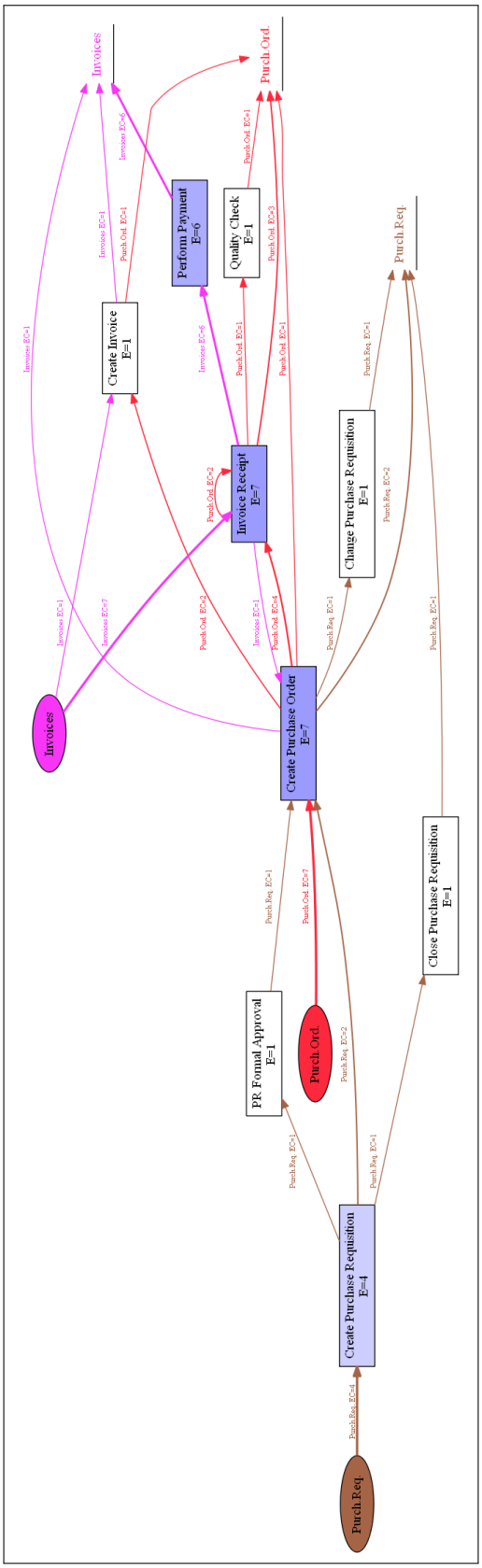


Figure 6.2: OCDFG discovered on the example event log provided in Table 3.3 (the frequency on the activities is π_{measn_E}; the frequency on the arcs is $\pi_{measef_{EC}}$).

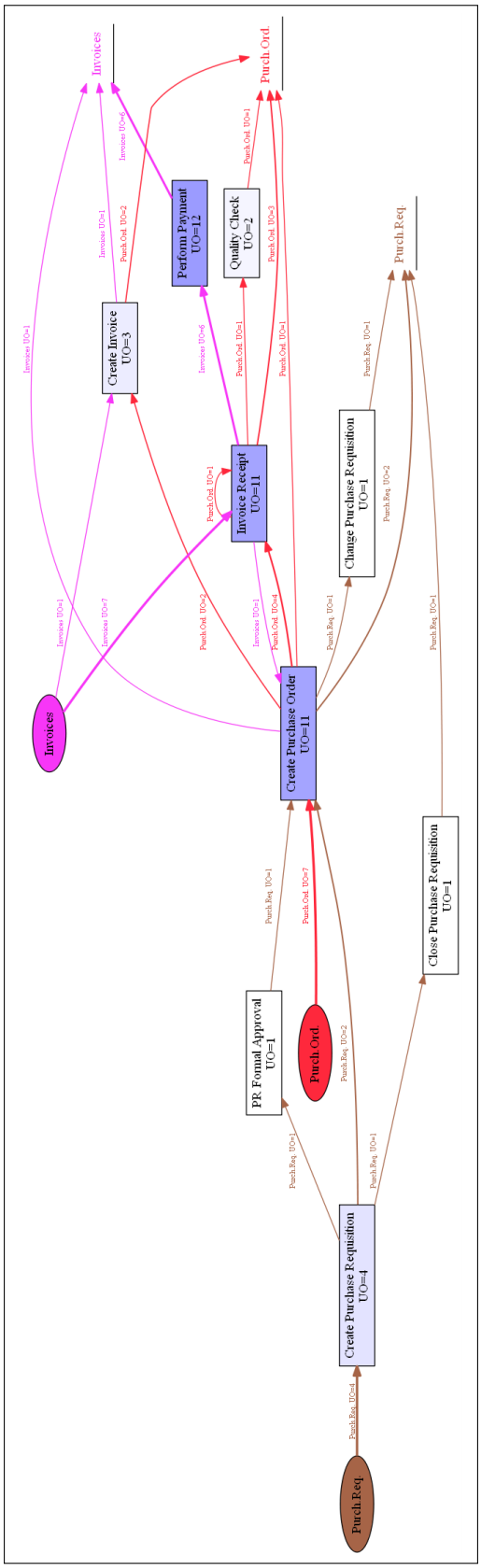


Figure 6.3: OCDFG discovered on the example event log provided in Table 3.3 (the frequency on the activities is $\pi_{measn_{UO}}$; the frequency on the arcs is $\pi_{measef_{UO}}$).

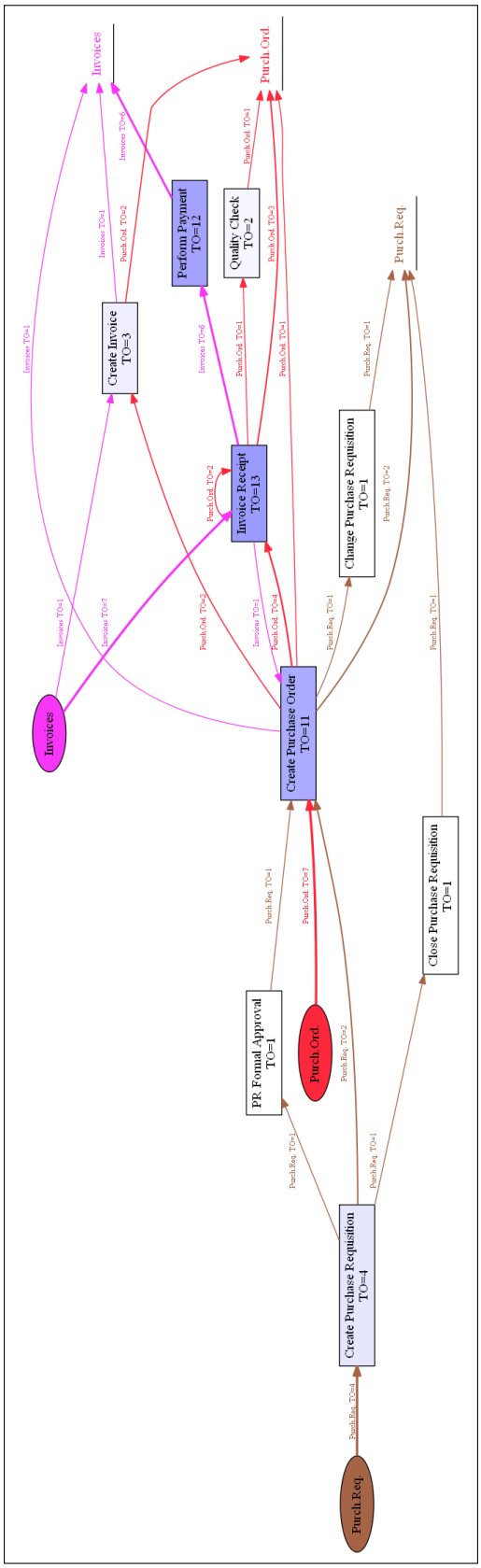


Figure 6.4: OCDFG discovered on the example event log provided in Table 3.3 (the frequency on the activities is $\pi_{measn_{TO}}$; the frequency on the arcs is $\pi_{measef_{TO}}$).

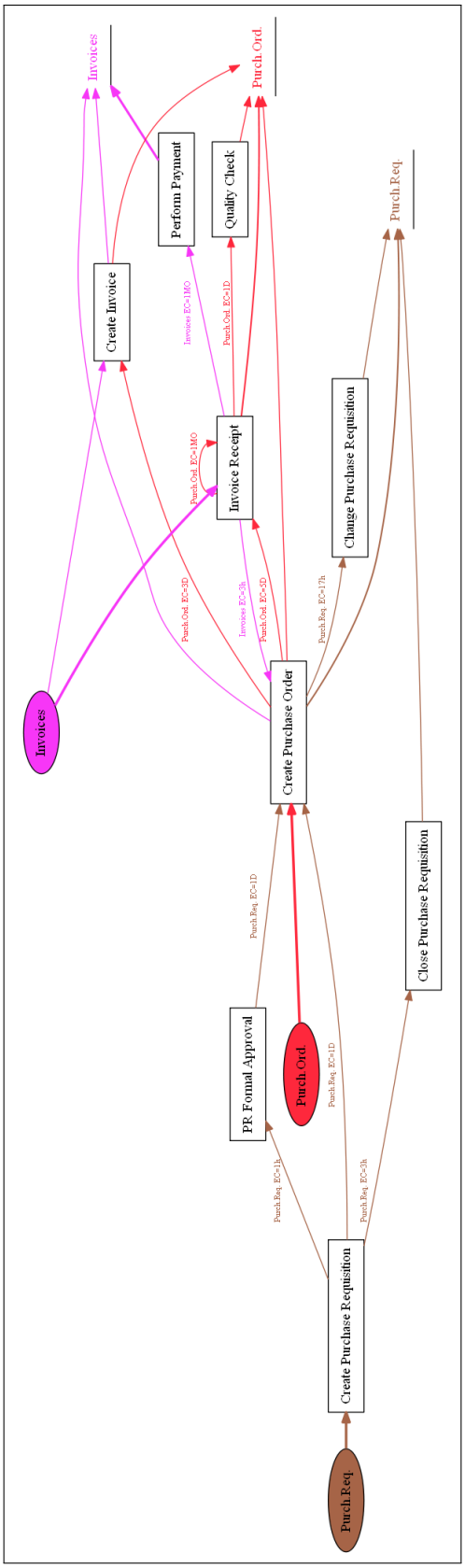


Figure 6.5: OCDFG discovered on the example event log provided in Table 3.3 (the frequency on the activities is π_{measn_E}; the frequency on the arcs is $\pi_{measе^p_{EC}}$).

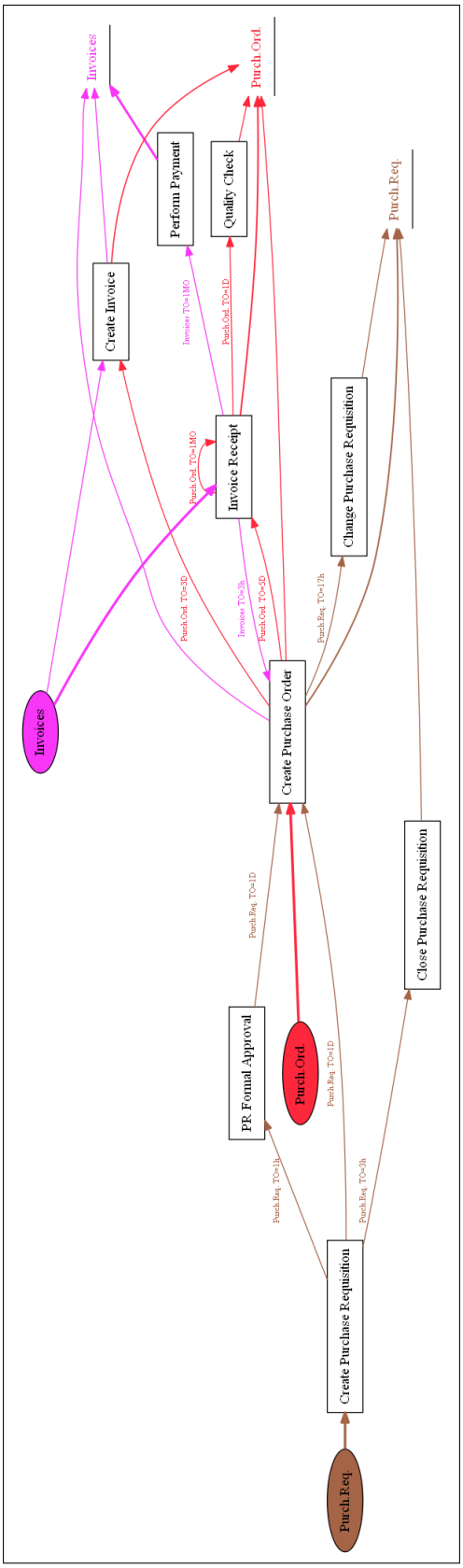


Figure 6.6: OCDFG discovered on the example event log provided in Table 3.3 (the frequency on the activities is $\pi_{measn_{TO}}$; the frequency on the arcs is $\pi_{measep_{TO}}$).

We include some representations of OCDFGs:

- Figure 6.2: This figure displays the OCDFG derived from the example event log shown in Table 3.3. The frequency of the activities is determined by the number of events (π_{measn_E}), while the frequency of the arcs is determined by the number of event couples ($\pi_{mease^f_{EC}}$).
- Figure 6.3: This figure presents the OCDFG extracted from the example event log provided in Table 3.3. The frequency of the activities is calculated based on the number of unique objects ($\pi_{measn_{UO}}$), and the frequency of the arcs is calculated based on the number of unique objects ($\pi_{mease^f_{UO}}$).
- Figure 6.4: This figure shows the OCDFG obtained from the example event log in Table 3.3. The frequency of the activities is computed by counting the total number of objects ($\pi_{measn_{TO}}$), and the frequency of the arcs is measured by counting the total number of objects ($\pi_{mease^f_{TO}}$).
- Figure 6.5: This figure illustrates the OCDFG identified from the example event log described in Table 3.3. The frequency of the activities is determined by the number of events (π_{measn_E}), and the performance of the arcs is evaluated by the average time between events ($\pi_{mease^p_{EC}}$).
- Figure 6.6: This figure depicts the OCDFG discovered from the example event log in Table 3.3. The frequency of the activities is assessed by the total number of objects ($\pi_{measn_{TO}}$), and the performance of the arcs is calculated by the average time in the flattened log ($\pi_{mease^p_{TO}}$).

Definition 49 describes how to apply filtering on an OCDFG based on thresholding the frequency of activities and edges.

Definition 49 (Frequency Thresholding)**.** *Given an OCDFG* $OCDFG = (A, OT, F, \pi_{measn}, \pi_{mease})$*, and two thresholds* $n, m \in \mathbb{R}^{\geq 0}$*, we define* $OCDFG' = (A', OT, F', \pi'_{measn}, \pi'_{mease})$ *in which:*

- $A' = \{a \in A \mid a \notin dom(\pi_{measn}) \ \vee \ \pi_{measn}(a) \geq n\}$
- $F' = \{f \in F \mid f \notin dom(\pi_{mease}) \ \vee \ \pi_{mease}(f) \geq m\}$
- $\pi_{measn'} = \pi_{measn_{\restriction A'}}$
- $\pi_{mease'} = \pi_{mease_{\restriction F'}}$

The main goal of Definition 49 is to keep the most frequent activities and arcs in the OCDFG, without any ensurance on the connectness of the graph.

6.2.5 Textual Abstraction of OCDFGs

When it comes to OCELs, we often utilize OCDFGs as part of the abstraction process. An example of such an abstraction is depicted in Listing 6.1. In this case, each type of object within the log has a corresponding set of arcs in its Directly-Follows Graph. These arcs illustrate the flow and sequence of activities involving that particular object type. However, similar to traditional Directly-Follows Graphs, we must be mindful of the context window limitation. As a result, arcs with fewer occurrences may be omitted from the textual abstraction to maintain a manageable size. Regardless of this omission, the critical insight here is the pattern of activities and interactions concerning each object type.

Like the previously discussed abstractions, the way in which these object-specific arcs are represented in the abstraction does not significantly affect how advanced LLMs interpret them. As long as the arcs are clearly stated, the LLM can efficiently process and generate useful insights from them. This flexibility underlines the applicability and utility of advanced LLMs in understanding and analyzing complex object-centric artifacts.

Listing 6.1: Textual abstraction of the OCDFG obtained from Table 3.3 (obtained using the method `pm4py.llm.abstract_ocel_ocdfg`).

```
If I have an OCEL with the following directly follows graph (split between the different object types):

Object type: Invoices
  Invoice Receipt -> Perform Payment (frequency (number of events) = 6, frequency (number of objects) = 6, duration =
      4768890.00)
  Invoice Receipt -> Create Purchase Order (frequency (number of events) = 1, frequency (number of objects) = 1, duration =
       10800.00)

Object type: Purch.Ord.
  Create Purchase Order -> Invoice Receipt (frequency (number of events) = 4, frequency (number of objects) = 4, duration =
       514710.00)
  Invoice Receipt -> Invoice Receipt (frequency (number of events) = 2, frequency (number of objects) = 1, duration =
      2635230.00)
  Create Purchase Order -> Create Invoice (frequency (number of events) = 2, frequency (number of objects) = 2, duration =
      296850.00)
  Invoice Receipt -> Quality Check (frequency (number of events) = 1, frequency (number of objects) = 1, duration =
      153000.00)

Object type: Purch.Req.
  Create Purchase Requisition -> Create Purchase Order (frequency (number of events) = 2, frequency (number of objects) =
      2, duration = 93360.00)
  PR Formal Approval -> Create Purchase Order (frequency (number of events) = 1, frequency (number of objects) = 1,
      duration = 110700.00)
  Create Purchase Order -> Change Purchase Requisition (frequency (number of events) = 1, frequency (number of objects) =
      1, duration = 61200.00)
  Create Purchase Requisition -> Close Purchase Requisition (frequency (number of events) = 1, frequency (number of objects)
       = 1, duration = 12600.00)
  Create Purchase Requisition -> PR Formal Approval (frequency (number of events) = 1, frequency (number of objects) = 1,
      duration = 3600.00)
```

6.3 Object-Centric Petri Nets (OCPNs)

An *Object-Centric Petri Net (OCPN)* is a special kind of Petri net designed to model processes involving multiple types of objects, capturing the interactions between them. Traditional Petri nets are typically used to model the flow of a single case type, but OCPNs extend this by incorporating multiple object types, like orders, items, and customers in an online shop.

In this section, we introduce OCPNs with an emphasis on their semantics, especially the concept of *variable arcs*, which enable the model to capture many-to-one or one-to-many relationships between objects and activities.

6.3.1 Definition and Semantics of OCPNs

First, we formalize the concept of OCPNs.

Definition 50 (OCPN)**.** *An* Object-Centric Petri Net (OCPN) *is a tuple* $ON = (N, pt, F_{var})$ *where:*

1. $N = (P, T, F, \lambda)$ *is a labeled Petri net.*
2. $pt : P \to U_{otype}$ *maps places onto object types.*
3. $F_{var} \subseteq F$ *is the set of* variable arcs*.*

In an OCPN:

- *Object Types*: Each place represents a type of object, such as "Order", "Item", or "Customer". Tokens in these places represent specific instances of those objects (e.g., order #123, item #456).
- *Activities/Transitions*: Transitions represent activities in the process, connecting the object-type places.
- *Regular Arcs*: These arcs connect places and transitions and usually represent a one-to-one relationship. For example, an arc from "Order" to "Process Payment" might mean that processing payment requires one order.
- *Variable Arcs*: These arcs are represented by double lines and indicate that an activity can consume or produce *multiple* objects of a certain type. For example:
  - An arc from "Item" to "Pack Order" with a variable arc signifies that the "Pack Order" activity consumes multiple items.
  - Conversely, a variable arc from "Pack Order" to "Package" would indicate that packing an order produces one package.

Non-variable arcs can be traversed by a single token, whereas variable arcs can be traversed by a variable number of tokens, reflecting event data associated with multiple objects. Variable arcs enable the model to capture the many-to-one or one-to-many relationships between objects and activities that are common in real-world processes.

An illustrative example of an OCPN is provided in Figure 6.7.

To properly define the execution semantics of OCPNs, we introduce the concept of *bindings*, which are similar to the bindings in colored Petri nets but adapted to the object references in the event logs.

A token, represented as (p, oi), is located in place p and is associated with the object oi. A *marking* M is a multiset of such tokens.

Definition 51 (Marking)**.** *Let* $ON = (N, pt, F_{var})$ *be an OCPN with* $N = (P, T, F, \lambda)$*.*
$Q_{ON} = \{(p, oi) \in P \times U_{obj} \mid type(oi) = pt(p)\}$ *is the set of possible tokens.*
A marking M *of ON is a multiset of tokens, i.e.,* $M \in \mathcal{B}(Q_{ON})$*.*

The semantics of an OCPN are defined in terms of the movement of tokens across the net based on enabled transitions and their bindings.

Definition 52 (Binding Execution)**.** *Let* $ON = (N, pt, F_{var})$ *be an OCPN with* $N = (P, T, F, \lambda)$*. Let* $tpl(t)$ *denote the set of object types associated with transition* t*. Let* $tpl_{nv}(t)$ *denote the set of object types associated with non-variable arcs of* t*. Let* U_{omap} *be the set of all possible object mappings, where an object mapping* $b \in U_{omap}$ *is a function from object types to sets of object instances, i.e.,* $b : \mathbb{OT} \to 2^{U_{obj}}$*. Let* $type(oi)$ *be a function that returns the object type of an object instance* oi*.*

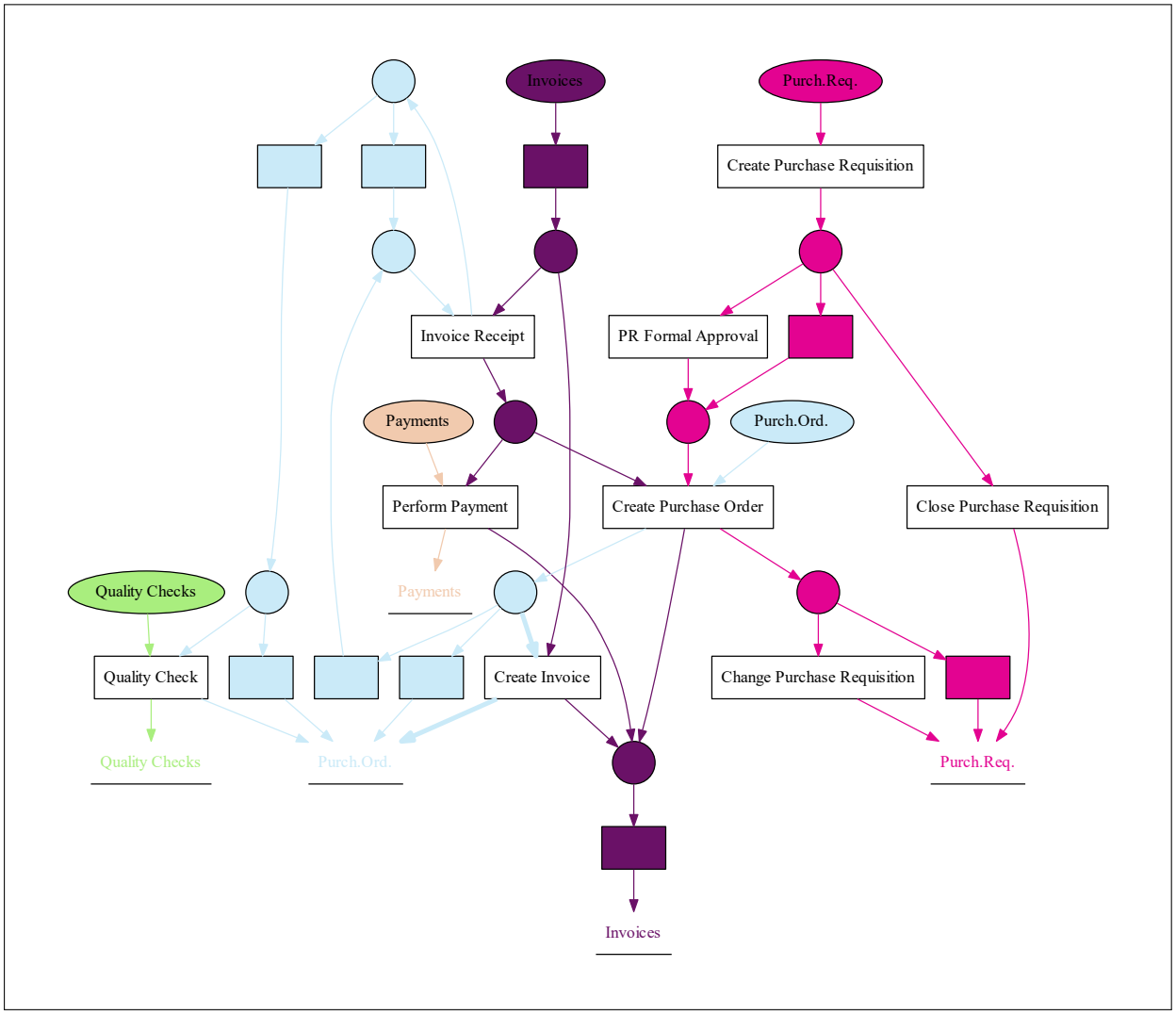


Figure 6.7: An OCPN computed on the example event log provided in Table 3.3 using the Inductive Miner as the underlying process discovery algorithm (no filtering applied).

The set of all possible bindings *is defined as:*

$$B = \{(t, b) \in T \times U_{omap} \mid dom(b) = tpl(t) \ \wedge \ \forall_{ot \in tpl_{nv}(t)} \ |b(ot)| = 1\}$$

Given a binding (t, b)*, we define:*

- $cons(t, b) = [(p, oi) \in Q_{ON} \mid p \in \bullet t \ \wedge \ oi \in b(pt(p))]$,
- $prod(t, b) = [(p, oi) \in Q_{ON} \mid p \in t \bullet \ \wedge \ oi \in b(pt(p))]$.

The binding (t, b) *is* enabled *in marking* $M \in \mathcal{B}(Q_{ON})$ *if* $cons(t, b) \leq M$*. The occurrence of an enabled binding* (t, b) *in marking* M *leads to the new marking* $M' = M - cons(t, b) + prod(t, b)$*. This is denoted as* $M \xrightarrow{(t,b)} M'$*.*

In the above, U_{omap} denotes the set of object mappings b where each object type in $tpl(t)$ is mapped to a set of object identifiers of that type. Non-variable arcs require that exactly one object is consumed or produced, i.e., for $ot \in tpl_{nv}(t)$, $|b(ot)| = 1$.

We define the following for a transition t:

- $pl(t) = \bullet t \cup t \bullet$ are the places connected to t.
- $pl_{var}(t) = \{p \in pl(t) \mid (p, t) \in F_{var} \vee (t, p) \in F_{var}\}$ are the places connected to t through variable arcs.
- $pl_{nv}(t) = pl(t) \setminus pl_{var}(t)$ are the places connected through non-variable arcs.
- $tpl_{var}(t) = \{pt(p) \mid p \in pl_{var}(t)\}$ are the object types of places connected through variable arcs.
- $tpl_{nv}(t) = \{pt(p) \mid p \in pl_{nv}(t)\}$ are the object types of places connected through non-variable arcs.

A *well-formed* OCPN enforces that for any given transition t, the arcs connected to places of the same object type are either all variable or all non-variable.

Definition 53 (Well-Formed OCPN)**.** *An OCPN* $ON = (N, pt, F_{var})$ *is* well-formed *if for each transition* $t \in T$*, the sets of object types associated with variable and non-variable arcs are disjoint, i.e.,* $tpl_{var}(t) \cap tpl_{nv}(t) = \emptyset$*.*

We can define the initial and final markings to establish an accepting OCPN.

Definition 54 (Accepting OCPN)**.** *An accepting OCPN is a tuple* $AN = (ON, M_{init}, M_{final})$*, where* ON *is a well-formed OCPN, and* M_{init}*,* M_{final} *are initial and final markings respectively.*

The behavior of an accepting OCPN can be described by the sequences of visible bindings from the initial marking to the final marking.

Definition 55 (Language of an OCPN)**.** *An accepting OCPN* $AN = (ON, M_{init}, M_{final})$ *defines a language* $\phi(AN) = \{\sigma_v \mid M_{init} \xrightarrow{\sigma} M_{final}\}$*, where* σ_v *is the sequence of visible bindings (transitions with labels) from* M_{init} *to* M_{final}*.*

6.3.2 Discovery of OCPNs

Having established the definitions and execution semantics of OCPNs, we proceed to discuss the discovery of OCPNs from event logs.

While *Object-Centric Directly-Follows Graphs (OCDFGs)* provide the ability to depict the frequency and performance of interconnections between varied activities within an *Object-Centric Event Log (OCEL)*, their capacity to handle concurrency is limited. To address this limitation, we utilize OCPNs for representing business processes involving concurrency.

The discovery of OCPNs involves several primary steps:

- We first *flatten* the OCEL for each object type and apply a standard process discovery algorithm, such as the Inductive Miner.
- A *Token-Based Replay (TBR)* is then performed between the flattened log and the resulting model.

- The discovery of an OCPN involves the amalgamation of the places, transitions, and arcs derived from the collection of Petri nets. While we incorporate the places and the invisible transitions from the object-type-specific Petri nets, we retain only a single visible transition for each label, adding the arcs accordingly.
- To discern the variable arcs, we use the outcomes of the TBR. The arcs connected to places of object type *ot* and the transitions, which according to the TBR results consistently consume one object of the object type *ot*, are regarded as non-variable. Otherwise, they are marked as variable arcs.

This process presents a systematic and efficient approach to the discovery of OCPNs.

Definition 56 (Discovery of OCPNs)**.** *Given an OCEL L, and a process discovery algorithm $disc_{APN}$ that produces an accepting Petri net per object type, we define the discovered OCPN ON as follows:*

- *For each object type ot in L, we flatten L to obtain a traditional event log for ot.*
- *Apply $disc_{APN}$ to each flattened log to obtain an accepting Petri net N_{ot}.*
- *Combine the nets $\{N_{ot}\}$ into a single net N, merging transitions with the same label into a single transition, and retaining places, silent transitions, and arcs.*
- *Define the mapping pt from places to object types based on their origin.*
- *Perform TBR on the flattened logs and the corresponding models to obtain metrics on the usage of tokens.*
- *For each arc f in N, determine whether it is variable based on the TBR metrics: if the number of consumed or produced tokens varies across events, the arc is variable ($f \in F_{var}$); otherwise, it is non-variable.*

Formally, an OCPN $ON = (N, pt, F_{var})$ is constructed where $N = (P, T, F, \lambda)$.

Indeed, the method detailed in Definition 56 is designed to be compatible with any conventional process discovery algorithm, such as the Alpha Miner or the Inductive Miner. We opt to utilize the Inductive Miner due to its robust theoretical underpinnings.

6.3.3 Visualization and Annotation of OCPNs

The visualization of OCPNs can be annotated by frequency and performance metrics obtained during the TBR (an operation needed to identify the variable arcs). The TBR operation returns metrics at the place level (number of missing, consumed, produced, and remaining tokens), associates each transition with the list of corresponding events, and measures the time from the activation of the transition to its firing.

Using this data, we can effectively annotate the OCPN. Figure 6.8 presents an OCPN decorated with frequency information derived from the flattened event logs.

Similarly, Figure 6.9 showcases the same Petri net but annotated with performance metrics.

6.3.4 Conversion to Object-Centric BPMN

Another common procedural modeling notation used in business process management is the *Business Process Model and Notation (BPMN 2.0)* [1]. It is a rich, graphical representation for specifying business processes in a business process model, offering a more extensive set of symbols compared to Petri nets, thereby allowing more precise business communication [65, 64]. However, the expressiveness of BPMN can lead to ambiguity due to the potential for different interpretations of its constructs [42, 92].

An object-centric BPMN diagram has the distinct advantage of presenting the same information as an OCPN but does so using fewer nodes and edges. This reduced complexity not only aids in readability but also streamlines the modeling process. Given this efficiency, it is advantageous to convert an OCPN to an object-centric BPMN diagram. The conversion process entails transforming the Petri nets of individual object types into BPMN diagrams and subsequently collating these diagrams into a cohesive representation.

To illustrate the utility of object-centric BPMN diagrams, Figure 6.10 showcases such a diagram, which is decorated with frequency information derived from flattened event logs.

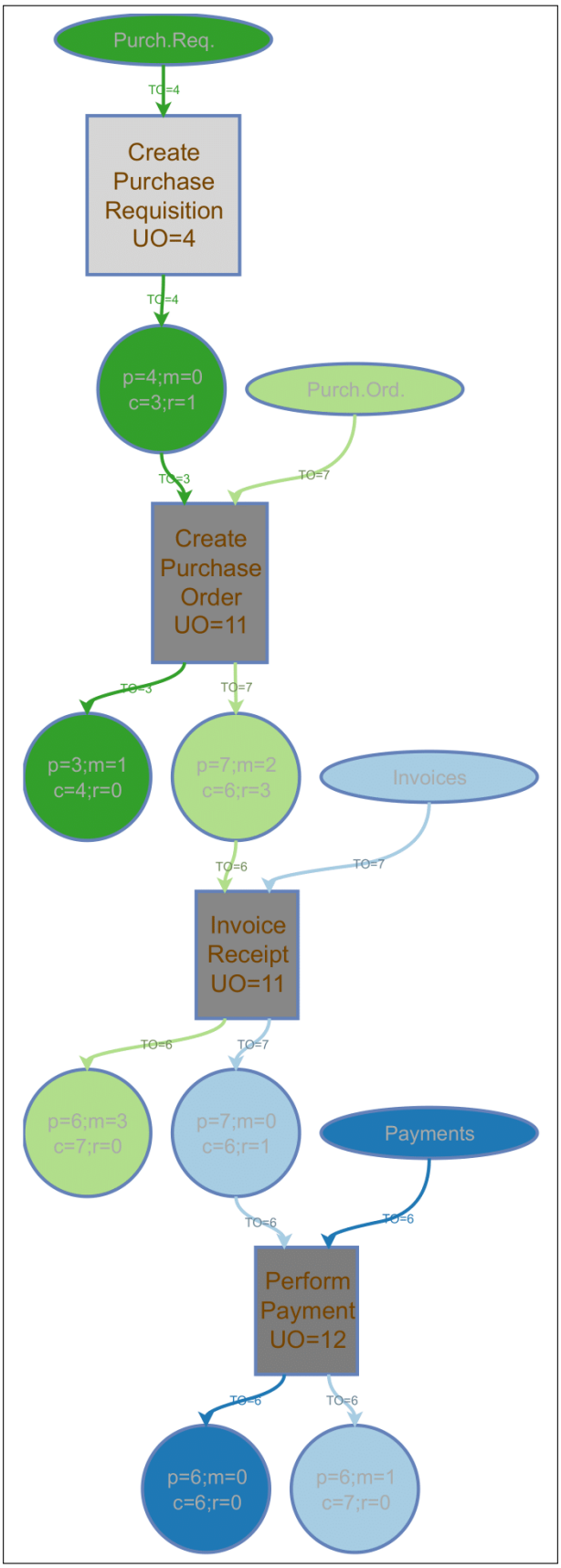


Figure 6.8: OCPN decorated with frequency information (measured on the flattened event logs).

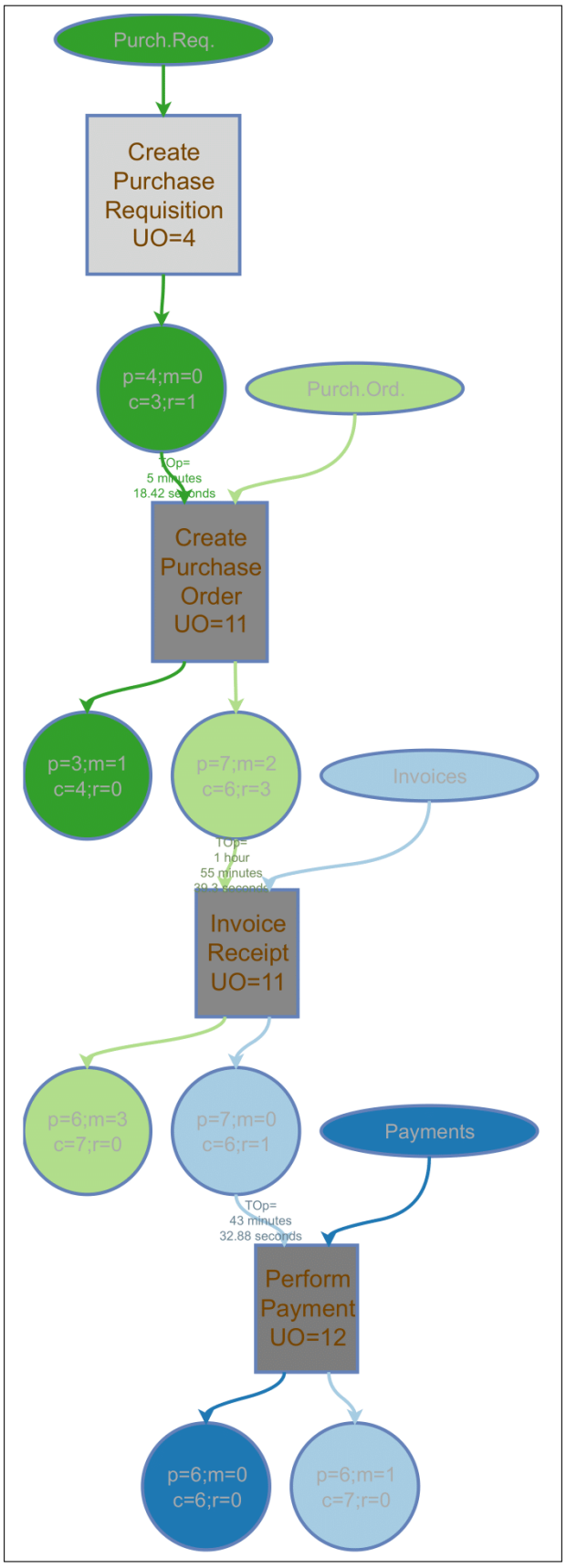


Figure 6.9: OCPN decorated with performance information (measured on the flattened event logs).

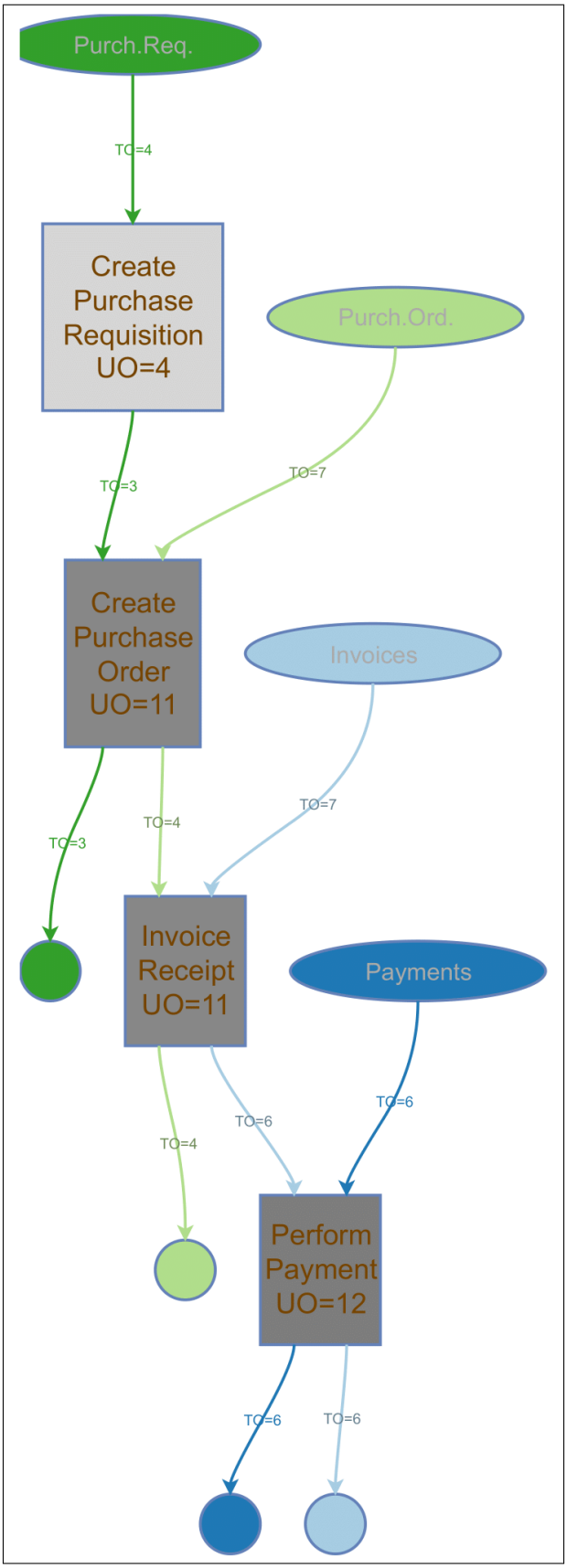


Figure 6.10: Object-centric BPMN diagram decorated with frequency information (measured on the flattened event logs).

Similarly, Figure 6.11 presents an object-centric BPMN diagram adorned with performance metrics. By referencing such metrics, the diagram can shed light on the efficiency and effectiveness of the depicted business processes.

6.3.5 Projection to Traditional Petri Nets

The process of mapping an OCPN to a collection of models for the individual object types is straightforwardly delineated in Definition 57. This operation facilitates the disentanglement of the overarching process into distinct, object-specific models.

Definition 57 (Projection of an Accepting OCPN to a Collection of Traditional Accepting Petri Nets)**.** *Given an accepting OCPN* $AN = ((P, T, F, \lambda), pt, F_{var}, M_{init}, M_{final})$*, and* $OT = \{pt(p) \mid p \in P\}$*, we define* $coll : OT \rightarrow U_{APN}$*, where for each* $ot \in OT$*, the accepting Petri net* $coll(ot) = (P_{ot}, T_{ot}, F_{ot}, \lambda_{ot}, M_{init,ot}, M_{final,ot})$ *is defined as:*

- $P_{ot} = \{p \in P \mid pt(p) = ot\}$,
- $T_{ot} = \{t \in T \mid \exists p \in P_{ot} : (p, t) \in F \vee (t, p) \in F\}$,
- $F_{ot} = F \cap ((P_{ot} \times T_{ot}) \cup (T_{ot} \times P_{ot}))$,
- λ_{ot} *is the restriction of* λ *to* T_{ot},
- *For* $p \in P_{ot}$, $M_{init,ot}(p) = \sum_{oi \in U_{obj}} M_{init}(p, oi)$,
- *For* $p \in P_{ot}$, $M_{final,ot}(p) = \sum_{oi \in U_{obj}} M_{final}(p, oi)$.

6.3.6 Conclusion

In summary, OCPNs provide a powerful formalism for modeling processes involving multiple interacting object types. By incorporating variable arcs and defining precise execution semantics, OCPNs can represent complex real-world processes that involve many-to-one and one-to-many relationships between activities and objects. The discovery of OCPNs from event logs enables analysts to extract accurate representations of such processes, which can be visualized and analyzed using annotations derived from performance and frequency metrics.

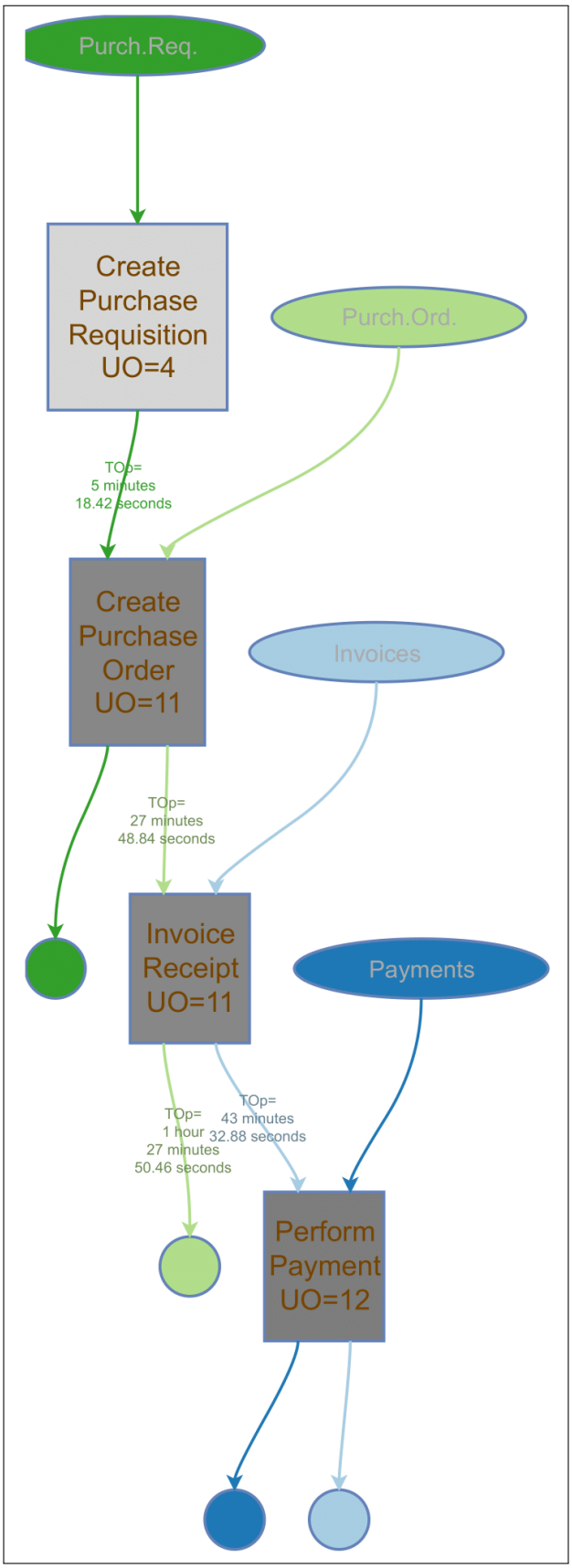


Figure 6.11: Object-centric BPMN diagram decorated with performance information (measured on the flattened event logs).

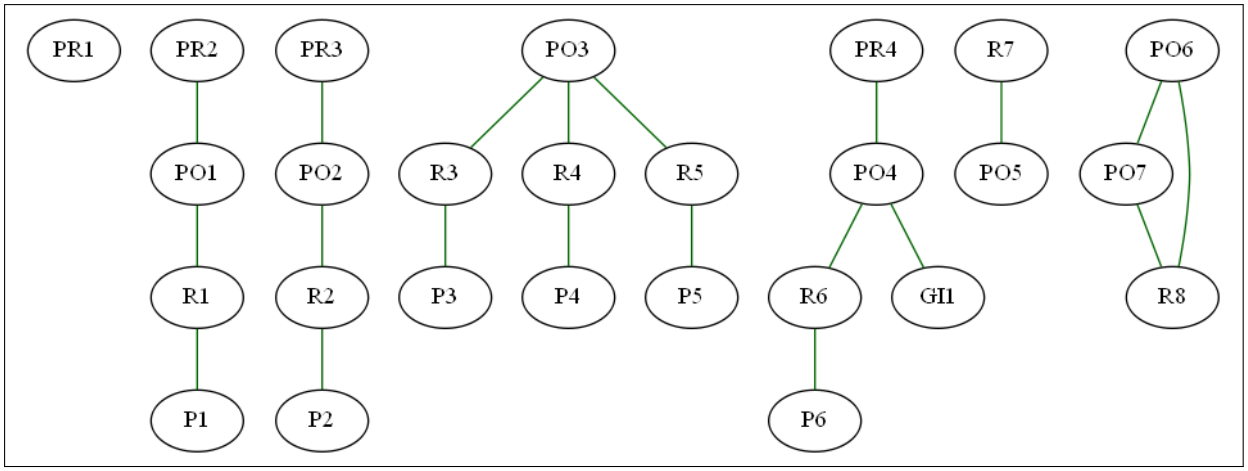


Figure 6.12: Object Interaction Enrichment built on top of the event log described in Table 3.3.

6.4 Discovering Object Graph Enrichments

In process analysis, the relationships between objects can take various forms, each revealing a different aspect of the process dynamics. Therefore, to provide a comprehensive overview of a process, it is essential to explore these different types of object relationships. In the following, we introduce several distinct types of Object Graph Enrichments, each capturing a unique dimension of object interaction within a process. This is done in Definition 58.

Definition 58 (Some Types of Object Graph Enrichments)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$*, we define:*

- *(Object Interaction Enrichment)* $GE_{IG} = (L, \Lambda_{IG})$ *where* $\Lambda_{IG} = \{(o_1, o_2) \in O \times O \mid \exists_{e \in E} o_1, o_2 \in \pi_{omap}(e)\}$.
- *(Object Creation Enrichment)* $GE_{CRG} = (L, \Lambda_{CRG})$ *where* $\Lambda_{CRG} = \{(o_1, o_2) \in \Lambda_{IG} \mid start(o_1) < start(o_2)\}$.
- *(Object Continuation Enrichment)* $GE_{COG} = (L, \Lambda_{COG})$ *where* $\Lambda_{COG} = \{(o_1, o_2) \in \Lambda_{IG} \mid start(o_1) < start(o_2) \wedge end(o_1) = start(o_2)\}$.
- *(Object Cobirth Enrichment)* $GE_{CB} = (L, \Lambda_{CB})$ *where* $\Lambda_{CB} = \{(o_1, o_2) \in \Lambda_{IG} \mid start(o_1) = start(o_2)\}$.
- *(Object Codeath Enrichment)* $GE_{CD} = (L, \Lambda_{CD})$ *where* $\Lambda_{CD} = \{(o_1, o_2) \in \Lambda_{IG} \mid end(o_1) = end(o_2)\}$.

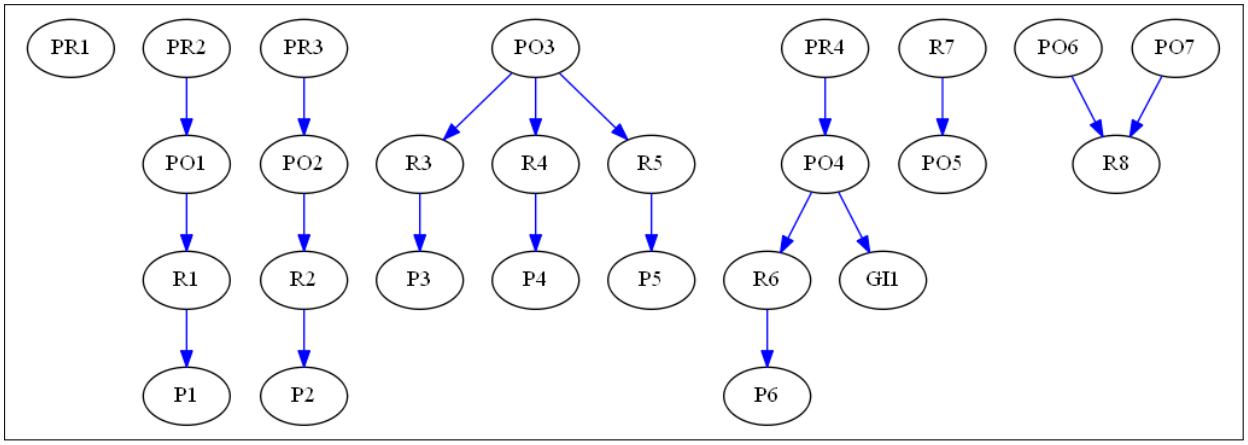


Figure 6.13: Object Creation Enrichment built on top of the event log described in Table 3.3.

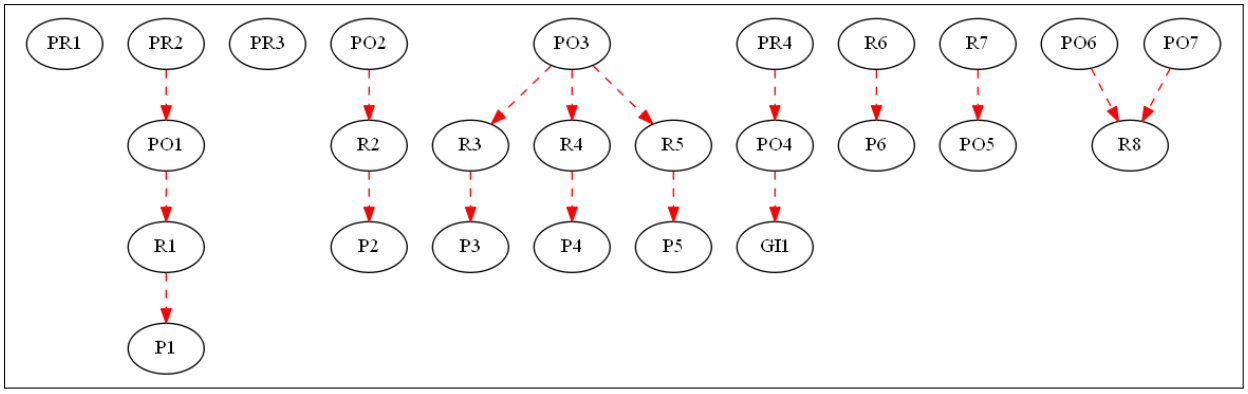


Figure 6.14: Object Continuation Enrichment built on top of the event log described in Table 3.3.

In Definition 58, we introduce several types of Object Graph Enrichments:

- Firstly, we introduce the concept of *Object Interaction Enrichment*. This graph maps out all the objects that are related in the same event, highlighting the direct interactions that can take place within a single event. It gives a sense of the degree of concurrency or interdependence that characterizes the process.
- Next, we consider the *Object Creation Enrichment*. This graph captures the interactions between objects where the lifecycle of one starts before the other. It can give insights into the temporal dependencies between objects, such as the prerequisite relationships that require one object to be initiated before another.
- We also define the *Object Continuation Enrichment*. This graph is a refinement of the Object Creation Enrichment, where the lifecycle of the first object ends exactly when the second one starts. It shows the seamless handovers in the process, marking points where the completion of one task immediately triggers the start of another.
- Additionally, we introduce the concept of an *Object Cobirth Enrichment*. This graph connects objects that start their lifecycle in the same event. It indicates the simultaneous creation of objects, highlighting the points in the process where multiple tasks or activities are launched concurrently.
- Finally, we define the *Object Codeath Enrichment*. This graph links objects that end their lifecycle in the same event. It identifies points in the process where multiple objects are completed at once, indicating synchronized closures of tasks or activities.

Together, these different types of Object Graph Enrichments provide a multi-faceted view of the process, reflecting the diversity of object interactions that can influence process execution and performance. In Figure 6.12, Figure 6.13 and Figure 6.14, we show respectively the object interaction, creation and continuation graphs starting from the event log presented in Table 3.3.

6.4.1 Discovering Object Graph Enrichments for the Purchase-to-Pay Process

In the realm of business process modeling, understanding the intricate relationships between various entities or objects is paramount. These relationships often reflect the operational flow and interdependencies between different stages of a process. As an exemplification of this, we discuss the Purchase-to-Pay process, a fundamental procedure in the business domain which we previously introduced in Section 3.2.1. In the ensuing content, we uncover some of the pivotal object relationships within the Purchase-to-Pay process, categorizing them based on their nature: creation, continuation, and cobirth.

Some Object Creation relationships in the P2P Process

- "Purchase Requisition" → "Purchase Order": A Purchase Requisition is a request or instruction to Purchasing to procure a certain quantity of a material or a service for a certain date. Upon its approval, a Purchase Order is created. The Purchase Requisition lifecycle doesn't conclude until the Purchase Order is created, establishing a creation relationship.

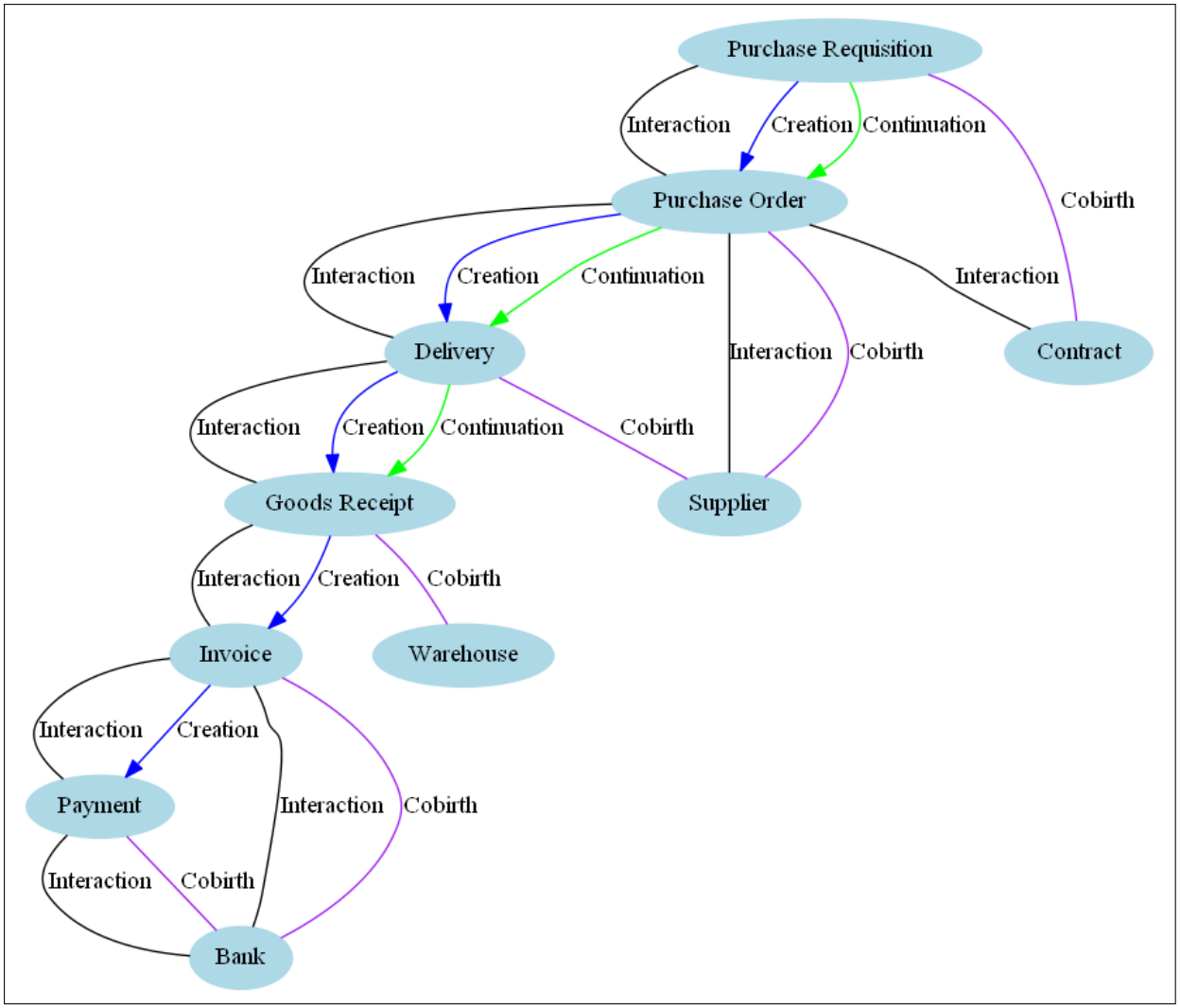


Figure 6.15: Abstraction showing the different relationships between the object types of a Purchase-to-Pay process.

- "Purchase Order" → "Delivery": Once a Purchase Order is issued to a supplier, the next step is the delivery of the goods or services that were ordered. The lifecycle of the Purchase Order continues until the goods or services are delivered, and thus the delivery lifecycle begins, establishing a creation relationship.
- "Delivery" → "Goods Receipt": The Goods Receipt process begins when the ordered goods are delivered. The completion of the delivery (and hence its lifecycle) is not marked until the goods are received and checked, triggering the start of the Goods Receipt lifecycle.
- "Goods Receipt" → "Invoice": Once the goods are received and checked, the supplier issues an invoice. The lifecycle of the Goods Receipt continues until the Invoice is created, thus establishing a creation relationship between the two objects.
- "Invoice" → "Payment": After the invoice is created, the next step is the payment of the invoice. The lifecycle of the Invoice continues until the Payment has been initiated.

In each case, the lifecycle of the first object is not complete until the lifecycle of the second object has begun, which is why these are considered "creation" relationships.

Some Object Continuation relationships in the P2P Process

- "Purchase Requisition" → "Purchase Order": In many organizations, a Purchase Requisition's completion immediately kicks off the creation of a Purchase Order. The Purchase Order effectively

"continues" the intent of the Purchase Requisition, transitioning the request from an internal desire for goods or services to an external order with a supplier.

- "Purchase Order" → "Delivery": The lifecycle of a Purchase Order concludes when the goods or services ordered have been delivered, at which point the Delivery lifecycle starts. The order details in the Purchase Order continue directly into the Delivery, as the Delivery fulfills the requirements laid out in the Purchase Order.
- "Delivery" → "Goods Receipt": Once the Delivery is complete and goods have arrived, the Goods Receipt process begins. This marks the continuation from Delivery to Goods Receipt. The act of receiving the goods is the final act of the Delivery process, and the first act of the Goods Receipt process.

The continuity and direct relationship between these objects indicate a high degree of interdependence.

Some Object Cobirth relationships in the P2P Process

- "Purchase Requisition" → "Contract": In some business environments, when a Purchase Requisition is made, it may coincide with the creation of a Contract, especially if it is a new agreement with the supplier.
- "Purchase Order" → "Supplier": Often, when a Purchase Order is created for the first time for a specific item or service, the Supplier who provides that item or service might be registered (if it is a new supplier) or marked for reference in the system at the same time.
- "Delivery" → "Supplier": When a Delivery is initiated for a new supplier, the Supplier object is also "born" in the system. This may occur when the company is dealing with a supplier for the first time or if it is a new branch of a supplier.
- "Goods Receipt" → "Warehouse": The Warehouse object may be created or designated at the same time as a Goods Receipt is issued, particularly if it is a new warehouse or storage location that is being used for the first time.
- "Invoice" → "Bank": An Invoice could cause the "birth" of a Bank object if it is the first time that a specific bank account is being used for invoicing purposes.
- "Payment" → "Bank": Similarly, the Payment process might coincide with the creation of a Bank object if it is the first time a particular bank or account is being used for making payments.

6.4.2 Discovering Object Graph Enrichments for the Order-to-Cash Process

In Section 3.2.2, we introduced the Order-to-Cash (O2C) process. This section goes deeper into the relationships between the various objects within the O2C workflow. Using Figure 6.16 as a reference, we will categorize and detail these relationships to better understand the overall process.

Some Object Creation relationships in the O2C Process

- "Customer Order" → "Goods Issue": In the O2C process, the customer order is the beginning of the cycle. The customer order includes details like what goods the customer wants, how many they want, and where they should be delivered. Once the customer order is received and validated, it triggers the creation of a goods issue. The goods issue involves taking the requested goods from the inventory and preparing them for delivery. In other words, the customer order directly leads to the creation of the goods issue.
- "Goods Issue" → "Delivery Note": After the goods have been prepared for delivery, a delivery note is created. This note is an essential document that accompanies the goods throughout the transportation process. It includes information like the goods included in the shipment, their quantities, and the destination. Therefore, the process of issuing goods creates the delivery note.
- "Delivery Note" → "Transportation": Once the delivery note is created, transportation is arranged. The delivery note will accompany the goods during transportation and will be used to confirm the delivery once it reaches the customer. In this context, the creation of the delivery note triggers the creation of a transportation order or request.

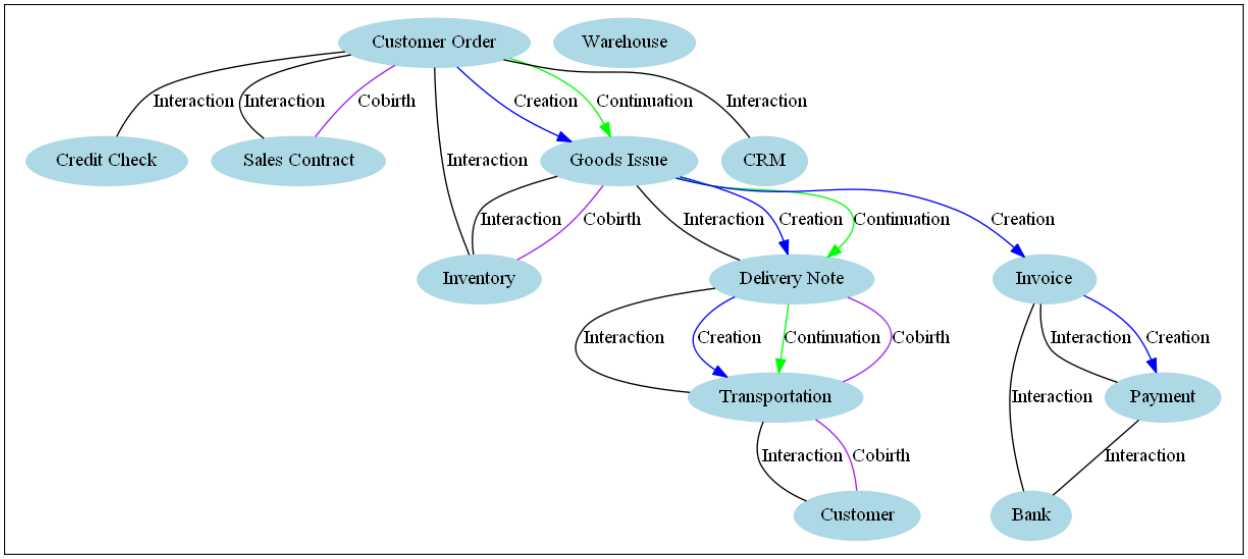


Figure 6.16: Abstraction showing the different relationships between the object types of an Order-to-Cash process.

- "Goods Issue" → "Invoice": At the same time as the goods are being prepared for delivery (or sometimes after delivery confirmation), an invoice is generated. This invoice will detail the cost of the goods, any taxes, delivery fees, and the total amount due from the customer. It is crucial to note that while the invoice is created after the goods issue, it might be sent to the customer at a different time depending on the business's invoicing policies.
- "Invoice" → "Payment": After the invoice has been sent to the customer, it is expected that the customer will make a payment. The invoice thus directly leads to the creation of a payment. Once the payment is received, it will be processed and recorded in the company's financial system.

Some Object Continuation relationships in the O2C Process

- "Customer Order" → "Goods Issue": In the O2C process, the customer order doesn't merely trigger the goods issue; it also sets the stage for the goods issue process. The details within the customer order, such as product descriptions, quantities, and delivery locations, continue to be pertinent during the goods issue process. Thus, the lifecycle of the customer order continues into the goods issue, and the two are directly linked in a sequence.
- "Goods Issue" → "Delivery Note": The information from the goods issue process, like what items are being shipped and their quantities, is vital in creating the delivery note. Therefore, the lifecycle of the goods issue continues into the delivery note. This continuation means that the process isn't complete with just issuing the goods; the detailed documentation in the form of a delivery note also needs to be created.
- "Delivery Note" → "Transportation": The delivery note provides all the necessary details for the transportation of the goods, such as destination and items to be transported. It is used for validation upon delivery. Therefore, the lifecycle of the delivery note continues into the transportation process, indicating that the delivery note is not just an isolated document but a part of the broader transportation and delivery process.

Some Object Cobirth relationships in the O2C Process

- "Customer Order" → "Sales Contract": This cobirth relationship reflects the possibility that a new sales contract may be created concurrently with a customer order. While it is certainly possible to have sales contracts pre-existing or being created independently, there are situations where a unique sales contract may be created for a specific customer order. For example, in the case of a customized order or unique business arrangement, the customer order and sales contract

would be "born" together, marking the first involvement of that specific sales contract in the O2C process. In subsequent operations, the same sales contract may be reused without being in a cobirth relationship with another customer order.

- "Goods Issue" → "Inventory": The cobirth relationship here represents scenarios where a new inventory item is first recognized at the point of goods issue. This might be the case for drop-shipping scenarios or with just-in-time inventory systems where an item enters the company's inventory system only when it is being issued. As the goods issue process commences, the inventory record "comes to life" alongside it. However, once an inventory item has been introduced this way, it may be involved in numerous other transactions, not necessarily tied to a goods issue event.
- "Delivery Note" → "Transportation": This cobirth relationship acknowledges that a specific transportation event might be initiated in sync with the creation of a delivery note. A delivery note generally prompts the physical transfer of goods, thus the specific act of transportation tied to this delivery note comes into existence simultaneously. It is important to note that while the transportation entity is born with the delivery note, the same transportation resource (like a vehicle or a shipping partner) may participate in many such events, not always related to a cobirth context.
- "Transportation" → "Customer": The cobirth relationship here is indicative of the possibility that a new customer's first interaction with the company is through a delivery, i.e., a transportation event. This could happen in case of direct delivery models where the product is delivered to the customer as soon as they place the order. In such a scenario, the customer and the transportation event are "born" together in the company's O2C process. However, this doesn't prevent the customer from having more interactions with the company that are not directly tied to a transportation event.

6.5 Methodological Framework for OCPD

We initiate with a deep dive into the algorithmic choices available for OCPD in Section 6.5.1. Here, a comparative analysis between the OCDFGs and OCPNs is laid out, presenting the strengths, weaknesses, and contextual applicability of each. Following this, Section 6.5.2 focuses on the pivotal step of selecting the most pertinent object types, emphasizing criteria that ensure the resultant process models are both meaningful and insightful.

6.5.1 Choice of the OCPD Algorithm

Characteristic/Performance Metric	**OCDFGs**	**OCPNs**
Scalability	Superior due to the lightweight computation of Directly-Follows Graphs for an object type.	Moderate, as computations can be intensive, especially when employing complex discovery algorithms and TBR.
Handling Concurrency	Limited effectiveness, especially when processes are highly concurrent.	Robust mechanism to manage concurrency.
Frequency and Performance Metrics Calculation	Straightforward and direct calculations.	Requires TBR operation, making it more complex and potentially resource-intensive.
Process Discovery Phase	Efficient due to inherent simplicity.	Could be computationally demanding, sometimes rendering it inefficient for certain applications.

Table 6.1: Comparison between OCDFGs and OCPNs

In the realm of OCPD, practitioners often struggle with the decision of which algorithmic approach to employ.

The *OCDFGs* exhibit considerable advantages when it comes to scalability. Given that the computational overhead of deriving a Directly-Follows Graph for a specific object type is relatively minimal, this methodology is particularly beneficial for scenarios with large datasets or when rapid iterative analysis is essential. Furthermore, when it comes to metrics—specifically frequency and performance—the calculations within this framework are direct and uncomplicated, allowing for straightforward interpretations and actionable insights.

The primary limitation of the OCDFGs is their somewhat constrained ability to handle concurrency effectively. As processes become more intertwined and concurrent operations become prevalent, the clarity and accuracy of this approach may diminish.

On the other hand, the *OCPNs* bring a different set of attributes to the table. While they provide a robust mechanism to address concurrency, this strength comes at a cost. Evaluating frequency and performance metrics within the ambit of an OCPN is far from straightforward. It necessitates a TBR operation—a computational endeavor that can be quite resource-intensive. To make matters more challenging, when attempting to apply well-established process discovery algorithms like the inductive miner to this framework, the computational demands can surge, rendering the approach inefficient for certain applications.

In summation, the choice between these two methodologies should be guided by the specific requirements of the analytical task at hand. For scenarios where scalability and rapid computations are paramount, the OCDFGs might be more fitting. However, when the intricate handling of concurrency is a core concern, despite the computational overhead, the OCPNs may emerge as the more prudent choice. Table 6.1 summarizes these findings.

6.5.2 Selection of the Relevant Object Types

Selecting the appropriate object types from an event log for process discovery is a crucial step in Object-Centric Process Mining (OCPM). This selection aims at identifying the object types for which process discovery will yield meaningful and insightful models. The decision is multifaceted and goes beyond simple frequency counts, demanding a holistic understanding of the event log's structure and the underlying process dynamics. Here are some criteria to consider:

1. *Activity Frequency*: An object type should be included if it is involved in a sufficient number of activities within the log. Too few activities might result in a simplistic or uninteresting process model, while extremely high frequencies might indicate noise or overly generic behavior. A balance must be struck, potentially using frequency thresholds or statistical analysis (e.g., considering the distribution of activity counts across object types).

2. *Interaction with other Objects*: If an object type is heavily interconnected with other object types, it can be a significant factor in process behavior and thus should be included in the process discovery. This interaction can be further analyzed by:

    - *Object-to-Object Relationships*: Analyze direct relationships between objects, such as compositions (e.g., an 'order' object containing 'order item' objects), dependencies (e.g., a 'task' object depending on a 'resource' object), or associations (e.g., a 'customer' object related to multiple 'orders'). High interconnectedness suggests a central role in the process.
    - *Event-to-Object Relationships*: Consider how events link different object types. For example, an 'order shipped' event might connect 'order', 'shipment', and 'customer' objects. Examining the cardinality and types of these relationships (one-to-one, one-to-many, many-to-many) provides insights into the process flow and object dependencies.

3. *Lifecycle Analysis*: Objects that exhibit clear lifecycle stages (create, update, complete, etc.) can make a substantial contribution to the discovery of a process model. The stages in an object's lifecycle often map directly to process steps, making these objects ideal candidates for inclusion. The presence of well-defined start and end events is a strong indicator. This analysis should also consider:

    - *Completeness of Lifecycles*: Incomplete lifecycles might indicate data quality issues or process exceptions. Assessing the proportion of completed lifecycles helps determine the reliability of the object type for process discovery.
    - *Lifecycle Variability*: Different instances of the same object type might follow different lifecycle paths. Understanding this variability is crucial for capturing the full complexity of the process.

4. *Domain Relevance*: Some objects might be more important based on domain-specific knowledge. It is advisable to include object types that are considered significant within the context of the business process being studied, even if their frequency or interaction metrics are not exceptionally high. This requires close collaboration with domain experts.

5. *Variability in Behavior*: Object types that exhibit variability in behavior across different instances can be valuable for discovering different process paths or alternative sequences of activities. If an object type is always involved in the same sequence of activities, it might not add much information to the process model. Conversely, high variability can also indicate noise or lack of process standardization. Techniques like entropy measures or clustering of object behavior can help quantify variability.

6. *Temporal Dynamics*: Object types that have significant temporal properties, such as time constraints or patterns in activity sequences, can be valuable to include, as they may provide insight into process timing and scheduling aspects. This includes:

    - *Inter-arrival Times*: Analyzing the time between events related to an object type can reveal bottlenecks or inefficiencies.
    - *Activity Durations*: The time an object spends in certain states or activities can be indicative of process performance.
    - *Temporal Dependencies*: Identifying temporal constraints or dependencies between activities involving different objects (e.g., object A must be completed before object B can start).

7. *Event Types and Event Type Signatures*: The types of events associated with an object type are crucial. Some event types might indicate critical process steps, while others might be less informative.

    - *Event Type Significance*: Prioritize object types associated with events that mark significant milestones or transitions in the process.
    - *Event Type Diversity*: A diverse set of event types for an object type might indicate a richer and more complex process behavior.
    - *Event Type Signatures (Event Type per Object Type Matrix)*: Analyzing the co-occurrence of event types and object types can reveal important relationships. This can be represented as a matrix where rows are event types, columns are object types, and entries indicate the presence or frequency of associations. This matrix can highlight central object types and the events that shape their behavior. Analyzing this matrix can help to see which object types share similar event signatures and potentially aggregate them or identify those that are unique and thus process-relevant.

8. *Data Quality Considerations*: The quality of the event data associated with an object type can significantly impact the reliability of the discovered process model.

    - *Completeness*: Object types with incomplete or missing data should be treated with caution.
    - *Consistency*: Inconsistencies in event data or object attributes can lead to inaccurate process models.
    - *Noise*: High levels of noise or irrelevant events associated with an object type can obscure the underlying process.

Selecting the right object types is an iterative process that often involves experimentation and refinement. It is recommended to start with a broader set of object types and then narrow it down based on the criteria mentioned above, potentially using a combination of automated techniques and domain expertise. The goal is to identify a set of object types that provides a comprehensive and insightful view of the underlying process, while avoiding unnecessary complexity and noise.

6.6 Assessment

In the endeavor to gauge the efficiency and scalability of various methods under the OCPD umbrella, an assessment was conducted, centered around their execution times.

The experiments were executed on a notebook with an I7-7500U CPU, 16 GB DDR4 RAM, and the PM4Py 2.7.5.1 library (introduced in Chapter 9). Still, we are confident that the findings are generic and are also applicable to other configurations/tools.

6.6.1 Execution time of the OCDFG's Discovery

The results for the discovery of the OCDFG can be found in Table 6.2 and are visually represented in Figure 6.17.

In our observations, the complexity of this discovery method exhibited a linear growth pattern with respect to several factors: the number of events, objects, object types, and the Event-to-Object relationships within the OCEL. Interestingly, when it came to the number of activities, the complexity showcased a logarithmic growth. These findings suggest a relatively scalable performance, especially when one considers the multifaceted nature of object-centric processes. In terms of absolute execution times, the method performed admirably, yielding good execution times that make it a practical choice for real-world applications.

Table 6.2: Execution times for the (complete) discovery of the OCDFG.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79334	50	50	9.9989679
100000	10000	158529	50	50	19.3358028
200000	10000	316252	50	50	40.6315356
250000	10000	394532	50	50	47.7114672
300000	10000	474015	50	50	58.2313785
Varying the number of objects					
10000	50000	15767	50	50	7.4006251
10000	100000	15769	50	50	5.8893235
10000	200000	15818	50	50	7.4490436
10000	250000	15750	50	50	8.3237062
10000	300000	15748	50	50	9.2625536
Varying the number of object types					
10000	10000	15686	50	50	2.2700013
10000	10000	15824	100	50	2.5989102
10000	10000	15674	200	50	3.2902072
10000	10000	15788	300	50	3.9549183
Varying the number of activities					
10000	10000	16005	50	50	2.2849185
10000	10000	15893	50	100	2.2755908
10000	10000	15858	50	200	2.2950811
10000	10000	15670	50	300	2.4775128
Varying the number of related objects					
10000	10000	15873	50	50	2.4208826
10000	10000	25254	50	50	2.4780450
10000	10000	35032	50	50	2.6884402
10000	10000	45758	50	50	2.9693739
10000	10000	55654	50	50	3.6149181

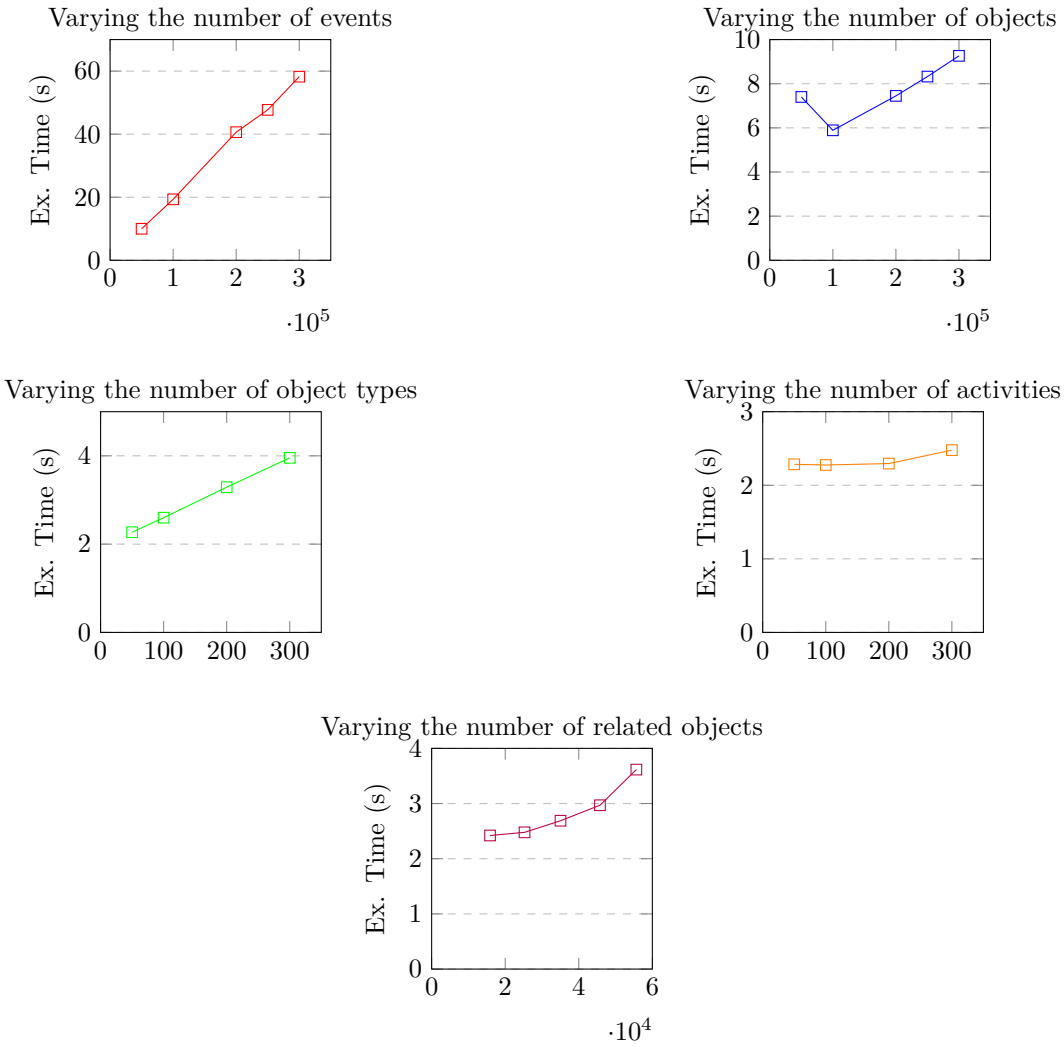


Figure 6.17: Execution times for the (complete) discovery of the OCDFG.

6.6.2 Execution time of the OCPN Discovery

Proceeding to the discovery of the OCPN without the TBR, the empirical data is consolidated in Table 6.3, with a complementary graphical representation in Figure 6.18.

Again, our findings indicated that the complexity in this method grows linearly across all considered dimensions of the OCEL. This consistent linear trend underscores the method's robustness, irrespective of the intricate interplays and relationships in the data. The execution times, in line with the previous method, were found to be commendable, further solidifying the utility of this approach in various application scenarios.

Finally, when we shifted our attention to the annotation of the OCPN via TBR, the results were tabulated in Table 6.4, and the trends were graphically depicted in Figure 6.19.

The experiments for this method unveiled a mixed bag of results in terms of complexity. Notably, the complexity skyrocketed exponentially with the number of events within the OCEL. However, a more predictable linear growth was observed concerning the number of objects, object types, Event-to-Object relationships, and activities. This exponential trend, particularly for events, emphasizes the need for caution when dealing with large event logs, as the computational demands could be steep.

Table 6.3: Execution times for the discovery of an OCPN.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79244	50	50	8.6180656
100000	10000	158281	50	50	14.8746944
200000	10000	316145	50	50	24.1265743
250000	10000	395529	50	50	34.6905827
300000	10000	474519	50	50	42.4510047
Varying the number of objects					
10000	50000	15804	50	50	4.5960451
10000	100000	15854	50	50	5.6567329
10000	200000	15851	50	50	8.7660005
10000	250000	15808	50	50	9.6429751
10000	300000	15825	50	50	10.9071395
Varying the number of object types					
10000	10000	15798	50	50	4.911101
10000	10000	15833	100	50	6.756428
10000	10000	15831	200	50	9.3146081
10000	10000	15861	300	50	11.5613695
Varying the number of activities					
10000	10000	16023	50	50	5.3901469
10000	10000	15962	50	100	7.3736042
10000	10000	16025	50	200	10.7486131
10000	10000	15853	50	300	10.3408823
Varying the number of related objects					
10000	10000	15693	50	50	4.7351841
10000	10000	25394	50	50	4.7984414
10000	10000	35701	50	50	5.3273096
10000	10000	45340	50	50	4.8957335
10000	10000	54353	50	50	7.8225067

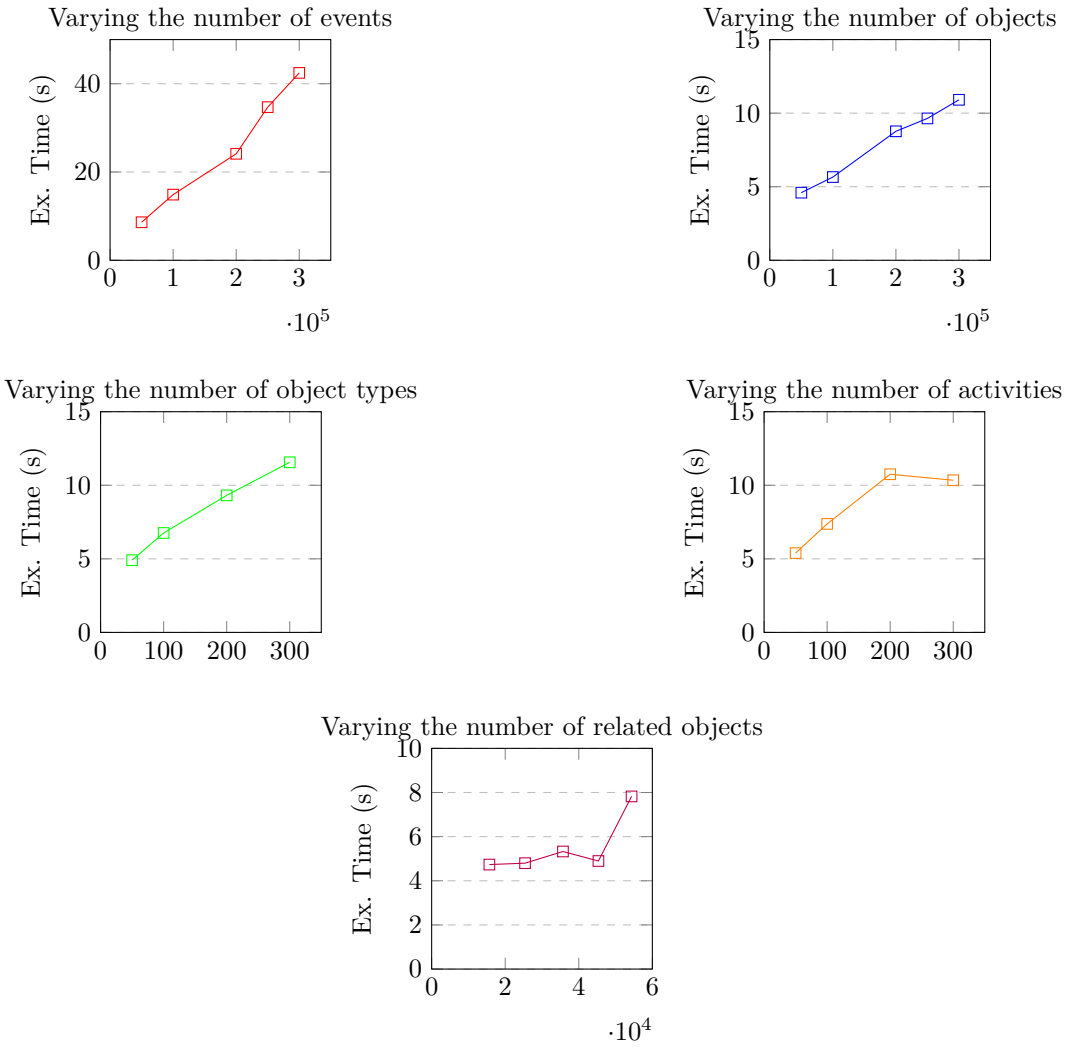


Figure 6.18: Execution times for the discovery of an OCPN.

Table 6.4: Execution times for the annotation (via TBR) of an OCPN.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	78676	50	50	28.4022595
100000	10000	158599	50	50	58.1410163
200000	10000	316480	50	50	142.6987869
250000	10000	394530	50	50	216.4297574
300000	10000	473741	50	50	456.2423821
Varying the number of objects					
10000	50000	16009	50	50	7.4174876
10000	100000	15860	50	50	7.4331230
10000	200000	15757	50	50	9.8557481
10000	250000	15861	50	50	10.9753322
10000	300000	15939	50	50	13.0213402
Varying the number of object types					
10000	10000	15906	50	50	9.4877269
10000	10000	15793	100	50	10.3810137
10000	10000	15854	200	50	12.9235588
10000	10000	15860	300	50	16.2318602
Varying the number of activities					
10000	10000	15827	50	50	8.7196110
10000	10000	15800	50	100	12.0125144
10000	10000	15745	50	200	17.6411984
10000	10000	15683	50	300	18.3971866
Varying the number of related objects					
10000	10000	15785	50	50	9.6429870
10000	10000	25384	50	50	10.4247165
10000	10000	34680	50	50	13.5367084
10000	10000	45392	50	50	15.0955733
10000	10000	55287	50	50	18.4198908

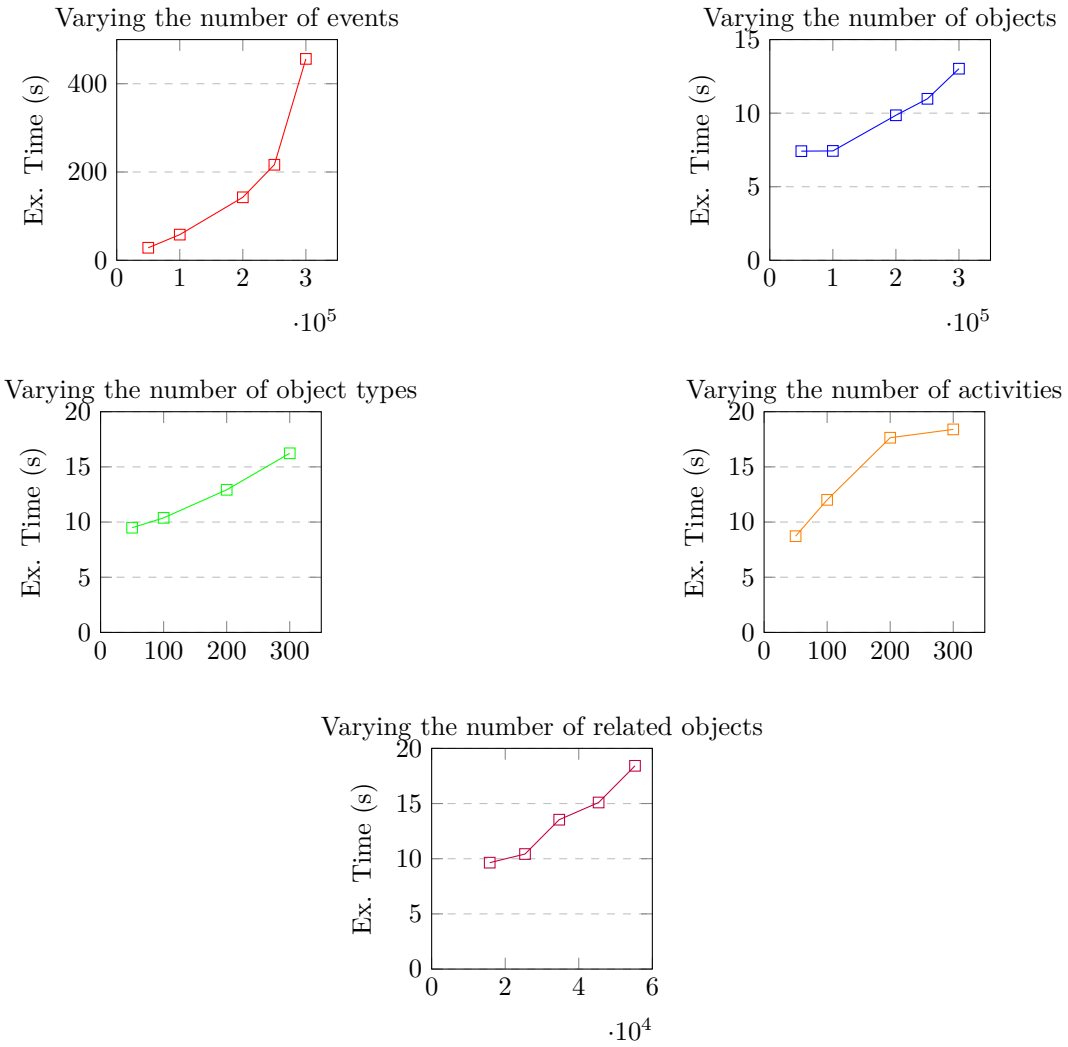


Figure 6.19: Execution times for the annotation (via TBR) of an OCPN.

6.6.3 Execution time of the Object Graph Enrichments Discovery

In the quest to understand the behavior and interactions of various Object Graph Enrichments, a series of experiments were undertaken. These studies were primarily focused on the relationship between execution times and the number of events, alongside other influencing factors like the number of objects, the types of these objects, the activities involved, and the Event-to-Object relationships.

Central to our findings is the observation that the execution time consistently grows in a linear fashion with the rise in the number of events. This behavior is attributed to the PM4Py implementation that necessitates an iteration over the set of events to compute the respective graphs. Interestingly, while the number of events played a pivotal role in determining the computation time, other parameters such as the quantity of objects, their types, specific activities, and Event-to-Object relationships seemed to exert a comparatively milder influence on the computation duration.

Delving deeper into the specifics:

- The execution times for the discovery of the Object Interaction Enrichment can be referred to in Table 6.5 and its corresponding visual representation in Figure 6.20.
- For the Object Creation Enrichment, the data is tabulated in Table 6.6, and its graphical elucidation is showcased in Figure 6.21.
- The results concerning the Object Continuation Enrichment are captured in Table 6.7 and Figure 6.22.
- For those keen on the nuances of the Object Cobirth Enrichment, pertinent data is housed in Table 6.8 with its graphical counterpart found in Figure 6.23.
- Lastly, the Object Codeath Enrichment yields insights presented in Table 6.9 and graphically delineated in Figure 6.24.

Table 6.5: Execution times for the discovery of the Object Interaction Enrichment.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79320	50	50	1.0632479
100000	10000	158470	50	50	2.1910442
200000	10000	316336	50	50	4.4199972
250000	10000	395239	50	50	5.8346664
300000	10000	475074	50	50	6.9346699
Varying the number of objects					
10000	50000	15807	50	50	0.2128277
10000	100000	15692	50	50	0.2418600
10000	200000	16019	50	50	0.2373974
10000	250000	15842	50	50	0.2077019
10000	300000	15832	50	50	0.2415340
Varying the number of object types					
10000	10000	15721	50	50	0.2393621
10000	10000	15666	100	50	0.2317085
10000	10000	15879	200	50	0.2115367
10000	10000	15632	300	50	0.2394102
Varying the number of activities					
10000	10000	15714	50	50	0.2430799
10000	10000	15780	50	100	0.2054818
10000	10000	15743	50	200	0.2324343
10000	10000	15932	50	300	0.2424954
Varying the number of related objects					
10000	10000	15782	50	50	0.2363361
10000	10000	25584	50	50	0.2142516
10000	10000	35320	50	50	0.2722809
10000	10000	44939	50	50	0.2743355
10000	10000	54084	50	50	0.3294501

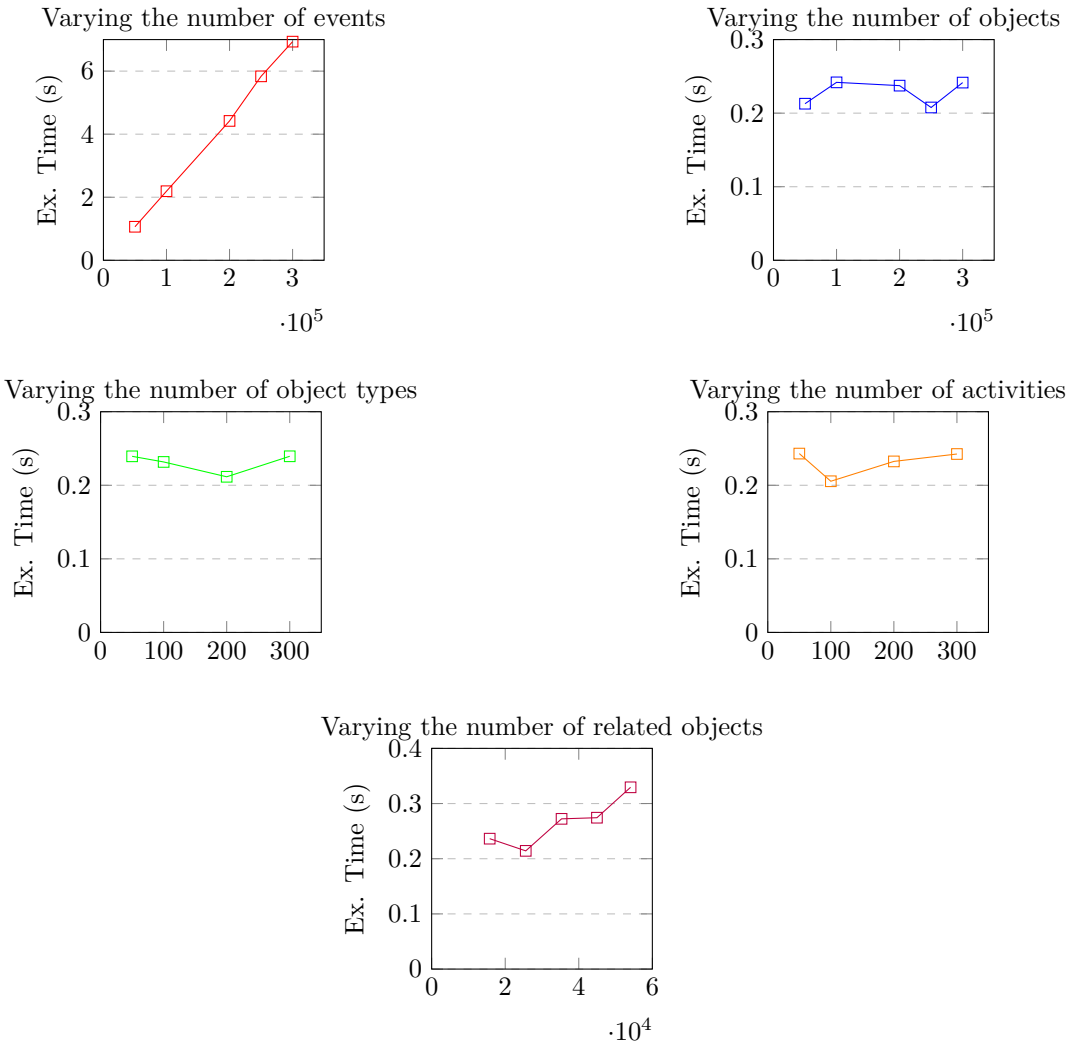


Figure 6.20: Execution times for the discovery of the Object Interaction Enrichment.

Table 6.6: Execution times for the discovery of the Object Creation Enrichment.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79055	50	50	1.1416385
100000	10000	158657	50	50	2.3744270
200000	10000	316556	50	50	4.6282279
250000	10000	395214	50	50	5.5803505
300000	10000	474363	50	50	6.9704797
Varying the number of objects					
10000	50000	15783	50	50	0.2485605
10000	100000	15685	50	50	0.2393862
10000	200000	15765	50	50	0.2206899
10000	250000	15877	50	50	0.2499782
10000	300000	15709	50	50	0.2144595
Varying the number of object types					
10000	10000	15727	50	50	0.2174273
10000	10000	15812	100	50	0.2430007
10000	10000	15815	200	50	0.2573430
10000	10000	15749	300	50	0.2159478
Varying the number of activities					
10000	10000	15812	50	50	0.2479893
10000	10000	15650	50	100	0.2473708
10000	10000	15675	50	200	0.2153236
10000	10000	15911	50	300	0.2404341
Varying the number of related objects					
10000	10000	16028	50	50	0.2183837
10000	10000	25175	50	50	0.2512068
10000	10000	35383	50	50	0.2503241
10000	10000	44958	50	50	0.2623304
10000	10000	55228	50	50	0.2373689

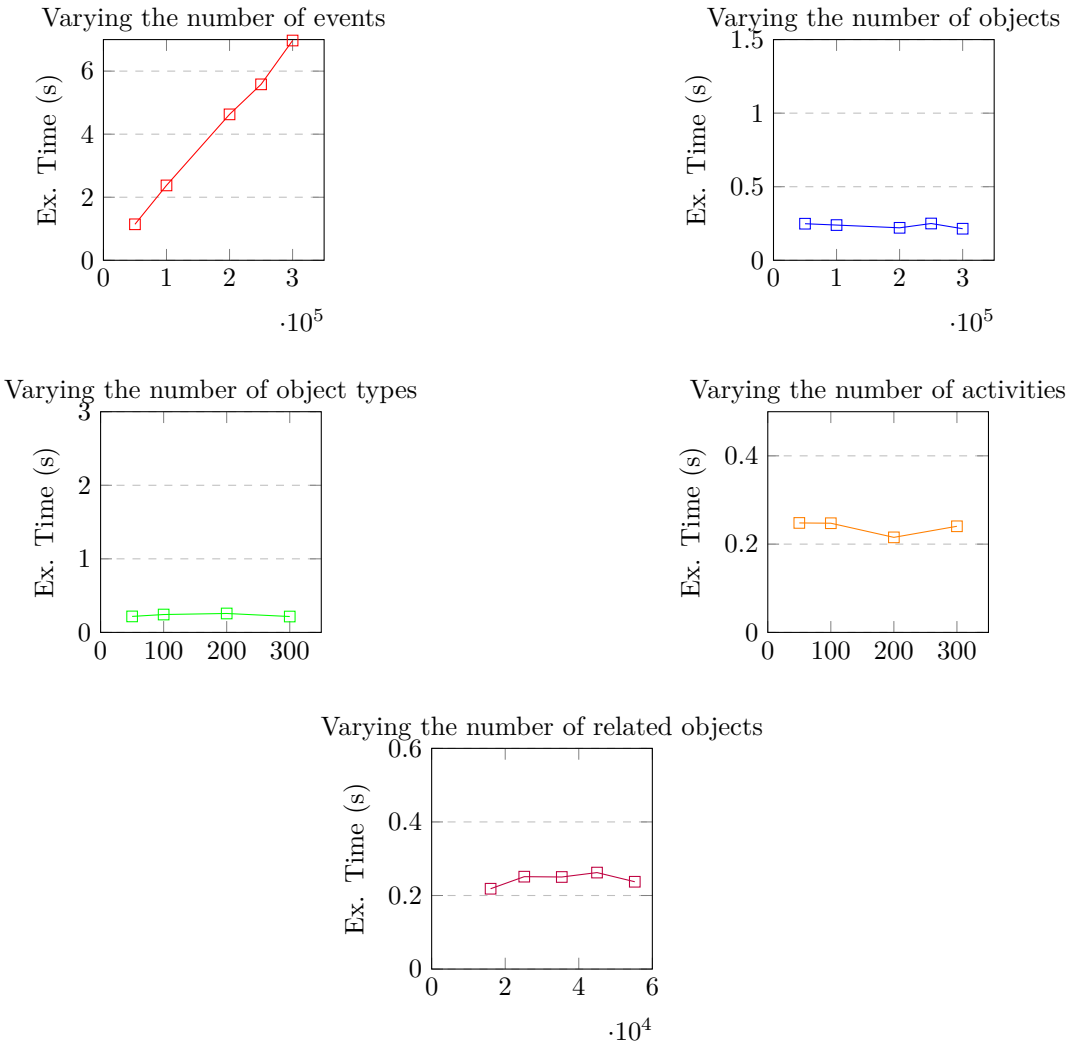


Figure 6.21: Execution times for the discovery of the Object Creation Enrichment.

Table 6.7: Execution times for the discovery of the Object Continuation Enrichment.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79060	50	50	1.8611197
100000	10000	158581	50	50	2.5878750
200000	10000	316002	50	50	5.1607249
250000	10000	395602	50	50	6.9657898
300000	10000	474370	50	50	8.0419183
Varying the number of objects					
10000	50000	15851	50	50	0.2632948
10000	100000	15876	50	50	0.2558091
10000	200000	15650	50	50	0.4168834
10000	250000	15925	50	50	0.2783517
10000	300000	15859	50	50	0.3059711
Varying the number of object types					
10000	10000	15857	50	50	0.237366
10000	10000	15741	100	50	0.2643976
10000	10000	15871	200	50	0.2404972
10000	10000	15865	300	50	0.225401
Varying the number of activities					
10000	10000	15732	50	50	0.2712406
10000	10000	15715	50	100	0.2738194
10000	10000	15833	50	200	0.2350025
10000	10000	15813	50	300	0.2674970
Varying the number of related objects					
10000	10000	15818	50	50	0.2722514
10000	10000	25485	50	50	0.2533215
10000	10000	35418	50	50	0.2836054
10000	10000	45086	50	50	0.2943195
10000	10000	54975	50	50	0.2762306

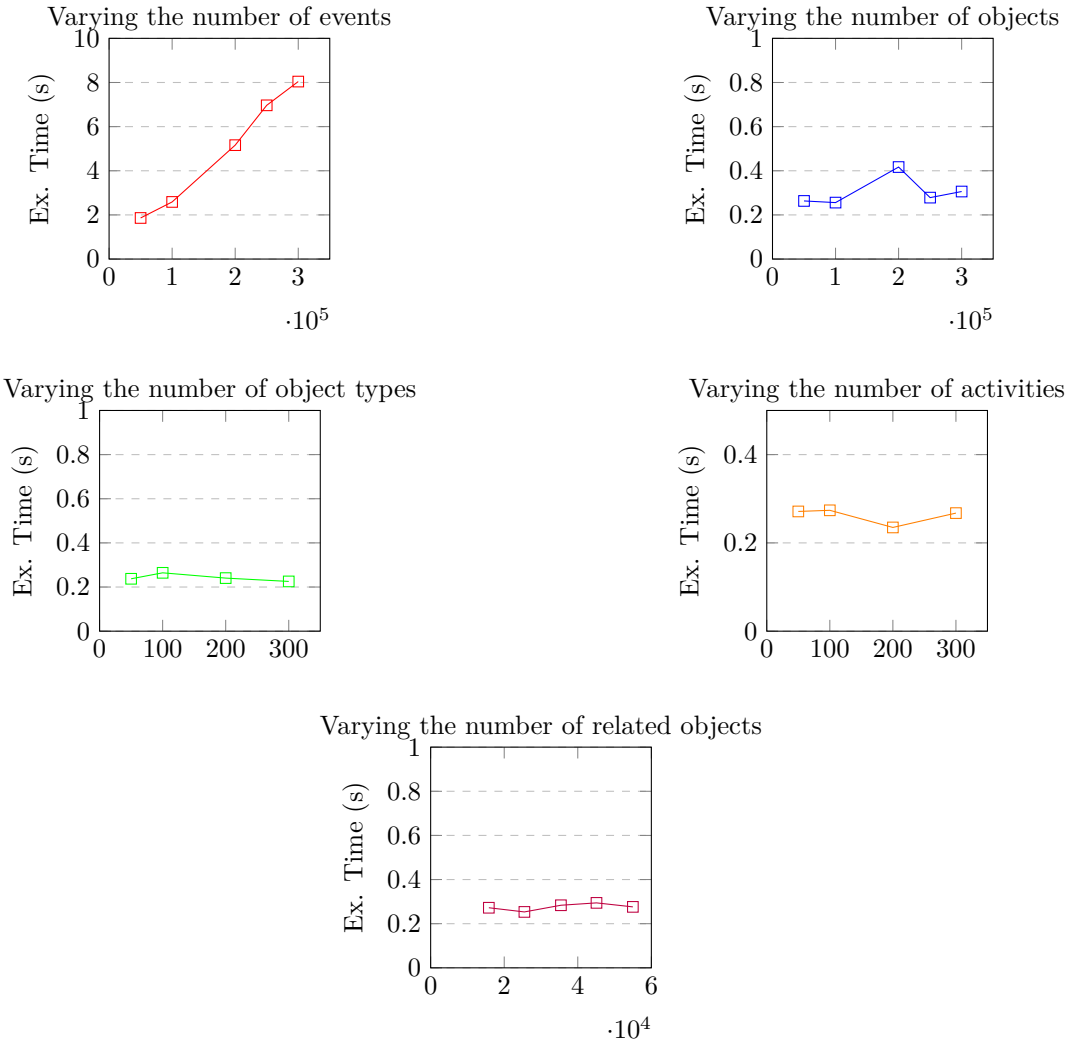


Figure 6.22: Execution times for the discovery of the Object Continuation Enrichment.

Table 6.8: Execution times for the discovery of the Object Cobirth Enrichment.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	78974	50	50	1.1838358
100000	10000	158173	50	50	2.3153614
200000	10000	316417	50	50	4.7558060
250000	10000	395634	50	50	5.9624541
300000	10000	473836	50	50	7.1723412
Varying the number of objects					
10000	50000	15823	50	50	0.2383703
10000	100000	15725	50	50	0.2184494
10000	200000	15810	50	50	0.2256589
10000	250000	15683	50	50	0.2368044
10000	300000	15764	50	50	0.2563428
Varying the number of object types					
10000	10000	15842	50	50	0.2311399
10000	10000	15735	100	50	0.2553477
10000	10000	15929	200	50	0.2323784
10000	10000	15932	300	50	0.2523271
Varying the number of activities					
10000	10000	15857	50	50	0.2742662
10000	10000	15747	50	100	0.2652889
10000	10000	15846	50	200	0.2373665
10000	10000	15891	50	300	0.2593147
Varying the number of related objects					
10000	10000	15735	50	50	0.2689829
10000	10000	25800	50	50	0.2353707
10000	10000	35298	50	50	0.2673202
10000	10000	45571	50	50	0.2772591
10000	10000	55510	50	50	0.2539511

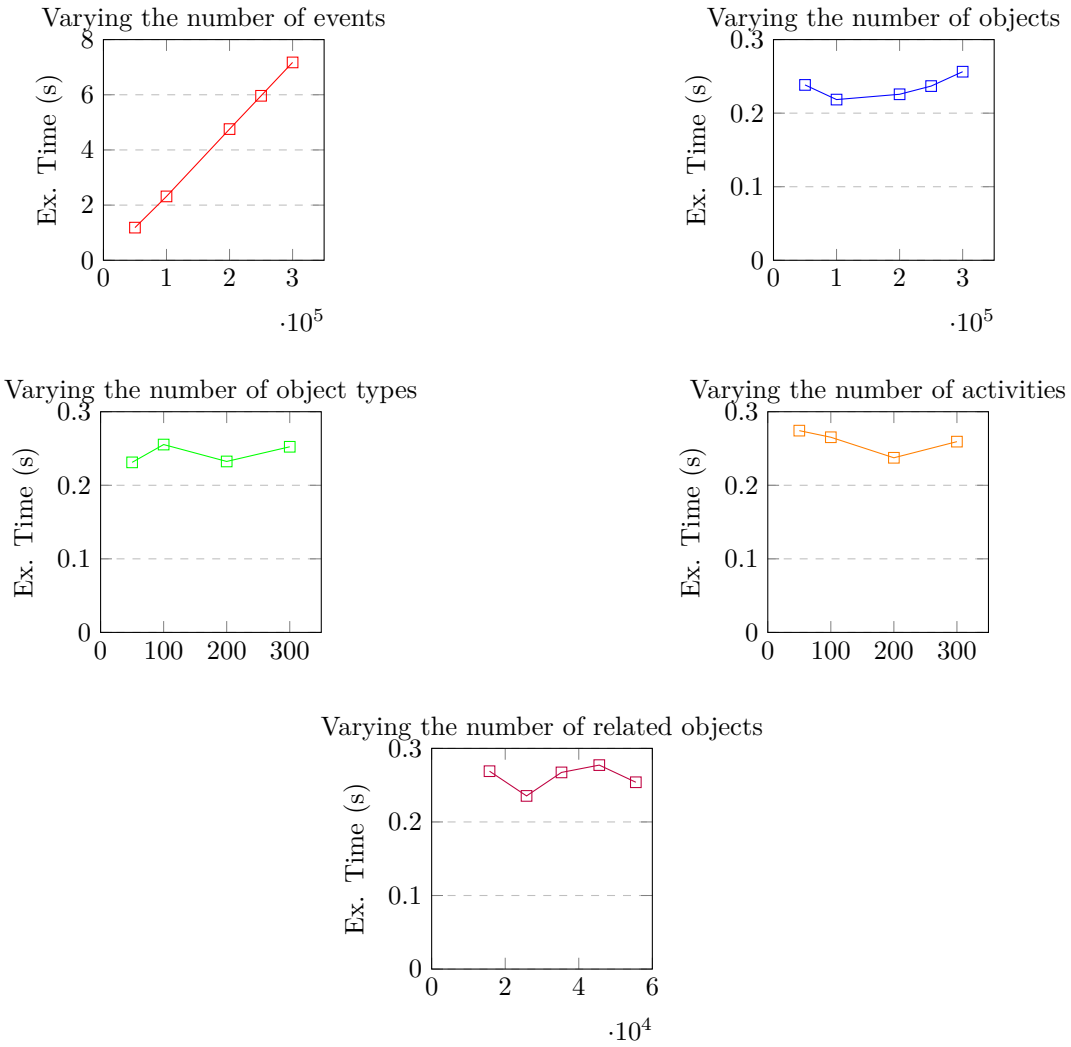


Figure 6.23: Execution times for the discovery of the Object Cobirth Enrichment.

Table 6.9: Execution times for the discovery of the Object Codeath Enrichment.

# Events	# Objects	# E-O Relations	# Object Types	# Activities	Execution Time (s)
Varying the number of events					
50000	10000	79070	50	50	1.1752827
100000	10000	158250	50	50	2.3627996
200000	10000	316003	50	50	4.8694325
250000	10000	395766	50	50	6.0514137
300000	10000	474240	50	50	7.3069175
Varying the number of objects					
10000	50000	15849	50	50	0.2652916
10000	100000	15888	50	50	0.2399566
10000	200000	15806	50	50	0.2812486
10000	250000	15703	50	50	0.2400150
10000	300000	15740	50	50	0.2420852
Varying the number of object types					
10000	10000	15713	50	50	0.2652902
10000	10000	15823	100	50	0.2512983
10000	10000	15906	200	50	0.2669229
10000	10000	15974	300	50	0.2802488
Varying the number of activities					
10000	10000	15761	50	50	0.2912541
10000	10000	15752	50	100	1.0569380
10000	10000	15802	50	200	0.3326528
10000	10000	15918	50	300	0.2922242
Varying the number of related objects					
10000	10000	15673	50	50	0.2364007
10000	10000	25536	50	50	0.2652944
10000	10000	35620	50	50	0.3229823
10000	10000	45478	50	50	0.3081768
10000	10000	55846	50	50	0.3593774

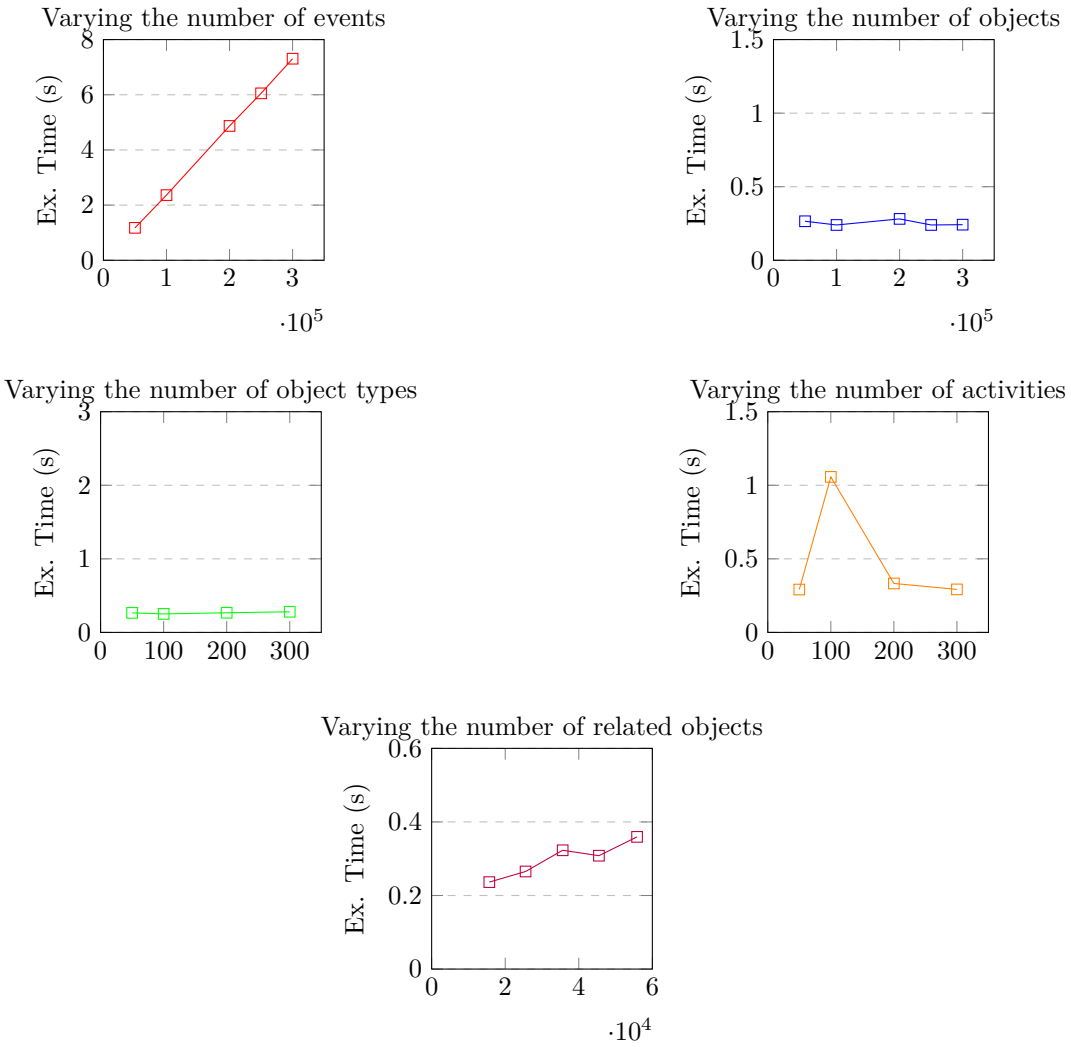


Figure 6.24: Execution times for the discovery of the Object Codeath Enrichment.

Conclusion

This chapter has delved into the core task of Object-Centric Process Discovery (OCPD), introducing and evaluating various techniques for uncovering process models from OCELs. We have explored the collation of traditional process models as a starting point and introduced OCDFGs and OCPNs as specialized object-centric representations, highlighting their respective strengths and limitations. The incorporation of textual abstractions for analysis with LLMs demonstrates the potential for integrating advanced language technologies with process mining. The detailed exploration of Object Graph Enrichment discovery, illustrated with examples from P2P and O2C processes, showcases the practical application of these techniques for understanding object interactions. Finally, the assessment of execution times for different discovery methods provides valuable insights into their performance characteristics. This chapter provides a robust foundation for the subsequent exploration of object-centric conformance checking, which will build upon the discovered process models to assess the alignment between observed behavior and normative expectations. The insights gained from OCPD, particularly the understanding of object interactions and process flow captured by OCDFGs, OCPNs, and Object Graph Enrichments, will be crucial for evaluating conformance and identifying deviations in the next chapter. Moreover, the efficient discovery methods presented here pave the way for scalable and practical applications of OCPM in real-world settings.

Chapter 7

Object-Centric Conformance Checking (OCCC)

"*In every complex system, there is a pattern; in every pattern, an anomaly.*"
Nassim Nicholas Taleb, The Black Swan

Introduction

This chapter presents Object-Centric Conformance Checking (OCCC), focusing on verifying the alignment between observed process behavior in Object-Centric Event Logs (OCELs) and expected normative models. The approaches described here are grounded in and extend the author's contributions in the field:

- [24] provides refined techniques that support handling complexity and obtaining advanced diagnostics in conformance checking.
- [23] offers foundational insights into token-based replay methods, instrumental for adapting conformance checking to object-centric Petri nets.
- [15] introduces methodologies for explainable object-centric anomaly detection, enabling the integration of domain knowledge into the analysis.

Building on these contributions, this chapter explores three perspectives for OCCC: OCDFG-based, Object-Type-Graph-based, and Event-Type-to-Object-Type Graph-based conformance checking. Each perspective emphasizes different aspects of object interactions and activity flows, complemented by definitions and fitness functions for quantifying conformance. An Order-to-Cash (O2C) example illustrates how deviations can be identified and interpreted, providing actionable insights. The chapter concludes by presenting a comprehensive approach to object-centric anomaly detection, integrating dimensionality reduction, feature selection, anomaly detection algorithms, correlation analysis, and domain expertise.

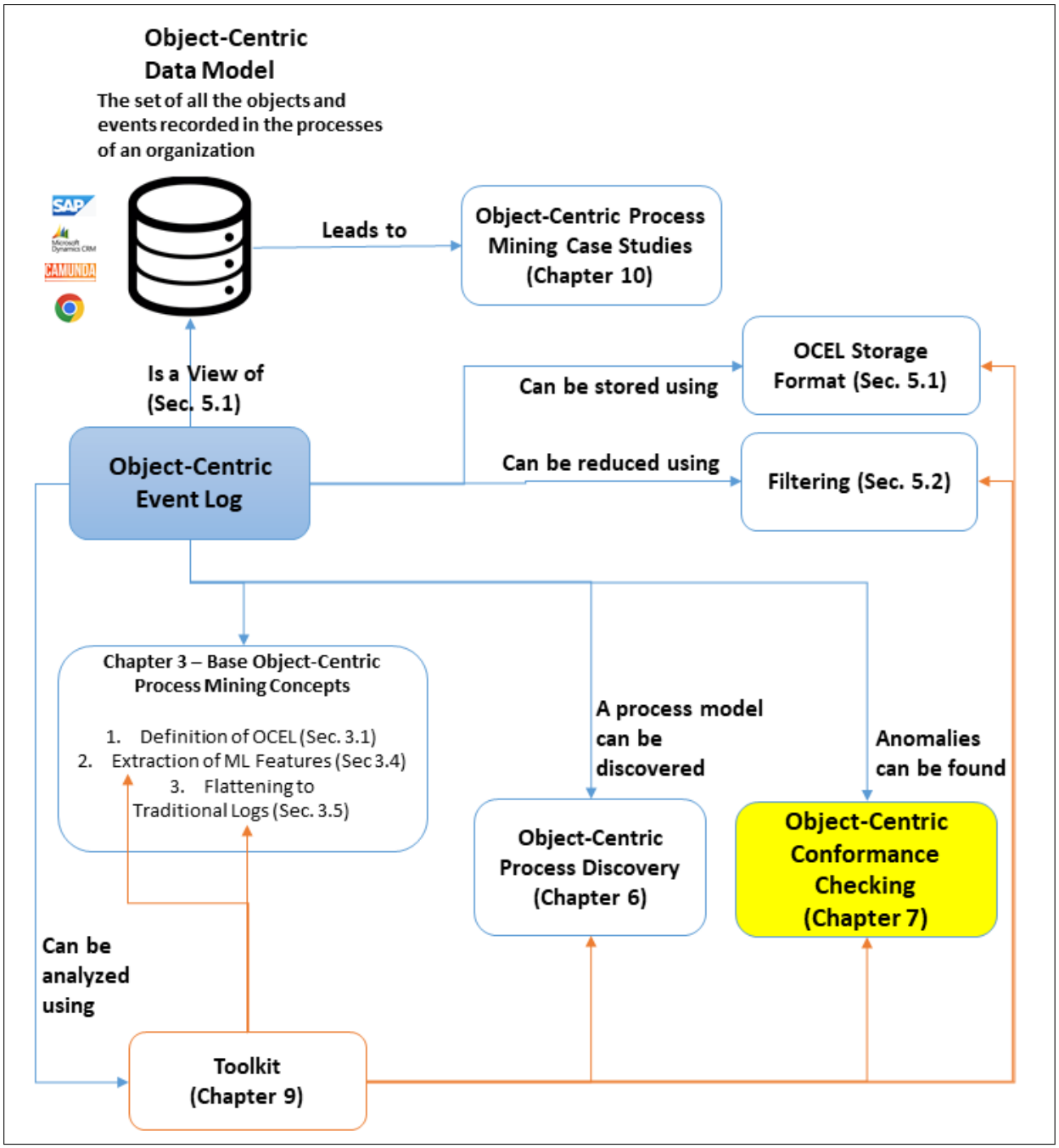


Figure 7.1: Highlight of the problems tackled in this specific chapter.

7.1 OCDFG-Based Conformance Checking

Ensuring that the observed behavior in processes captured by Object-Centric Event Logs (OCEL) aligns with the expected behavior defined by a normative model is crucial for process compliance and improvement. The Object-Centric Directly Follows Graph (OCDFG) acts as such a normative model, representing the ideal sequences of activities and their relationships with respect to different object types.

To assess conformance between an OCEL and an OCDFG, a systematic approach is needed to compare the observed activities and flows in the event log with those prescribed by the normative model. This comparison helps identify discrepancies such as missing or additional activities and flows, significant deviations in activity and flow frequencies, and provides a quantitative measure of overall conformance.

The following definition formalizes this conformance checking approach, detailing how to compare activities and flows, evaluate measure differences, and compute a fitness value that quantifies the alignment between the observed process behavior and the normative model.

Definition 59 (Conformance Checking between OCEL and OCDFG)**.** *Let* $L = (A_L, OT_L, E_L, O_L, EA_L, OA_L, evtype_L, time_L, objtype_L, eatype_L, eaval_L, oaval_L, E2O_L, O2O_L, \pi^L_{omap}, \leq_L)$ *be an OCEL, as in Definition 13.*

Let $M = (A_M, OT_M, F_M, \pi^M_{measn}, \pi^M_{mease})$ *be a normative OCDFG, as in Definition 45.*

Let $disc_{OCDFG}$ *be a discovery operator that derives an OCDFG from an OCEL. Applying this operator to* L*, we obtain* $disc_{OCDFG}(L) = (A'_L, OT'_L, F'_L, \pi^{L'}_{measn}, \pi^{L'}_{mease})$*, where* $A'_L \subseteq A_L$ *and* $OT'_L \subseteq OT_L$*.*

We define the conformance between L *and* M *as follows:*

1. ***Activity Conformance***

   - Missing Activities*:*
     $$A_{missing} = A_M \setminus A'_L$$
     These are activities present in the normative OCDFG M *but not in the discovered OCDFG from* L*.*
   - Additional Activities*:*
     $$A_{additional} = A'_L \setminus A_M$$
     These are activities present in the discovered OCDFG from L *but not in the normative OCDFG* M*.*

2. ***Flow (Edge) Conformance***

   - Missing Flows*:*
     $$F_{missing} = F_M \setminus F'_L$$
     These are flows present in M *but not in the discovered OCDFG from* L*.*
   - Additional Flows*:*
     $$F_{additional} = F'_L \setminus F_M$$
     These are flows present in the discovered OCDFG from L *but not in* M*.*

3. ***Measure Conformance***

   - *For each activity* $a \in A_M \cap A'_L$*, define the* measure difference*:*
     $$\Delta_{act}(a) = \left|\pi^M_{measn}(a) - \pi^{L'}_{measn}(a)\right|$$
     The activity a *is considered* non-conforming in measure *if:*
     $$\Delta_{act}(a) > \theta_{act}$$
     where θ_{act} *is a predefined threshold.*
   - *For each flow* $f \in F_M \cap F'_L$*, define the* measure difference*:*
     $$\Delta_{flow}(f) = \left|\pi^M_{mease}(f) - \pi^{L'}_{mease}(f)\right|$$
     The flow f *is considered* non-conforming in measure *if:*
     $$\Delta_{flow}(f) > \theta_{flow}$$
     where θ_{flow} *is a predefined threshold.*

4. ***Fitness Value***

We define a fitness function $\phi(L, M)$ to quantify the overall conformance between the OCEL L and the normative OCDFG M:

$$\phi(L, M) = 1 - \frac{\alpha \cdot |A_{missing}| + \beta \cdot |F_{missing}| + \gamma \cdot \sum_{a \in A_M \cap A'_L} \delta_{act}(a) + \delta \cdot \sum_{f \in F_M \cap F'_L} \delta_{flow}(f)}{N}$$

where:

- *α, β, γ, δ are weighting factors for missing activities, missing flows, non-conforming activities, and non-conforming flows, respectively.*
- $\delta_{act}(a) = \begin{cases} 1, & \text{if } \Delta_{act}(a) > \theta_{act} \\ 0, & \text{otherwise} \end{cases}$
- $\delta_{flow}(f) = \begin{cases} 1, & \text{if } \Delta_{flow}(f) > \theta_{flow} \\ 0, & \text{otherwise} \end{cases}$
- *N is a normalization constant defined as:*

$$N = \alpha \cdot |A_M| + \beta \cdot |F_M| + \gamma \cdot |A_M| + \delta \cdot |F_M|$$

ensuring that $0 \leq \phi(L, M) \leq 1$.

A higher value of $\phi(L, M)$ indicates better conformance between the OCEL L and the normative OCDFG M.

Definition 59 provides a systematic approach for conformance checking between an OCEL and a normative OCDFG. The approach consists of the following key components:

- **Activity Conformance**: Identifying discrepancies in the set of activities executed in the process. Specifically, we detect:
  - *Missing Activities*: Activities present in the normative OCDFG M but absent in the discovered OCDFG from L.
  - *Additional Activities*: Activities present in the discovered OCDFG from L but not defined in the normative OCDFG M.
- **Flow (Edge) Conformance**: Detecting discrepancies in the process flow by comparing the directly follows relationships (flows) between activities:
  - *Missing Flows*: Flows present in M but not observed in the discovered OCDFG from L.
  - *Additional Flows*: Flows observed in the discovered OCDFG from L but not prescribed by the normative model M.
- **Measure Conformance**: For activities and flows present in both models, assessing quantitative differences in associated measures, such as frequency counts:
  - Computing the *measure difference* for each activity and flow.
  - Identifying activities and flows as *non-conforming in measure* if the measure difference exceeds predefined thresholds θ_{act} and θ_{flow}, respectively.
- **Fitness Function**: Aggregating the identified discrepancies into a single *fitness value* $\phi(L, M)$, which quantifies the overall conformance level between the observed behavior and the normative model. A value closer to 1 indicates better conformance. The weighting factors α, β, γ, and δ allow for adjusting the emphasis on different aspects of conformance.

Before applying this conformance checking approach, we use the discovery operator disc_{OCDFG} to derive an OCDFG from the OCEL L. This discovered OCDFG, denoted as $\text{disc}_{OCDFG}(L)$, represents the actual behavior observed in the event log.

The conformance checking process involves comparing the normative OCDFG M with the discovered OCDFG from L across the aspects outlined above.

7.1.1 Example Application of OCDFG-Based Conformance Checking to an O2C Process

In this section, we apply the OCDFG-based conformance checking approach to an Order-to-Cash (O2C) process. We compare a hypothetical normative OCDFG (Figure 7.2) with a realistic OCDFG extracted from an object-centric event log (Figure 7.3) for the O2C process, and elaborate on the non-conformities identified.

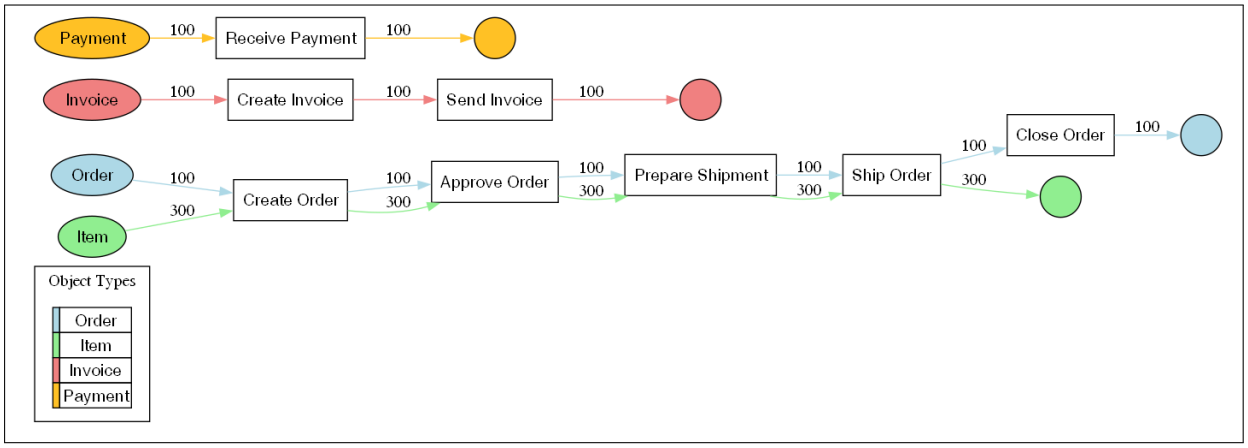


Figure 7.2: Hypothetical normative OCDFG for an O2C process. The arcs are decorated with the expected frequencies of the directly-follows relationships for the object types.

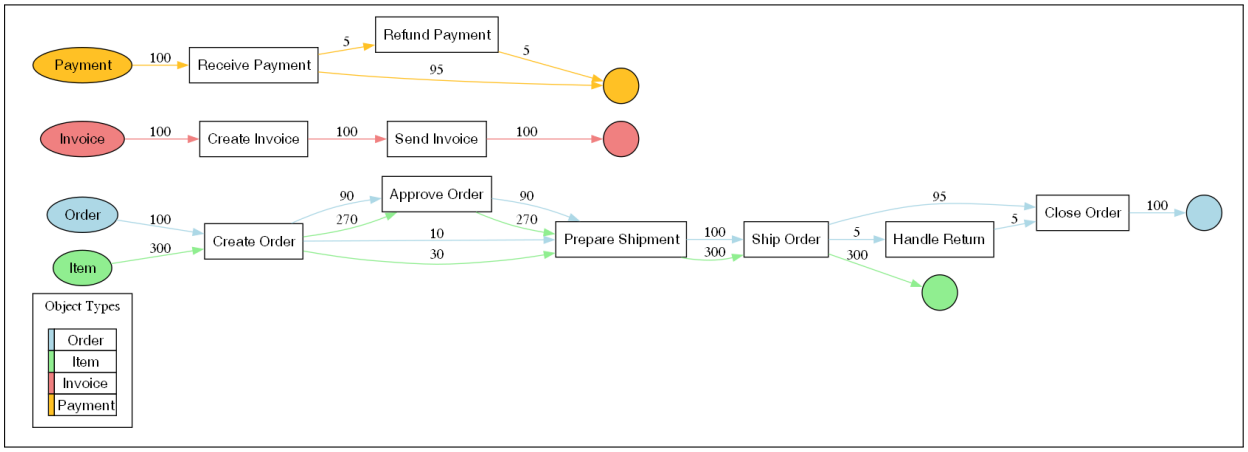


Figure 7.3: Realistic OCDFG extracted from an object-centric event log for an O2C process. The arcs are decorated with the observed frequencies of the directly-follows relationships for the object types.

Comparative Analysis of the Normative and Real-Life OCDFGs

Figures 7.2 and 7.3 present the normative and real-life OCDFGs, respectively. The O2C process involves several activities and object types, including *Order*, *Item*, *Invoice*, and *Payment*. Each object type is represented with a specific color, and the arcs represent the directly follows relationships, annotated with their frequencies.

Using the conformance checking approach defined in Definition 59, we proceed to identify non-conformities between the two OCDFGs.

Activity Conformance **Missing Activities** (A_{missing}):

- **None**. All activities present in the normative OCDFG are also present in the real-life OCDFG.

Additional Activities ($A_{\text{additional}}$):

- *Handle Return*
- *Refund Payment*

These activities appear in the real-life OCDFG but are not defined in the normative OCDFG. This indicates that in practice, there are cases where orders are returned and payments are refunded—exceptions not accounted for in the normative model.

Flow (Edge) Conformance **Missing Flows** (F_{missing}):

- **None**. All flows present in the normative OCDFG exist in the real-life OCDFG, though frequencies may differ.

Additional Flows ($F_{\text{additional}}$):

- *Order* Object Type (Blue):
  - **Create Order** → **Prepare Shipment** (frequency: 10)
  - **Ship Order** → **Handle Return** (frequency: 5)
  - **Handle Return** → **Close Order** (frequency: 5)
- *Payment* Object Type (Orange):
  - **Receive Payment** → **Refund Payment** (frequency: 5)
  - **Refund Payment** → **Payment End** (frequency: 5)

These additional flows represent deviations from the expected process sequences defined in the normative model.

Measure Conformance We assess the quantitative differences in measures (frequencies) associated with activities and flows.

Activities ($A_M \cap A'_L$):
For each activity a present in both models, we compute the measure difference:

$$\Delta_{\text{act}}(a) = \left| \pi^{M}_{\text{measn}}(a) - \pi^{L'}_{\text{measn}}(a) \right|$$

- **Approve Order**:
  - Normative frequency: 100
  - Real-life frequency: 90 (since 10 orders skip approval)
  - Measure difference: $\Delta_{\text{act}}(\text{Approve Order}) = |100 - 90| = 10$
- **Receive Payment**:
  - Normative frequency: 100
  - Real-life frequency: 100
  - Since frequencies are equal, $\Delta_{\text{act}}(\text{Receive Payment}) = 0$

Flows ($F_M \cap F'_L$):
For each flow f present in both models, we compute the measure difference:

$$\Delta_{\text{flow}}(f) = \left| \pi^{M}_{\text{mease}}(f) - \pi^{L'}_{\text{mease}}(f) \right|$$

- **Approve Order** → **Prepare Shipment** [Order]:
  - Normative frequency: 100
  - Real-life frequency: 90
  - Measure difference: $\Delta_{\text{flow}}(f) = |100 - 90| = 10$
- **Receive Payment** → **Payment End** [Payment]:

  - Normative frequency: 100
  - Real-life frequency: 95
  - Measure difference: $\Delta_{\text{flow}}(f) = |100 - 95| = 5$

Non-Conforming Measures:
Assuming predefined thresholds:

- Activity threshold (θ_{act}): 5
- Flow threshold (θ_{flow}): 5

We identify activities and flows where the measure difference exceeds the thresholds:

- **Non-Conforming Activities**:
  - **Approve Order**: $\Delta_{\text{act}}(\text{Approve Order}) = 10 > \theta_{\text{act}}$
- **Non-Conforming Flows**:
  - **Approve Order** $\rightarrow$ **Prepare Shipment** [Order]: $\Delta_{\text{flow}}(f) = 10 > \theta_{\text{flow}}$

Fitness Value Computation

Using the fitness function defined in Definition 59:

$$\phi(L, M) = 1 - \frac{\alpha \cdot |A_{\text{missing}}| + \beta \cdot |F_{\text{missing}}| + \gamma \cdot \sum_{a \in A_M \cap A'_L} \delta_{\text{act}}(a) + \delta \cdot \sum_{f \in F_M \cap F'_L} \delta_{\text{flow}}(f)}{N}$$

Where:

- $\delta_{\text{act}}(a) = \begin{cases} 1, & \text{if } \Delta_{\text{act}}(a) > \theta_{\text{act}} \\ 0, & \text{otherwise} \end{cases}$
- $\delta_{\text{flow}}(f) = \begin{cases} 1, & \text{if } \Delta_{\text{flow}}(f) > \theta_{\text{flow}} \\ 0, & \text{otherwise} \end{cases}$
- $N = \alpha \cdot |A_M| + \beta \cdot |F_M| + \gamma \cdot |A_M| + \delta \cdot |F_M|$

Assuming weights:

- $\alpha = 1$ (weight for missing activities)
- $\beta = 1$ (weight for missing flows)
- $\gamma = 1$ (weight for non-conforming activities)
- $\delta = 1$ (weight for non-conforming flows)

Compute Components:

- $|A_{\text{missing}}| = 0$ (no missing activities)
- $|F_{\text{missing}}| = 0$ (no missing flows)
- $\sum_{a \in A_M \cap A'_L} \delta_{\text{act}}(a) = 1$
  - $\delta_{\text{act}}(\text{Approve Order}) = 1$ (measure difference exceeds threshold)
  - Other activities have $\delta_{\text{act}} = 0$
- $\sum_{f \in F_M \cap F'_L} \delta_{\text{flow}}(f) = 1$
  - $\delta_{\text{flow}}(\text{Approve Order} \rightarrow \text{Prepare Shipment[Order]}) = 1$
  - Other flows have $\delta_{\text{flow}} = 0$

- $|A_M| = 8$ (number of activities in the normative model)
- $|F_M| = 7$ (number of flows in the normative model)
- $N = 1 \cdot 8 + 1 \cdot 7 + 1 \cdot 8 + 1 \cdot 7 = 30$

Compute Fitness Value:

$$\phi(L, M) = 1 - \frac{1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1 + 1 \cdot 1}{30} = 1 - \frac{2}{30} = 1 - 0.0667 = 0.9333$$

Thus, the fitness value is $\phi(L, M) = 0.9333$, indicating a high level of conformance between the real-life OCEL and the normative OCDFG, with some deviations.

Summary of Non-Conformities

Process Deviations:

- **Skipping Approval**: 10% of orders skip the *Approve Order* activity, directly proceeding to *Prepare Shipment*.
- **Returns Handling**: 5% of orders involve a return process, introducing the *Handle Return* and *Refund Payment* activities not present in the normative model.

Additional Activities:

- *Handle Return*
- *Refund Payment*

Non-Conforming Measures:

- **Approve Order** activity has a significant reduction in frequency (10 fewer occurrences), suggesting potential non-compliance with approval policies.
- The flow from **Approve Order** $\rightarrow$ **Prepare Shipment** [Order] has a decreased frequency (90 instead of 100), aligning with the skipped approvals.

Potential Compliance Issues:

- **Bypassing Approval**: Skipping the approval step may violate organizational policies or regulatory requirements, posing a risk of unauthorized orders being processed.
- **Unaccounted Returns**: The occurrence of returns and refunds suggests that the normative model does not fully capture the real-life process, indicating a need to update the model to include these exceptions.

Conclusion

The comparison between the normative and real-life OCDFGs reveals areas where the actual process deviates from the expected behavior. The identified non-conformities include additional activities, deviations in process sequences, and significant differences in activity and flow measures.

By applying the OCDFG-based conformance checking approach, the organization can gain valuable insights:

- **Investigate Process Shortcuts**: Examine the reasons behind orders bypassing the approval step and address any compliance risks associated with this deviation.
- **Update Normative Models**: Incorporate common exceptions, such as returns and refunds, into the normative model to better reflect the real-life process.
- **Enhance Process Compliance**: Implement controls and training to ensure that all orders undergo the necessary approval process, reducing the likelihood of unauthorized transactions.

Overall, this approach enables organizations to systematically identify and address deviations, leading to improved process efficiency, compliance, and alignment between prescribed procedures and actual execution.

7.2 Object-Type-Graph-Based Conformance Checking

Understanding how different object types interact in complex processes captured by Object-Centric Event Logs (OCEL) is crucial for analyzing and improving process performance. *Object Graph Enrichments* enable us to define various types of relationships between objects based on their interactions within events. However, to perform *conformance checking* by comparing these interactions against a normative model, we need an abstraction that not only summarizes these relationships at the level of object types but also includes the *frequency* of these interactions.

This need motivates the introduction of the *Object Type Graph* (Definition 60), which provides a high-level view of how object types are related through various kinds of interactions, explicitly incorporating the frequency of these interactions between object types.

Definition 60 (Object Type Graph with Edge Frequencies)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *and a set of Object Graph Enrichments* $\{GE_\sigma = (L, \Lambda_\sigma) \mid \sigma \in \Sigma\}$ *as per Definition 58, we define the* Object Type Graph $OTG = (\mathcal{OT}, \mathcal{E}, f)$ *as a weighted multigraph where:*

- $\mathcal{OT} = OT$ *is the set of object types.*
- $\mathcal{E} \subseteq \mathcal{OT} \times \Sigma \times \mathcal{OT}$ *is the set of edges, where:*
  - Σ *is the set of relationship types (e.g., Interaction, Creation, Continuation, Cobirth, Codeath).*
  - *Each edge* $(ot_1, \sigma, ot_2) \in \mathcal{E}$*, with* $ot_1, ot_2 \in \mathcal{OT}$ *and* $\sigma \in \Sigma$*, represents that there are interactions of type* σ *between objects of types* ot_1 *and* ot_2*.*
- $f : \mathcal{E} \to \mathbb{N}$ *assigns a* frequency *to each edge, indicating how many times the interaction of type* σ *between* ot_1 *and* ot_2 *occurs in the log.*

The Object Type Graph serves as an aggregated representation of the relationships between object types derived from individual object interactions, explicitly capturing the frequency of these interactions. By focusing on object types and their interaction frequencies, we can compare the overall structure and patterns of interactions in the observed data against a normative model without getting overwhelmed by the details of individual object instances.

To construct the Object Type Graph from an OCEL and a set of Object Graph Enrichments, we follow the procedure described in Definition 61.

Definition 61 (Construction of Object Type Graph from OCEL)**.** *Given an OCEL* L *and a set of Object Graph Enrichments* $\{GE_\sigma = (L, \Lambda_\sigma) \mid \sigma \in \Sigma\}$*, where* Σ *is the set of relationship types as defined in Definition 58, the Object Type Graph* $OTG = (\mathcal{OT}, \mathcal{E}, f)$ *is constructed as follows:*

- *For each object type* $ot \in OT$*, include a node in* $\mathcal{OT}$*.*
- *Initialize the set of edges* $\mathcal{E}$ *as empty.*
- *For each relationship type* $\sigma \in \Sigma$*, and for each pair* $(o_1, o_2) \in \Lambda_\sigma$*:*
  - *Let* $ot_1 = objtype(o_1)$ *and* $ot_2 = objtype(o_2)$*.*
  - *Create or update the edge* (ot_1, σ, ot_2) *in* $\mathcal{E}$*:*
    - *If* $(ot_1, \sigma, ot_2) \notin \mathcal{E}$*, add it to* $\mathcal{E}$ *and set* $f((ot_1, \sigma, ot_2)) = 1$*.*
    - *If* $(ot_1, \sigma, ot_2) \in \mathcal{E}$*, increment* $f((ot_1, \sigma, ot_2))$ *by 1.*

This construction process aggregates the object-level relationships captured by the Object Graph Enrichments into type-level relationships with explicit frequencies. The resulting Object Type Graph encapsulates how different object types interact across various relationship types and how often these interactions occur, providing a concise summary of the process's structural properties.

To perform conformance checking between the observed interactions in the OCEL and the normative model, we compare the discovered Object Type Graph with the normative Object Type Graph. This comparison involves identifying missing and additional object types and edges, as well as significant differences in edge frequencies.

Definition 62 (Conformance Checking between Object Type Graphs)**.** *Let $OTG_M = (\mathcal{OT}_M, \mathcal{E}_M, f_M)$ be a* normative *Object Type Graph, and $OTG_L = (\mathcal{OT}_L, \mathcal{E}_L, f_L)$ be the Object Type Graph* discovered *from an OCEL L using Definition 61.*

We define the conformance between OTG_L and OTG_M as follows:

1. ***Object Type Conformance***

   - Missing Object Types*:*
   $$OT_{missing} = \mathcal{OT}_M \setminus \mathcal{OT}_L$$
   These are object types present in the normative OTG but not in the discovered OTG.
   - Additional Object Types*:*
   $$OT_{additional} = \mathcal{OT}_L \setminus \mathcal{OT}_M$$
   These are object types present in the discovered OTG but not in the normative OTG.

2. ***Edge Conformance***

   - *For each relationship type $\sigma \in \Sigma$:*
     - Missing Edges*:*
     $$\mathcal{E}_{\sigma,missing} = \{(ot_1, \sigma, ot_2) \in \mathcal{E}_M \mid (ot_1, \sigma, ot_2) \notin \mathcal{E}_L\}$$
     These are edges of type σ present in the normative OTG but not in the discovered OTG.
     - Additional Edges*:*
     $$\mathcal{E}_{\sigma,additional} = \{(ot_1, \sigma, ot_2) \in \mathcal{E}_L \mid (ot_1, \sigma, ot_2) \notin \mathcal{E}_M\}$$
     These are edges of type σ present in the discovered OTG but not in the normative OTG.

3. ***Edge Frequency Conformance***

   - *For each edge $e = (ot_1, \sigma, ot_2) \in \mathcal{E}_M \cap \mathcal{E}_L$:*
     - *The frequency in the normative OTG:*
     $$f_M(e) = \textit{frequency assigned to edge } e \textit{ in } f_M$$
     - *The frequency in the discovered OTG:*
     $$f_L(e) = \textit{frequency assigned to edge } e \textit{ in } f_L$$
     - *The* relative frequency difference*:*
     $$\Delta_{edge}(e) = \left| \frac{f_L(e) - f_M(e)}{f_M(e)} \right|$$
     - *An edge e is considered* non-conforming in frequency *if:*
     $$\Delta_{edge}(e) > \theta_\sigma$$
     where θ_σ is a predefined threshold (e.g., a percentage) for relationship type σ.

4. ***Fitness Value***

   We define a fitness function $\phi(OTG_L, OTG_M)$ to quantify the conformance:

$$\phi(OTG_L, OTG_M) = 1 - \frac{\alpha \cdot |OT_{missing}| + \beta \cdot \sum_{\sigma \in \Sigma} |\mathcal{E}_{\sigma,missing}| + \gamma \cdot \sum_{\sigma \in \Sigma} \sum_{e \in \mathcal{E}_M \cap \mathcal{E}_L} \delta_{edge}(e)}{N}$$

where:

- *α, β, γ are weighting factors for missing object types, missing edges, and non-conforming edge frequencies, respectively.*

- $\delta_{edge}(e) = \begin{cases} 1, & \text{if } \Delta_{edge}(e) > \theta_\sigma \\ 0, & \text{otherwise} \end{cases}$
- *N is a normalization constant defined as:*

$$N = \alpha \cdot |\mathcal{OT}_M| + \beta \cdot \sum_{\sigma \in \Sigma} |\mathcal{E}_{M,\sigma}| + \gamma \cdot \sum_{\sigma \in \Sigma} |\mathcal{E}_{M,\sigma}|$$

where $\mathcal{E}_{M,\sigma} = \{e \in \mathcal{E}_M \mid e = (ot_1, \sigma, ot_2)\}$ is the set of edges of type σ in OTG_M.

The fitness value $\phi(OTG_L, OTG_M)$ ranges from 0 to 1, where a higher value indicates better conformance.

Through this conformance checking approach, we systematically compare the normative expectations expressed in the normative Object Type Graph OTG_M with the actual observations captured in the discovered Object Type Graph OTG_L, explicitly considering the frequencies of edges. This comparison highlights discrepancies in terms of missing or unexpected object types and interactions, as well as significant deviations in the frequency of interactions.

Understanding the interactions between different object types, along with how frequently they occur, is critical in processes where multiple entities collaborate to achieve process goals. By abstracting these interactions to the level of object types and including frequency information, we can capture the structural patterns of the process and how different entities are expected to relate to each other quantitatively. This abstraction enables us to:

- Identify discrepancies in the process design versus execution, including underutilization or overutilization of certain interactions.
- Detect potential compliance issues, such as missing mandatory interactions or excessive occurrence of prohibited interactions.
- Analyze the structural robustness and complexity of the process.
- Provide actionable insights for process improvement and governance.

7.2.1 Example Application of Object-Type-Graph-Based Conformance Checking to an O2C Process

In this section, we apply the Object-Type-Graph-Based Conformance Checking approach to an Order-to-Cash (O2C) process. We define a hypothetical normative Object Type Graph (OTG) representing the expected interactions and frequencies in the process and a realistic OTG that might occur in practice. We then elaborate on the non-conformities based on the described approach.

The normative OTG, illustrated in Figure 7.4, captures the expected object types and their interactions in the O2C process. The arcs are annotated with the expected frequencies of the interactions.

The realistic OTG, shown in Figure 7.5, represents the observed interactions and frequencies discovered from an object-centric event log, including possible deviations from the normative model. The arcs are annotated with the observed frequencies of the interactions.

Non-Conformities Based on Conformance Checking Approach

Using the conformance checking approach described in Definition 62, we compare the normative OTG with the realistic OTG to identify non-conformities.

1. Object Type Conformance *Missing Object Types*

- None. All object types present in the normative OTG are also present in the realistic OTG.

Additional Object Types

- **Return**: This object type is present in the realistic OTG but not in the normative OTG.

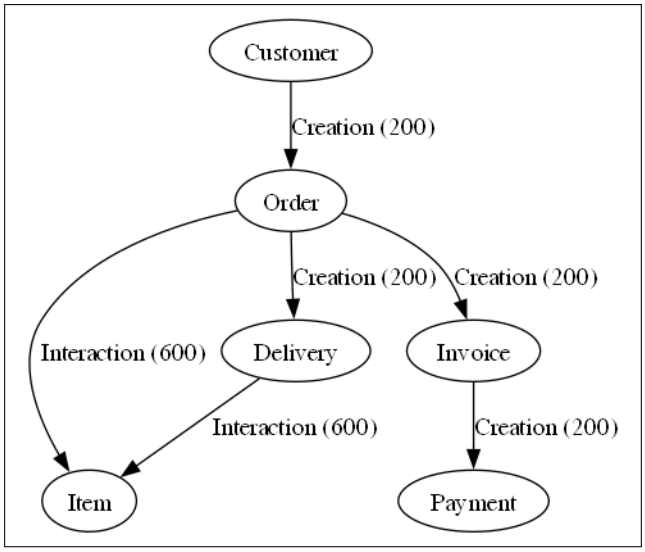


Figure 7.4: Hypothetical normative OTG for an O2C process. The arcs are decorated with the expected frequencies of the different interactions.

2. Edge Conformance *Missing Edges*

- None. All edges present in the normative OTG are also present in the realistic OTG.

Additional Edges

- **Order** *leads to* **Return** (*Interaction*, frequency: 30)
- **Payment** *interacts with* **Order** (*Interaction*, frequency: 40)
- **Customer** *interacts with* **Return** (*Interaction*, frequency: 30)

These additional edges represent interactions not accounted for in the normative model, indicating deviations or extensions in the actual process.

3. Edge Frequency Conformance For edges present in both OTGs, we compute the relative frequency difference and identify those exceeding the threshold (assumed to be 15% for both *Creation* and *Interaction* relationships).

Edges Exceeding the Threshold

1. **Customer** *creates* **Order**

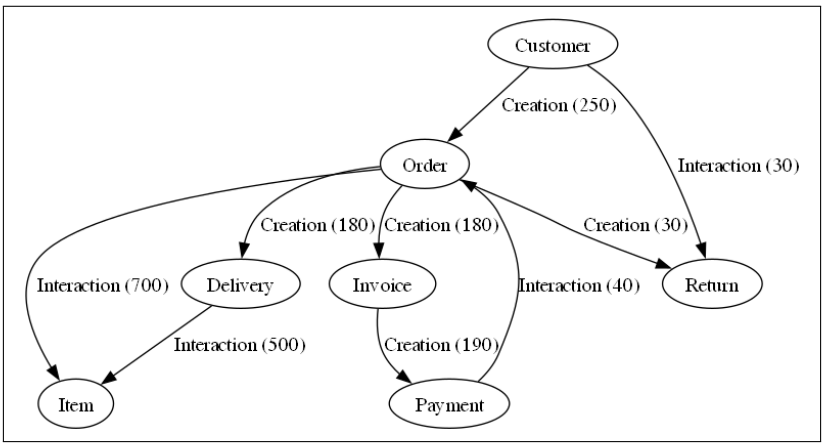


Figure 7.5: Realistic OTG extracted from an object-centric event log for an O2C process. The arcs are decorated with the observed frequencies of the different interactions.

- Normative frequency: 200
- Realistic frequency: 250
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{250 - 200}{200} \right| = 25\% \quad \text{(exceeds 15\% threshold)}$$

2. **Order** *includes* **Item**

- Normative frequency: 600
- Realistic frequency: 700
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{700 - 600}{600} \right| = 16.67\% \quad \text{(exceeds 15\% threshold)}$$

3. **Delivery** *fulfills* **Item**

- Normative frequency: 600
- Realistic frequency: 500
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{500 - 600}{600} \right| = 16.67\% \quad \text{(exceeds 15\% threshold)}$$

Edges Within the Threshold

1. **Order** *leads to* **Invoice**

- Normative frequency: 200
- Realistic frequency: 180
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{180 - 200}{200} \right| = 10\% \quad \text{(within threshold)}$$

2. **Invoice** *leads to* **Payment**

- Normative frequency: 200
- Realistic frequency: 190
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{190 - 200}{200} \right| = 5\% \quad \text{(within threshold)}$$

3. **Order** *leads to* **Delivery**

- Normative frequency: 200
- Realistic frequency: 180
- Relative difference:

$$\Delta_{\text{edge}} = \left| \frac{180 - 200}{200} \right| = 10\% \quad \text{(within threshold)}$$

4. Fitness Value Calculation Using the fitness function defined in Definition 62, we compute the overall fitness value.

Assuming weights:

- $\alpha = 1$
- $\beta = 1$
- $\gamma = 1$

Total normative object types ($|\mathcal{OT}_M|$): 6
Total normative edges ($|\mathcal{E}_M|$): 6
Normalization constant:

$$N = \alpha \cdot |\mathcal{OT}_M| + \beta \cdot |\mathcal{E}_M| + \gamma \cdot |\mathcal{E}_M| = 1 \cdot 6 + 1 \cdot 6 + 1 \cdot 6 = 18$$

Calculating the numerator:

- Missing object types: 0
- Missing edges: 0
- Non-conforming edge frequencies: 3 (edges exceeding threshold)

Therefore, the fitness value is:

$$\phi(OTG_L, OTG_M) = 1 - \frac{0 + 0 + 3}{18} = 1 - \frac{3}{18} = 0.8333$$

5. Interpretation of Non-Conformities *Additional Object Types and Edges*

- **Return** object type and its interactions indicate that returns are occurring in practice, which were not included in the normative model. This suggests either an oversight in the normative model or that returns represent exceptional cases not modeled.
- **Payment** interacting directly with **Order** suggests that payments are sometimes made directly for orders, possibly indicating prepayments or immediate payments upon order placement.
- **Customer** interacting with **Return** suggests customer involvement in the return process.

Edge Frequency Deviations

- The higher frequency of **Customer** *creates* **Order** (25% increase) indicates more orders are being placed than expected, possibly due to an increased customer base or promotional activities.
- The higher frequency of **Order** *includes* **Item** (16.67% increase) suggests that orders contain more items on average than anticipated, which might indicate changes in customer purchasing behavior or effective upselling strategies.
- The lower frequency of **Delivery** *fulfills* **Item** (16.67

Conclusion The realistic OTG aligns with the normative OTG in terms of the overall structure but exhibits several non-conformities:

- **Process Extensions**: The presence of **Return** processes not accounted for in the normative model suggests that returns are a part of the process that need to be formally included in the model.
- **Process Deviations**: Direct interactions between **Payment** and **Order** indicate deviations or alternate payment procedures, such as advance payments, which are not captured in the normative process.
- **Frequency Variances**: Significant differences in expected frequencies highlight changes in process execution or customer behavior. For instance, increased order and item frequencies suggest higher demand, while decreased delivery fulfillment rates may point to operational challenges.

By applying the Object-Type-Graph-Based Conformance Checking approach, organizations can identify and analyze these deviations to improve process alignment with expectations, update normative models to reflect actual practices, and address operational issues revealed by frequency variances.

7.3 Event-Type-to-Object-Type Graph-Based Conformance Checking

Understanding the relationships between event types (activities) and object types is crucial in object-centric process analysis. The *Event-Type-to-Object-Type Graph* (ETOT Graph) provides a high-level abstraction of these relationships, indicating which activities involve which kinds of objects and how frequently these associations occur. By comparing this graph derived from observed data with a normative model, we can perform *conformance checking* to identify deviations and assess the alignment between actual process behavior and expected behavior.

This section introduces the formal definitions necessary for establishing this conformance checking approach, detailing how to construct the ETOT Graph from an OCEL, compare it against a normative model, and compute a fitness value that quantifies the level of conformance.

Definition 63 (Event-Type-to-Object-Type Graph)**.** *Given an OCEL* $L = (A, OT, E, O, EA, OA, evtype, time, objtype, eatype, eaval, oaval, E2O, O2O, \pi_{omap}, \leq)$ *as per Definition 13, an* Event-Type-to-Object-Type Graph *(ETOT Graph) is a weighted bipartite graph* $G = (A, OT, R, w)$ *where:*

- A *is the set of event types (activities).*
- OT *is the set of object types.*
- $R \subseteq A \times OT$ *is the set of relationships, where an edge* $(a, ot) \in R$ *indicates that events of type* a *are associated with objects of type* ot*.*
- $w : R \to \mathbb{N}$ *assigns a* frequency *to each relationship, representing the number of times events of type* a *are associated with objects of type* ot *in the log.*

The ETOT Graph captures the associations between event types and object types, along with the frequency of these associations. This information is valuable for understanding which activities involve which kinds of objects and identifying patterns or anomalies in process execution.

To extract the ETOT Graph from an OCEL, we follow a systematic discovery approach.

Definition 64 (Discovery of Event-Type-to-Object-Type Graph from OCEL)**.** *Given an OCEL* L*, the* discovery *of the Event-Type-to-Object-Type Graph* $G = (A, OT, R, w)$ *proceeds as follows:*

1. *Initialize the set of relationships* R *as empty.*
2. *For each event* $e \in E$*:*
   (a) *Let* $a = evtype(e)$*.*
   (b) *For each object* $o \in \pi_{omap}(e)$*:*
      i. *Let* $ot = objtype(o)$*.*
      ii. *If* $(a, ot) \notin R$*, add* (a, ot) *to* R *and set* $w(a, ot) = 1$*.*
      iii. *If* $(a, ot) \in R$*, increment* $w(a, ot)$ *by 1.*

This approach processes each event, recording the associations between its event type and the types of objects it references, building up the relationship set R and computing the frequencies w.

To assess conformance between the observed ETOT Graph and a normative ETOT Graph, we define a conformance checking method that compares the two graphs and computes a fitness value indicating the degree of alignment.

Definition 65 (Conformance Checking between ETOT Graphs)**.** *Let* $G_M = (A_M, OT_M, R_M, w_M)$ *be a* normative *ETOT Graph defining the expected associations between event types and object types, and let* $G_L = (A_L, OT_L, R_L, w_L)$ *be the ETOT Graph* discovered *from an OCEL* L *using Definition 64.*

We define the conformance between G_L *and* G_M *as follows:*

1. ***Node Conformance***
   - Missing Activities*:*
     $$A_{missing} = A_M \setminus A_L$$
     Activities present in the normative ETOT Graph but not observed in the discovered ETOT Graph.

- Additional Activities*:*

$$A_{additional} = A_L \setminus A_M$$

Activities observed in the discovered ETOT Graph but not defined in the normative ETOT Graph.

- Missing Object Types*:*

$$OT_{missing} = OT_M \setminus OT_L$$

Object types present in the normative ETOT Graph but not observed in the discovered ETOT Graph.

- Additional Object Types*:*

$$OT_{additional} = OT_L \setminus OT_M$$

Object types observed in the discovered ETOT Graph but not defined in the normative ETOT Graph.

2. ***Edge Conformance***

- Missing Relationships*:*

$$R_{missing} = R_M \setminus R_L$$

Relationships present in the normative ETOT Graph but not observed in the discovered ETOT Graph.

- Additional Relationships*:*

$$R_{additional} = R_L \setminus R_M$$

Relationships observed in the discovered ETOT Graph but not defined in the normative ETOT Graph.

3. ***Edge Frequency Conformance***

- *For each relationship* $r = (a, ot) \in R_M \cap R_L$*:*
  - *Compute the* relative frequency difference*:*

$$\Delta_{rel}(r) = \left| \frac{w_L(r) - w_M(r)}{w_M(r)} \right|$$

  - *The relationship* r *is considered* non-conforming in frequency *if:*

$$\Delta_{rel}(r) > \theta_{rel}$$

where θ_{rel} *is a predefined threshold (e.g., a percentage) indicating acceptable relative deviations.*

4. ***Fitness Value***

We define the fitness function $\phi(G_L, G_M)$ *as:*

$$\phi(G_L, G_M) = 1 - \frac{\alpha \cdot (|A_{missing}| + |OT_{missing}|) + \beta \cdot |R_{missing}| + \gamma \cdot \sum_{r \in R_M \cap R_L} \delta_{rel}(r)}{N}$$

where:

- α*,* β*,* γ *are weighting factors for missing nodes (activities and object types), missing relationships, and non-conforming frequencies, respectively.*
- $\delta_{rel}(r) = \begin{cases} 1, & \text{if } \Delta_{rel}(r) > \theta_{rel} \\ 0, & \text{otherwise} \end{cases}$
- N *is a normalization constant defined as:*

$$N = \alpha \cdot (|A_M| + |OT_M|) + \beta \cdot |R_M| + \gamma \cdot |R_M|$$

ensuring that $0 \leq \phi(G_L, G_M) \leq 1$*.*

A higher value of $\phi(G_L, G_M)$ *indicates better conformance between the discovered ETOT Graph and the normative model.*

This conformance checking method systematically compares the normative ETOT Graph G_M with the discovered ETOT Graph G_L by evaluating the presence and absence of nodes and edges, as well as significant deviations in the frequencies of relationships. The fitness function aggregates these discrepancies into a single value that quantifies the overall conformance level.

7.3.1 Example Application of Event-Type-to-Object-Type Graph-Based Conformance Checking to an O2C Process

In this section, we apply the ETOT Graph-based conformance checking approach to an Order-to-Cash (O2C) process. We construct a hypothetical normative ETOT Graph representing the expected process model (Figure 7.6) and a realistic ETOT Graph derived from observed data, which includes some deviations (Figure 7.7). We then identify and analyze the non-conformities based on the approach described in Definition 65.

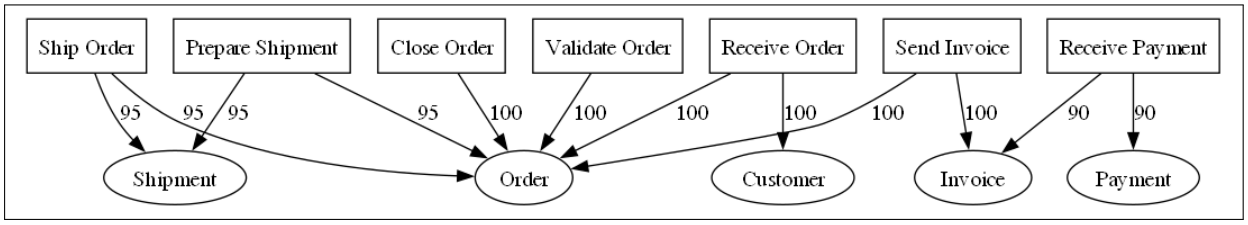


Figure 7.6: Hypothetical normative ETOT Graph for an O2C process. The arcs are annotated with the expected frequencies of the associations.

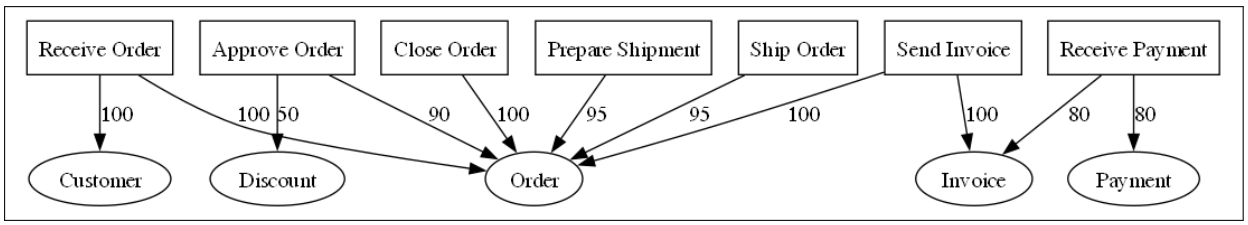


Figure 7.7: Realistic ETOT Graph extracted from an object-centric event log for an O2C process. The arcs are annotated with the observed frequencies of the associations.

Analysis of the Normative and Realistic ETOT Graphs

The normative ETOT Graph (Figure 7.6) represents the expected interactions between event types and object types in the O2C process. The realistic ETOT Graph (Figure 7.7) reflects the actual process execution, including deviations from the normative model.

Node Conformance **Missing Activities** (A_{missing}):

- `Validate Order`: Expected but not observed in the real-life ETOT Graph.

Additional Activities ($A_{\text{additional}}$):

- `Approve Order`: Observed in the real-life ETOT Graph but not defined in the normative model.

Missing Object Types (OT_{missing}):

- `Shipment`: Expected in the normative model but not observed in the real-life ETOT Graph.

Additional Object Types ($OT_{\text{additional}}$):

- `Discount`: Observed in the real-life ETOT Graph but not defined in the normative model.

Edge Conformance **Missing Relationships** (R_{missing}):

- (`Validate Order`, `Order`)
- (`Prepare Shipment`, `Shipment`)
- (`Ship Order`, `Shipment`)

These relationships are expected in the normative model but not observed in the real-life ETOT Graph.

Additional Relationships ($R_{\text{additional}}$):

- (`Approve Order`, `Order`)
- (`Approve Order`, `Discount`)

These relationships are observed in the real-life ETOT Graph but not defined in the normative model.

Edge Frequency Conformance For relationships present in both graphs ($R_M \cap R_L$), we compute the relative frequency differences and identify those exceeding a predefined threshold ($\theta_{\text{rel}} = 10\%$).

Relationship	**Description**	$w_M(r)$	$w_L(r)$	$\Delta_{\text{rel}}(r)$
(`Receive Order`, `Order`)	Expected	100	100	0.00%
(`Receive Order`, `Customer`)	Expected	100	100	0.00%
(`Prepare Shipment`, `Order`)	Expected	95	95	0.00%
(`Ship Order`, `Order`)	Expected	95	95	0.00%
(`Send Invoice`, `Order`)	Expected	100	100	0.00%
(`Send Invoice`, `Invoice`)	Expected	100	100	0.00%
(`Receive Payment`, `Invoice`)	Expected	90	80	11.11%
(`Receive Payment`, `Payment`)	Expected	90	80	11.11%
(`Close Order`, `Order`)	Expected	100	100	0.00%

Table 7.1: Relative frequency differences for relationships present in both graphs.

Non-Conforming Relationships in Frequency:
The following relationships exceed the 10% threshold:

- (`Receive Payment`, `Invoice`): Relative difference of 11.11%.
- (`Receive Payment`, `Payment`): Relative difference of 11.11%.

Fitness Value Calculation Using the fitness function with weighting factors $\alpha = 1$, $\beta = 1$, $\gamma = 1$:

- Total normative activities ($|A_M| = 7$)
- Total normative object types ($|OT_M| = 5$)
- Total normative relationships ($|R_M| = 12$)
- Normalization constant:

$$N = \alpha(|A_M| + |OT_M|) + \beta|R_M| + \gamma|R_M| = 1 \cdot (7 + 5) + 1 \cdot 12 + 1 \cdot 12 = 36$$

- Numerator:

$$\alpha \cdot (|A_{\text{missing}}| + |OT_{\text{missing}}|) + \beta \cdot |R_{\text{missing}}| + \gamma \cdot \sum_r \delta_{\text{rel}}(r) = 1 \cdot (1 + 1) + 1 \cdot 3 + 1 \cdot 2 = 7$$

- Fitness value:

$$\phi(G_L, G_M) = 1 - \frac{7}{36} \approx 0.8056$$

Interpretation The fitness value of approximately 0.806 indicates that the observed ETOT Graph conforms to the normative model to a significant extent, with deviations accounting for approximately 19.44% of the total possible discrepancies.

Summary of Non-Conformities

Structural Deviations:

- **Missing Activity**: `Validate Order`
- **Additional Activity**: `Approve Order`
- **Missing Object Type**: `Shipment`
- **Additional Object Type**: `Discount`
- **Missing Relationships**:
  - (`Validate Order`, `Order`)
  - (`Prepare Shipment`, `Shipment`)
  - (`Ship Order`, `Shipment`)
- **Additional Relationships**:
  - (`Approve Order`, `Order`)
  - (`Approve Order`, `Discount`)

Behavioral Deviations:

- Significant frequency deviations in relationships involving `Receive Payment`, indicating a higher rate of unpaid invoices than expected.

Conclusion

By comparing the normative ETOT Graph with the observed ETOT Graph, we have identified several non-conformities, including missing and additional activities and object types, missing and additional relationships, and significant deviations in relationship frequencies. The calculated fitness value reflects the degree of alignment between the actual process behavior and the expected behavior. This analysis provides valuable insights for process stakeholders to investigate the causes of these deviations and implement corrective measures to improve process conformance.

7.4 Object-Centric Conformance Checking Using Petri Nets

In addition to graph-based approaches, object-centric conformance checking can be conducted using Petri nets adapted to handle multiple object types. Traditionally, conformance checking techniques, such as Token-Based Replay (TBR), have been applied to event logs where each trace represents a single case. In these scenarios, TBR simulates the firing of transitions on a Petri net and tracks produced, consumed, missing, and remaining tokens to evaluate how well the model aligns with the observed behavior.

When shifting from single-case (trace-based) logs to object-centric event logs (OCELs), each event may relate to multiple objects of different types. This complexity can be captured by *object-centric Petri nets*, where tokens can be differentiated by the object types they represent. Instead of tracking a single token flow corresponding to a single process instance, multiple tokens of various object types can move through the network, each modeling the lifecycle of a particular object or set of objects within the process.

For example, consider an order-to-cash scenario where an event may relate both to an *order* object and to one or more *item* objects. An object-centric Petri net would include places and transitions that reflect the relationships and constraints among different object types. During replay, if a transition representing the shipment of an item fires, it consumes tokens from places corresponding to both the item and the order contexts if both must be present. If the required tokens (e.g., an order token) are missing, then a missing token must be inserted, indicating a deviation. Likewise, if tokens remain unconsumed at the end of the process execution, they signal parts of the model that are never completed as observed in the log.

Applying TBR in this object-centric setting follows the same principles outlined for the traditional single-case scenario: tokens are produced and consumed according to transition firings, missing tokens indicate misalignments, and remaining tokens highlight incomplete executions. The key difference is that

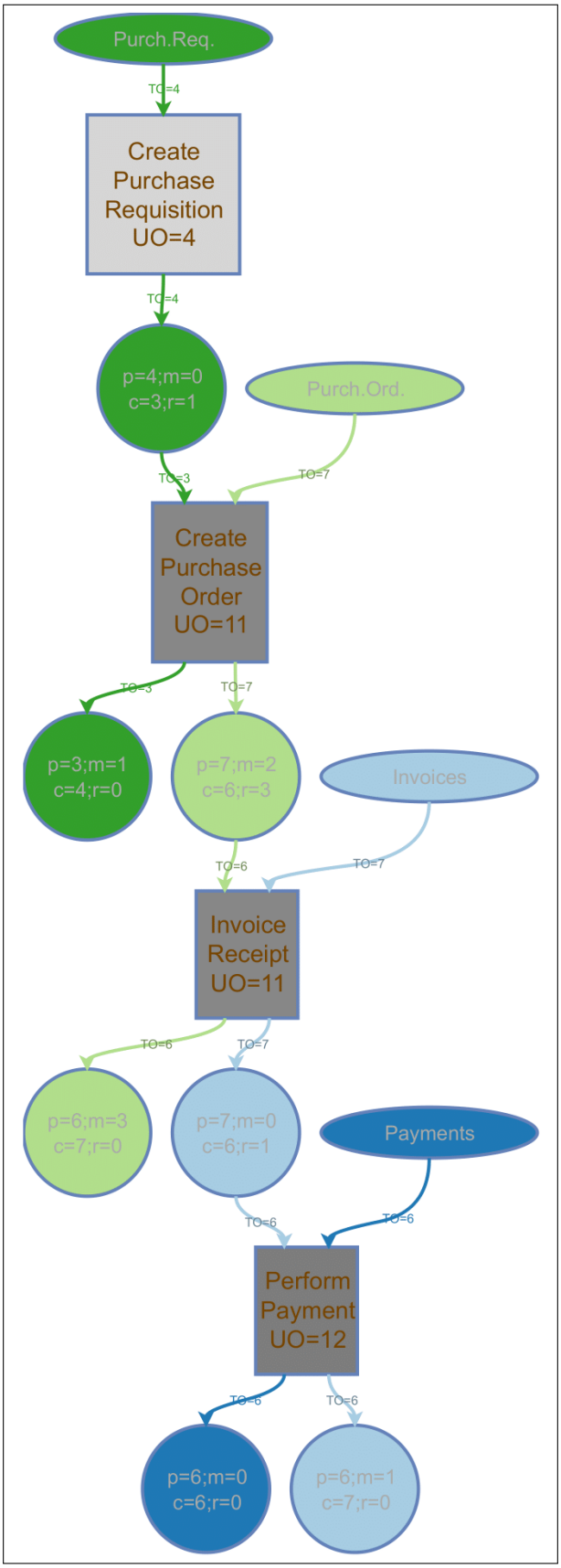


Figure 7.8: OCPN reporting the number of consumed, produced, missing, and remaining tokens for each place.

these tokens now carry the meaning of objects and their relationships, enabling a nuanced understanding of where and how multiple object lifecycles deviate from the normative model. Figure 7.8 illustrates how these counts of tokens are recorded during the replay and how discrepancies manifest in terms of token movements.

By extending TBR to object-centric Petri nets, analysts can leverage the intuitive token-based diagnostic approach to identify whether observed object behaviors align with expected patterns. They can pinpoint precisely which object types or interactions are problematic, leading to more informed remediation strategies. Moreover, because the method aligns closely with the traditional TBR concepts, it can be integrated with existing toolsets and methodologies, while enriching the diagnostic capabilities to reflect the more realistic, multi-dimensional nature of modern event data.

7.5 Object-Centric Anomaly Detection

Anomaly detection in object-centric processes presents unique challenges due to the complex interactions among multiple object types. Unlike traditional event logs where each event is associated with a single case identifier, Object-Centric Event Logs (OCEL) involve events linked to multiple objects of different types. This complexity necessitates specialized methods to detect anomalies that may not be apparent when analyzing individual object lifecycles in isolation.

In this section, we introduce a comprehensive approach to anomaly detection tailored for object-centric event logs. The methodology encompasses:

- **Dimensionality Reduction**: Techniques to manage high-dimensional data while preserving essential information.
- **Feature Selection**: Identifying the most relevant features contributing to anomalies.
- **Anomaly Detection Algorithms**: Selecting algorithms suited for the nature of object-centric data.
- **Correlation Analysis**: Quantifying the impact of features on anomaly scores.
- **Domain Knowledge Integration**: Leveraging expert insights to interpret anomalies within the process context.

7.5.1 Method Selection for Dimensionality Reduction, Feature Selection, and Anomaly Detection

Selecting appropriate methods is crucial for effectively detecting anomalies in object-centric data. Table 7.2 compares various techniques based on their suitability for handling the complexities of object-centric event logs.

Dimensionality Reduction

Principal Component Analysis (PCA) PCA transforms high-dimensional data into a lower-dimensional space by identifying the directions (principal components) that capture the maximum variance in the data. While PCA is effective for reducing dimensionality, it assumes linear relationships among features, which may not fully represent the complex interactions in object-centric data.

t-Distributed Stochastic Neighbor Embedding (t-SNE) t-SNE is a non-linear dimensionality reduction technique that excels at preserving local structures in high-dimensional data. It is particularly useful for visualizing complex patterns and clusters in object-centric event logs. However, t-SNE can be computationally intensive and may not scale well to very large datasets.

Feature Selection

Unsupervised Feature Selection Unsupervised methods select features based on statistical properties such as variance or redundancy. These methods are suitable when there is no specific target variable or prior knowledge about anomalies. They help in reducing noise and focusing on features that contribute most to data variability.

Table 7.2: Comparison of Methods for Dimensionality Reduction, Feature Selection, and Anomaly Detection

Aspect	Method A	Method B
Dimensionality Reduction	**Principal Component Analysis (PCA)**: • Effective for linear correlations. • Reduces dimensionality by capturing variance. • May not capture non-linear relationships inherent in object interactions.	**t-Distributed Stochastic Neighbor Embedding (t-SNE)**: • Preserves local structures in data. • Captures complex non-linear patterns. • Computationally intensive for large datasets.
Feature Selection	**Unsupervised Methods**: • Variance Thresholding. • Selects features based on intrinsic properties. • Useful when no target variable is specified.	**Supervised Methods**: • Mutual Information, Correlation Coefficients. • Considers relationship with target variable. • Effective when anomalies are defined concerning specific outcomes.
Anomaly Detection Algorithms	**Isolation Forest**: • Efficient for high-dimensional data. • Isolates anomalies based on random partitions. • Suitable for individual object analysis.	**Local Outlier Factor (LOF)**: • Detects anomalies based on local density deviations. • Effective for capturing anomalies from object interactions. • Sensitive to choice of neighborhood size.

Supervised Feature Selection Supervised methods leverage the relationship between features and a target variable, such as known anomalies. Techniques like mutual information or correlation analysis identify features that have strong predictive power regarding the target. This approach is beneficial when anomalies are defined concerning specific outcomes or behaviors.

Anomaly Detection Algorithms

Isolation Forest Isolation Forest is an ensemble-based algorithm that isolates anomalies by constructing random decision trees. It operates on the principle that anomalies are data points that are easier to separate from the rest of the data. The algorithm is efficient and scales well to large, high-dimensional datasets.

Local Outlier Factor (LOF) LOF measures the local deviation of a data point's density relative to its neighbors. An object is considered an anomaly if it has a significantly lower density than its neighbors. LOF is effective in detecting anomalies arising from unusual interactions between objects but may require careful tuning of parameters like the number of neighbors.

7.5.2 Correlating Features with Anomaly Scores

To understand how different features contribute to the anomalies detected, we compute anomaly scores for each object and analyze the correlation between these scores and the features.

Definition 66 (Anomaly Scores and Ranking)**.** *Given an OCEL L and an anomaly detection model that assigns a score $score(o) \in \mathbb{R}$ to each object $o \in O$, we rank the objects based on their scores. Higher scores indicate higher likelihoods of being anomalous.*

Definition 67 (Feature Normalization)**.** *For each feature f, we normalize the values across all objects to ensure comparability:*

$$f_{norm}(o) = \begin{cases} 0, & \text{if } max_{\ o'} f(o') - min_{\ o'} f(o') = 0 \\ \dfrac{f(o) - min_{\ o'} f(o')}{max_{\ o'} f(o') - min_{\ o'} f(o')} & \text{otherwise} \end{cases}$$

This scales the feature values to the range $[0, 1]$.

Definition 68 (Feature-Anomaly Correlation)**.** *The correlation between a normalized feature f_{norm} and the anomaly scores score is calculated using Pearson's correlation coefficient:*

$$corr(f, score) = \frac{\sum_{o \in O}(f_{norm}(o) - \overline{f_{norm}})(score(o) - \overline{score})}{\sqrt{\sum_{o \in O}(f_{norm}(o) - \overline{f_{norm}})^2}\sqrt{\sum_{o \in O}(score(o) - \overline{score})^2}}$$

where $\overline{f_{norm}}$ and $\overline{score}$ are the mean values of the normalized feature and the anomaly scores, respectively.

Features with high absolute correlation values are considered to have a significant impact on the anomaly detection results.

7.5.3 Incorporating Domain Knowledge

Domain expertise is essential for interpreting anomalies detected in object-centric event logs. By examining the process execution related to anomalous objects, experts can uncover underlying issues that automated methods may not detect. For an object $o \in O$, the *process execution* includes all events and related objects that are connected to o through direct or indirect interactions. This subset of the OCEL provides context for analyzing the object's behavior within the process. Analyzing the process execution helps in:

- Identifying unusual event sequences or patterns.
- Detecting deviations in lifecycle durations.
- Understanding the impact of interactions with other objects.
- Providing insights into potential process inefficiencies or compliance issues.

7.5.4 Application: Anomaly Detection in a Purchase-to-Pay Process

We apply the proposed anomaly detection approach to a Purchase-to-Pay (P2P) process, which involves multiple object types such as purchase orders, invoices, and goods receipts.

Data Preparation

An OCEL is extracted from a P2P system (Oracle EBS), focusing on key object types:

- Purchase Order
- Purchase Requisition
- Invoice
- Goods Receipt
- Payment

Relevant features are engineered for each object, including:

- **Lifecycle Duration**: Time between the first and last events related to the object.
- **Event Count**: Number of events associated with the object.
- **Interaction Count**: Number of other objects the object interacts with.
- **Activity Frequencies**: Counts of specific activities involving the object.

Anomaly Detection and Feature Correlation

We apply the Isolation Forest algorithm to the prepared data to compute anomaly scores for each object. Features are normalized, and correlations with anomaly scores are calculated.

Table 7.3: Features Highly Correlated with Anomaly Scores

Feature	**Correlation Coefficient**
Lifecycle Duration	0.68
Event Count	0.55
Interaction Count	0.72
Frequency of *Cancel Purchase Order* Activity	0.80

The high correlation between the frequency of the *Cancel Purchase Order* activity and anomaly scores suggests that orders with cancellations are more likely to be anomalous.

Analyzing Anomalous Objects

For objects with high anomaly scores, we extract their process executions and analyze them in detail.

Consider a purchase order with an unusually high anomaly score. Examination of its process execution reveals:

- **Multiple Cancellations and Re-approvals**: The order was canceled and re-approved several times, which is atypical.
- **Extended Lifecycle Duration**: The order remained open for an extended period compared to the average lifecycle.
- **High Interaction Count**: The order interacted with a larger number of invoices and goods receipts than usual.

These findings may indicate process inefficiencies, potential compliance issues, or fraudulent activities requiring further investigation.

Domain Expert Interpretation

By involving domain experts, the anomalies detected can be contextualized within the business process:

- **Process Deviations**: The repeated cancellations and re-approvals may violate company policies.
- **Compliance Risks**: Extended lifecycles and unusual interactions may expose the organization to financial risks.
- **Operational Inefficiencies**: Identifying bottlenecks or errors in the procurement process.

Experts can recommend corrective actions, such as process audits, staff training, or system improvements.

7.5.5 Conclusion

Object-centric anomaly detection provides valuable insights into complex processes involving multiple interacting objects. By combining advanced analytical techniques with domain knowledge, organizations can identify and address anomalies that may affect efficiency, compliance, and overall performance.

7.6 Related Work

This chapter introduced novel object-centric conformance checking approaches using OCDFGs, Object Type Graphs, and ETOT Graphs. While building on the foundation of traditional process mining and conformance checking, these approaches address the unique challenges of object-centric event data in ways that differ from existing research. Several key related works are discussed below, highlighting their distinct contributions and contrasting them with the methods presented in this chapter.

In [7], conformance checking techniques for object-centric process mining are introduced, utilizing the concept of "event context" to capture the history of related objects. While this acknowledges the importance of object interplay, their approach differs from ours by using object-centric Petri nets and employing a replay algorithm for conformance checking calculations. In contrast, this chapter leverages graph-based representations (OCDFGs, Object Type Graphs, and ETOT Graphs) and utilizes graph comparison techniques, offering a different perspective on representing and analyzing object-centric processes.

The paper [98] addresses constraint monitoring in object-centric business processes using Object-Centric Constraint Graphs (OCCGs). While their focus on quantifying relationships between activities and object involvement aligns with the object-centric perspective of this chapter, their work centers on constraint violation detection rather than conformance checking. Our approaches, on the other hand, aim to quantify the overall alignment between observed behavior and normative models, providing a different type of process analysis.

Object-centric alignments are introduced in [97], extending the concept of alignments to handle the complexities of multiple interacting objects. This work provides a trace-level analysis of deviations using a synchronous product net approach. In contrast, the methods presented in this chapter operate at a higher level of abstraction, focusing on aggregate behavior represented by graphs rather than individual trace alignments. This difference in granularity provides complementary insights into process conformance.

Another contribution found in [97] introduces a technique for conformance checking and performance analysis using projected DFGs. This approach decomposes the object-centric problem into multiple standard conformance checking problems, one for each object type. While we also utilize the concept of projected DFGs, our methods differ by re-combining the individual object-type analyses to provide a holistic object-centric perspective, whereas their work primarily focuses on separate analyses for each object type. Furthermore, their utilization of workflow nets and the A* algorithm contrasts with the direct graph comparison methods employed in this chapter.

In conclusion, these related works demonstrate the diverse approaches being explored in object-centric process mining. The techniques presented in this chapter offer distinct methods for representing and analyzing object interactions, quantifying conformance, and detecting deviations, providing valuable complements to existing techniques and opening avenues for future research in object-centric process analysis.

Conclusion

This chapter presented a comprehensive framework for Object-Centric Conformance Checking (OCCC) and anomaly detection, addressing the inherent complexities of processes involving multiple interacting objects. We demonstrated how OCDFGs, Object Type Graphs, and ETOT Graphs provide valuable perspectives for analyzing process conformance, revealing discrepancies between observed and expected behavior in terms of activities, object interactions, and their frequencies. The introduced fitness functions allow for quantifying conformance levels, facilitating objective comparisons and highlighting areas of deviation. The O2C process examples illustrated the practical application of these methods, showcasing how the identified non-conformities can inform process improvement initiatives. Furthermore, the chapter detailed an approach for object-centric anomaly detection, emphasizing the importance of combining analytical techniques with domain knowledge to interpret unusual object behavior and pinpoint potential process inefficiencies, compliance risks, or even fraudulent activities. By leveraging the methods presented in this chapter, organizations can gain a deeper understanding of their object-centric processes, identify areas for improvement, and ultimately enhance process efficiency, compliance, and overall performance. Future research directions include developing more sophisticated anomaly detection techniques tailored to object-centric data and exploring the integration of OCCC with other process mining tasks such as predictive monitoring and process optimization.

Chapter 8

Real-world Applications and Case Studies

"In theory, there is no difference between theory and practice. But, in practice, there is."
Jan L. A. van de Snepscheut

Introduction

In the preceding Chapters 3–7, the theoretical and methodological foundations of Object-Centric Process Mining (OCPM) were established: object-centric event logs, techniques for object extraction and linking (Chapter 5), approaches to object-centric process discovery (Chapter 6), object-centric conformance checking and compliance analysis (Chapter 7), and object-centric performance analysis. These chapters introduced the formalisms and preprocessing methods that ensure data quality and analytical depth.

Building upon these foundations, this chapter applies the discussed techniques to real-world data extracted from SAP ECC ERP systems. In particular, the author of this thesis has worked on two contributions that are included in this chapter:

- [19] provides a semi-automated methodology for extracting object-centric event data from relational databases supporting SAP ERP, incorporating scalability and quality assessments to achieve robust data preparation before process discovery and analysis.
- [14] demonstrates how to practically apply OCPM in a Purchase-to-Pay scenario from SAP ECC ERP data, applying the modeling and analysis frameworks introduced earlier in the thesis to gain actionable insights into procurement processes.

These contributions directly inform the selection of techniques and frameworks applied in this chapter. They illustrate how the object-centric methods previously discussed can be used to examine complex processes, such as Purchase-to-Pay (P2P) and Order-to-Cash (O2C), in a realistic setting. General ERP and SAP ECC concepts are presented, and then object-centric methods are systematically employed on both a generic scenario and a dedicated case study at ECE Group. By integrating process models and conformance checking results directly into the analysis, the chapter shows how to transition from theory to tangible, data-driven improvements.

This approach ensures that P2P and O2C are not treated as isolated examples but rather as contexts in which the object-centric techniques perform under realistic conditions. It further clarifies the difference between generic SAP/P2P/O2C elements and the specific details of the ECE environment, thereby offering valuable lessons for both practitioners and researchers.

8.1 SAP ECC ERP

In Chapter 3, we introduced the fundamental concepts of event data extraction and object identification from enterprise systems. Here, we focus on SAP ECC ERP, a real-world system that encapsulates the complexity and richness of integrated processes. Before we apply the object-centric modeling, discovery, and conformance checking techniques from Chapters 5–7, this section provides a high-level overview of SAP ECC and its database structure. We distinguish between general ERP-level concepts (applicable

to any SAP ECC environment) and the particular processes and configurations that will be used in subsequent case studies.

By clearly outlining the various modules, tables, and data-change tracking mechanisms (e.g., change documents) in SAP ECC, we set the stage for applying our previously discussed object-centric event extraction methods. In subsequent sections, we will refer back to this overview when demonstrating how to extract object-centric event logs (OCELs) and apply the process discovery and conformance checking techniques introduced in earlier chapters.

SAP ECC supports a vast array of functional modules, each designed to handle specific business areas or operations:

- *Finance (FI)*: Manages financial transactions and generates financial reports.
- *Controlling (CO)*: Supports planning, reporting, and monitoring of business operations.
- *Sales and Distribution (SD)*: Manages all sales and distribution activities including quotations, sales orders, shipping, billing, and more.
- *Material Management (MM)*: Deals with procurement and inventory management.
- *Production Planning (PP)*: Manages all activities related to production, from planning to execution.
- *Quality Management (QM)*: Deals with quality control and assurance processes.
- *Plant Maintenance (PM)*: Manages maintenance processes in an organization.
- *Human Resources (HR)*: Handles recruitment, personnel administration, payroll, training, and more.
- *Project System (PS)*: Helps plan, manage and track projects.
- *Customer Service (CS)*: Deals with after-sales support services for customers.

SAP ECC can be customized to cater to industry-specific requirements, providing the ability to handle unique business processes and compliance standards. Many businesses of various sizes across different sectors use SAP ECC to efficiently manage their operations and adapt to changing market needs.

An important aspect of SAP's database design is the use of change tables (or change documents) to capture changes made to master data or transaction data. This is a peculiar and highly useful feature that lets administrators track all modifications made to important data fields.

When changes are made to monitored fields, the system automatically creates a change document that records the following details:

- The name of the table and the field that was changed.
- The type of change (e.g., insert, modify, delete).
- The old and new values of the field.
- The user who made the change.
- The date and time of the change.

The main tables associated with change documents are CDHDR (Change Document Header) and CDPOS (Change Document Items). CDHDR stores information about the changes at the header level, including the transaction code, the user who made the changes, and the time of the change. CDPOS stores the detailed changes at the field level, including the table and field names and the old and new field values.

This feature enables a high level of auditability and traceability, as it provides a complete history of changes, which is critical for many business processes and compliance requirements.

8.1.1 P2P process (Case Study)

The Purchase-to-Pay (P2P) process, introduced in a general sense in Chapter 3, now becomes a focal point for applying the object-centric methodologies from Chapters 6–7. In this subsection, we first present the generic P2P process as implemented in SAP ECC. Then, we highlight how the techniques developed in the previous chapters—such as object identification, event-to-object mappings, object-centric modeling, and conformance checking—can be systematically applied to a real P2P scenario.

This sets the stage for introducing the ECE Group case study. We will explicitly distinguish between the generic P2P implementation in SAP (applicable to many organizations) and the specific ECE case details. This distinction allows us to show how the general methods introduced earlier can be tailored and refined to handle the complexities of a particular organizational environment.

Implementation of the Process in SAP ERP

In Chapters 5 and 6, we described how to move from raw database tables to object-centric event logs and object-centric process models. Here, we apply these principles by detailing how the P2P process is realized within SAP ECC. We identify the relevant modules, transactions, and database tables, and we map these to object types and attributes, as introduced in our earlier theoretical chapters.

Crucially, this section remains process-generic. It does not yet focus on the ECE case study but shows how any SAP ECC system supports the P2P cycle. Later, we will apply our extraction techniques (Chapter 5) to retrieve OCELs from these tables and then use object-centric discovery (Chapter 6) and conformance checking methods (Chapter 7) to analyze the actual execution of the process.

The Purchase-to-Pay (P2P) intersects with various SAP ECC modules:

- *Material Management (MM)*: The P2P process is fundamentally grounded in the MM module. It starts with a need for materials or services, leading to the creation of a purchase requisition. This then gets converted into a purchase order, which is sent to the supplier. The MM module effectively manages these procurement processes, from vendor selection to purchase order management.
- *Finance (FI)*: Once the goods or services have been received and checked, an invoice is received from the supplier. The FI module comes into play here, as it handles the processing and reconciliation of the supplier's invoice. When the invoice is approved, the FI module also handles the payment processing to the supplier, closing out the P2P process.
- *Sales and Distribution (SD)*: Although the SD module is mainly focused on the company's outbound processes, it is indirectly related to the P2P process. For example, in a manufacturing organization, the materials procured through the P2P process feed into the production of goods, which then become part of the SD process.
- *Controlling (CO)*: The CO module assists in monitoring and controlling the costs associated with the P2P process. It can track costs at each stage, from the initial purchase requisition to the final payment, aiding in budget control and financial reporting.
- *Quality Management (QM)*: Once the goods are received from the supplier, they need to be inspected for quality before they can be used in the organization's operations. The QM module manages these quality inspection processes, ensuring that the received goods meet the requisite quality standards.
- *Plant Maintenance (PM) and Project System (PS)*: The PM and PS modules can also be connected to the P2P process. For instance, the procurement of materials or services could be directly related to a maintenance task or project, and these modules would manage those processes.

The Purchase-to-Pay process in SAP relies on several key database tables, each housing specific fields pertinent to the process. These tables and their fields contribute significantly to the process flow, providing a structured, logical, and efficient mechanism for managing and tracking the procurement and payment activities.

- **EKKO, EKPO (Purchase Order)**: The EKKO table contains the Purchase Order Header data and EKPO table contains Purchase Order Item data. They are integral parts of the Purchasing module in SAP.
  - **EBELN (EKKO)**: This field stores the Purchase Order Number. It uniquely identifies a Purchase Order in the SAP system.

  - **EBELP (EKPO)**: This field represents the line item number in the Purchase Order. Each line item represents a distinct material or service being ordered.

- **LIPS (Delivery)**: This table in SAP contains the Delivery Document Line Item Data.
  - **VBELN**: This field represents the Delivery Document number. It uniquely identifies a Delivery Document in the system.
  - **POSNR**: This field represents the line item number in the Delivery Document. Each line item represents a distinct material or service being delivered.

- **EKBE (Goods Receipt)**: This table in SAP contains the History per Purchasing Document. It is used to store changes in Purchasing Document like Goods Receipt.
  - **EBELN**: This field stores the Purchase Order Number. It uniquely identifies a Purchase Order in the SAP system.
  - **EBELP**: This field represents the line item number in the Purchase Order. Each line item represents a distinct material or service being ordered.
  - **GJAHR**: This field corresponds to the Fiscal Year in which the document was posted.
  - **BELNR**: This field is the Document Number for Accounting documents.

- **RBKP, RSEG (Invoice)**: The RBKP table holds the Invoice Receipt Header data and the RSEG table holds the Invoice Receipt Item data.
  - **BELNR (RBKP)**: This field corresponds to the Accounting Document Number. It uniquely identifies an Invoice in the SAP system.
  - **GJAHR (RSEG)**: This field corresponds to the Fiscal Year of the Document in the SAP system.
  - **BUKRS (RBKP)**: This field represents the Company Code. The Company Code is an organizational unit within the financial accounting module.

- **BSEG (Payment)**: The BSEG table in SAP holds the Accounting Document Segment data, which is integral to financial transactions in the system.
  - **BELNR**: This field corresponds to the Accounting Document Number. It uniquely identifies a financial transaction in the SAP system.
  - **GJAHR**: This field represents the Fiscal Year of the Document.
  - **BUKRS**: This field represents the Company Code. The Company Code is an organizational unit within the financial accounting module.

- **LFA1 (Supplier)**: The LFA1 table in SAP contains the Vendor Master data (General section). This includes information about vendors that supply a company.
  - **LIFNR**: This field corresponds to the Account Number of the Vendor or Creditor in SAP. It uniquely identifies a supplier in the system.

- **LFBK (Bank)**: This table in SAP stores the Bank Details for the Vendor Master.
  - **BANKS**: This field represents the Bank Keys, which are used to define bank details in the system.
  - **BANKN**: This field stores the Bank Number within the bank key.

- **EKKO, EKPO (Contract)**: The EKKO table contains the Purchasing Document Header data and the EKPO table contains the Purchasing Document Item data. They also store contract data in the SAP system.
  - **KONNR (EKKO)**: This field represents the Contract Number in SAP. It uniquely identifies a contract in the system.
  - **KTPNR (EKPO)**: This field represents the Item Number of Contract in SAP. It identifies a specific item within a contract.

- **LAGP (Warehouse)**: The LAGP table in SAP contains the Storage Bin Master Record data. It is used to manage and track the storage locations within a warehouse.
  - **LGPLA**: This field corresponds to the Storage Bin. It uniquely identifies a particular storage location in a warehouse in the system.
- **MCHB (Inventory)**: The MCHB table in SAP contains the Batch Stock data. It is used to manage batch-level inventory in the system.
  - **MATNR**: This field corresponds to the Material Number. It uniquely identifies a material in the SAP system.
  - **CHARG**: This field corresponds to the Batch Number. It uniquely identifies a batch of a specific material.
- **QALS (Quality Control)**: The QALS table in SAP contains the Inspection Lot Record data. It is used for quality management and inspection in the system.
  - **AUFNR**: This field corresponds to the Order Number. It uniquely identifies an order for inspection in the quality management module.

Table 8.1 presents a comprehensive overview of the various object types involved in the Purchase-to-Pay process within the SAP system. It highlights the corresponding SAP tables and identifies the specific object identifiers that play an essential role in the flow of the P2P process.

Object Type	**SAP Table**	**Object Identifier**
Purchase Requisition	EBAN	BANFN (Number) + BNFPO (Item)
Purchase Order	EKKO, EKPO	EKKO.EBELN (Number) + EKPO.EBELP (Item)
Delivery	LIPS	LIPS.VBELN (Number) + LIPS.POSNR (Item)
Goods Receipt	EKBE	EKBE.EBELN (PO Number) + EKBE.EBELP (PO Item) + EKBE.GJAHR (Fiscal Year) + EKBE.BELNR (Document Number)
Invoice	RBKP, RSEG	RBKP.BELNR (Document Number) + RSEG.GJAHR (Fiscal Year) + RBKP.BUKRS (Company Code)
Payment	BSEG	BSEG.BELNR (Document Number) + BSEG.GJAHR (Fiscal Year) + BSEG.BUKRS (Company Code)
Supplier	LFA1	LFA1.LIFNR (Account Number)
Bank	LFBK	LFBK.BANKS (Bank Key) + LFBK.BANKL (Bank number)
Contract	EKKO, EKPO	EKKO.KONNR (Contract Number) + EKPO.KTPNR (Item)
Warehouse	LAGP	LAGP.LGPLA (Storage Bin)
Inventory	MCHB	MCHB.MATNR (Material) + MCHB.CHARG (Batch)
Quality Control	QALS	QALS.AUFNR (Order Number)

Table 8.1: Table summarizing the extraction process for objects of different object type in the P2P process.

Table 8.2 showcases the extraction process of qualified Object-to-Object relationships within the P2P process in SAP. It details the link between different SAP tables and their respective object identifiers, capturing the sequence of actions that transform a purchase requisition into a successful payment.

Table 8.3 outlines the extraction process of the events and their related objects for the P2P process in SAP. Each row in the table represents an activity, elaborating on its associated tables, the specific activity fields, the timestamp fields, and the set of related objects involved in each activity. This table provides a detailed roadmap for the Event-to-Object relationships that make up the backbone of the P2P process.

Qualifier	SAP Table	Source Object Identifier	Target Object Identifier
Purchase Requisition Generates Purchase Order	EBAN, EKKO, EKPO	EBAN.BANFN + EBAN.BNFPO	EKKO.EBELN + EKPO.EBELP
Purchase Order Triggers Delivery	EKES, LIPS	EKES.EBELN + EKES.EBELP	LIPS.VBELN + LIPS.POSNR
Purchase Order Sent To Supplier	EKKO, LFA1	EKKO.EBELN + EKPO.EBELP	LFA1.LIFNR
Supplier Fulfills Delivery	LFA1, EKKO, EKPO	LFA1.LIFNR	EKKO.EBELN + EKPO.EBELP
Delivery Results In Goods Receipt	EKKO, EKPO, EKBE	EKKO.EBELN + EKPO.EBELP	EKBE.EBELN + EKBE.EBELP + EKBE.GJAHR + EKBE.BELNR
Goods Receipt Triggers Invoice	EKBE, RBKP	EKBE.EBELN + EKBE.EBELP + EKBE.GJAHR + EKBE.BELNR	RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS
Invoice Requires Payment	RBKP, BSEG	RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS
Invoice Sent To Bank	RBKP, LFBK	RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS	LFBK.BANKS + LFBK.BANKL
Bank Processes Payment	LFBK, BSEG	LFBK.BANKS + LFBK.BANKL	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS
Purchase Order Bound By Contract	EKKO, EKKO	EKKO.EBELN + EKPO.EBELP	EKKO.KONNR + EKPO.KTPNR
Contract Applies To Supplier	EKKO, LFA1	EKKO.KONNR + EKPO.KTPNR	LFA1.LIFNR
Goods Receipt Updates Stock In Warehouse	EKBE, LAGP	EKBE.EBELN + EKBE.EBELP + EKBE.GJAHR + EKBE.BELNR	LAGP.LGPLA
Warehouse Holds Inventory	LAGP, MCHB	LAGP.LGPLA	MCHB.MATNR + MCHB.CHARG
Supplier Subjected To Quality Control	LFA1, QALS	LFA1.LIFNR	QALS.AUFNR
Quality Control Approves Delivery	QALS, LIPS	QALS.AUFNR	LIPS.VBELN + LIPS.POSNR

Table 8.2: Table summarizing the extraction process of the qualified Object-to-Object relationships for the P2P process.

Research Questions and the P2P Process at ECE Group

Having established a generic P2P process view in SAP ECC, we now focus on the specific application at ECE Group. Here, we formulate research questions that leverage the techniques introduced in earlier chapters to address ECE's unique challenges. While Chapter 3 discussed how to frame process-related inquiries and Chapters 6–7 explained how to discover and analyze object-centric models, we now put these methods into practice.

We carefully separate what is generally true for SAP-based P2P processes from the particularities of ECE's organizational structure, data quality, and business constraints. By doing so, we illustrate how the generic methods and models can be tailored to elicit meaningful insights in a real-world setting.

The ECE Group, a major player in the European shopping center sector since 1965, manages assets worth 31 billion Euros across 200 shopping centers in 13 countries. In 2020, aiming to optimize its data-intensive operations, ECE focused on its Purchase-to-Pay (P2P) process—a critical link between procurement and Accounts Payable (AP). They established a dedicated "process insights" team and adopted Celonis as a process mining tool. However, challenges arose in managing process variations across multiple countries and understanding complex process patterns. To address this, a team of experts leveraged Object-Centric Process Mining (OCPM) to analyze the P2P process, focusing on Process Performance, Compliance, and Quality.

To pinpoint areas for improvement using OCPM, particularly where traditional process mining fell short in managing interdependencies, the following research questions were established:
Process Performance (PP): Focusing on efficiency and throughput.

PP1 What is the current processing capacity of the accounts payable department?

PP2 How long does it take to process/verify an invoice?

PP3 What percentage of purchase orders are "no-touch orders"?

PP4 What is the average throughput time from purchase order to goods receipt or payment for finalized orders?

Activity	Involved Tables	Activity Field	Timestamp Field	Set of Related Objects
Create Purchase Requisition	EBAN	EBAN.BANFN	EBAN.ERDAT	(EBAN.BANFN + EBAN.BNFPO)
Approve Purchase Requisition	EBAN	EBAN.BANFN	EBAN.FRGDT	(EBAN.BANFN + EBAN.BNFPO)
Create Purchase Order	EKKO, EBAN	EKKO.EBELN	EKKO.BEDAT	(EBAN.BANFN + EBAN.BNFPO, EKKO.EBELN + EKPO.EBELP)
Goods Receipt	LIPS, EKKO	LIPS.VBELN	LIPS.ERDAT	(EKKO.EBELN + EKPO.EBELP, LIPS.VBELN + LIPS.POSNR)
Check Goods Receipt	EKBE, EKKO	EKBE.EBELN	EKBE.BUDAT	(EKKO.EBELN + EKPO.EBELP, EKBE.EBELN + EKBE.EBELP + EKBE.GJAHR + EKBE.BELNR)
Invoice Receipt	RBKP, EKBE	RBKP.BELNR	RBKP.BLDAT	(EKBE.EBELN + EKBE.EBELP + EKBE.GJAHR + EKBE.BELNR, RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS)
Invoice Posted	RBKP	RBKP.BELNR	RBKP.BUDAT	(RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS)
Clearance Posted (Payment)	BSEG, LFBK, RBKP	BSEG.BELNR	BSEG.AUGDT	(RBKP.BELNR + RBKP.GJAHR + RBKP.BUKRS, LFBK.BANKS + LFBK.BANKL, BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS)

Table 8.3: Table summarizing the extraction process of the events, and the Event-to-Object relationships, for the P2P process.

PP5 What is the end-to-end P2P process performance?

PP6 Which activities correlate with high processing times?

PP7 Does high workload in purchasing or AP extend processing times?

Process Compliance (PC): Ensuring adherence to established procedures.

PC1 Is there maverick buying in the process?

PC2 How many purchase requisitions were modified after the purchase order to match information?

Process Quality (PQ): Assessing task quality and identifying deviations.

PQ1 How many orders have multiple invoices, causing extra work for AP?

PQ2 What are the unforeseen patterns in the P2P process execution?

Techniques Used During the Case Study

During our study, we undertook a methodological and robust approach towards both preprocessing and analyzing our data. We present these techniques and the insights drawn from each of them.

Data Preprocessing Upon obtaining an OCEL for the P2P process, we identified that it might contain incomplete executions, which might affect the reliability of our subsequent analyses. Our aim was to gain a clear and complete understanding of the process. Thus, it became essential to eliminate these incomplete executions. For this purpose, we developed an Object Interaction Enrichment. In this graph, each node represents an object from the OCEL, and the edges signify the interactions between these objects. These interactions were determined based on events in the log where multiple objects were involved. We could then segment our event log based on the connected components in this graph. By doing so, we achieved a cleaner dataset by retaining only the events associated with objects that are part of a single connected component. Moreover, our focus was on the end-to-end process ranging from purchase requisition to payment. To align our data with this focus, we particularly concentrated on components that had at least a purchase order object and a payment object.

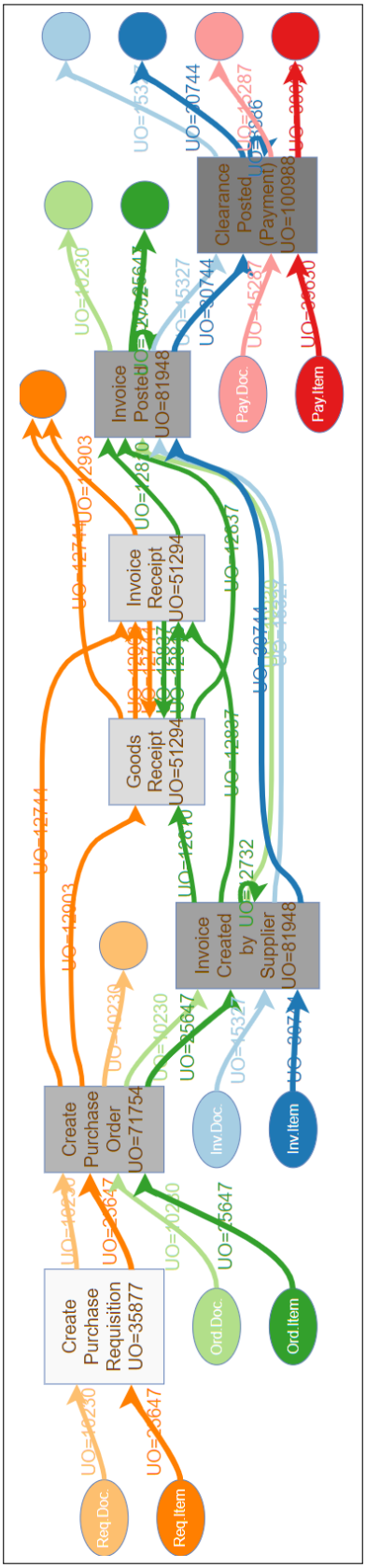


Figure 8.1: OCDFG discovered on the OCEL extracted during the case study.

Analysis Techniques The analysis started with the application of traditional object-centric process discovery techniques (see Figure 8.1). Then, our analysis was structured into two main methodologies: *graph-based* and *statistical analyses*. The decision to not prioritize object-centric directly-follows graphs or object-centric Petri nets (presented in Chapter 6) in the case study was driven by their limited analytical capabilities for the specific research goals. While valuable for understanding control flow and object interactions, these techniques were insufficient for addressing questions related to throughput times, correlations with processing time and workload, and the detection of outlier patterns. The research prioritized the development and application of novel analysis techniques specifically designed for object-centric event logs. These included new graph-based representations and statistical analyses that offered a more promising path towards achieving the research objectives.
Graph-Based Analyses:

While our preprocessing introduced the concept of the Object Interaction Enrichment, our graph-based analysis added another layer to this, the Object Creation Enrichment. This new graph allowed us to understand the logical progression of objects. It showed how objects linked over time, capturing the essence of the lifecycle of each object. It was particularly useful in identifying the long-term dependencies between objects, which gave us a structured view of object creation and linkage.

The Object Creation Enrichment was pivotal in addressing various goals:

- Analyzing long-term dependencies between objects to inform process performance.
- Identifying deviations from the standard order-to-invoice flow.
- Studying the time duration between the creation of purchase requisitions and their corresponding orders.
- Pinpointing orders that have been continued into more than one invoice, an indicator of potential quality issues.

Statistical Analyses:

To answer specific questions that couldn't be discerned from the graph-based approach alone, we turned to statistical methods. Using the OC-PM tool, we performed:

- Correlation studies to understand activities associated with high processing time and assess if a high workload led to increased processing times.
- Anomaly detection to spot outlier patterns in the business process execution. Using techniques like isolation forests, we could categorize objects as anomalous or non-anomalous. This helped in identifying unique patterns associated with these anomalies.

Furthermore, to address specific analysis goals, we executed SQL queries in the OC-PM tool. These queries facilitated detailed insights into:

- The processing capacity of the accounts payable department.
- The average time required by the accounts payable department to process a single invoice document.
- Identifying procurement orders that were correctly inserted at the inception, eliminating the need for later changes.

Insights Obtained During the Case Study

The analytical juxtaposition reveals certain advantages of OCPM over traditional process mining methodologies:

- We could ascertain the exact number of documents at every stage of the process (**PP1**). This particularly pertains to the distinct count of purchase requisitions. Some purchase requisitions contained items that were subsequently procured through different purchase orders. Consequently, the count of purchase requisitions deduced through the Celonis software was considerably larger than the actual count of purchase requisitions, a result of the convergence problem.
- We determined that the end-to-end performance (**PP5**) of the Purchase-to-Pay process was distorted by the reversal of some payments. This functionality was unsupported by the traditional extractor due to the computational burden of additional table joins, but it was feasible in the object-centric setting.

Furthermore, we discovered some intriguing compliance/quality patterns during the assessed timespan:

- Maverick Buying (**PC1**) is a harmful practice where standard purchase order approval steps are bypassed and the order is directly placed with the supplier. Following the receipt of an invoice from the supplier, the purchase order is generated. We could detect a significant occurrence of maverick buying in our process.
- Post-Mortem Alterations to Purchase Requisitions (**PC2**): another harmful behavior noted in the process is the modification of purchase requisitions after their approval to match the amounts or quantities of the purchase order.
- Orders with Duplicate Invoices (**PQ1**): we could identify some orders with duplicate invoices in our system. Also, some orders with an abundance of different invoices exist. Upon investigation, we found that they are *maintenance contracts*. An instance of a maintenance contract is a cleaning contract, repeated weekly under the same conditions and invoiced monthly.

Additionally, the statistical analyses also provided us with intriguing insights:

- We could detect a significant correlation (**PP6**) between the presence of change activities (for both orders and invoices) and the processing time. Specifically, modifications in the invoice amount payable or the expected amounts for the orders, have a particularly detrimental impact on the processing time.
- In our process and within the evaluated timespan, the workload has a high correlation (correlation coefficient 0.89) with the processing time (**PP7**). Therefore, a high workload in the process (number of open documents) contributes to an extended processing time.
- Considering **PQ2**, we could pinpoint some purchase orders with a notable count of change activities (>= 150) performed on the order. Additionally, we can identify invoices associated with more than 20 orders.

8.1.2 Process Improvement Actions

The analysis results are currently being leveraged to enhance the execution of the business process. In particular, the following improvements have been incorporated in the company:

- Our study explored the concerns surrounding maverick buying (**PC1**) and post-mortem adjustments in purchase requisitions (**PC2**). By investigating these areas, we managed to identify steps to mitigate these issues. Such measures included refining internal documents with supplementary instructions and making modifications to internal training programs to better the purchasing process.
- Identifying activities that had a correlation with extended processing times (**Pp6**) facilitated us in holding a series of workshops with business users. Applying the Pareto principle, we pinpointed the main underlying causes. This approach enabled us to focus on the largest contributors to the processing time and develop effective solutions.
- Our analysis highlighted instances of duplicate invoices (**PQ1**), providing an understanding of the master data management issues within the company. A deeper investigation indicated that these duplicates often resulted from individual users implementing workarounds to expedite the invoice entry process. Addressing these root causes allowed the company to enhance its master data management and decrease the instances of duplicate invoices.
- By investigating the relationship between high workload and extended processing times (**PP7**), we pinpointed peak times of maximum workload. This information led to the development of temporary resource solutions to balance the workload and ensure a more efficient process.

Other considerations are presently under evaluation but have yet to be implemented:

- By scrutinizing the results of **PP1** and **PP2**, we managed to identify distinct invoice groups that displayed particularly lengthy or brief lifecycle times. While certain cases might be justified due to factors like long-term contracts or emergency maintenance orders, we also identified outliers that

couldn't be accounted for by these reasons. From the perspective of business users, these outliers might be deemed excessive. Potential remedial actions to this issue could include updating relevant master data and conducting interdepartmental workshops to resolve underlying issues.

- By considering the outcomes of **PP4** and **PP5**, we determined whether invoices with extended lifecycle times were affected by purchase order related issues, such as protracted processing times or bottlenecks. This information guided the design of improved communication strategies between the departments responsible for invoice payments and ordering goods, aiming to tackle these identified issues and anticipate them in future.
- The identification and scrutiny of outliers in interconnected processes (**PQ2**) kickstarted a discussion about the expected behavior in these processes and what constitutes non-compliant behavior. This ultimately led to a more profound understanding of the processes and how to refine them.

8.1.3 P2P Process (Automated Extraction)

In Chapter 5, we introduced a semi-automated methodology for creating OCELs from underlying relational database structures. Here, we apply those concepts to the P2P process data within SAP ECC using a largely automated extraction procedure. We clarify how the blueprint-based extraction, table-to-object mapping, and domain knowledge filtering—techniques all previously discussed—facilitate the automated generation of high-quality object-centric logs.

This section illustrates the scalability and reproducibility of the approaches defined in earlier chapters, making it clear how theoretical extraction principles hold up under realistic conditions and large data volumes.

Constructing the Database Abstraction

Here, we build a GoR on top of an instance of SAP ERP. This is done automatically (without user interaction) in three different steps:

- Extraction of the set of nodes.
- Extraction of the set of arcs.
- Association of a set of table entries.
- Definition of the relationships between the different object identifiers.

Table 8.4: SQL queries to extract different categories of nodes of the GoR built on top of SAP ERP.

Node Type	**SQL query**
tables	SELECT DISTINCT TABNAME FROM DD02L
transactions	SELECT DISTINCT TCODE FROM TSTCT
attributes	SELECT DISTINCT FIELDNAME FROM DD03L
domains	SELECT DISTINCT DOMNAME FROM DD03L
object types	SELECT DISTINCT OBJECT FROM TCDOB

Extraction of the set of nodes: The various categories of nodes, along with some queries that can be utilized for their extraction, are illustrated in Table 8.4. Specifically, five distinct categories of nodes are recognized: *tables* (such as EKKO, RBKP, BKPF), *transactions* (such as MIRO, MR1M, VA21N), *attributes* (like BELNR, representing the invoice number, and GJAHR, denoting the fiscal year), *domains* (such as a timestamp (DATUM), organizational resource (USNAM)), and *object types* (for instance, EINKBELEG corresponds to the purchase order documents and VERKBELEG refers to the sales order document).

Extraction of the set of arcs: The various categories of arcs, along with some queries that can be utilized for their extraction, are outlined in Table 8.5. Specifically, seven different categories of arcs are recognized: *attribute arcs* (connecting the tables to some attributes that are neither primary nor foreign keys for the given table), *primary key arcs* (linking the tables to some attributes that are primary keys, but not foreign keys, for the given table), *foreign key arcs* (connecting the tables to some attributes that are foreign keys), *domain arcs* (linking the attributes to the corresponding domain(s)), *object type arcs* (connecting the tables to the corresponding object type(s)), *transaction arcs* (connecting the tables to

the corresponding transaction(s)), *relationship arcs* (linking a table to another related table, i.e., two tables having a pair of related objects).

Table 8.5: SQL queries to extract different categories of arcs of the GoR built on top of SAP ERP.

Category	**SQL query**
attribute arcs	SELECT TABNAME,FIELDNAME FROM DD03L WHERE KEYFLAG!='X' AND CHECKTABLE=' '
primary key arcs	SELECT TABNAME,FIELDNAME FROM DD03L WHERE KEYFLAG='X' AND CHECKTABLE=' '
foreign key arcs	SELECT TABNAME,FIELDNAME FROM DD03L WHERE CHECKTABLE!=' '
domain arcs	SELECT DISTINCT FIELDNAME,DOMNAME FROM DD03L
object type arcs	SELECT DISTINCT TABNAME,OBJECT FROM TCDOB
transaction arcs	SELECT DISTINCT TABNAME,TCODE FROM CDHDR a JOIN CDPOS b ON a.MANDANT = b.MANDANT AND a.CHANGENR = b.CHANGENR
relationship arcs	SELECT DISTINCT TABNAME,CHECKTABLE FROM DD03L

Table entries association: Here, we assign each table to its entries. An approach for this is to view each row of each table in SAP as a separate entry. In this case, the attributes of the table entry and the transactions executed against it rely solely on the attributes of each row.

Relationships between different object identifiers: Some tables in the graph can depict relationships between object identifiers. This is achieved by connecting all the object identifiers referred to by the foreign keys of the rows of such a table.

Process Identification and Selection

The database abstraction crafted in the preceding section lays the groundwork for extracting an OCEL from an instance of SAP ERP. SAP ERP houses a variety of processes (such as Order-to-Cash and Procure-to-Pay) that involve different tables. Consequently, a selection of table subsets is required for the extraction of an OCEL. Here is a method for selecting the table subsets:

- Generate the GoR from the database.
- Begin with a table representative of the process (for instance, EKKO for Procure-to-Pay) or the tables related to a given object type (for example, EINKBELEG is linked with various tables of the Procure-to-Pay process, including EKKO, EKET, EKPA).
- Enlarge the set of tables based on the relationships exhibited in the GoR.

The expansion step explores the relationship arcs that target one of the tables in the initial set. Then, any table serving as the source of such arcs is incorporated into the expanded table set. This expansion procedure can be repeated multiple times, gradually increasing the set of tables.

At this juncture, the user needs to identify the representative tables and consider which tables from the proposed expansion could be intriguing for extraction.

Preprocessing the Table Entries

The preprocessing step helps to reduce the number of entries associated with the tables of the provided set by checking the values allowed by some attributes. For example, in a Procure-to-Pay process, we might be interested in considering only the orders having a given material (in this case, the attribute is MATNR).

This step is fully manual (e.g., the user selects the values admitted for the attributes).

Extracting an OCEL

The OCEL can be extracted with the provision (from the user) of a blueprint.

Assessment - Scalability

In this particular subsection, we evaluate the scalability of the approach proposed. Specifically, we conduct measurements to ascertain the time required to construct the database abstraction, which is the primary source of computational complexity.

Table 8.6: Extraction of the different nodes of the GoR. For every category, we report the number of nodes and the time in seconds needed for the extraction.

Category	**Number of Nodes**	**Query Time(s)**
tables	662161	0.43 s
transactions	103060	0.80 s
attributes	934546	4.05 s
domains	115484	2.29 s
object types	1885	0.15 s

Table 8.7: Extraction of the arcs of the GoR. For every category, we report the number of arcs, the time in seconds needed for the extraction, and the postprocessing time.

Arc Type	**Number of Arcs**	**Query Time(s)**	**Postprocessing Time(s)**
attribute arcs ($R_{T,A}$)	9628155	6.71 s	0.0 s
domain arcs ($R_{A,D}$)	1311489	9.64 s	0.0 s
object type arcs ($R_{T,OT}$)	4803	0.12 s	0.0 s
relationship arcs ($R_{T,T}$)	1671478	6.83 s	16.28 s
primary key arcs ($R_{T,PR}$)	412183	6.71 s	0.0 s
foreign key arcs ($R_{T,FR}$)	1931064	1.73 s	0.0 s

We conducted our experiments on an instructional instance of SAP ERP loaded with demo data. Despite the limited number of distinct documents contained in this instance, the relational structure remains quite comprehensive. Therefore, the number of nodes/edges is fairly representative. Our experimental findings don't include the time taken for graph insertion (database). Instead, we focus on the times necessitated to retrieve information from the relational database underpinning SAP ERP, along with the pre-processing time needed to discern the connections between concepts prior to graph database insertion.

Table 8.8: Extraction of the table entries for some tables in our educational instance in SAP ERP. For every table, we report the number of extracted entries and the time needed for the query.

Table	**Description**	**Num. Entries**	**Query Time(s)**
EBAN	Purchase Requisitions (Master)	4281	0.18 s
EBKN	Purchase Requisitions (Detail)	1042	0.15 s
EKKO	Purchase Orders (Master)	16214	0.20 s
EKPO	Purchase Orders (Detail)	30985	0.25 s
RBKP	Invoices (Master)	5720	0.18 s
RSEG	Invoices (Detail)	14802	0.22 s
BKPF	Payments (Master)	664021	0.41 s
BSEG	Payments (Detail)	1844532	1.25 s

Tables 8.6 and 8.7 assess the retrieval of the GoR's nodes/edges. The extraction of relationships between tables requires postprocessing. In our implementation, it is extracted by associating the tables that share a foreign key with a given table, and all pairwise relationships between the tables are taken into account. The execution time for these queries and postprocessing operations is excellent, making the identification of database abstraction quite timely (24 seconds in our experiment).

Table 8.8 pertains to some tables critical for the Procure-to-Pay process (such as purchase requisitions, purchase orders, invoices, and payments) and the time required to extract entries from such tables. The number of entries, and the execution time, are influenced by the educational nature of our instance. Table 8.9 reveals the time required to extract relationships between object identifiers given the table containing these relationships. For instance, EKPO relates purchase requisitions and purchase orders, while RSEG connects invoices and purchase orders. The lion's share of time is consumed in discovering the relationships between the object identifiers, thus forming the largest bottleneck in the overall process.

Table 8.9: Extraction of the relationships between object identifiers defined in some tables of our educational instance in SAP ERP. For every table, we report the number of relationships and the query/post-processing times.

Table	Description	Num. Relationships	Query Time(s)	Postprocessing Time(s)
EBKN	Purchase Requisitions (Detail)	8964	0.15 s	0.06 s
EKPO	Purchase Orders (Detail)	748750	0.25 s	3.72 s
RSEG	Invoices (Detail)	157086	0.22 s	0.58 s
BSEG	Payments (Detail)	2323792	1.25 s	137.79 s

Assessment - Quality

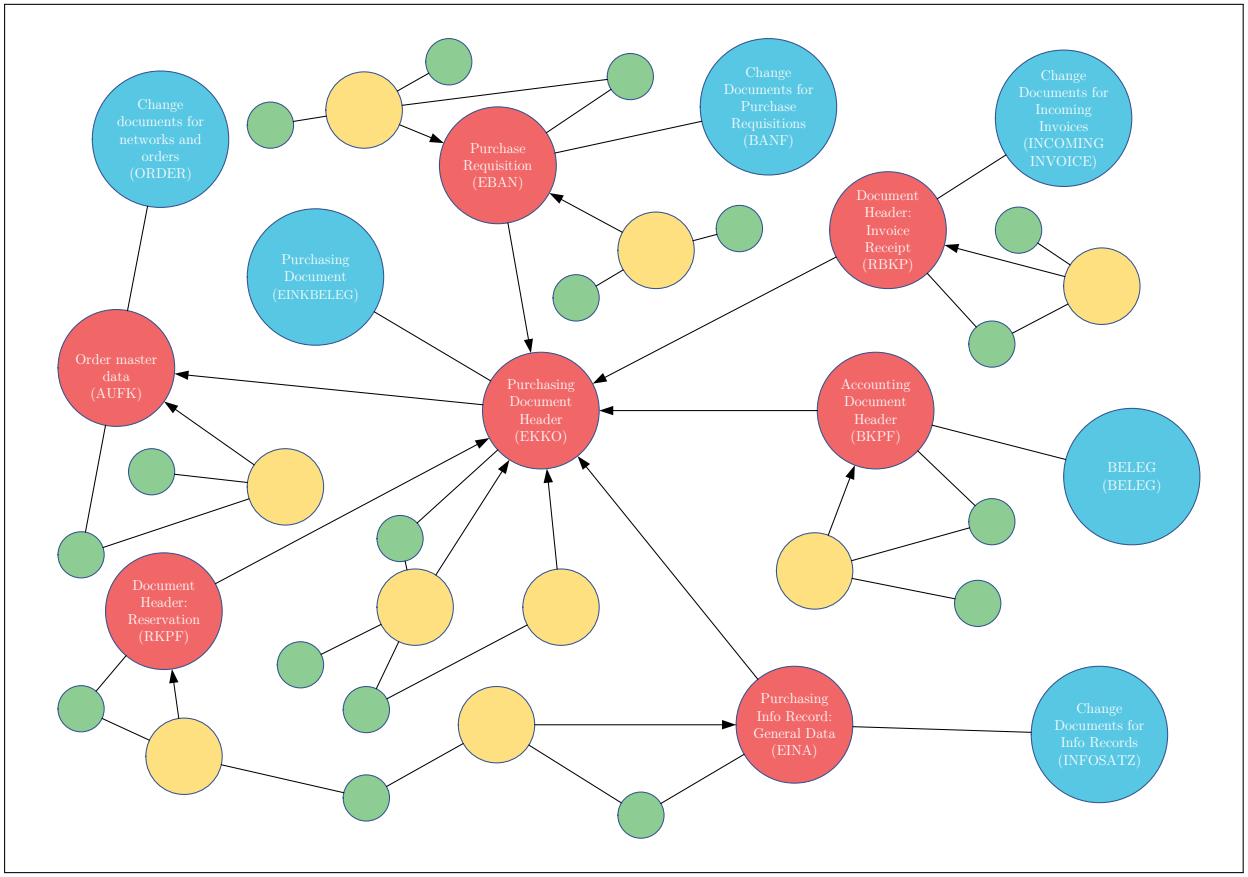


Figure 8.2: Subgraph showing the tables related to the EINKBELEG object type (purchase orders).

The extraction process begins with identifying and selecting the process. In this stage, we identify a set of interconnected tables starting from an object type, which is then utilized to extract the OCEL. We evaluate the set of interconnected tables derived from the purchasing order type (EINKBELEG). This object type is associated with the Procure-to-Pay process. Consequently, we anticipate the discovery of tables related to purchase requisitions, invoices, and payments in this step. As illustrated in Figure 8.2, the purchasing order type is indeed linked to tables of other object types, encompassing invoices, payments, and purchase requisitions, which aligns with our expectations.

Upon executing the tool on our instructional SAP ERP instance, the process identification step discloses the object types outlined in Table 8.10. As we move forward with the subsequent table selection and extraction stages, we are capable of connecting each object type to a set of tables and extracting events from those tables.

A primary concern arises in the naming of activities when utilizing the basic blueprint. Purchase orders are indiscriminately linked with the same activity, invoices are tied to the transaction conducted on the respective document, and alterations are associated with the set of fields that undergo changes without contrasting their values. Consequently, the naming of the activities falls short in quality compared to a more bespoke extraction.

Table 8.10: Processes identified on the educational SAP ERP instance.

Object type	Number of events	Number of tables	Description of the process
VERKBELEG	412228	11	Management of Sales Document (O2C)
IFLO	133263	3	PM: Functional Location
EINKBELEG	95490	6	Management of Purchasing Document (P2P)
VTBFHA	77200	9	Treasury: Transaction
EQUI	66827	4	Equipment change document objects
STUE	65430	10	Object list change documents
BUPA_BUP	45810	6	Business partner: Use of BUP
PROJ	26067	5	Project structure plan (PSP)
FTR_TCORT_CO	6586	8	Treasury: Correspondence
IMAK	4080	5	Appropriation requests with variant
EHFND_REGLIST	1447	4	Regulatory List Revision
FARR_CONTRACT	521	4	Financial Accounting Revenue Recognition

We showcase some logs extracted on an educational instance of SAP ERP related to the O2C process (*VERKBELEG* object type) and the P2P process (*EINKBELEG* object type). The two OCELs are in the JSON-OCEL implementation and can be retrieved at the addresses:

- `https://www.ocpm.info/o2c.jsonocel` for the O2C process.
- `https://www.ocpm.info/p2p.jsonocel` for the P2P process.

8.1.4 O2C Process

Like P2P, the Order-to-Cash (O2C) process provides another context to apply the object-centric methodologies from Chapters 5–6. First, we present a generic O2C implementation in SAP ECC and identify the relevant tables, object types, and events. Then, building on the frameworks from earlier chapters, we demonstrate how to extract the O2C-specific OCEL, model its execution using object-centric process models, and perform conformance checking.

As with P2P, we maintain a clear distinction between the general O2C principles and any case-specific nuances. This allows us to illustrate the portability and generality of the methods introduced previously, ensuring that our approach is not confined to a single process type.

The Order-to-Cash (O2C) process intersects with various SAP ECC modules:

- *Sales and Distribution (SD)*: The SD module is the central pillar of the O2C process. It manages sales order processing, shipping and delivery of products, and billing to the customers. From the initial receipt of a customer order to the creation of an invoice, the SD module handles the key steps in the O2C process.
- *Finance (FI)*): After the invoice is issued to the customer, the FI module comes into play. It manages the accounting processes associated with the sale, such as posting the revenue from the sale and managing the accounts receivable process. The FI module also handles the receipt of payment from the customer, closing out the O2C process.
- *Material Management (MM)*: The MM module indirectly supports the O2C process by ensuring the availability of goods to be sold. It manages the inventory levels and, in case of a manufacturing organization, the procurement of raw materials needed for production.
- *Production Planning (PP)*: For manufacturing companies, the PP module plays a significant role in the O2C process. Once a sales order is received, the PP module plans and controls the production process to meet this demand. It schedules the production runs, manages the use of resources, and ensures the timely availability of the product for delivery to the customer.
- *Controlling (CO)*: The CO module helps monitor and control the costs associated with the O2C process. It tracks costs and profitability at each stage, providing valuable input for pricing and cost control strategies.
- *Quality Management (QM)*: The QM module ensures that the products being sold meet the requisite quality standards. It manages the quality checks before the product is shipped to the customer, preventing defective or subpar products from being delivered.

- *Plant Maintenance (PM) and Project System (PS)*: While the direct connection might not be as apparent, the efficient running of machines (PM) and management of customer-specific projects (PS) contribute to the successful and timely execution of the O2C process.

The Order-to-Cash process in SAP is orchestrated through a sequence of database tables that hold relevant fields for each activity. These tables and their specific fields capture the essence of every step in the O2C process, from order placement to cash realization, enabling a smooth, traceable, and efficient process execution.

- **KNA1** - General Data in Customer Master: It stores general data about customers in SAP.
  - *KUNNR*: Customer Number
  - *NAME1*: Name of the customer
- **VBAK** - Sales Document: Header Data: It contains the header-level data of sales documents.
  - *VBELN*: Sales Document Number
  - *ERDAT*: Date of document creation
  - *BSTNK*: Customer purchase order number
- **VBAP** - Sales Document: Item Data: It contains the item-level data of sales documents.
  - *VBELN*: Sales Document Number
  - *POSNR*: Sales document item
- **CRMD_ORDERADM_H** - CRM Order Header Table: It stores the header-level data of CRM orders.
  - *GUID*: CRM Order GUID
  - *PROCESS_TYPE*: CRM Order Process Type
- **MCHB** - Batch Stock: It stores data about the inventory stock at batch level.
  - *MATNR*: Material number
  - *LFGJA*: Fiscal Year
- **MKPF** - Material Document Header: It contains information about the material document that is created when goods are issued.
  - *MBLNR*: Material Document Number
  - *BUDAT*: Posting Date in the Document
- **MSEG** - Material Document Item: It contains information about each individual item in the goods issue.
  - *MBLNR*: Material Document Number
  - *MATNR*: Material Number
- **LAGP** - Storage Bin: It stores data about the storage bins in the warehouse.
  - *LGPLA*: Storage bin
- **VEKP** - Handling Unit - Header Table: It contains data about the transportation units used in logistics.
  - *EXIDV*: External Handling Unit Identification
- **LIKP** - Delivery Header Data: It contains the header-level data of delivery documents.
  - *VBELN*: Delivery Number
  - *ERDAT*: Date of document creation
- **LIPS** - Delivery: Item Data: It contains the item-level data of delivery documents.

  - *VBELN*: Delivery Number
  - *POSNR*: Delivery item
- **VBRK** - Billing: Header Data: It contains the header-level data of billing documents.
  - *VBELN*: Billing Document Number
  - *ERDAT*: Date of document creation
- **VBRP** - Billing: Item Data: It contains the item-level data of billing documents.
  - *VBELN*: Billing Document Number
  - *POSNR*: Billing item
- **LFBK** - Bank Details for Vendor Master: It stores bank details for vendors and customers.
  - *BANKL*: Bank keys
  - *BANKN*: Bank account number
- **BSEG** - Accounting Document Segment: It contains line item data for accounting documents.
  - *BELNR*: Accounting document number
  - *AUGDT*: Clearing Date

Table 8.11 provides a comprehensive overview of the various object types involved in the Order-to-Cash (O2C) process, presenting the corresponding SAP tables, and their specific object identifiers, offering a detailed understanding of the extraction process.

Object Type	**SAP Table**	**Object Identifier**
Customer Order	VBAK, VBAP	VBAK.VBELN (Order Number) + VBAP.POSNR (Item)
Customer	KNA1	KNA1.KUNNR (Customer Number)
Sales Contract	VBAK, VBAP	VBAK.VBELN (Contract Number) + VBAP.POSNR (Item)
Inventory	MCHB	MCHB.MATNR (Material) + MCHB.CHARG (Batch)
Goods Issue	MSEG	MSEG.MATNR (Material Number) + MSEG.LGORT (Storage Location)
Delivery Note	LIKP, LIPS	LIKP.VBELN (Delivery Number) + LIPS.POSNR (Item)
Transportation	VEKP	VEKP.HUOBJ (Handling Unit)
Invoice	VBRK, VBRP	VBRK.VBELN (Invoice Number) + VBRP.POSNR (Item)
Bank	LFBK	LFBK.BANKS (Bank Key) + LFBK.BANKL (Bank number)
Payment	BSEG	BSEG.BELNR (Document Number) + BSEG.GJAHR (Fiscal Year) + BSEG.BUKRS (Company Code)
Warehouse	LAGP	LAGP.LGPLA (Storage Bin)
CRM	CRMD_ORDERADM_H	CRMD_ORDERADM_H.GUID (Global Unique Identifier)

Table 8.11: Table summarizing the extraction process for objects of different object type in the O2C process.

As shown in Table 8.12, the complexities of the relationships between different objects within the O2C process are depicted. It elucidates the qualifier roles between source and target objects, and their respective SAP tables.

Table 8.13 delineates the assortment of activities involved in the O2C process, outlining the associated tables, activity-specific fields, and timestamps. It further specifies the related objects for each activity, highlighting the interconnectivity within the process.

Qualifier	Source SAP Table	Source Object	Target SAP Table	Target Object
Checks	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR	MCHB	MCHB.MATNR + MCHB.CHARG
Determines	MCHB	MCHB.MATNR + MCHB.CHARG	MSEG	MSEG.MATNR + MSEG.LGORT
Generates	MSEG	MSEG.MATNR + MSEG.LGORT	LIKP, LIPS	LIKP.VBELN + LIPS.POSNR
Initiates	LIKP, LIPS	LIKP.VBELN + LIPS.POSNR	VEKP	VEKP.HUOBJ
Delivers To	VEKP	VEKP.HUOBJ	KNA1	KNA1.KUNNR
Triggers	MSEG	MSEG.MATNR + MSEG.LGORT	VBRK, VBRP	VBRK.VBELN + VBRP.POSNR
Sent To	VBRK, VBRP	VBRK.VBELN + VBRP.POSNR	LFBK	LFBK.BANKS + LFBK.BANKL
Processes	LFBK	LFBK.BANKS + LFBK.BANKL	BSEG	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS
Received From	BSEG	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS	KNA1	KNA1.KUNNR
Placed By	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR	KNA1	KNA1.KUNNR
Bound By	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR
Applies To	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR	KNA1	KNA1.KUNNR
Stored In	MCHB	MCHB.MATNR + MCHB.CHARG	LAGP	LAGP.LGPLA
Dispatches	LAGP	LAGP.LGPLA	MSEG	MSEG.MATNR + MSEG.LGORT
Managed By	KNA1	KNA1.KUNNR	CRMD_ORDERADM_H	CRMD_ORDERADM_H.GUID
Informs	CRMD_ORDERADM_H	CRMD_ORDERADM_H.GUID	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR
Updates	VBAK, VBAP	VBAK.VBELN + VBAP.POSNR	CRMD_ORDERADM_H	CRMD_ORDERADM_H.GUID
Tracks	CRMD_ORDERADM_H	CRMD_ORDERADM_H.GUID	BSEG	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS
Satisfies	BSEG	BSEG.BELNR + BSEG.GJAHR + BSEG.BUKRS	VBRK, VBRP	VBRK.VBELN + VBRP.POSNR

Table 8.12: Table summarizing the extraction process of the qualified Object-to-Object relationships for the O2C process.

Activity	Involved Tables	Activity Table & Field	Timestamp Table & Field	Related Objects
Create Customer Order	VBAK, VBAP, CRMD_ORDERADM_H, KNA1	VBAK.VBELN	VBAK.ERDAT	Customer Order, Customer, CRM
Confirm Order	VBAK, VBAP, CRMD_ORDERADM_H	VBAK.BSTNK	VBAK.ERDAT	Customer Order, Sales Contract, CRM
Check Inventory	MCHB, VBAK, VBAP	MCHB.MATNR	MCHB.LFGJA	Inventory, Customer Order
Issue Goods	MSEG, MCHB, VBAK, VBAP	MSEG.MATNR	MSEG.BUDAT	Goods Issue, Inventory, Customer Order
Pack Goods	MSEG, LAGP, VBAK, VBAP	MSEG.MATNR	MSEG.BUDAT	Goods Issue, Warehouse, Customer Order
Dispatch Goods	MSEG, VEKP, VBAK, VBAP	MSEG.MATNR	MSEG.BUDAT	Goods Issue, Transportation, Customer Order
Generate Delivery Note	LIKP, LIPS, MSEG, VBAK, VBAP	LIKP.VBELN	LIKP.ERDAT	Delivery Note, Goods Issue, Customer Order
Create Invoice	VBRK, VBRP, LIKP, LIPS, VBAK, VBAP	VBRK.VBELN	VBRK.ERDAT	Invoice, Delivery Note, Customer Order
Send Invoice	VBRK, VBRP, LFBK, VBAK, VBAP	VBRK.VBELN	VBRK.ERDAT	Invoice, Bank, Customer Order
Receive Payment	BSEG, LFBK, VBRK, VBRP, CRMD_ORDERADM_H	BSEG.BELNR	BSEG.AUGDT	Payment, Bank, Invoice, CRM

Table 8.13: Table summarizing the extraction process of the events, and the Event-to-Object relationships, for the O2C process.

Conclusion

This chapter has demonstrated the practical application and tangible benefits of Object-Centric Process Mining through real-world case studies within the SAP ECC ERP system. The detailed examination of the P2P and O2C processes, including the specific techniques employed and the insights obtained, has showcased the power of OCPM in uncovering valuable information about process performance, compliance, and quality. The comparison of manual and automated data extraction methods has provided practical insights into the trade-offs involved in choosing the appropriate approach for different scenarios. The process improvement actions undertaken as a result of the OCPM analyses demonstrate the real-world impact and potential for optimizing complex business processes. These case studies not only validate the object-centric methodologies and techniques introduced in earlier chapters, but also highlight the practical challenges and considerations involved in applying OCPM in real-world settings. The insights gained from these real-world applications provide valuable lessons for practitioners and researchers alike, paving the way for further development and refinement of OCPM techniques and their application to an even broader range of processes and industries.

Chapter 9

Tooling for OCPM

"*Tools, of course, can be the subtlest of traps.*"
Neil Gaiman

Introduction

This chapter explores the practical application of Object-Centric Process Mining (OCPM) through three key contributions developed by the author of this thesis:

- Extending OCPM support within the Python-based PM4Py library [26, 27], enabling robust handling of OCEL data and facilitating object-centric process discovery.
- Creating a JavaScript-based OC-PM tool [25], offering a fully in-browser environment to visualize, analyze, and interact with object-centric models.
- Developing a converter that transforms OCELs into the Celonis Object-Centric Data Model, bridging open standards with a commercial platform for comprehensive, enterprise-level OCPM analyses.

These contributions form the backbone of this chapter, demonstrating how OCPM methodologies can be applied in practice and providing the means to compare different tooling options. By examining their features, performance, scalability, and usability, this chapter offers guidance for researchers and practitioners, as well as suggestions for future development, ensuring that OCPM continues to evolve and address a wide range of academic and industrial scenarios.

9.1 Tools for OCPM

A wide array of tools and libraries have emerged in recent years to support OCPM. These tools aim to facilitate every step of the OCPM lifecycle—from data ingestion, transformation, and filtering, to advanced analytics, visualization, and conformance checking—thereby allowing researchers and practitioners to gain deeper insights from complex event data involving multiple, interconnected object types. Building on open standards, such as the Object-Centric Event Log (OCEL) [27], and leveraging established process mining methodologies, modern OCPM tooling significantly lowers the barriers to analyzing heterogeneous datasets. As a result, it empowers analysts to uncover hidden patterns, dependencies, and performance bottlenecks that traditional case-centric models often fail to capture.

In this chapter, we present three main contributions to the OCPM tooling landscape. First, we describe how OCPM functionalities have been integrated into the PM4Py library [26, 27], a widely-used Python-based open-source ecosystem for process mining. Second, we introduce a novel JavaScript-based OC-PM tool [25] that enables data exploration, discovery, and conformance checking directly in the browser, without the need for additional server-side installations. Finally, we illustrate a converter that transforms standard OCELs into the Celonis Object-Centric Data Model, bridging a popular open standard with a commercial platform for large-scale business analyses.

Operation	Format	Function
Reading (OCEL)	.csv	read_ocel_csv()
	.jsonocel	read_ocel_json()
	.xmlocel	read_ocel_xml()
	.sqlite	read_ocel_sqlite()
Reading (OCEL2.0)	.xmlocel	read_ocel2_xml()
	.sqlite	read_ocel2_sqlite()
Writing (OCEL)	.csv	write_ocel_csv()
	.jsonocel	write_ocel_json()
	.xmlocel	write_ocel_xml()
	.sqlite	write_ocel_sqlite()
Writing (OCEL2.0)	.xmlocel	write_ocel2_xml()
	.sqlite	write_ocel2_sqlite()

Table 9.1: Reading/Writing OCELs in PM4Py

9.1.1 Overview of PM4Py Library (Process Mining for Python)

PM4Py [27] is a state-of-the-art open-source process mining library in Python, underpinning the critical objective of bridging the gap between the wealth of data produced by modern information systems and the need for data-driven decision-making and optimization in process-oriented domains. As its name suggests, PM4Py provides advanced tooling for process mining, a powerful set of techniques to extract insights about the flow of processes from event data. The library's broad goal is to offer a complete suite of process mining techniques and algorithms that can be leveraged by researchers, analysts, and organizations alike. PM4Py provides functionality for a wide array of process mining tasks, including but not limited to: process discovery, conformance checking, and model enhancement, each of which contribute to an improved understanding of organizational processes. As an open-source library, PM4Py has been designed with flexibility and extensibility in mind. It has been built around the principle of modularity, which means individual components or algorithms can be replaced or updated with minimal impact on the overall system. The library makes use of standard data formats and has a well-documented API, which makes it accessible and easy to integrate into a variety of software environments. Additionally, PM4Py is backed by a strong scientific foundation, with its algorithms and methods being based on established research in the process mining field. This grounding in academic research, coupled with its practical usability, has made PM4Py a popular choice among researchers and practitioners interested in harnessing the power of process mining.

The list of approaches implemented in PM4Py is quite extensive, indicating the library's comprehensive support for process mining. Here is a brief summary of them:

- *Process Discovery:* Process discovery refers to the task of deriving a process model from event log data. Various algorithms are used, each with their own characteristics and best-use scenarios.
  - *Inductive Miner* [71] discovers block-structured process models from event logs.
  - *Inductive Miner Infrequent* [72] improves upon the Inductive Miner by discovering models from logs containing infrequent behaviour.
  - *Inductive Miner Directly-Follows* [73] is a scalable variant of the Inductive Miner, focusing on performance.
  - *Heuristics Miner* [135] is an algorithm designed to discover heuristic knowledge about the process execution.
  - *Correlation Miner* [105] focuses on discovering process models and event correlations without case identifiers.
- *Conformance Checking:* Conformance checking verifies if the real execution of a process, as recorded in an event log, conforms to a model.
  - *TBR* [24] is a novel technique that speeds up conformance checking.
  - *Alignments* [10] provide a unified approach to quantify the conformance of a trace.
  - *Decomposed/Recomposed Alignments* [70] offer a more efficient way to perform alignments.
  - *Alignments Approximation* [111] focuses on approximation for process trees.

  - *Log Skeleton* [131] uses a classification approach to discover processes.
  - *Temporal Profile* [113] provides a temporal perspective for conformance checking.

- *Evaluation Log-Model:* Several techniques evaluate the discovered process models with respect to various quality dimensions.
  - *Fitness* [24, 32] measures how much of the behavior in the log is allowed by the model.
  - *ETConformance precision* [87, 9] offers a fresh look at precision in process conformance.
  - *Generalization* [32] measures how well the model can deal with unseen behavior.
  - *Simplicity* [130] evaluates how simple, and thus understandable, the model is.
  - *Anti-Alignments and Multi-Alignments* [35, 29] provide additional tools for measuring the precision of process models and event logs.

- *Other Approaches:* PM4Py also supports several other techniques, such as:
  - *Process Tree Generation* [63] for artificial event data.
  - *Decision Mining* [38] for understanding and predicting dynamic behavior.
  - *Soudness Checking (WOFLAN)* [132] for diagnosing workflow processes.
  - *Trace Clustering* [128] for grouping similar process instances.
  - *Performance Spectrum* [41] for fine-grained performance analysis.
  - *LTL Checker* [120] for verifying properties based on temporal logic.
  - *Process Tree Converter* [129] for translating workflow nets to process trees.
  - *Social Network Analysis* [121] for discovering social networks from event logs.
  - *Roles Discovery* [33] for enhancing business models.
  - *Resource Profiles* [100] for mining resource profiles from event logs.
  - *Organizational Mining* [140] for mining organizational models from event logs.
  - *Differential Privacy* [50] for privacy-preserving event log publishing.
  - *Batch Detection* [83] for identifying batch processing in event logs.
  - *Temporal Feature Extraction* [104] for automatic system dynamics model generation for simulation.

Traditional Log Structure in PM4Py

The traditional logs structures in PM4Py are Pandas dataframes. Pandas is an open-source data analysis and manipulation library that provides flexible data structures for working with structured data in Python. It is notably suitable for process mining due to its high-performance, easy-to-use data structures and data analysis tools. Here's why:

- The primary data structure in Pandas, the dataframe, aligns naturally with the event log structure required in process mining. An event log is essentially a tabular dataset, where each row corresponds to an event and each column to an attribute of the event such as case ID, event name, and timestamp. These attributes are crucial in process mining for constructing process models, checking conformance, and conducting other analysis tasks.

- Pandas excels in handling large datasets, which is essential in process mining, given the increasing size of event logs produced by modern information systems. The library offers efficient and memory-friendly operations for filtering, sorting, aggregating, and transforming data, which are operations commonly performed in process mining. The computational efficiency of Pandas becomes particularly useful when working with logs containing millions of events.

- The library offers extensive functionality for time series manipulation, a fundamental aspect of process mining. This includes capabilities for parsing timestamps, resampling time series data, handling timezones, and calculating time differences. These functions are beneficial when conducting temporal analysis in process mining, such as computing the duration of process instances or the waiting time between activities.

- Pandas is well-integrated with other libraries in the Python ecosystem, such as NumPy for numerical computations and Matplotlib for visualization, which are also often used in process mining. The combination of these libraries allows for a seamless analysis pipeline, from data preprocessing and analysis to visualization and reporting.

OCELs in PM4Py

In PM4Py, the OCEL data structure is represented using a collection of three Pandas dataframes: the events dataframe, the objects dataframe, and the relations dataframe. The events dataframe records each event with attributes including an event identifier, an activity, a timestamp, and other event-specific attributes. The objects dataframe provides a register of each object, accompanied by an object identifier, a type, and a suite of object-specific attributes. The relations dataframe maps the connection between each event and object, providing identifiers and types for each.

The application of Pandas dataframes to represent the OCEL offers several substantial advantages.

- *Increased filtering capabilities:* it affords a well-structured, intuitive, and easily accessible means of housing and manipulating the OCEL data. This is valuable because each component of the OCEL can be isolated and interacted with in a more precise and flexible manner. This aspect is particularly beneficial in scenarios involving filtering, as users can effortlessly apply conditions to single or multiple attributes within each dataframe, thereby enabling highly granular control and refinement of the data to meet specific analytical requirements.

- *Increased performance:* the use of dataframes enables the efficient computation of operations on the data due to the implementation of vectorized operations in Pandas, which can be significantly faster than Python's native operations. This can lead to substantial time savings when dealing with large event logs, enhancing the scalability of the process mining tasks.

- *Robust Pandas ecosystem:* leveraging Pandas dataframes also introduces the benefits of the robust ecosystem around the Pandas library. This includes its extensive functionalities for data analysis and manipulation, its seamless interoperability with other popular data analysis libraries in the Python ecosystem, and its strong support for reading from and writing to various data formats.

Overall, the use of Pandas dataframes for representing the OCEL data structure in PM4Py is a well-considered design choice that capitalizes on the powerful features of Pandas, consequently empowering users with more refined control and flexibility in their process mining tasks. In PM4Py, a variety of methods are available for reading and writing OCELs. As shown in Table 9.1, the library supports importing OCEL from four different formats: CSV (`read_ocel_csv()`), JSON (`read_ocel_json()`), XML (`read_ocel_xml()`), and SQLite (`read_ocel_sqlite()`). Moreover, it also provides support for importing OCEL2.0 event logs from XML (`read_ocel2_xml()`) and SQLite (`read_ocel2_sqlite()`) formats. On the other hand, exporting OCEL is facilitated through corresponding functions that allow for writing logs in CSV (`write_ocel_csv()`), JSON (`write_ocel_json()`), XML (`write_ocel_xml()`), and SQLite (`write_ocel_sqlite()`) formats. Similar to the reading operations, exporting OCEL2.0 logs is supported in XML (`write_ocel2_xml()`) and SQLite (`write_ocel2_sqlite()`) formats. This wide range of reading and writing capabilities in PM4Py enables users to handle OCELs effectively across different data storage and interchange formats.

Table 9.2: Main computations possible on OCELs in PM4Py

Function	**Description**
ocel_get_object_types	Retrieves the object types from an OCEL.
ocel_get_attribute_names	Retrieves the attribute names from an OCEL.
ocel_flattening	Flattens OCEL with the selection of an object type.
ocel_object_type_activities	Gets the activities related to an object type in an OCEL.
ocel_objects_ot_count	Counts the objects for an object type.
ocel_temporal_summary	Returns the temporal summary from an OCEL.
ocel_objects_summary	Returns the objects summary from an OCEL.
ocel_objects_interactions_summary	Returns the objects interactions from an OCEL.
sample_ocel_objects	Returns a sampled OCEL picking a subset of the objects of the original one.
sample_ocel_connected_components	Returns a sampled OCEL containing the provided number of connected components.
ocel_drop_duplicates	Drops relations between events and objects happening at the same time.
ocel_merge_duplicates	Merge events in the OCEL which are happening with the same activity at the same timestamp.
ocel_o2o_enrichment	Enriches the O2O table of the OCEL with the graph-based relationships.
ocel_e2o_lifecycle_enrichment	Enriches the relations table of the OCEL with lifecycle-based information.
cluster_equivalent_ocel	Performs a clustering of the objects of an OCEL based on lifecycle/interactions similarity.
discover_ocdfg	Discovers an OCDFG from the OCEL.
discover_oc_petri_net	Discovers an OCPN from the OCEL.
discover_objects_graph	Discovers an object-based graph from the OCEL.

Table 9.3: OCEL Filters in PM4Py

Function	**Description**
filter_ocel_event_attribute	Filters the events of an OCEL having a given value for an attribute.
filter_ocel_object_attribute	Filters the objects of an OCEL having a given value for an attribute.
filter_ocel_object_types_allowed_activities	Filters the relations between events (activities) and objects (object types) in an OCEL.
filter_ocel_object_per_type_count	Filters the objects of an OCEL having at least the specific amount of objects per object type.
filter_ocel_start_events_per_object_type	Filters the events of an OCEL that start the lifecycle of an object of a given object type.
filter_ocel_end_events_per_object_type	Filters the events of an OCEL that end the lifecycle of an object of a given object type.
filter_ocel_events_timestamp	Filters the events of an OCEL based on a timestamp range.
filter_ocel_object_types	Filters a specified collection of object types from the OCEL.
filter_ocel_events	Filters a specified collection of event identifiers from the OCEL.
filter_ocel_objects	Filters a specified collection of object identifiers from the OCEL.
filter_ocel_cc_object	Filters a connected component from the OCEL to which the object with the provided identifier belongs.

Table 9.4: PM4Py Connectors

Connector	**Description**
extract_log_outlook_mails	Extracts a traditional Pandas dataframe representing the Outlook mails
extract_log_outlook_calendar	Extracts a traditional Pandas dataframe representing the Outlook calendar
extract_log_windows_events	Extracts a traditional Pandas dataframe containing the Windows events registry
extract_log_chrome_history	Extracts a traditional Pandas dataframe containing the Chrome navigation history
extract_log_firefox_history	Extracts a traditional Pandas dataframe containing the Firefox navigation history
extract_log_github	Extracts a traditional Pandas dataframe of a Github repository (issues management)
extract_log_camunda_workflow	Extracts a traditional Pandas dataframe from the database supporting Camunda
extract_log_sap_o2c	Extracts a traditional Pandas dataframe from the database supporting SAP (O2C process)
extract_log_sap_accounting	Extracts a traditional Pandas dataframe from the database supporting SAP (Accounting process)
extract_ocel_outlook_mails	Extracts an OCEL representing the Outlook mails
extract_ocel_outlook_calendar	Extracts an OCEL representing the Outlook calendar
extract_ocel_windows_events	Extracts an OCEL representing the Windows events
extract_ocel_chrome_history	Extracts an OCEL representing the Chrome history
extract_ocel_firefox_history	Extracts an OCEL representing the Firefox history
extract_ocel_github	Extracts an OCEL of a Github repository (issues management)
extract_ocel_camunda_workflow	Extracts an OCEL from the database supporting Camunda
extract_ocel_sap_o2c	Extracts an OCEL from the database supporting SAP (O2C process)
extract_ocel_sap_accounting	Extracts an OCEL from the database supporting SAP (Accounting process)

Filtering and Available Operations on OCELs

Here we describe the filtering and statistical operations possible using pm4py.

Filtering: as shown in Table 9.3, PM4Py also offers a multitude of filtering functions for OCELs (OCELs). These allow for precise control over the data included in analysis and range from filtering events or objects based on attribute values (`filter_ocel_event_attribute()` and `filter_ocel_object_attribute()`), to managing relations between events and objects (`filter_ocel_object_types_allowed_activities()`). It is possible to filter objects based on their count per type (`filter_ocel_object_per_type_count()`) and to filter events based on their role in the lifecycle of objects (`filter_ocel_start_events_per_object_type()` and `filter_ocel_end_events_per_object_type()`). Filtering is also supported based on a timestamp range (`filter_ocel_events_timestamp()`) or specified collections of object types, event identifiers, and object identifiers (`filter_ocel_object_types()`, `filter_ocel_events()`, and `filter_ocel_objects()`). Additionally, PM4Py can filter out a connected component from the OCEL based on an object identifier (`filter_ocel_cc_object()`), enabling more focused exploration and analysis of OCEL data.

Statistics: the PM4Py library provides a robust suite of functions for computations on OCELs (OCELs), as outlined in Table 9.2. These include functions for retrieving object types (`ocel_get_object_types()`) and attribute names (`ocel_get_attribute_names()`), flattening an OCEL (`ocel_flattening()`), and counting objects for a specific type (`ocel_objects_ot_count()`). Further, PM4Py offers functions to obtain various summaries such as the temporal summary (`ocel_temporal_summary()`), object summary (`ocel_objects_summary()`), and interaction summary (`ocel_objects_interactions_summary()`). It also provides capabilities for sampling OCELs based on objects (`sample_ocel_objects()`) or connected components (`sample_ocel_connected_components()`). Additional functionality includes removing and merging duplicates, enriching OCEL tables with graph-based and lifecycle-based information, clustering objects based on similarity, and discovering object-centric graphs and Petri nets. This comprehensive set of computations empowers users to perform detailed analysis and manipulation of OCELs in diverse ways.

Connectors:

PM4Py offers a plethora of connectors tailored to integrate data from various platforms and applications. These connectors ensure the seamless extraction of data logs, both in traditional formats like Pandas dataframes and the emerging OCEL representation. The table provided lists these connectors along with their descriptions:

- *Mail and Calendar Systems*:
  - The `extract_log_outlook_mails` and `extract_log_outlook_calendar` connectors enable users to obtain event logs from the popular Microsoft Outlook application. This can be useful for analyzing communication patterns, meeting distributions, or work habits.
  - The object-centric versions, `extract_ocel_outlook_mails` and `extract_ocel_outlook_calendar`, cater to the object-centric paradigm, focusing on multi-dimensional aspects of these events.
- *Operating System Events*:
  - With '`extract_log_windows_events`', users can mine system-level events registered by the Windows operating system. Such logs often provide insights into software behavior, system usage, or potential security incidents.
  - The '`extract_ocel_windows_events`' connector offers an object-centric version of the same.
- *Web Browsing History*:
  - The connectors '`extract_log_chrome_history`' and '`extract_log_firefox_history`' pull browsing data from Google Chrome and Mozilla Firefox, respectively. This data can reveal browsing habits, website preferences, or the frequency of particular web-based tasks.
  - Their object-centric counterparts, '`extract_ocel_chrome_history`' and '`extract_ocel_firefox_history`', provide a multidimensional view of this browsing history.
- *Software Development Activities*:

  - For those managing software projects on Github, the '`extract_log_github`' connector furnishes an event log related to issue management, which is instrumental in understanding the development lifecycle, issue resolution times, and collaborative practices.
  - '`extract_ocel_github`' provides an object-centric perspective on the same data.

- *Workflow Management Systems*:
  - Camunda, a workflow and decision automation platform, can be connected via the '`extract_log_camunda_workflow`' connector. This allows analysts to study workflow patterns, process bottlenecks, and task distributions.
  - The object-centric version, '`extract_ocel_camunda_workflow`', provides a more intricate view of these workflows.

- *Enterprise Resource Planning Systems*:
  - SAP, one of the leading ERP systems, is catered to with connectors like '`extract_log_sap_o2c`' and '`extract_log_sap_accounting`' for the Order-to-Cash and Accounting processes, respectively. Such logs are vital for understanding enterprise-level operations, financial flows, and customer interactions.
  - Their object-centric versions, '`extract_ocel_sap_o2c`' and '`extract_ocel_sap_accounting`', offer deeper insights into these complex processes.

In conclusion, these PM4Py connectors stand as bridges to a wide variety of data sources, ensuring that the vast realm of process mining can be applied to diverse domains, from individual software usage to large-scale enterprise operations.

Example Code Snippets

Process discovery: in Listing 9.1, the PM4Py library, is employed to perform OCPD. The code encompasses a series of steps including the importation of an OCEL, the discovery of an OCDFG, the visualization of this graph with frequency and performance metrics, and the discovery and visualization of an OCPN. Firstly, an OCEL is read from a JSON OCEL file through the `pm4py.read_ocel()` function. This event log is the basis for all subsequent operations. The `pm4py.discover_ocdfg()` function is then utilized to derive an OCDFG [25] from the imported event log. This graph represents the relations between events based on their chronological order and the interconnectedness of their associated objects. This graph is visualized twice with the help of the `pm4py.save_vis_ocdfg()` function. The first visualization decorates the graph with frequency metrics, offering insights into the commonality of different event sequences. The second visualization provides a performance-oriented view, outlining the efficiency of various process pathways. Lastly, an OCPN is discovered using the `pm4py.discover_oc_petri_net()` function. This mathematical modeling language allows for a more comprehensive representation of the concurrent, asynchronous, distributed, and nondeterministic elements of the process. The Petri net is then visualized and saved with the `pm4py.save_vis_ocpn()` function.

Anomaly detection: the provided Python code snippet in Listing 9.2 employs the PM4Py library and additional machine learning libraries, specifically scikit-learn and pandas, to perform anomaly detection on an OCEL. The code carries out several steps including importing the event log, filtering it based on object types, computing a feature table, applying Principal Component Analysis (PCA), implementing an anomaly detection algorithm, and finally, printing the resultant dataframe. Initially, the script reads an OCEL from a CSV file using `pm4py.read_ocel()`. Subsequently, the event log is filtered to retain only specific object types ("orders", "items", and "packages") using `pm4py.filter_ocel_object_types()`. The code then extracts features from the event log with the `pm4py.extract_ocel_features()` function, focusing specifically on "orders". The result is a feature table that represents each order as a feature vector in a high-dimensional space. To alleviate the curse of dimensionality and improve the efficiency of subsequent computations, PCA, a dimensionality reduction technique, is applied to the feature table, retaining only the first five principal components. The transformed data is stored in a new Pandas dataframe. Following the data transformation, the script applies the Isolation Forest algorithm, an effective anomaly detection method particularly suited to high-dimensional datasets. The anomaly scores generated by this model are added to the dataframe, and each row is assigned the corresponding order object ID. The dataframe is then sorted based on the anomaly scores and the order IDs. Finally, the resultant dataframe, featuring order IDs and their respective anomaly scores, is printed, providing a clear, concise overview of potential anomalies in the event log data.

Listing 9.1: OCPD using pm4py.

```
import pm4py

ocel = pm4py.read_ocel("tests/input_data/ocel/newocel.jsonocel")

# discovers an OCDFG
ocdfg = pm4py.discover_ocdfg(ocel)

# views the OCDFG decorated with frequency metrics.
pm4py.save_vis_ocdfg(ocdfg, "PM4Py_ocdfg_freq.png", annotation="frequency",
                         act_metric="events", edge_metric="event_couples")

# views the OCDFG decorated with performance metrics.
pm4py.save_vis_ocdfg(ocdfg, "PM4Py_ocdfg_perf.png", annotation="performance")

# discovers an OCPN
ocpn = pm4py.discover_oc_petri_net(ocel)

# saves the visualization of the OCPN.
pm4py.save_vis_ocpn(ocpn, "PM4Py_ocpn.png")
```

Listing 9.2: OCAD using pm4py.

```
import pm4py
from sklearn.decomposition import PCA
from sklearn.ensemble import IsolationForest

import pandas as pd

ocel = pm4py.read_ocel("tests/input_data/ocel/ocel_order_simulated.csv")

# filter the OCEL on a subset of object types
ocel = pm4py.filter_ocel_object_types(ocel, ["orders", "items", "packages"])

# computes the feature table and keeps (in the table) only the rows of the objects
# of type ''order''.
df = pm4py.extract_ocel_features(ocel, "orders")

# applies the PCA algorithm to keep a subset of the columns/features
pca = PCA(n_components=5)
df2 = pd.DataFrame(pca.fit_transform(df))

# applies anomaly detection (based on the isolation forest algorithm)
model = IsolationForest()
model.fit(df2)
df2["scores"] = model.decision_function(df2)
df2["object"] = list(ocel.objects[ocel.objects[ocel.object_type_column] == "orders"]
                         [ocel.object_id_column])
# sorts the dataframe on the anomaly score
df2.sort_values(["scores", "object"])

# prints the resulting dataframe
print(df2)
```

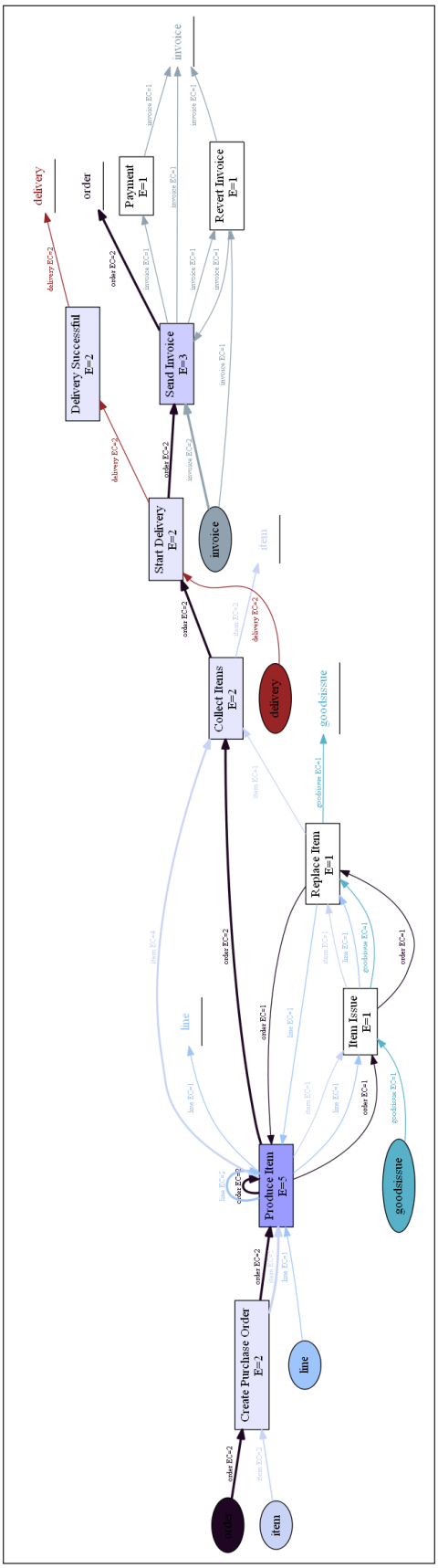


Figure 9.1: OCDFG, annotated with frequency metrics, discovered in PM4Py using the code snippet proposed in Listing 9.1.

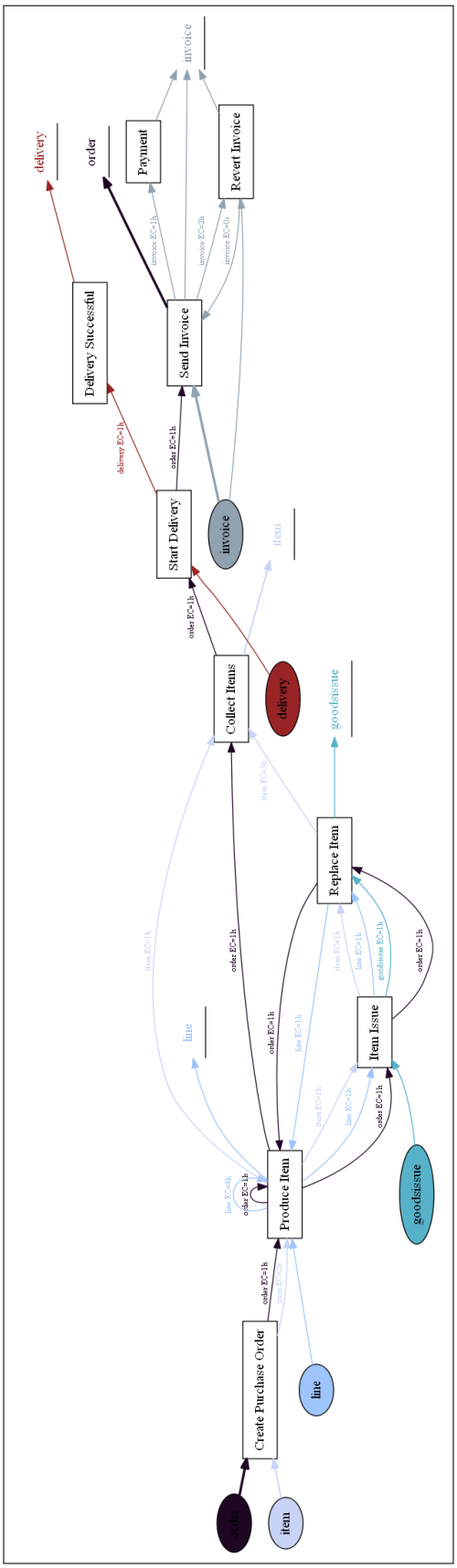


Figure 9.2: OCDFG, annotated with performance metrics, discovered in PM4Py using the code snippet proposed in Listing 9.1.

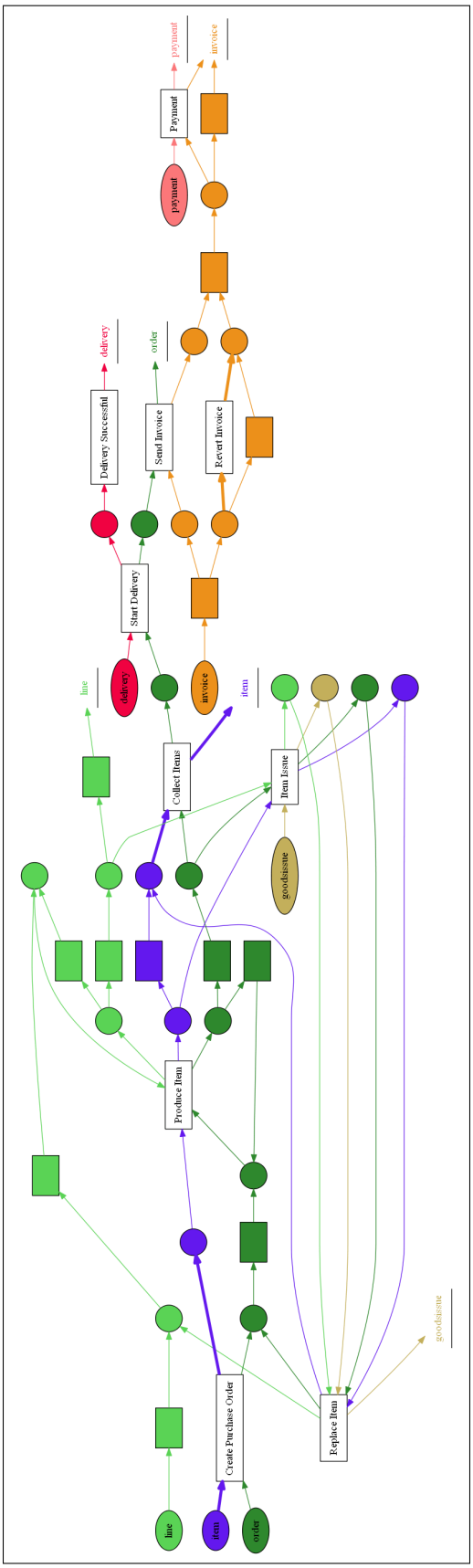


Figure 9.3: OC-Petri net, discovered in PM4Py using the code snippet proposed in Listing 9.1.

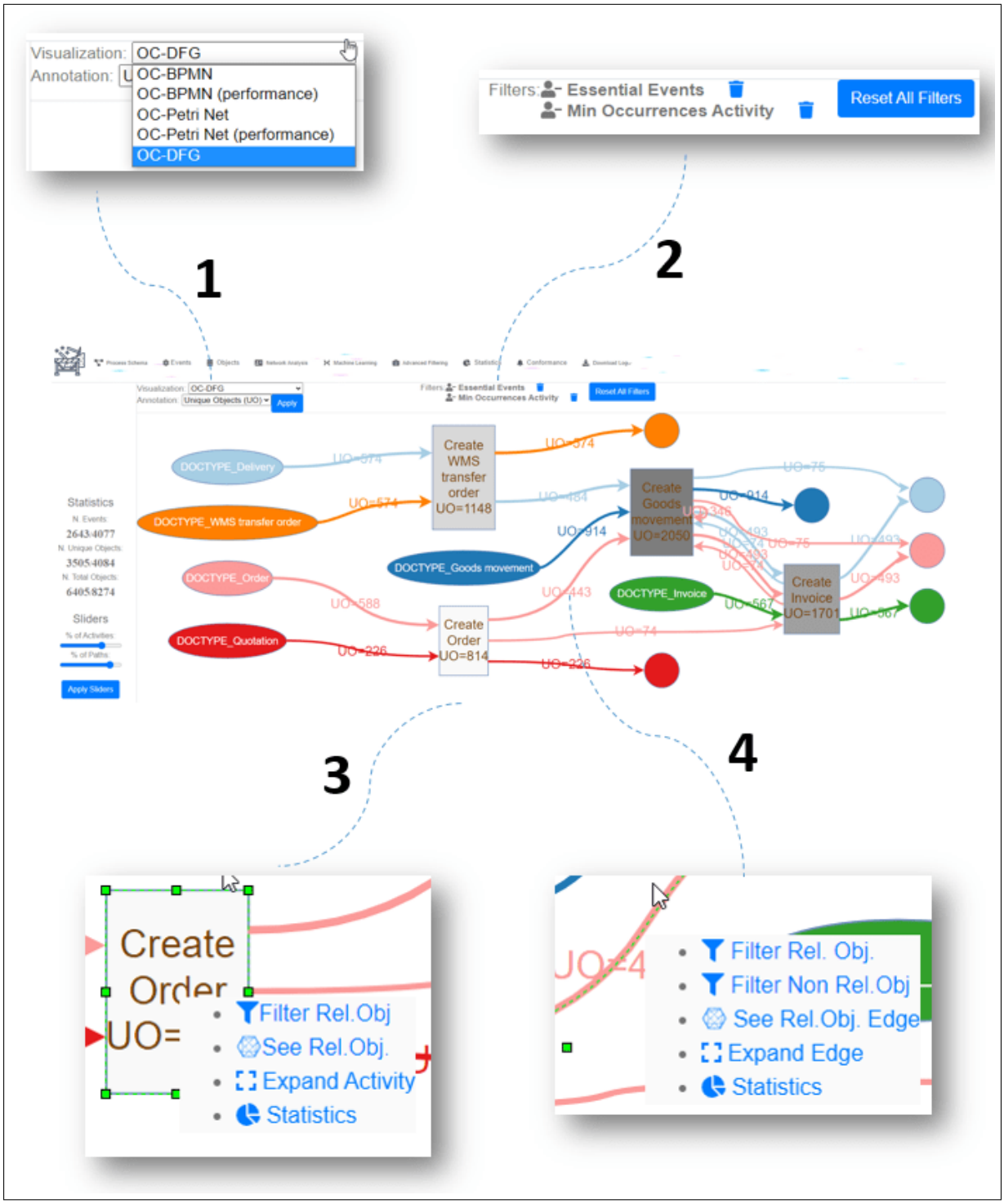


Figure 9.4: Overall view over the process model page of the OC-PM tool.

Please select the correct values for the separator and the quote character of the CSV. Moreover, please provide a mapping for the columns (i.e., which column is the activity, timestamp, object type, event attribute, event identifier).

Separator: , Quote characters: " Object Quotes: Classic (' or ") Re-Build Table

ocel:eid	ocel:timestamp	ocel:activity	ocel:type:element	ocel:type:order	ocel:type:delivery
Event ID	Timestamp	Activity	Object Type	Object Type	Object Type
e1	1980-01-01	Create Order	['i4', 'i1', 'i3', 'i2']	['o1']	
e2	1980-01-02	Confirm Order		['o1']	
e3	1980-01-03	Item out of Stock	['i3']		
e4	1980-01-04	Create Delivery	['i2', 'i1']		['d1']
e5	1980-01-05	Item back in Stock	['i3']		
e6	1980-01-08	Delivery Failed			['d1']
e7	1980-01-09	Retry Delivery			['d1']
e8	1980-01-10	Delivery Successful			['d1']
e9	1980-01-11	Invoice Sent		['o1']	

Figure 9.5: Functionality to convert an unstructured CSV to a structured OCEL.

9.1.2 Overview of the OC-PM Tool

The OCPM (OC-PM) tool [25] is a web-based graphical application for OCPM. It leverages pm4js, a robust JavaScript library for process mining that enables comprehensive in-browser data processing without requiring API calls to external web services.

The pm4js library provides broad capabilities in the field of traditional process mining, offering support to different types of objects:

- Handling of traditional event logs, allowing users to import XES and CSV logs and export logs in XES and CSV formats. Furthermore, users can convert these logs to an event stream for further analysis.
- Creation and management of Petri nets, offering functionality for creating a new Petri net and understanding its execution semantics. Users can import and export Petri nets and visualize them using Graphviz.
- Process trees can also be managed within pm4js, with support for importing, exporting, and visualizing process trees. Importantly, pm4js supports the conversion of process trees into accepting Petri nets.
- The library also deals with Directly-Follows Graphs (DFGs), enabling users to work with both frequency and performance DFGs, import and export frequency DFGs, and maximize DFG capacity.
- BPMN objects are supported, with functionalities including importing and exporting BPMN diagrams, converting BPMN to accepting Petri nets and vice versa.

Support to several traditional process mining algorithms is offered in pm4js, including:

- Process discovery techniques, including the Inductive Miner [71], the Inductive Miner Directly-Follows [73], and the log skeleton [131]. Conformance checking capabilities are also provided, including TBR [24], alignments on Petri nets [10] and DFGs, and conformance checking using the Log Skeleton [131]. Furthermore, it provides functionality for temporal profile conformance checking [113].
- Various metrics are available, including replay fitness [24], ETConformance precision [87], generalization [32], and simplicity[130].
- Support for filtering of event logs and sliding directly follows graphs, and the simulation feature enables playout of a DFG.
- With the feature extraction capabilities [38], users can extract features on event logs and OCELs based on both events and objects.
- Statistics feature offers general log statistics, and for interoperability with Celonis, pm4js provides a Celonis Connector.

The OC-PM tool provides a rich set of OCPM features, including:

- Ingestion/exporting of OCELs in the OCEL standard format (JSON-OCEL and XML-OCEL).
- Flattening the OCELs into traditional event logs with the choice of a case notion.
- Advanced preprocessing features (filtering, sampling).
- Discovering object-centric process models: OCDFGs, and OCPNs.
- Conformance checking on OCELs based on declarative and temporal constraints (log skeleton, temporal profile).
- Exploration of the events/objects of the OCEL.
- Machine learning (anomaly detection, correlation analytics, advanced conformance checking).

Figure 9.6: Visualization of the list of events (along with their attributes and related objects) offered by the OC-PM tool.

Figure 9.7: Visualization of the list of objects for a given object type (along with their relevant dates) offered by the OC-PM tool.

Figure 9.8: Visualization of the features computed on an OCEL using the OC-PM tool.

Figure 9.9: Anomaly detection using the OC-PM tool and the isolation forest method.

Figure 9.10: Planar visualization of the features after the application of a dimensionality reduction technique.

Figure 9.11: Correlation statistics between a feature and all the other features extracted from an OCEL.

Figure 9.12: Decision tree showing the most influent factors for the values of a given feature.

The OC-PM tool's intuitive interface ensures that users can effectively utilize its wide array of functionalities. To aid the understanding of its various features, a series of illustrations are provided:

Conversion of Data Formats: The tool enables users to transform unstructured CSV files into structured OCELs, thus ensuring a seamless data ingestion process. This feature is particularly beneficial when dealing with disparate data sources, as illustrated in Figure 9.5.

Event-centric and Object-centric Views: Visualization is a significant part of any analysis process. In the OC-PM tool, users can explore both the events and the associated attributes as well as the connections to related objects (Figure 9.6). Moreover, the tool provides an overview of objects for a particular object type, showcasing their related timestamps and aiding in understanding the lifecycle of each object type, as seen in Figure 9.7.

Machine Learning and Analytics Features: The integration of machine learning [13] is vital in modern process mining tools. The OC-PM tool offers a range of analytics functionalities:

- Feature computation on object-centric logs enables users to derive crucial insights from the data, as presented in Figure 9.8.
- Anomaly detection using methods such as isolation forests helps in identifying potential outliers or uncommon patterns in the logs. An example of this can be found in Figure 9.9.
- Dimensionality reduction techniques, as shown in Figure 9.10, allow users to visualize complex feature sets in a 2D space, aiding in pattern identification.
- The tool also provides correlation statistics between features, helping users understand relationships and dependencies between different event attributes or object attributes, as evidenced by Figure 9.11.
- Decision trees (Figure 9.12) offer users insights into the most influential factors for a specific feature's values, allowing for a deeper understanding of feature interactions.

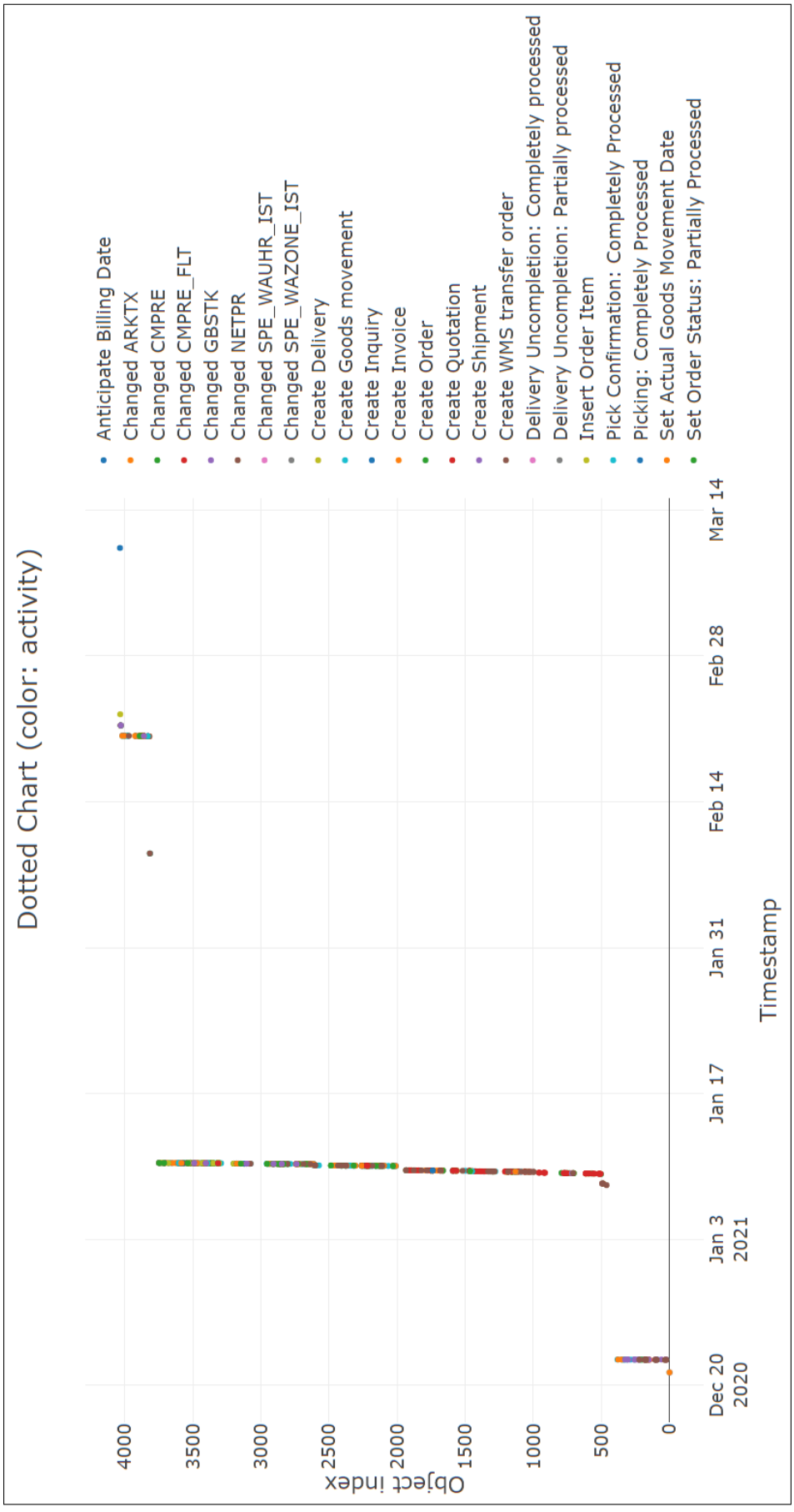


Figure 9.13: Dotted chart visualization (where the Y axis contains all the objects of the OCEL).

We comment into the statistical visualizations provided by the OC-PM tool:

Dotted Chart Visualization: A crucial aspect of process analysis is understanding the temporal distribution of events. The dotted chart visualization, illustrated in Figure 9.13, presents a timeline representation where each dot corresponds to an event. The Y-axis groups these events by the respective objects from the OCEL. This visualization assists in identifying patterns, sequences, or gaps in event occurrences across different objects, aiding in trend analysis and potential bottleneck identification.

Objects per Type Statistic: To understand the distribution and importance of various object types in a process, the OC-PM tool offers the "Objects per type" statistic. As showcased in Figure 9.14, this visualization provides a pie chart representing the count of different object types in the log. It aids in identifying the most frequent and the least frequent object types, enabling analysts to focus on predominant objects or investigate rare ones.

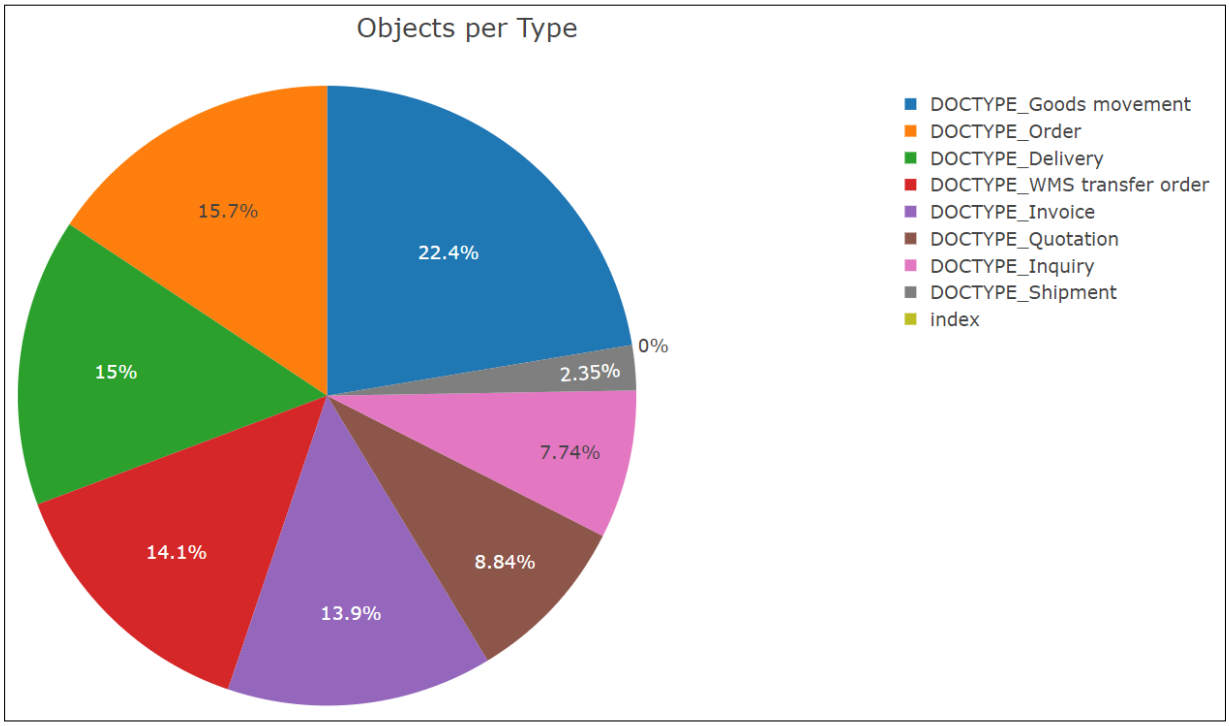


Figure 9.14: Objects per type statistic as computed by the OC-PM tool.

Objects Lifecycle Length Statistic: The duration that each object remains active in the process is an essential metric for several analyses, such as performance analysis or compliance checking. The "Objects lifecycle length" statistic, demonstrated in Figure 9.15, visualizes the time span of each object type from its initiation to completion. This can reveal insights on which objects have longer lifecycles, potentially due to delays, waiting times, or inherent complexity.

Events per Activity Statistic: Analyzing the frequency of activities in a process can provide valuable insights into which tasks are predominant or require optimization. Figure 9.16 depicts the "Events per activity" statistic, where each activity's frequency is displayed. This visualization can help pinpoint the most common activities, thereby allowing for streamlining or optimization opportunities. Conversely, less frequent activities may be candidates for automation or further investigation.

Events per Time Statistic: Understanding how events are distributed over time is vital for workload analysis, forecasting, and identifying seasonal patterns. The "Events per time" statistic, represented in Figure 9.17, plots the number of events against a time axis (e.g., days, weeks, months). Peaks in this graph can indicate busy periods, while troughs can indicate idle times. Such insights are instrumental in resource planning, capacity management, and optimizing process performance.

In summary, these statistical visualizations offered by the OC-PM tool enrich the analytical capabilities of analysts, researchers, and professionals. They not only provide a granular view of the process data but also furnish actionable insights that can be pivotal for process improvement and optimization endeavors.

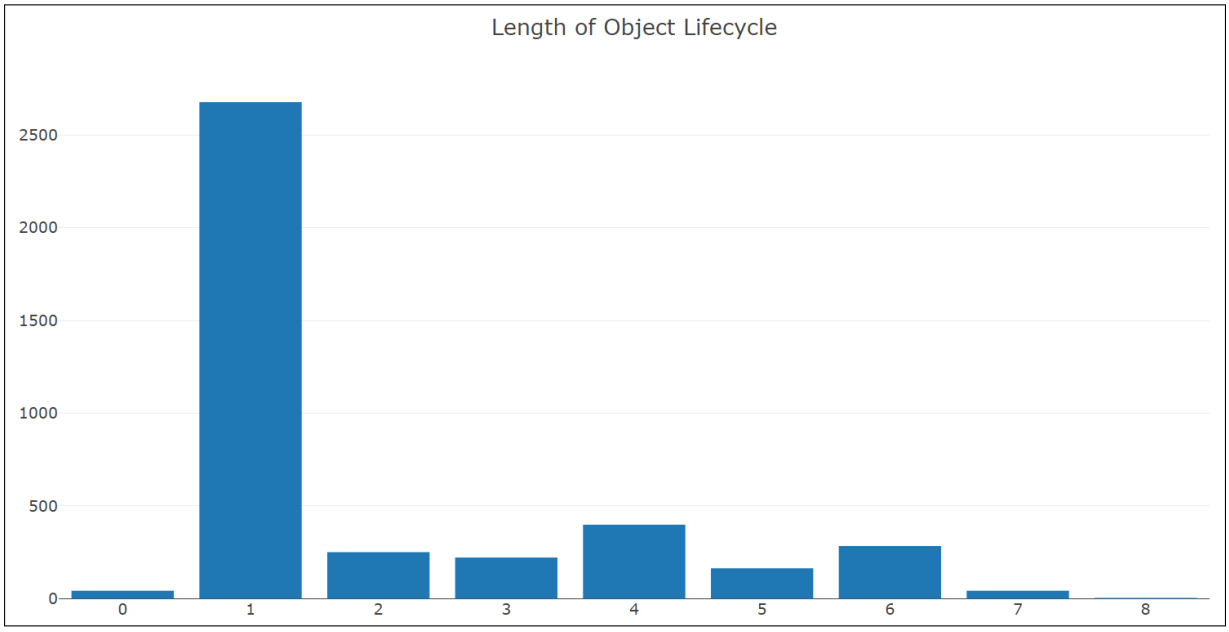


Figure 9.15: Objects lifecycle length statistic as computed by the OC-PM tool.

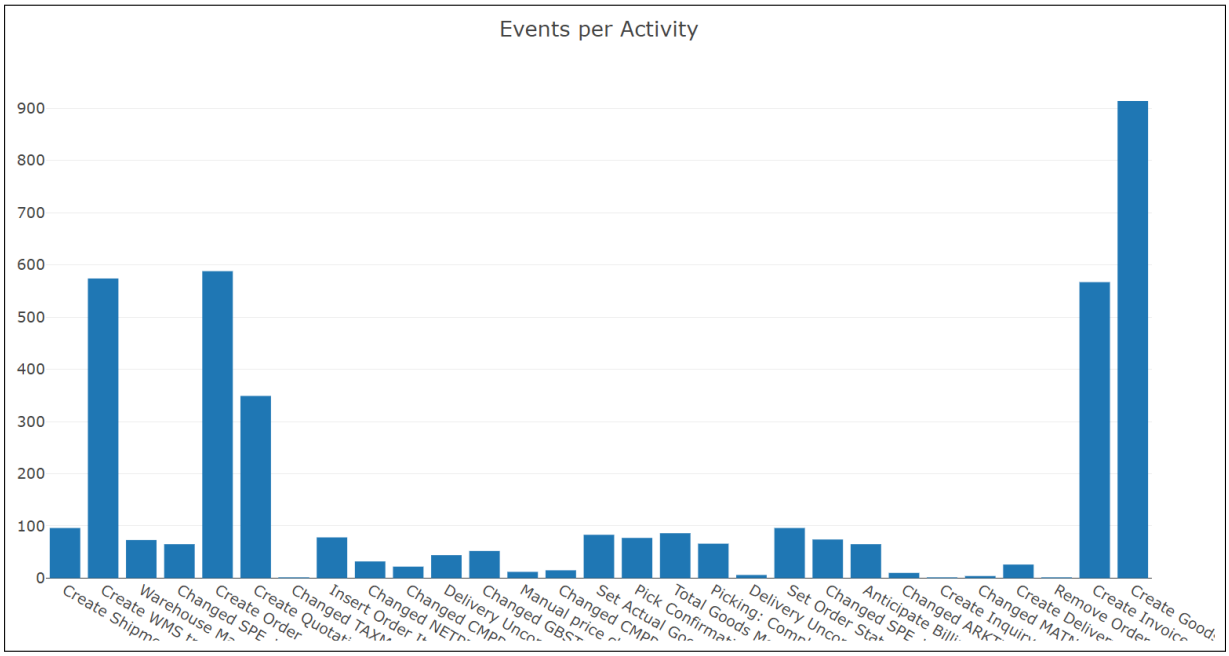


Figure 9.16: Events per activity statistic as computed by the OC-PM tool.

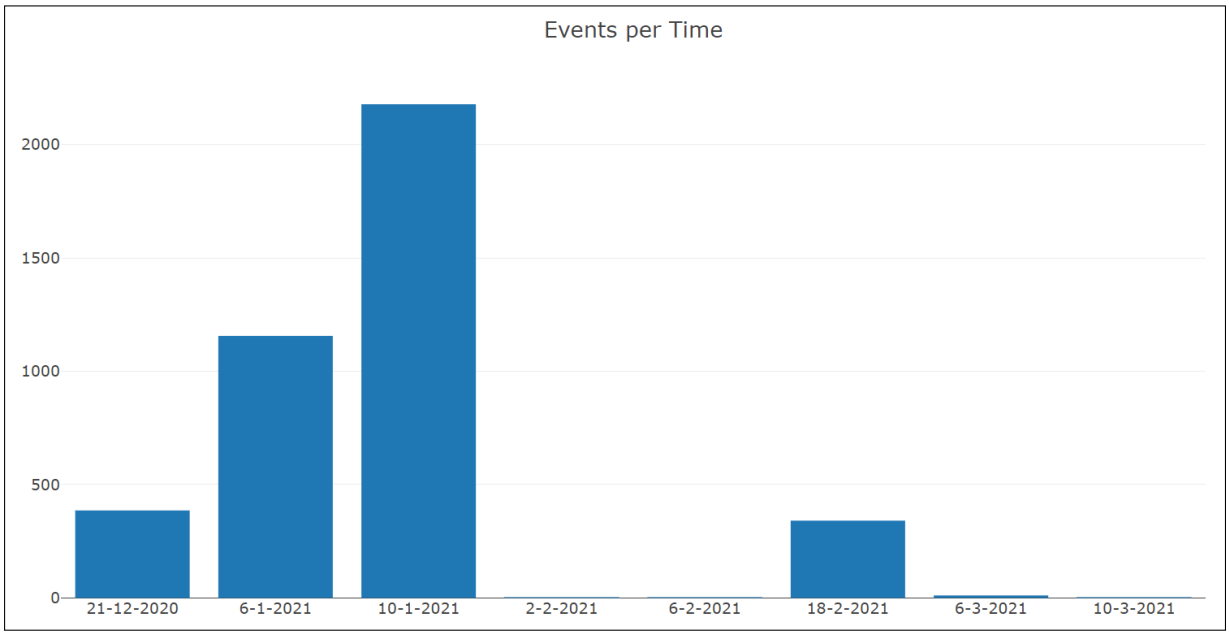


Figure 9.17: Events per time statistic as computed by the OC-PM tool.

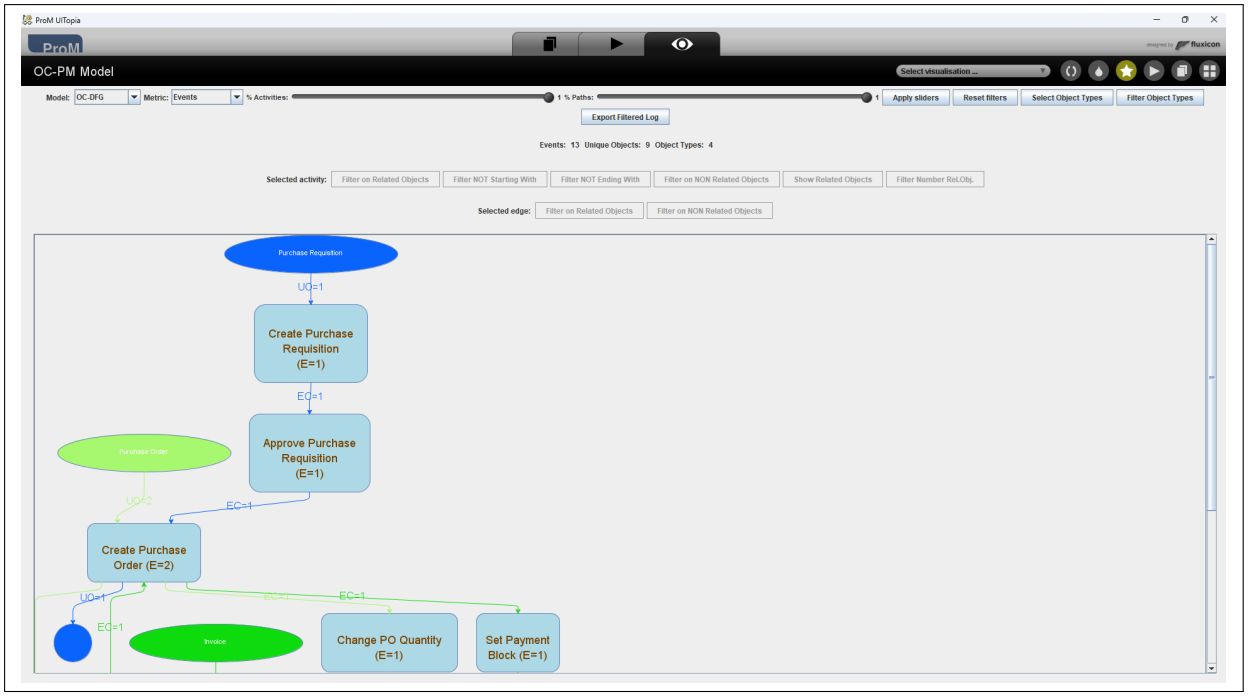


Figure 9.18: Screenshot of the OC-PM Visualizer (OCDFG visualization) in the ProM framework

OC-PM (ProM Framework)

The ProM framework is a versatile and extensible open-source platform primarily used for process mining. It offers a wide range of functionalities through its plugin-based architecture, allowing users to perform various analyses on event logs, from process discovery and conformance checking to performance analysis and prediction. The ProM framework provides a robust foundation upon which specialized packages can be built, and the OCELStandard package is one such extension, tailored for the analysis of Object-Centric Event Logs (OCELs).

The **OCELStandard** package is a collection of plugins for the ProM framework specifically designed to handle and analyze OCELs. This package empowers users to leverage the ProM framework's capabilities in the context of object-centric process mining. The plugins within the OCELStandard package mirror many of the functionalities found in the web-based OC-PM tool, offering a cohesive experience for users who prefer a desktop environment or need to integrate OCEL analysis into larger ProM-based workflows.

The OCELStandard package for ProM includes the following plugins, which align with the functionalities provided by the OC-PM web-based tool:

Data Import/Export:

- *Import OCEL from JSON/XML/CSV*: This plugin enables importing OCEL data from JSON-OCEL, XML-OCEL, CSV files.
- *Export OCEL to CSV/JSON/XML file*: Enables exporting OCEL data to CSV-OCEL, JSON-OCEL, and XML-OCEL formats.
- *Import OCEL 2.0 from JSON/XML/SQLite*: These plugins facilitate the ingestion of OCELs in the standard JSON-OCEL, XML-OCEL, and SQLite formats.
- *Export OCEL 2.0 to JSON/XML/SQLite file*: Allows users to export processed or analyzed OCELs back into the standard formats.

Log Transformation and Filtering:

- *Flatten OCEL to traditional event log* and *Advanced flattening of OCEL log (SAP document flow)*: These plugins convert OCELs into traditional event logs by choosing a case notion, allowing for

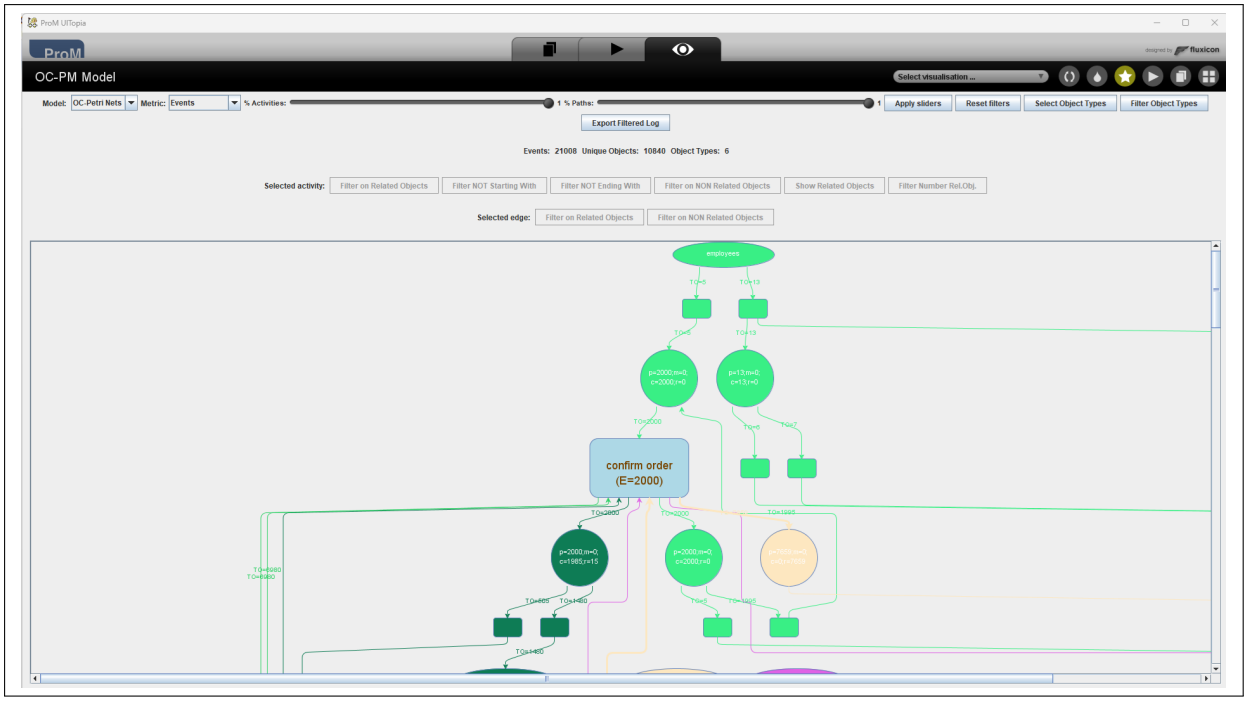


Figure 9.19: Screenshot of the OC-PM Visualizer (OCPN visualization) in the ProM framework

the application of traditional process mining techniques. The advanced version provides support for specific use cases, like SAP document flows.

- *Classic Event Log to OCEL (simple conversion)*: This plugin converts classic event logs into OCEL format, enabling object-centric analysis on traditional data.
- *Filter OCEL on activities - object types correspondence*: Allows filtering based on the relationship between activities and object types.
- *Filter OCEL on activities occurrences*: Enables filtering based on the number of times activities occur.
- *Filter OCEL on object types' objects*: Provides the capability to filter based on specific objects of given object types.
- *Filter OCEL on specified activities* and *Filter OCEL on specified object types*: These plugins allow users to focus on relevant parts of the OCEL by filtering based on specific activities or object types.
- *Filtered log export*: Exports the filtered OCEL for further analysis or storage.

Process Discovery and Visualization:

- *OC-PM Discovery*: Discovers object-centric process models such as OCDFGs and OCPNs (see Figure 9.18 and Figure 9.19).
- *OC-PM Visualizer*: Provides visualizations tailored for OCELs, allowing users to understand the object-centric process flows.
- *Visualize Events from Object-Centric Model*: Visualizes events from the object-centric model.
- *Visualize Lifecycle Duration from Object-Centric Model* and *Visualize Object Lifecycle Length from Object-Centric Model*: Provide insights into the duration of object lifecycles.
- *Visualize Objects from Object-Centric Model*: Enables visualization of objects involved in the process model.

Conformance Checking:

- *Perform Token-Based Replay*: Allows for conformance checking using token-based replay techniques on OCELs.
- *Petri Net with TBR result projected*: Projects the results of token-based replay onto a Petri net for visual inspection.

These plugins collectively provide a comprehensive suite for object-centric process mining within the ProM framework, mirroring the functionalities offered by the web-based OC-PM tool and allowing for in-depth analysis of OCELs in a desktop environment.

9.1.3 Overview of Celonis Object-Centric Process Mining (PAM)

Celonis, a globally recognized process mining software provider, has been at the forefront of innovative developments in the field, particularly regarding OCPM. The object-centric approach takes a broader view compared to traditional event log analysis, accounting for the interconnected nature of different object types through the use of multi-event logs.

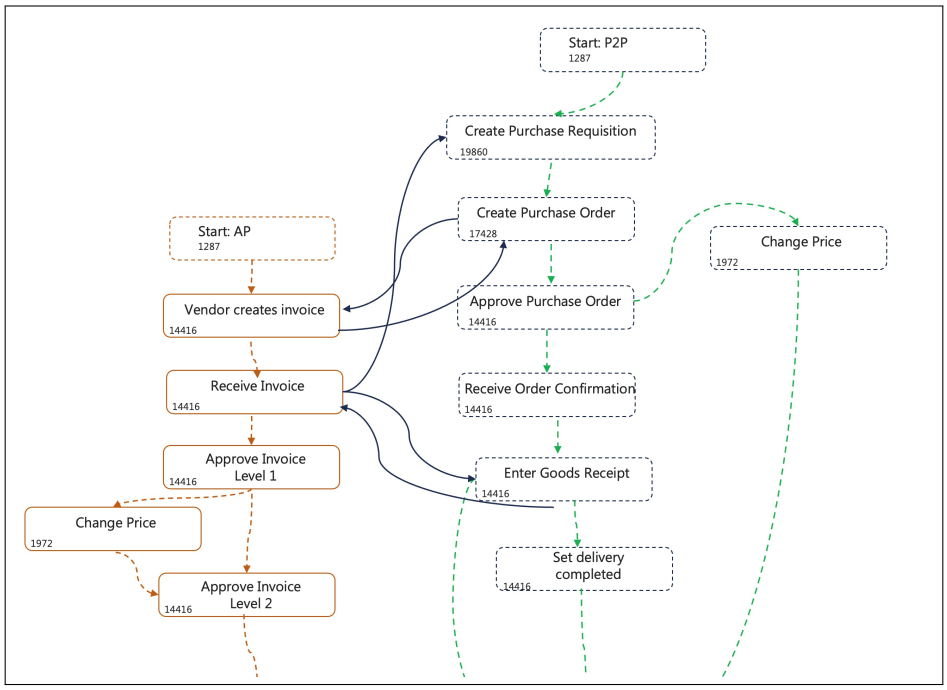


Figure 9.20: Multi-event log visualization showing the interaction between different logical components of the process (procurement and accounts payable).

The first implementation related to OCPM is called *Multi Event Log*. It consists in a data model where event logs of different object types are connected thanks to Object-to-Object relationships. The Object-to-Object relationships within these multi-event logs serve as a crucial source of information for deriving insights about event relations. To this end, Celonis offers four unique miners that enable users to explore these relationships in different ways:

- The Match Miner uses attribute matching to identify and connect events related to interconnected objects. This technique ensures that only those events which share the same attribute values are linked, offering a level of precision in the mining process.
- The Manual Miner, as the name suggests, facilitates user-driven mining. It allows users to manually determine the connections between events based on their activity names. This provides the user with a high degree of control and customization, enabling them to guide the mining process according to their unique needs or understanding of the processes involved.
- The Interleaved Miner and Non-Interleaved Miner focus on extracting temporal order between events of interconnected objects. The Interleaved Miner is more flexible, permitting a single event to serve as both the source and target of a connection. The Non-Interleaved Miner, on the other hand, adopts a more restrictive approach and does not allow for this kind of dual-role event.

The insights are visualized on top of arcs interconnecting the activities of different process maps (one for each event log included in the data model).

Figure 9.20 contains an example multi-event log, in which the interleaved miner has been used to mine temporal interactions between the two process maps (the right one related to the "procurement" log, the left one related to the "accounts payable" log). We could see different interactions, including:

- An interaction between the *Create Purchase Requisition*, the *Create Purchase Order*, and the *Vendor Creates Invoice* activities, considering the order in which they are created. If goods are ordered

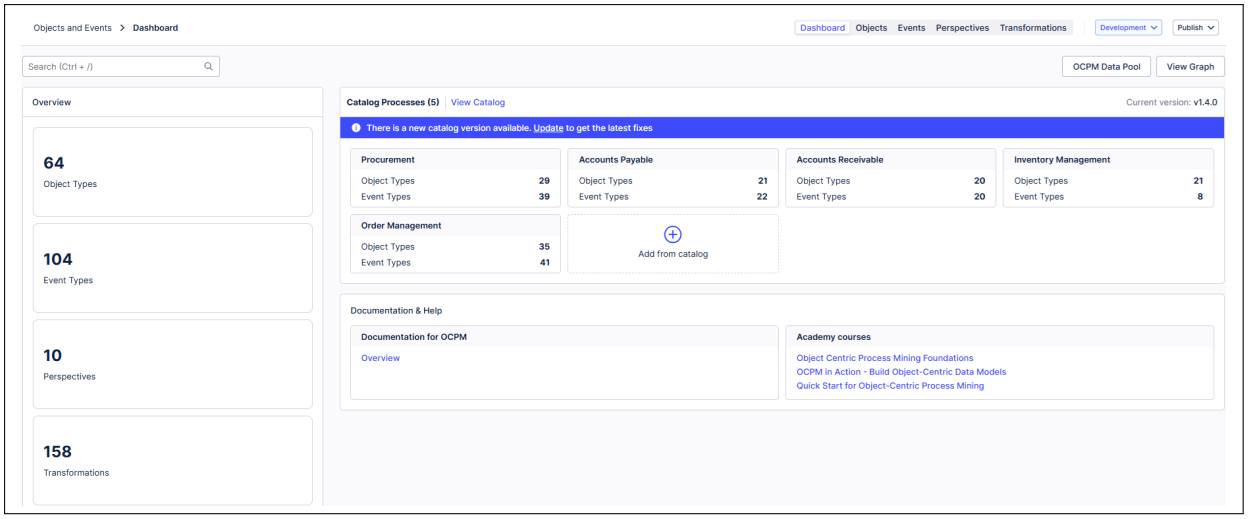


Figure 9.21: "Objects and Events" feature in Celonis, allowing for the ingestion of the Object-Centric Data Model (OCDM).

prior to issuing of Purchase Requisition and Purchase Order, then *maverick buying* happens. Problems connected to maverick buying can lead to a number of issues, such as overspending or lack of proper approvals. Tracking the temporal interactions between different process maps using the interleaved miner can further improve visibility and understanding of the overall purchase process.

- An examination of the interactions between the *Enter Goods Receipt* and *Receive Invoice* activities is crucial to understand how the business users are processing goods receipts in practice. In theory, the goods receipt processing transaction (MIGO, in SAP) should be used to confirm the quantity and quality of received materials. However, if it is observed that there is only a minimal time gap between the receipt of an invoice and entering the goods receipt activity, the users might be entering both invoice information and goods receipt information simultaneously. This interaction can be used to identify categories of goods for which the goods receipt activity is unnecessary and only pro forma. This can help organizations to optimize their processes, eliminate unnecessary steps and reduce costs. Additionally, it can also help to identify potential errors or discrepancies in the invoicing and goods receipt process and improve the accuracy of financial and inventory records.

However, while the meaning of some interactions discovered by the method was clear, some others are quite difficult to interpret, since event-to-event relationships are approximated by the interleaved miner.

Celonis has significantly advanced its support for Object-Centric Process Mining (OCPM) through several key innovations:

- *Object-Centric Data Model (OCDM)*: The OCDM serves as a comprehensive, extensible representation of an entire business, acting as a single source of truth for process intelligence. This model simplifies data integration by allowing businesses to work with familiar terms—such as invoices, orders, and deliveries—rather than the technical language of source systems. It is system-agnostic, enabling companies to leverage Celonis applications and process content regardless of their underlying ERP, SCM, or CRM systems. The OCDM also offers flexibility, allowing organizations to dynamically adjust process analyses without revisiting source data, thereby accelerating onboarding and insight generation[1].

- *Objects and Events Feature*: This feature enables the extraction and transformation of data into objects and events, forming the foundation of the object-centric data model. Users can connect to various source systems to create these objects and events, facilitating a comprehensive view of complex, interrelated business processes. Celonis provides prebuilt extractions and transformations for systems like SAP ECC and Oracle EBS, streamlining the setup process[2].

[1] https://www.celonis.com/blog/celonis-announces-next-generation-mri-process-mining-technology-with-process-sphere/
[2] https://techcrunch.com/2022/11/09/celonis-can-now-map-multiple-processes-and-present-them-in-subway-style-map/

- *PAM (former Process Sphere)*: Introduced as a capability within the Celonis platform, PAM leverages OCPM to provide end-to-end visibility of business operations. It allows users to visualize and analyze the relationships between objects and events across interconnected processes, offering a dynamic, three-dimensional view of operations[3].
- *Multi-Object Process Explorer*: Building upon the existing Process Explorer, this enhanced tool allows users to explore how activities are connected across multiple objects. It enables the visualization of relationships between different objects and how events relate to multiple objects within a process. This capability is particularly valuable for initial exploratory analyses and for creating custom views that address complex, multi-object relationships.

The PAM introduces a novel "subway map" visualization that redefines how organizations view their processes:

1. *Process as Subway Lines:*
   - Each distinct process (e.g., Order-to-Cash or Procure-to-Pay) is represented as a line in the subway map.
   - Intersections or junctions represent touchpoints where multiple processes converge, showing the interrelations between business objects (e.g., orders, deliveries, invoices).
2. *Clarity and Accessibility:*
   - This subway-style abstraction simplifies understanding of even the most complex processes, offering users an intuitive way to grasp relationships without requiring deep technical expertise.
   - The map provides a real-time, interactive overview of how different process lines and objects are connected, highlighting bottlenecks or inefficiencies.
3. *Performance Metrics on Nodes and Paths:*

[3] `https://www.celonis.com/process-sphere/`

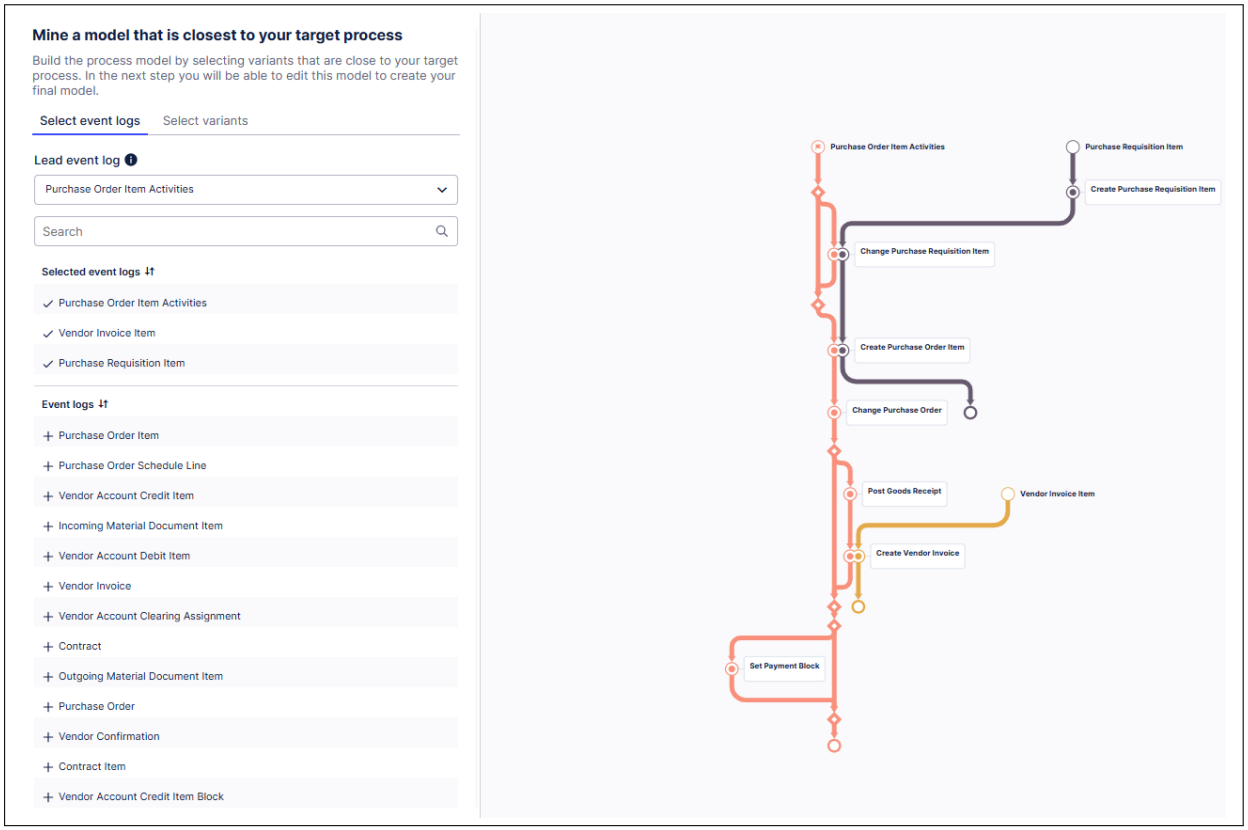


Figure 9.22: "PAM" feature in Celonis, allowing the discovery of "subway maps" from the OCDM.

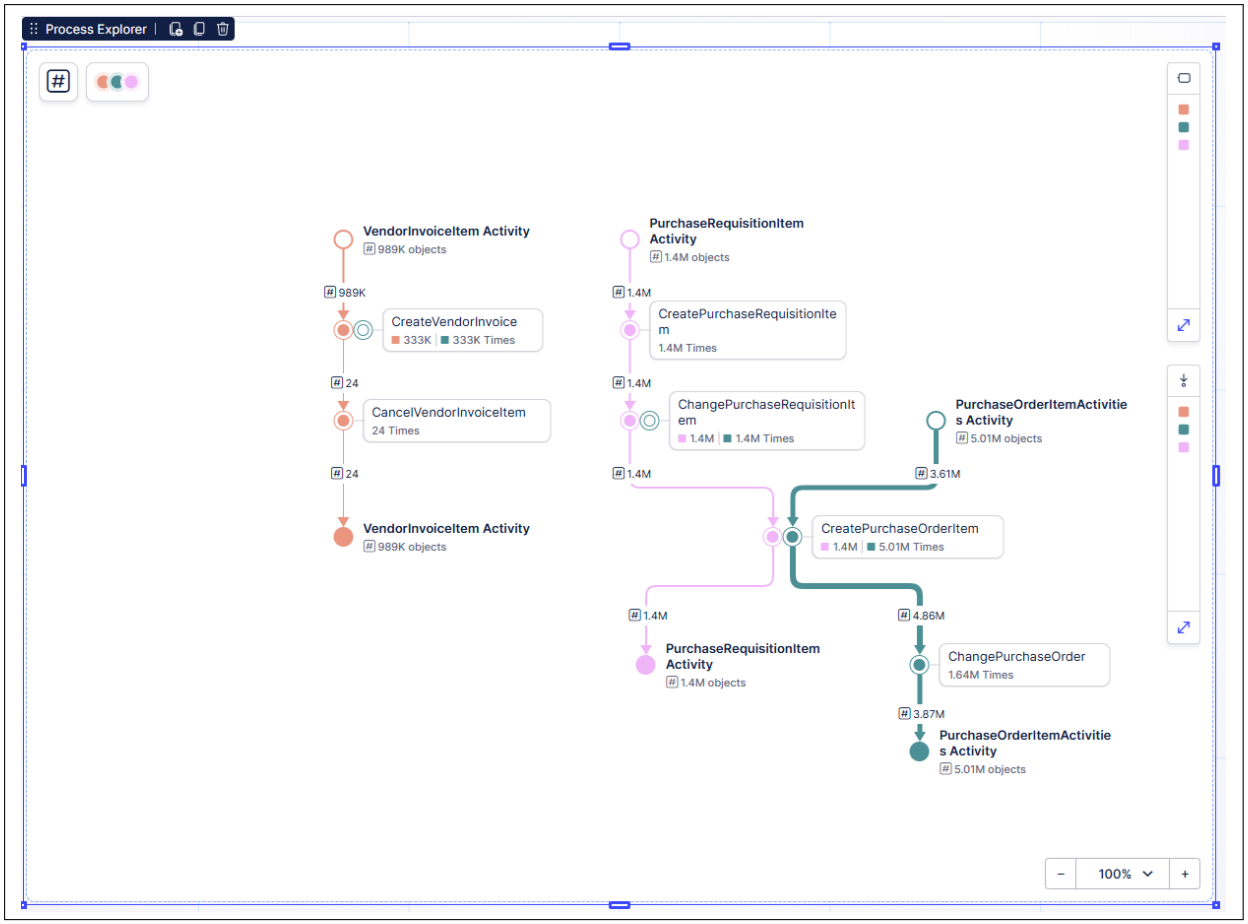


Figure 9.23: "Multi-perspective Process Explorer" in Celonis, allowing the discovery of object-centric directly-follows graphs from the OCDM.

- Each "station" (representing process activities) and "line" (indicating flows between activities) can be enriched with performance metrics, such as average processing time, delays, or throughput, making inefficiencies immediately visible.
- Metrics are contextualized by the relationships between different objects and their associated events.

PAM's *dynamic filtering and exploration tools* empower users to tailor the visualization and drill down into specific process details:

1. *Interactive Path Filtering:*
   - Users can dynamically filter paths based on specific criteria, such as activity type, object relationships, or performance thresholds.
   - For example, users might filter for processes where invoice creation exceeds a particular duration or where specific delivery conditions are met.
2. *Customizable Object Relationships:*
   - PAM enables users to focus on specific object types and their interactions, providing the ability to isolate or combine views of processes involving multiple objects (e.g., connecting orders to shipments).
3. *Exploration Through Zoom and Layering:*
   - Users can zoom in to explore granular details of individual activities or zoom out for a high-level overview of process interconnections.

- Layering tools allow users to view specific process dimensions (e.g., financial performance or compliance) on top of the subway map.

4. *Bottleneck and Opportunity Identification:*

   - The dynamic capabilities allow users to pinpoint bottlenecks, such as overloaded process segments, or uncover opportunities, such as underutilized paths.

9.1.4 Translating Object-Centric Event Logs into a Celonis OCDM

The Celonis Object-Centric Data Model (OCDM) provides a flexible, system-agnostic framework for representing complex business operations, allowing analysts to deal with multiple object types and associated events in a single, coherent schema. Instead of being forced into a single case notion, this approach organizes data around diverse objects (such as orders, invoices, or deliveries), each with its own distinct lifecycle and attributes, while also linking events to objects. Celonis requires a strict naming convention and schema structure for ingesting these data sets: object tables are prefixed with "o_" to store object identifiers and attributes, event tables are prefixed with "e_" to store event records, and relationship tables—connecting events to objects or objects to objects—are prefixed with "r_e_" or "r_o_," respectively, making all these interconnected components align into a consistent and navigable model.

Since Celonis expects data that follows these strict conventions, a crucial preparatory step for leveraging the OCDM in practice often involves starting from an object-centric logging standard like the Object-Centric Event Log (OCEL). OCEL formats (including versions 1.0 and 2.0) represent events connected to multiple object instances, but they do not by themselves conform to Celonis' ingestion rules. In this thesis project, I have implemented a transformation process that standardizes naming, renames activities and object types to strip away non-alphanumeric characters and follow Celonis' naming rules, and then translates the OCEL data into separate event, object, and relationship tables that reflect the Celonis OCDM structure. This process, described in detail in the provided code repository[4], ensures that the resulting data model contains all required tables named according to Celonis conventions (for example, "e_namespace_activityname" for events, "o_namespace_objtype" for objects, and "r_e_namespace_activityname_objtype" or "r_o_namespace_objtype1_objtype2" for relationships) and that the tables are properly linked through foreign keys.

By carrying out this translation, the code developed for this thesis project not only cleans and maps these object-centric event logs into the OCDM, but also sets the stage for more advanced analytics that Celonis provides on top of the model. With an OCDM properly set up, organizations can take advantage of Celonis' multi-object process exploration, more intuitive visualization techniques (including the "subway map" abstraction), and dynamic filtering capabilities. Thus, while OCEL itself is a convenient, general-purpose format for object-centric event data, performing this translation to Celonis' OCDM structure represents a critical enabling step. By adhering to the naming conventions, setting up event-object relationships, and ensuring compatibility with Celonis' table structure, the translated data can fully power Celonis' innovative process intelligence features. In doing so, the code serves as a bridge, turning a neutral, standards-based object-centric event log into an analytically rich OCDM that can drive deeper insights and more agile business decision-making.

[4] `https://github.com/Javert899/ocel20-celonis-connector/`

9.2 Comparison of the Tools

This section provides a comparative analysis of the three previously introduced process mining tools: the PM4Py library, the OC-PM tool, and Celonis Object-Centric Process Mining (OCPM).

First, the similarities and differences in the features and functionalities of the three tools are examined. This segment explores the capabilities of each tool, enabling readers to better understand how these tools can meet requirements in various contexts.

Subsequently, the discussion focuses on the performance, scalability, and ease of use of the tools. The assessment includes how well these tools perform under different data volumes, their scalability in accommodating growing datasets, and the user-friendliness of their interfaces and functionalities.

Finally, the suitability of these tools for different use cases and scenarios is discussed, providing insights into how each tool can be utilized effectively in diverse practical and research-oriented contexts.

Feature/Quality	**Celonis OCPM**	**OC-PM**	**PM4Py**
Functionalities			
Discovery of Object-centric PNs/BPMNs	✓	✓	✓
Activities and paths sliding	✓	✓	✓
Performance analysis	✓	✓	✓
Customization or scripting			✓
Performance & Scalability			
Scalable in-memory engine	✓		
Integration with Python ecosystem			✓
Ease of data model creation		✓	
Ease of Use			
User-friendly interaction		✓	
Requires scripting			✓

Table 9.5: Comparison of OCPM Tools

9.2.1 Similarities and Differences in Features and Functionalities

Celonis OCPM, OC-PM, and PM4Py differ in terms of their features, functionalities, and usability.

Celonis OCPM provides functionalities such as discovering object-centric BPMNs and DFGs, performing activities and path sliding, and enabling performance analysis. These features are essential in an OCPM context, offering a simplified approach for users.

OC-PM is a graphical tool designed to facilitate interaction and accessibility in process mining. It provides features that enable users to interact with activities and edges of diagrams, offering an intuitive and hands-on experience. However, the OC-PM tool does not readily support customization or scripting, which may limit its use for tasks requiring a tailored or programmable approach.

PM4Py is a Python library that offers a similar set of features to the OC-PM tool but requires scripting, which may present a higher learning curve. Unlike OC-PM, PM4Py does not offer interactive functionalities but provides increased flexibility and integration capabilities. As a Python library, PM4Py integrates with the Python ecosystem, enabling the process mining capabilities to be combined with other computational and analytical libraries, thus supporting the development of complex and customizable process mining workflows.

Table 9.5 summarizes the results of this comparison.

9.2.2 Performance, Scalability, and Ease of Use

In terms of performance, scalability, and ease of use, Celonis OCPM, PM4Py, and OC-PM exhibit different characteristics and trade-offs.

Celonis OCPM is integrated with the Celonis in-memory engine, which is designed for handling large volumes of data efficiently, enhancing its scalability. However, establishing an object-centric data model in Celonis OCPM requires the configuration of multiple tables and thorough documentation, which may require considerable effort and a robust skill set.

PM4Py is built on Python, a language known for its versatility. PM4Py leverages the Python ecosystem to improve its performance and provide a range of additional functionalities. It employs libraries such as Pandas for structuring event logs and Scikit-Learn for various machine learning tasks,

enhancing its capabilities beyond process mining. However, as a Python library, it requires scripting, so users need to have knowledge of Python to utilize its features effectively.

OC-PM is based on JavaScript, resulting in lower scalability compared to Celonis OCPM and PM4Py. Nonetheless, it emphasizes ease of use and immediacy of features. The tool is designed with a focus on user experience, ensuring that functionalities are readily accessible and easy to interact with. This makes OC-PM suitable for users seeking an intuitive and hands-on approach to process mining without the need for scripting or complex configuration tasks.

Table 9.5 summarizes the results of this comparison.

Use Case/Scenario	Celonis OCPM	OC-PM	PM4Py
Basic Process Discovery	H	H	H
Detailed Performance Analysis	M	L	H
Custom Process Mining Research	L	L	H
Interactive Exploration	M	H	L
Large-Scale Process Mining	H	L	M

Table 9.6: Suitability of Tools for Different Use Cases and Scenarios (H=High; M=Medium; L=Low).

9.2.3 Suitability for Different Use Cases and Scenarios

When assessing the suitability of Celonis OCPM, PM4Py, and OC-PM for different use cases and scenarios, the varying attributes of each tool should be considered.

Celonis OCPM, with its high scalability and integration with the Celonis in-memory engine, is suitable for large-scale enterprise environments where large volumes of data are common. Its comprehensive data model supports complex scenarios where the relationships between various entities are examined in detail. However, the requirement for extensive configuration and a high level of expertise may render it less suitable for smaller projects or for analysts new to the field.

PM4Py, integrating with the Python ecosystem, offers functionalities beyond process mining, such as data wrangling and machine learning. This versatility makes it appropriate for scenarios requiring a combination of process mining and other data analysis tasks. The scripting requirement allows for extensive customization, suitable for analytical projects with unique or specific requirements. However, it may be less suitable for users lacking Python knowledge or those who prefer a graphical interface.

OC-PM emphasizes ease of use and offers a graphical, interactive environment. This makes it accessible for users who prefer an intuitive, hands-on experience or who need to quickly understand the concepts and techniques of process mining. It is appropriate for rapid prototyping, exploratory analysis, or teaching and learning contexts. However, its reduced scalability could limit its suitability for large-scale, high-volume scenarios.

Table 9.6 summarizes the results of this analysis.

9.3 Guidelines for Selecting and Using the Appropriate Tool for Scientific Research or Practical Tasks

When selecting the appropriate tool for scientific research or practical tasks, several key factors should be considered:

- *Data Volume and Complexity*: For large-scale, complex datasets, Celonis OCPM's scalability and in-memory engine may provide significant advantages. For smaller, less complex datasets, the straightforward and interactive approach of OC-PM or the versatility of PM4Py may be more appropriate.
- *Required Functionalities*: The specific functionalities required for the project should be considered. Celonis OCPM specializes in the discovery of object-centric BPMNs and DFGs, offering sliding activities and paths analysis and some performance analysis features. PM4Py, with its Python integration, offers a broader array of functionalities and benefits from the Python ecosystem. OC-PM provides a user-friendly, interactive environment.
- *Technical Skills*: The technical skills and familiarity with certain programming languages (such as Python or JavaScript) are important factors. For those well-versed in Python, PM4Py may be an

appropriate choice, while users who prefer an intuitive, graphical approach may find OC-PM more suitable.

- *Customization and Interactivity Needs*: If the work requires extensive customization or direct interaction with process diagrams, PM4Py or OC-PM should be considered.
- *Nature of the Task*: For rapid prototyping, exploratory analysis, or teaching and learning contexts, OC-PM's user-friendly interface can be beneficial. For tasks requiring in-depth analysis or combining process mining with other data analysis tasks, PM4Py can be advantageous due to its integration with the Python ecosystem. For large-scale enterprise environments, Celonis OCPM can handle substantial data volumes.

In summary, the decision should be driven by the specific needs of the project or task. It may be helpful to experiment with different tools to identify their strengths and weaknesses in relation to the requirements. The three tools discussed—Celonis OCPM, PM4Py, and OC-PM—each have unique strengths, and understanding these can assist in making an informed choice.

9.4 Recommendations for Future Tool Development and Improvement

Future Developments	Celonis OCPM	OC-PM	PM4Py
Scalability Improvement	-	✓	✓
Increase Customizability	✓	✓	-
Enhance Interactivity	-	-	✓
Ease of Data Ingestion	✓	-	-
Enhance Documentation	✓	-	-

Table 9.7: Directions for Future Tool Development

Based on the preceding discussions, several recommendations for future tool development and improvement in the realm of OCPM are identified:

1. *Enhancing Scalability*: With the increasing data volumes in organizations, scalability is a paramount concern. While Celonis OCPM integrates scalable solutions, other tools such as OC-PM and PM4Py could improve their capacity to handle larger datasets. Incorporating more efficient data structures or algorithms, or leveraging parallel computing or cloud technologies, may enhance scalability.
2. *Increasing Interactivity*: Tools like OC-PM emphasize user interaction, allowing direct manipulation of process diagrams, which enhances usability. Future development could focus on expanding interactivity in all tools, potentially integrating machine learning algorithms that adapt and learn from user interactions.
3. *Expanding Functionalities*: Although PM4Py offers an extensive range of functionalities through its integration with the Python ecosystem, there is potential for all tools to expand their features to address a wider array of use cases. This may involve incorporating advanced analytics capabilities or introducing new methods to visualize and explore process data.
4. *Improving Documentation and Usability*: Comprehensive documentation and user-friendly interfaces enhance a tool's utility. For tools like Celonis OCPM, which require significant knowledge to employ effectively, improving the clarity and completeness of documentation is important. User interface design is also crucial for making tools accessible and intuitive to both technical and non-technical users.
5. *Encouraging Interoperability*: As organizations rely on various software tools, interoperability is critical. Future developments could focus on how these process mining tools can effectively interact with other software applications and data formats, facilitating seamless data exchange and integration.

6. *Implementing Robust Performance Measures*: With the wider adoption of process mining in operational contexts, having robust, real-time performance measurement capabilities is important. Future tool development may include features that allow for the tracking of key performance indicators, process efficiencies, and other metrics relevant to operational success.

Table 9.7 summarizes possible lines of development for the three tools. In conclusion, while the current suite of tools for OCPM offers substantial capabilities, there is opportunity for future development and improvement. These recommendations outline potential areas where tool developers can innovate and evolve their offerings to meet the changing needs of the field.

Conclusion

This chapter has explored the essential role of tooling in Object-Centric Process Mining, providing a comprehensive overview and comparative analysis of three prominent tools: PM4Py, OC-PM, and Celonis OCPM. We have examined their respective features, functionalities, performance characteristics, and suitability for different use cases, offering practical guidance for tool selection. While each tool offers valuable capabilities for OCPM, their strengths and weaknesses cater to different needs and user profiles. PM4Py's extensibility within the Python ecosystem makes it a powerful choice for research and complex analytical tasks, while OC-PM's user-friendly interface promotes interactive exploration and rapid prototyping. Celonis OCPM, with its focus on scalability and enterprise integration, caters to large-scale process analysis and improvement initiatives. The recommendations for future tool development, including scalability enhancements, increased customizability, improved interactivity, and streamlined data ingestion, provide a roadmap for advancing the state-of-the-art in OCPM tooling.

Chapter 10

Conclusion and Future Work

"*Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world.*"
Louis Pasteur

10.1 Summary of the Main Contributions and Findings of the Thesis

Object-centric process mining (OCPM) offers significant advantages over traditional, process-centric approaches. First, it reduces the need to repeatedly go back to the source systems for information, streamlining the analysis process. Second, OCPM mitigates distortions commonly encountered in traditional process mining, such as deficiency, convergence, and divergence, by considering the interactions of multiple object types. Finally, and crucially, OCPM provides the ability to see and understand the complex interactions between different object types within a process, offering a more holistic and accurate view of operations.

The primary objective of this thesis was to explore, innovate, and develop methodologies for OCPM. Over the course of various chapters, this exploration covered multiple facets, ranging from the foundations of Object-Centric Event Logs (OCELs) to their application in real-world scenarios. Herein, we summarize the main contributions and findings derived from the contents:

1. *Theoretical Foundations and Preliminaries* (Chapter 2 and Chapter 3): This research began by establishing a robust background on traditional paradigms like relational databases and event logs. These foundations allowed for the clear differentiation of the OCPM domain and its intrinsic challenges, setting the stage for subsequent innovations. Furthermore, by integrating the concept of LLMs, especially in the context of business process management, the thesis ventured into the potential of combining state-of-the-art technologies with traditional BPM paradigms.

2. *Literature Review and Existing Approaches* (Chapter 4): A comprehensive review of existing methodologies and practices in OCPM was undertaken. This chapter not only set the thesis in the broader research context but also identified gaps and potential areas of enhancement, making it pivotal for the development of novel methodologies later in the research.

3. *Data Extraction and Preprocessing* (Chapter 5): Recognizing that the efficacy of any process mining technique is rooted in the quality and representativeness of data, this research dedicated an entire chapter to data extraction and preprocessing. The development of a structured approach for data extraction, especially in the context of OCELs, was a significant contribution. Notably, the three-step methodology – from table selection to formulating blueprints to actual event log extraction – showcased the intricacy of the process and the innovations therein. The introduction of the OCEL format added a standardized structure, facilitating better analytics in subsequent stages.

4. *Object-Centric Process Discovery* (Chapter 6): A core contribution of this thesis was the formulation of novel algorithms and methods to discover processes in an object-centric manner. This chapter dived deep into the intricacies of process discovery, from collations of traditional models to the creation of OCDFGs. The flexibility of these methods, evident from their application to diverse processes like Purchase-to-Pay and Order-to-Cash, underscored their robustness and versatility.

5. *Object-Centric Conformance Checking* (Chapter 7): Beyond discovery, ensuring that the identified processes conform to expectations and standards is crucial. This research addressed the unique challenges posed by OCPM in the realm of conformance checking, thereby extending its practical implications.

6. *Real-world Applications and Tooling* (Chapters 8 and 9): The ultimate test of any research lies in its applicability, and this thesis did not fall short. Through real-world case studies, the practicality and utility of the developed methodologies were exemplified. Furthermore, the development of dedicated tools tailored for OCPM bridged the academic-industry gap, enabling practitioners to leverage the research findings in tangible settings.

In summary, this thesis, through its exploration and innovation in the domain of OCPM, has not only advanced the state-of-the-art but also paved the way for practical, real-world applications of these methodologies.

10.2 Implications of the Research for the Process Mining Field

The implications of this research are important for the field of process mining, affecting both academic and industrial communities. Below, we discuss these implications in a structured manner:

1. *Shift Towards Object-Centricity*: Traditionally, process mining has been centered on flat event logs, wherein events are often sequenced in a linear manner. This thesis presents a comprehensive exploration into OCPM, which recognizes the interplay of multiple entities and their relationships in process executions. The shift towards OCPM could pave the way for a more holistic understanding of complex processes, especially in large enterprises where multiple entities interact simultaneously.

2. *Standardization of Data Structures*: With the introduction of the OCEL format, there is now a push towards standardization in the representation of OCELs. This could have a cascading effect on tool development, interoperability, and research reproducibility in the field. Standardized data structures reduce ambiguities and increase the efficiency of data processing, a critical aspect for real-time process mining applications.

3. *Enhanced Analytical Capabilities*: The methodologies introduced in this research, especially those related to process discovery and conformance checking, broaden the analytical spectrum of process mining. These methods, being tailored for object-centric data, can potentially uncover insights that traditional techniques might overlook, offering a more granular view of processes and their nuances.

4. *Improved Data Extraction and Preprocessing*: The systematic approach towards data extraction and preprocessing can serve as a blueprint for future research and applications. By addressing challenges specific to object-centric logs, the research ensures that subsequent analysis is based on accurate, comprehensive, and representative data.

5. *Interdisciplinary Integration*: By weaving in concepts of LLMs into process mining, the research showcases the potential of interdisciplinary integrations. Such integrations could be the key to unlocking novel solutions and insights, especially as artificial intelligence and machine learning continue to evolve and influence various research domains.

6. *Practical Applicability and Tooling*: The real-world case studies and dedicated tooling discussed in this thesis underscore the direct applicability of the research in industrial settings. This bridges the often-existing gap between academic research and industry practice, ensuring that the innovations and findings can be implemented and tested in tangible scenarios.

7. *Stimulus for Future Research*: By identifying gaps, challenges, and proposing novel solutions in the domain of OCPM, this research can serve as a stimulus for future explorations. The open questions and challenges posed will likely guide subsequent investigations, fostering an environment of continuous learning and innovation in the field of process mining.

10.3 Connection of Research Questions/Goals to the Objectives

This section ties the main objectives and research questions, as laid out in the introduction, to the respective sections and chapters of the thesis. It serves as a bridge, emphasizing how each portion of the content contributes to answering the formulated questions and achieving the set goals.

10.3.1 Main Objectives and Corresponding Sections

O1: Address Data Extraction and Preprocessing Challenges:

- Chapter 5: Primarily devoted to data extraction and preprocessing for OCELs.
- Sections 5.1 to 5.5: Provide methodologies, from data extraction to assessment, to address challenges associated with OCELs.

O2: Improve Process Discovery and Conformance Checking Methods:

- Chapter 6: Introduces OCPD, detailing methods ranging from traditional adaptations to object-centric specific approaches.
- Chapter 7: Dedicated to OCCC methods, from log-model comparisons to criteria for selecting relevant activities.

O3: Implement Real-world Case Studies and Applications:

- Chapter 8: Demonstrates the application of developed methodologies in real-world scenarios, providing insights into their practicality and efficacy.

O4: Develop and Provide Effective Tools for OCPM:

- Chapter 9: Focuses on the development and provision of software tools tailored specifically for OCPM.

10.3.2 Research Questions and Corresponding Sections

RQ1: Definition of OCELs:

- Addressed in Chapter 3, particularly in sections discussing examples like P2P and O2C.

RQ2: Data extraction and preprocessing techniques for OCELs:

- Thoroughly explored in Chapter 5.

RQ3: Novel method for process discovery in object-centric settings:

- Addressed in Chapter 6 with emphasis on OCDFGs, Petri nets, and Object Graph Enrichments.

RQ4: Unique approach for conformance checking in object-centric setting:

- Expounded upon in Chapter 7, with sections discussing the nuances of log-model comparison in object-centric logs.

RQ5: Adapting feature extraction and anomaly detection techniques for OCELs:

- Covered in Section 3.5 of Chapter 3 and in Chapter 7.

RQ6: Extending traditional event log definitions for object-centric complexities:

- Addressed in Chapter 3 with real-world and minimal examples.

RQ7: Tools and libraries for OCPM:

- Discussed in Chapter 9, focusing on specific tools and libraries designed for this purpose.

RQ8: Comparing the performance and functionality of OCPM tools:

- Explored within Chapter 5 in the assessment sections, and further evaluated in Chapter 9 where the developed tools are compared in terms of their performance and scalability.

10.4 Identification of Open Research Questions and Directions for Future Work

Throughout this thesis, we have taken comprehensive strides in understanding and enhancing the landscape of OCPM. While significant progress has been made in answering the outlined research questions, as with any scientific endeavor, our efforts have also highlighted new challenges and opened new research avenues. This section discusses the emergent open research questions and suggests potential directions for future endeavors in OCPM.

10.4.1 Open Research Questions

ORQ1 *Generalization across Industries*: While we focused on ERP processes, how can the methodologies developed be generalized across other industries or processes with unique characteristics?

ORQ2 *Process Enhancement and Prediction*: Our primary focus was on data extraction, process discovery, and conformance checking. What about the potential for object-centric process enhancement or predictive analytics in this domain?

ORQ3 *Quality and Availability of Data*: Given the limitations tied to the availability and quality of data in empirical research, how can we ensure the robustness and reliability of OCELs? What methods can be developed to assess and improve data quality for such logs?

ORQ4 *Transferability of Case Study Results*: How can we ensure that the results from specific case studies are more widely applicable? What methodologies can be introduced to test the generalizability of findings from one setting to another in OCPM?

ORQ5 *Scalability and Real-time Processing*: With the increasing volume of data in today's digital ecosystems, how can OCPM techniques be adapted for real-time processing and analysis on larger scales?

ORQ6 *Integration with Other Technologies*: With the proliferation of technologies like AI, IoT, and blockchain, how can OCPM be integrated with these advancements to provide more holistic solutions?

ORQ7 *Ethical Considerations*: As with all data-driven approaches, what are the ethical considerations in OCPM? How can we ensure privacy, transparency, and fairness?

10.4.2 Directions for Future Work

The open research questions provide a structured direction for future research endeavors. Some proposed directions include:

- *Cross-industry Collaboration*: Engage in collaborative studies across diverse industries to refine and generalize the OCPM methodologies.
- *Enhanced Data Quality Frameworks*: Introduce frameworks and tools specifically tailored to assess, enhance, and maintain the quality of OCELs.
- *Real-time Processing Solutions*: Research on algorithmic enhancements and optimizations to ensure that OCPM can handle real-time data streams.
- *User Experience (UX) Design*: Given the potential implications on human actors, a focus on UX design in tools and methodologies will ensure a seamless transition and adoption of OCPM.
- *Ethical Guidelines and Frameworks*: Formulate guidelines that specifically cater to the unique challenges posed by OCPM, ensuring ethical handling of data and processes.
- *Integration Research*: Explore ways to meaningfully integrate OCPM with emerging technologies, creating holistic solutions for modern business challenges.

In conclusion, while this thesis has provided significant insights into the domain of OCPM, the field remains ripe for exploration. As we stand on the cusp of this promising domain, these open research questions and future directions offer a roadmap for the next generation of researchers.

Chapter 11

Summary of Contributions

This thesis presents the findings from a series of research studies, that have been published in various academic journals and workshops. These publications have contributed to the advancement of the field of OCPM. The research findings have also been shared with the academic community, contributing to an exchange of ideas and continual improvement in the field.

Journal Publications

In this section, the peer-reviewed journal papers that have emanated from the research conducted during my Ph.D. studies are presented.

Journal Paper 1 (second iteration of OCDFG)

Alessandro Berti and Wil M. P. van der Aalst. Extracting multiple viewpoint models from relational databases. In Paolo Ceravolo, Maurice van Keulen, and María Teresa Gómez López, editors, *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13-14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers*, volume 379 of *Lecture Notes in Business Information Processing*, pages 24–51. Springer, 2019. [22]

The paper presented OCDFGs, "MVP models", computed from an OCEL. It also carried out a comprehensive evaluation of the model's usability and scalability. It is referenced in Chapter 6.

Journal Paper 2 (improved TBR)

Alessandro Berti and Wil M. P. van der Aalst. A novel TBR technique to speed up conformance checking and process enhancement. *Trans. Petri Nets Other Model. Concurr.*, 15:1–26, 2021. [24]

The paper offered a refined TBR technique for handling invisible transitions and the token explosion problem. It included a comparison with other techniques and an evaluation of parameter effects, as well as advanced process diagnostics. It is referenced in Chapter 6 and Chapter 7.

Journal Paper 3 (third iteration of OCDFG)

Alessandro Berti and Wil M. P. van der Aalst. OC-PM: analyzing OCELs and process models. *Int. J. Softw. Tools Technol. Transf.*, 25(1):1–17, 2023. [25]

The paper explored OCDFGs with extensive discussions on various activity and edge metrics and introduced tool support for OCPD. It is referenced in Chapter 6 and Chapter 9.

Journal Paper 4 (Object-Centric Feature Extraction)

Alessandro Berti, Johannes Herforth, Mahnaz Sadat Qafari, and Wil M. P. van der Aalst. Graph-based feature extraction on OCELs. *International Journal of Data Science and Analytics*, 2023. [13]

The paper proposed a feature extraction methodology on OCELs. The resulting features were a combination of lifecycle-related features and graph-based features, concerning object interactions. It is referenced in Chapter 3.

Journal Paper 5 (second iteration of Automatic Extraction of OCELs)

Alessandro Berti, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. A generic approach to extract object-centric event data from databases supporting sap erp. *Journal of Intelligent Information Systems*, 2023. [19]

This paper presented a semi-automated methodology for extracting OCELs from a class of relational databases, extending the workshop contribution. The paper also included scalability and quality assessment. It is referenced in Chapter 5 and Chapter 8.

Journal Paper 6 (Case Study of OCPM on SAP ECC ERP)

Alessandro Berti, Urszula Jessen, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. Analyzing interconnected processes: using object-centric process mining to analyze procurement processes. *International Journal of Data Science and Analytics*, 2023. [14]

The paper demonstrated an application of OCPM to a real-life Purchase-to-Pay process using a revised process mining project methodology. It is referenced in Chapter 8.

Journal Paper 7 (OCPNs)

Wil M. P. van der Aalst and Alessandro Berti. Discovering OCPNs. *Fundam. Informaticae*, 175(1-4):1–40, 2020. [119]

The paper introduced the discovery of OCPNs from OCELs, with TBR being instrumental in defining arc typing and providing quality metrics for the resultant model. It is referenced in Chapter 6.

Journal Paper 8 (PM4Py)

Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. PM4Py: A process mining library for python. *Software Impacts*, 17:100556, 2023. [26]

PM4Py is a versatile Python process mining library. This paper introduces its features, including OCPM, and highlights its profound influence in academia, industry, and the open-source realm. Its broad acceptance underscores its importance for both researchers and professionals in process mining. It is referenced in Chapter 9.

Workshop Contributions

In this section, my contributions to workshops which provided a platform for presenting preliminary results and receiving feedback are outlined.

Workshop Contribution 1 (first iteration of OCDFG)

Alessandro Berti and Wil M. P. van der Aalst. Starstar models: Using events at database level for process analysis. In Paolo Ceravolo, María Teresa Gómez López, and Maurice van Keulen, editors, *Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain, December 13-14, 2018*, volume 2270 of *CEUR Workshop Proceedings*, pages 60–64. CEUR-WS.org, 2018. [21]

This paper presented the initial version of OCDFGs, "StarStar Models", computed from an Event-to-Object Graph Enrichment. The models comprise of activities as nodes with varied interconnections based on the object types.

Workshop Contribution 2 (TBR)

Alessandro Berti and Wil M. P. van der Aalst. Reviving TBR: Increasing speed while improving diagnostics. In Wil M. P. van der Aalst, Robin Bergenthum, and Josep Carmona, editors, *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2019 Satellite event of the conferences: 40th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2019 and 19th International Conference on Application of Concurrency to System Design ACSD 2019, ATAED@Petri Nets/ACSD 2019, Aachen, Germany, June 25, 2019*, volume 2371 of *CEUR Workshop Proceedings*, pages 87–103. CEUR-WS.org, 2019. [23]

The paper introduced a novel TBR for handling invisible transitions and the token explosion problem, instrumental for OCPNs' discovery and conformance checking with Petri nets. It is referenced in Chapter 6 and Chapter 7.

Workshop Contribution 3 (first iteration of Automatic Extraction of OCELs)

Alessandro Berti, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. An event data extraction approach from SAP ERP for process mining. In Jorge Munoz-Gama and Xixi Lu, editors, *Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers*, volume 433 of *Lecture Notes in Business Information Processing*, pages 255–267. Springer, 2021. [18]

The paper proposed a semi-automated methodology for extracting OCELs from relational databases, specifically from SAP ERP, with user selection of pertinent tables for a given process. It is referenced in Chapter 5.

Workshop Contribution 4 (OCEL standard)

Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. OCEL: A standard for OCELs. In Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, Raimundas Matulevicius, Ahmed Awad, Matthias Weidlich, Mirjana Ivanovic, and Olaf Hartig, editors, *New Trends in Database and Information Systems - ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Tartu, Estonia, August 24-26, 2021, Proceedings*, volume 1450 of *Communications in Computer and Information Science*, pages 169–175. Springer, 2021. [53]

The paper introduced the OCEL standard for storing OCELs, with two proposed implementations based on JSON and XML. It is referenced in Chapter 3 and Chapter 5.

Workshop Contribution 5 (LLMs for Process Mining)

Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. In *Proceedings of the NLP4BPM 2023 Workshop*, 2023. [20]

Modern LLMs, such as GPT-4, can answer complex queries starting from textual abstractions of process mining artifacts (event logs, Directly-Follows Graphs, Petri nets). It is referenced in Chapter 6.

Workshop Contribution 6 (Object-Centric Anomaly Detection)

Alessandro Berti, Urszula Jessen, Wil M. P. van der Aalst, and Dirk Fahland. Explainable object-centric anomaly detection: the role of domain knowledge. In Adela del-Río-Ortega, Marco Montali, Stefanie Rinderle-Ma, Hajo A. Reijers, Jan vom Brocke, Mathias Weske, Benoît Depaire, Marta Indulska, Han van der Aa, Weronika T. Adrian, Laura Genga, Sander J. J. Leemans, Katarzyna Gdowska, María Teresa Gómez-López, Jana-Rebecca Rehse, and Simone Agostinelli, editors, *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum at BPM 2024 co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024*, volume 3758 of *CEUR Workshop Proceedings*, pages 162–168. CEUR-WS.org, 2024. [15]

The paper introduces three different methodologies for object-centric anomaly detection. It is referenced in Chapter 7.

Demo Paper Contributions

Demo papers can play an essential role in the research landscape, showcasing the application and functionality of novel systems or methods. The contributions derived from these practical demonstrations often foster a deeper understanding of the theory and can inspire future enhancements. The following demo papers have been submitted and accepted during the course of this research:

Demo Paper 1 (PM4Py library)

Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Process Mining for Python (PM4Py): Bridging the gap between process- and data science. In *Proceedings of the 1st International Conference on Process Mining (ICPM 2019), Demo Track*. CEUR, 2019. [27]

The paper presented the PM4Py library, a versatile process mining library in Python with a comprehensive set of OCPM features, maintained by the author of this thesis. It is referenced in Chapter 9.

Demo Paper 2 (MongoDB scalable support for OCELs)

Alessandro Berti, Anahita Farhang Ghahfarokhi, Gyunam Park, and Wil M. P. van der Aalst. A scalable database for the storage of OCELs. *CoRR*, abs/2202.05639, 2022. [12]

The paper proposed a new implementation of OCEL based on the MongoDB document database, which offers enhanced scalability compared to traditional JSON and XML implementations. Performance considerations were also discussed.

Demo Paper 3 (Tool for the Automatic Extraction of OCELs from SAP ERP)

Julian Weber, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. Interactive process identification and selection from SAP ERP (extended abstract). In Marwan Hassani, Agnes Koschmider, Marco Comuzzi, Fabrizio Maria Maggi, and Luise Pufahl, editors, *Proceedings of the ICPM Doctoral Consortium and Demo Track 2022 co-located with 4th International Conference on Process Mining (ICPM 2022), Bolzano, Italy, October, 2022*, volume 3299 of *CEUR Workshop Proceedings*, pages 61–64. CEUR-WS.org, 2022. [134]

The paper introduced a tool for the semi-automated extraction of OCELs from the SAP ECC ERP system. It is referenced in Chapter 5.

Demo Paper 4 (OCEL 2.0 Standard)

István Koren, Jan Niklas Adams, Alessandro Berti, and Wil M. P. van der Aalst. OCEL 2.0 resources - www.ocel-standard.org. In Jan Martijn E. M. van der Werf, Cristina Cabanillas, Francesco Leotta, and Laura Genga, editors, *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*, volume 3648 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. [67]

The paper introduces the OCEL 2.0 standard for the storage of object-centric event logs. It is referenced in Chapter 3 and Chapter 5.

Pre-Prints

Pre-Print 1 (Literature Review OCPM)

Alessandro Berti, Marco Montali, and Wil M. P. van der Aalst. Advancements and challenges in object-centric process mining: A systematic literature review. *CoRR*, abs/2311.08795, 2023. [17]

The paper introduces a systematic literature review on the topic of object-centric process mining. It is referenced in Chapter 4.

Declarations

This thesis is the culmination of my own work and all sources utilized, whether quoted directly or indirectly, have been acknowledged.

Sources

Throughout the research and writing process, I relied on numerous online sources to substantiate my arguments, validate my findings, and expand my understanding of complex topics. I have made every effort to credit these sources in the text and in the references section of this thesis.

Tools and Support

In the creation of this thesis, I also employed several digital tools that contributed significantly to my research and analysis.

Grammarly was used extensively for grammar checking and ensuring clarity of language.

Quetext was employed to check for potential plagiarism, ensuring the originality of the presented content.

OpenAI's GPT-4 was also used as an analytical assistant throughout the thesis. It played a vital role in the analysis of OCELs by assisting in different abstractions such as list of events/objects, OCDFGs, and machine learning features extracted from OCELs.

Despite the assistance received, I bear complete responsibility for the content of this thesis. I am entirely accountable for the information presented and the conclusions drawn. I affirm that this work is original and has not been submitted elsewhere for any other degree or qualification.

Acknowledgements

I would like to take a moment to express my sincere gratitude to all those who have played a vital part in the completion of this thesis.

First and foremost, I would like to thank my companion, Minoo. Your unwavering support and belief in me have been the pillars that held me upright during this challenging journey. You've been my source of inspiration and motivation, encouraging me to persevere even on the toughest days. Your constant love and care have truly made a world of difference, and for that, I am immensely grateful.

To my family - Dad Silvano, Mum Gabriella, and Grandma Giulietta - thank you for supporting me since the day I was born. Your love, guidance, and unwavering faith in me have been my beacon of hope and strength. You've taught me resilience, perseverance, and the importance of hard work. I am grateful for the values you've instilled in me and for the sacrifices you've made on my behalf. Without your consistent support and encouragement, this journey would have been considerably more challenging.

I extend my sincere gratitude to my friends, particularly Elia and Silvia, for their friendship, laughter, and shared moments. You've been an integral part of this journey, providing me with the respite I needed from the rigorous academic pursuits. You were there to celebrate my successes, help me navigate through difficult times, and offer your invaluable advice and support. Your camaraderie and companionship made this journey more enjoyable and less daunting.

To all my university mates, thank you for the shared experiences, lively discussions, and insightful perspectives. You've provided a stimulating and enriching environment for growth, pushing me to strive for excellence, explore new ideas, and challenge my assumptions.

I must make a special mention of Mahnaz, whose friendship and camaraderie have been irreplaceable. Mahnaz, thank you for being by my side, both in moments of joy and during the tougher phases of this journey. Your unwavering support, combined with your kindness, patience, and understanding, has been a source of strength and comfort. Your presence has been a reassuring constant, and for that, I am deeply thankful.

Sebastiaan J. van Zelst has not only been a colleague but also a mentor and guide in my professional journey. Co-founding the PM4Py process mining library and leading our team at Fraunhofer FIT, Sebastiaan has played a pivotal role in shaping my perspectives and skills. The time spent under his leadership has been transformative, teaching me invaluable lessons both in our field and about life. Thank you, Sebastiaan, for your trust, guidance, and for believing in my potential. Your influence on my personal and professional growth is immeasurable.

Finally, I want to express my gratitude to all my colleagues and superiors at work. Your belief in my capabilities, even when I was just starting out with no experience, has greatly contributed to my professional growth. Your guidance and support have not only boosted my confidence but also allowed me to further develop my skills and knowledge.

In sum, to everyone who has been a part of my journey, your influence, in various forms, has shaped me into the person I am today and the researcher I aspire to be. Your faith in me has powered my drive, and for this, I am eternally grateful. Thank you all.

And last, but most certainly not least, I wish to express my deepest gratitude to my supervisor, Prof. Dr. Wil van der Aalst. Under his mentorship, I've had the privilege of learning, growing, and expanding my horizons in ways I couldn't have imagined. His deep insights, expert guidance, and tireless dedication to my progress have been invaluable. I am grateful for his patience, his understanding, and his encouragement, even in the face of challenges. His wisdom and perspective have profoundly influenced my approach to research, continually inspiring me to strive for excellence and originality in my work.

I count myself incredibly fortunate to have been under his supervision, and for his unwavering belief in my potential. Thank you, Professor van der Aalst, for your mentorship, your support, and the faith you've shown in me. Your guidance has not only shaped this thesis, but my entire academic journey.

List of Figures

List of Tables

Bibliography

[1] Gustav Aagesen and John Krogstie. BPMN 2.0 for modeling business processes. In Jan vom Brocke and Michael Rosemann, editors, *Handbook on Business Process Management 1, Introduction, Methods, and Information Systems, 2nd Ed*, International Handbooks on Information Systems, pages 219–250. Springer, 2015.

[2] Milad Naeimaei Aali, Felix Mannhardt, and Pieter Jelle Toussaint. Discovering care pathways for multi-morbid patients using event graphs. In Jorge Munoz-Gama and Xixi Lu, editors, *Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers*, volume 433 of *Lecture Notes in Business Information Processing*, pages 352–364, New York City, 2021. Springer.

[3] Giovanni Acampora, Autilia Vitiello, Bruno N. Di Stefano, Wil M. P. van der Aalst, Christian W. Günther, and Eric Verbeek. 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.

[4] Jan Niklas Adams, Gyunam Park, Sergej Levich, Daniel Schuster, and Wil M. P. van der Aalst. A framework for extracting and encoding features from object-centric event data. In Javier Troya, Brahim Medjahed, Mario Piattini, Lina Yao, Pablo Fernández, and Antonio Ruiz-Cortés, editors, *Service-Oriented Computing - 20th International Conference, ICSOC 2022, Seville, Spain, November 29 - December 2, 2022, Proceedings*, volume 13740 of *Lecture Notes in Computer Science*, pages 36–53. Springer, 2022.

[5] Jan Niklas Adams, Gyunam Park, and Wil MP van der Aalst. ocpa: A python library for object-centric process analysis. *Software Impacts*, page 100438, 2022.

[6] Jan Niklas Adams, Daniel Schuster, Seth Schmitz, Günther Schuh, and Wil M. P. van der Aalst. Defining cases and variants for object-centric event data. In Andrea Burattin, Artem Polyvyanyy, and Barbara Weber, editors, *4th International Conference on Process Mining, ICPM 2022, Bolzano, Italy, October 23-28, 2022*, pages 128–135. IEEE, 2022.

[7] Jan Niklas Adams and Wil M. P. van der Aalst. Precision and fitness in object-centric process mining. In Claudio Di Ciccio, Chiara Di Francescomarino, and Pnina Soffer, editors, *3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021*, pages 128–135, New York City, 2021. IEEE.

[8] Jan Niklas Adams and Wil M. P. van der Aalst. Ocπ: Object-centric process insights. In Luca Bernardinello and Laure Petrucci, editors, *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*, volume 13288 of *Lecture Notes in Computer Science*, pages 139–150, New York City, 2022. Springer.

[9] Arya Adriansyah, Jorge Munoz-Gama, Josep Carmona, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Measuring precision of modeled behavior. *Inf. Syst. E Bus. Manag.*, 13(1):37–67, 2015.

[10] Arya Adriansyah, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance checking using cost-based fitness analysis. In *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011, Helsinki, Finland, August 29 - September 2, 2011*, pages 55–64. IEEE Computer Society, 2011.

[11] Alessandro Berti. Process mining on event graphs: a framework to extensively support projects. In Benoît Depaire, Johannes De Smedt, Marlon Dumas, Dirk Fahland, Akhil Kumar, Henrik Leopold, Manfred Reichert, Stefanie Rinderle-Ma, Stefan Schulte, Stefan Seidel, and Wil M. P. van der Aalst, editors, *Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 co-located with 17th International Conference on Business Process Management, BPM 2019, Vienna, Austria, September 1-6, 2019*, volume 2420 of *CEUR Workshop Proceedings*, pages 60–65, Aachen, 2019. CEUR-WS.org.

[12] Alessandro Berti, Anahita Farhang Ghahfarokhi, Gyunam Park, and Wil M. P. van der Aalst. A scalable database for the storage of object-centric event logs. *CoRR*, abs/2202.05639, 2022.

[13] Alessandro Berti, Johannes Herforth, Mahnaz Sadat Qafari, and Wil M. P. van der Aalst. Graph-based feature extraction on object-centric event logs. *International Journal of Data Science and Analytics*, 2023.

[14] Alessandro Berti, Urszula Jessen, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. Analyzing interconnected processes: using object-centric process mining to analyze procurement processes. *International Journal of Data Science and Analytics*, 2023.

[15] Alessandro Berti, Urszula Jessen, Wil M. P. van der Aalst, and Dirk Fahland. Explainable object-centric anomaly detection: the role of domain knowledge. In Adela del-Río-Ortega, Marco Montali, Stefanie Rinderle-Ma, Hajo A. Reijers, Jan vom Brocke, Mathias Weske, Benoît Depaire, Marta Indulska, Han van der Aa, Weronika T. Adrian, Laura Genga, Sander J. J. Leemans, Katarzyna Gdowska, María Teresa Gómez-López, Jana-Rebecca Rehse, and Simone Agostinelli, editors, *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Forum at BPM 2024 co-located with 22nd International Conference on Business Process Management (BPM 2024), Krakow, Poland, September 1st to 6th, 2024*, volume 3758 of *CEUR Workshop Proceedings*, pages 162–168. CEUR-WS.org, 2024.

[16] Alessandro Berti, Istvan Koren, Jan Niklas Adams, Gyunam Park, Benedikt Knopp, Nina Graves, Majid Rafiei, Lukas Liß, Leah Tacke Genannt Unterberg, Yisong Zhang, Christopher Schwanen, Marco Pegoraro, and Wil M. P. van der Aalst. Ocel (object-centric event log) 2.0 specification, 2024.

[17] Alessandro Berti, Marco Montali, and Wil M. P. van der Aalst. Advancements and challenges in object-centric process mining: A systematic literature review. *CoRR*, abs/2311.08795, 2023.

[18] Alessandro Berti, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. An event data extraction approach from SAP ERP for process mining. In Jorge Munoz-Gama and Xixi Lu, editors, *Process Mining Workshops - ICPM 2021 International Workshops, Eindhoven, The Netherlands, October 31 - November 4, 2021, Revised Selected Papers*, volume 433 of *Lecture Notes in Business Information Processing*, pages 255–267. Springer, 2021.

[19] Alessandro Berti, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. A generic approach to extract object-centric event data from databases supporting sap erp. *Journal of Intelligent Information Systems*, 2023.

[20] Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Abstractions, scenarios, and prompt definitions for process mining with llms: A case study. In *Proceedings of the NLP4BPM 2023 Workshop*, 2023.

[21] Alessandro Berti and Wil M. P. van der Aalst. Starstar models: Using events at database level for process analysis. In Paolo Ceravolo, María Teresa Gómez López, and Maurice van Keulen, editors, *Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain, December 13-14, 2018*, volume 2270 of *CEUR Workshop Proceedings*, pages 60–64. CEUR-WS.org, 2018.

[22] Alessandro Berti and Wil M. P. van der Aalst. Extracting multiple viewpoint models from relational databases. In Paolo Ceravolo, Maurice van Keulen, and María Teresa Gómez López, editors, *Data-Driven Process Discovery and Analysis - 8th IFIP WG 2.6 International Symposium, SIMPDA 2018, Seville, Spain, December 13-14, 2018, and 9th International Symposium, SIMPDA 2019, Bled, Slovenia, September 8, 2019, Revised Selected Papers*, volume 379 of *Lecture Notes in Business Information Processing*, pages 24–51. Springer, 2019.

[23] Alessandro Berti and Wil M. P. van der Aalst. Reviving token-based replay: Increasing speed while improving diagnostics. In Wil M. P. van der Aalst, Robin Bergenthum, and Josep Carmona, editors, *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2019 Satellite event of the conferences: 40th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2019 and 19th International Conference on Application of Concurrency to System Design ACSD 2019, ATAED@Petri Nets/ACSD 2019, Aachen, Germany, June 25, 2019*, volume 2371 of *CEUR Workshop Proceedings*, pages 87–103. CEUR-WS.org, 2019.

[24] Alessandro Berti and Wil M. P. van der Aalst. A novel token-based replay technique to speed up conformance checking and process enhancement. *Trans. Petri Nets Other Model. Concurr.*, 15:1–26, 2021.

[25] Alessandro Berti and Wil M. P. van der Aalst. OC-PM: analyzing object-centric event logs and process models. *Int. J. Softw. Tools Technol. Transf.*, 25(1):1–17, 2023.

[26] Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. Pm4py: A process mining library for python. *Software Impacts*, 17:100556, 2023.

[27] Alessandro Berti, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Process mining for python (pm4py): Bridging the gap between process- and data science. In *Proceedings of the 1st International Conference on Process Mining (ICPM 2019), Demo Track*. CEUR, 2019.

[28] Yannis Bertrand, Jochen De Weerdt, and Estefanía Serral. Assessing the suitability of traditional event log standards for iot-enhanced event logs. In Cristina Cabanillas, Niels Frederik Garmann-Johnsen, and Agnes Koschmider, editors, *Business Process Management Workshops - BPM 2022 International Workshops, Münster, Germany, September 11-16, 2022, Revised Selected Papers*, volume 460 of *Lecture Notes in Business Information Processing*, pages 63–75. Springer, 2022.

[29] Mathilde Boltenhagen, Thomas Chatain, and Josep Carmona. Optimized SAT encoding of conformance checking artefacts. *Computing*, 103(1):29–50, 2021.

[30] Marius Breitmayer, Lisa Arnold, and Manfred Reichert. Enabling conformance checking for object lifecycle processes. In Renata S. S. Guizzardi, Jolita Ralyté, and Xavier Franch, editors, *Research Challenges in Information Science - 16th International Conference, RCIS 2022, Barcelona, Spain, May 17-20, 2022, Proceedings*, volume 446 of *Lecture Notes in Business Information Processing*, pages 124–141, New York City, 2022. Springer.

[31] Marius Breitmayer and Manfred Reichert. Towards the discovery of object-aware processes. In Johannes Manner, Stephan Haarmann, Stefan Kolb, and Oliver Kopp, editors, *Proceedings of the 12th ZEUS Workshop on Services and their Composition, Potsdam, Germany, February 20-21, 2020*, volume 2575 of *CEUR Workshop Proceedings*, pages 1–4, Aachen, 2020. CEUR-WS.org.

[32] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014.

[33] Andrea Burattin, Alessandro Sperduti, and Marco Veluscek. Business models enhancement through discovery of roles. In *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013*, pages 103–110. IEEE, 2013.

[34] Diego Calvanese, Marco Montali, Montserrat Estañol, and Ernest Teniente. Verifiable UML artifact-centric business process models. In Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang, editors, *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1289–1298. ACM, 2014.

[35] Thomas Chatain, Mathilde Boltenhagen, and Josep Carmona. Anti-alignments - measuring the precision of process models and event logs. *Inf. Syst.*, 98:101708, 2021.

[36] Jan Claes and Geert Poels. Merging event logs for process mining: A rule based merging method and rule suggestion algorithm. *Expert Syst. Appl.*, 41(16):7291–7306, 2014.

[37] Priyam Das and Dhananjay R. Kalbande. Behavioural analysis of multi-source social network data using object-centric behavioural constraints and data mining technique. In *11th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2020, Kharagpur, India, July 1-3, 2020*, pages 1–8, New York City, 2020. IEEE.

[38] Massimiliano de Leoni, Wil M. P. van der Aalst, and Marcus Dees. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Inf. Syst.*, 56:235–257, 2016.

[39] Eduardo González López de Murillas, Hajo A. Reijers, and Wil M. P. van der Aalst. Case notion discovery and recommendation: automated event log building on databases. *Knowl. Inf. Syst.*, 62(7):2539–2575, 2020.

[40] Eduardo González López de Murillas, Wil M. P. van der Aalst, and Hajo A. Reijers. Process mining on databases: Unearthing historical data from redo logs. In Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich, editors, *Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings*, volume 9253 of *Lecture Notes in Computer Science*, pages 367–385, New York City, 2015. Springer.

[41] Vadim Denisov, Elena Belkina, Dirk Fahland, and Wil M. P. van der Aalst. The performance spectrum miner: Visual analytics for fine-grained performance analysis of processes. In Wil M. P. van der Aalst, Fabio Casati, Raffaele Conforti, Massimiliano de Leoni, Marlon Dumas, Akhil Kumar, Jan Mendling, Surya Nepal, Brian T. Pentland, and Barbara Weber, editors, *Proceedings of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018*, volume 2196 of *CEUR Workshop Proceedings*, pages 96–100. CEUR-WS.org, 2018.

[42] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Inf. Softw. Technol.*, 50(12):1281–1294, 2008.

[43] Ali Nour Eldin, Nour Assy, Meriana Kobeissi, Jonathan Baudot, and Walid Gaaloul. Enabling multi-process discovery on graph databases. In Mohamed Sellami, Paolo Ceravolo, Hajo A. Reijers, Walid Gaaloul, and Hervé Panetto, editors, *Cooperative Information Systems - 28th International Conference, CoopIS 2022, Bozen-Bolzano, Italy, October 4-7, 2022, Proceedings*, volume 13591 of *Lecture Notes in Computer Science*, pages 112–130, New York City, 2022. Springer.

[44] Stefan Esser and Dirk Fahland. Storing and querying multi-dimensional process event logs using graph databases. In Chiara Di Francescomarino, Remco M. Dijkman, and Uwe Zdun, editors, *Business Process Management Workshops - BPM 2019 International Workshops, Vienna, Austria, September 1-6, 2019, Revised Selected Papers*, volume 362 of *Lecture Notes in Business Information Processing*, pages 632–644, New York City, 2019. Springer.

[45] Stefan Esser and Dirk Fahland. Multi-dimensional event data in graph databases. *J. Data Semant.*, 10(1-2):109–141, 2021.

[46] Montserrat Estañol, Jorge Munoz-Gama, Josep Carmona, and Ernest Teniente. Conformance checking in UML artifact-centric business process models. *Softw. Syst. Model.*, 18(4):2531–2555, 2019.

[47] Dirk Fahland. Process mining over multiple behavioral dimensions with event knowledge graphs. In Wil M. P. van der Aalst and Josep Carmona, editors, *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, pages 274–319. Springer, New York City, 2022.

[48] Dirk Fahland, Massimiliano de Leoni, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Behavioral conformance of artifact-centric process models. In Witold Abramowicz, editor, *Business Information Systems - 14th International Conference, BIS 2011, Poznan, Poland, June 15-17, 2011. Proceedings*, volume 87 of *Lecture Notes in Business Information Processing*, pages 37–49, New York City, 2011. Springer.

[49] Dirk Fahland, Massimiliano de Leoni, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Conformance checking of interacting processes with overlapping instances. In Stefanie Rinderle-Ma, Farouk Toumani, and Karsten Wolf, editors, *Business Process Management - 9th International*

Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings, volume 6896 of *Lecture Notes in Computer Science*, pages 345–361, New York City, 2011. Springer.

[50] Stephan A. Fahrenkrog-Petersen, Han van der Aa, and Matthias Weidlich. PRIPEL: privacy-preserving event log publishing including contextual information. In Dirk Fahland, Chiara Ghidini, Jörg Becker, and Marlon Dumas, editors, *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 12168 of *Lecture Notes in Computer Science*, pages 111–128. Springer, 2020.

[51] Elio Ribeiro Faria Junior, Thais Rodrigues Neubauer, Marcelo Fantinato, and Sarajane Marques Peres. Clustering analysis and frequent pattern mining for process profile analysis: An exploratory study for object-centric event logs. In *International Conference on Process Mining*, pages 269–281. Springer, 2022.

[52] Riccardo Galanti, Massimiliano de Leoni, Nicolò Navarin, and Alan Marazzi. Object-centric process predictive analytics. *Expert Syst. Appl.*, 213(Part):119173, 2023.

[53] Anahita Farhang Ghahfarokhi, Gyunam Park, Alessandro Berti, and Wil M. P. van der Aalst. OCEL: A standard for object-centric event logs. In Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, Raimundas Matulevicius, Ahmed Awad, Matthias Weidlich, Mirjana Ivanovic, and Olaf Hartig, editors, *New Trends in Database and Information Systems - ADBIS 2021 Short Papers, Doctoral Consortium and Workshops: DOING, SIMPDA, MADEISD, MegaData, CAoNS, Tartu, Estonia, August 24-26, 2021, Proceedings*, volume 1450 of *Communications in Computer and Information Science*, pages 169–175. Springer, 2021.

[54] Anahita Farhang Ghahfarokhi and Wil M. P. van der Aalst. A python tool for object-centric process mining comparison. *CoRR*, abs/2202.05709, 2022.

[55] Wissam Gherissi, Joyce El Haddad, and Daniela Grigori. Object-centric predictive process monitoring. In Javier Troya, Raffaela Mirandola, Elena Navarro, Andrea Delgado, Sergio Segura, Guadalupe Ortiz, Cesare Pautasso, Christian Zirpins, Pablo Fernández, and Antonio Ruiz-Cortés, editors, *Service-Oriented Computing - ICSOC 2022 Workshops - ASOCA, AI-PA, FMCIoT, WESOACS 2022, Sevilla, Spain, November 29 - December 2, 2022 Proceedings*, volume 13821 of *Lecture Notes in Computer Science*, pages 27–39. Springer, 2022.

[56] Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin. Petri net-based object-centric processes with read-only data. *Inf. Syst.*, 107:102011, 2022.

[57] Alexandre Goossens, Johannes De Smedt, Jan Vanthienen, and Wil M. P. van der Aalst. Enhancing data-awareness of object-centric event logs. In Marco Montali, Arik Senderovich, and Matthias Weidlich, editors, *Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers*, volume 468 of *Lecture Notes in Business Information Processing*, pages 18–30. Springer, 2022.

[58] Alexander Guz and Jeremiah Rushchitsky. Scopus: A system for the evaluation of scientific journals. *International Applied Mechanics*, 45:351–362, 2009.

[59] Jaciel David Hernandez-Resendiz, Edgar Tello-Leal, Heidy Marisol Marin-Castro, Ulises Manuel Ramirez-Alcocer, and Jonathan Alfonso Mata-Torres. Merging event logs for inter-organizational process mining. In *New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques*, pages 3–26. Springer, New York City, 2021.

[60] Richard Hull. Artifact-centric business process models: Brief survey of research results and challenges. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part II*, volume 5332 of *Lecture Notes in Computer Science*, pages 1152–1163. Springer, 2008.

[61] Richard Hull, Elio Damaggio, Fabiana Fournier, Manmohan Gupta, Fenno F. Terry Heath III, Stacy Hobson, Mark H. Linehan, Sridhar Maradugu, Anil Nigam, Piyawadee Sukaviriya, and Roman Vaculín. Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In Mario Bravetti and Tevfik Bultan, editors, *Web Services and Formal Methods - 7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16-17, 2010. Revised Selected Papers*, volume 6551 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2010.

[62] Amin Jalali. Object type clustering using markov directly-follow multigraph in object-centric process mining. *IEEE Access*, 10:126569–126579, 2022.

[63] Toon Jouck and Benoît Depaire. Ptandloggenerator: A generator for artificial event data. In Leonardo Azevedo and Cristina Cabanillas, editors, *Proceedings of the BPM Demo Track 2016 Co-located with the 14th International Conference on Business Process Management (BPM 2016), Rio de Janeiro, Brazil, September 21, 2016*, volume 1789 of *CEUR Workshop Proceedings*, pages 23–27. CEUR-WS.org, 2016.

[64] Anna A. Kalenkova, Andrea Burattin, Massimiliano de Leoni, Wil M. P. van der Aalst, and Alessandro Sperduti. Discovering high-level BPMN process models from event data. *Bus. Process. Manag. J.*, 25(5):995–1019, 2019.

[65] Anna A. Kalenkova, Wil M. P. van der Aalst, Irina A. Lomazova, and Vladimir A. Rubin. Process mining using BPMN: relating event logs and process models. *Softw. Syst. Model.*, 16(4):1019–1048, 2017.

[66] Imran Khan, Joshua Zhexue Huang, Nguyen Thanh Tung, and Graham Williams. Ensemble clustering of high dimensional data with fastmap projection. In Wen-Chih Peng, Haixun Wang, James Bailey, Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, and Arbee L.P. Chen, editors, *Trends and Applications in Knowledge Discovery and Data Mining*, pages 483–493, Cham, 2014. Springer International Publishing.

[67] István Koren, Jan Niklas Adams, Alessandro Berti, and Wil M. P. van der Aalst. OCEL 2.0 resources - www.ocel-standard.org. In Jan Martijn E. M. van der Werf, Cristina Cabanillas, Francesco Leotta, and Laura Genga, editors, *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*, volume 3648 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[68] Akhil Kumar, Pnina Soffer, and Arava Tsoury. Normalizing object-centric process logs by applying database principles. *Inf. Syst.*, 115:102196, 2023.

[69] Vera Künzle and Manfred Reichert. Philharmonicflows: towards a framework for object-aware process management. *J. Softw. Maintenance Res. Pract.*, 23(4):205–244, 2011.

[70] Wai Lam Jonathan Lee, H. M. W. Verbeek, Jorge Munoz-Gama, Wil M. P. van der Aalst, and Marcos Sepúlveda. Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. *Inf. Sci.*, 466:55–91, 2018.

[71] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In José Manuel Colom and Jörg Desel, editors, *Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2013.

[72] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In Niels Lohmann, Minseok Song, and Petia Wohed, editors, *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer, 2013.

[73] Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Scalable process discovery and conformance checking. *Softw. Syst. Model.*, 17(2):599–631, 2018.

[74] Guangming Li and Renata Medeiros de Carvalho. Dealing with artifact-centric systems: a process mining approach. *EMISA Forum*, 38(1):61–65, 2018.

[75] Guangming Li and Renata Medeiros de Carvalho. Process mining in social media: Applying object-centric behavioral constraint models. *IEEE Access*, 7:84360–84373, 2019.

[76] Guangming Li, Renata Medeiros de Carvalho, and Wil M. P. van der Aalst. Automatic discovery of object-centric behavioral constraint models. In Witold Abramowicz, editor, *Business Information Systems - 20th International Conference, BIS 2017, Poznan, Poland, June 28-30, 2017, Proceedings*, volume 288 of *Lecture Notes in Business Information Processing*, pages 43–58, New York City, 2017. Springer.

[77] Guangming Li, Renata Medeiros de Carvalho, and Wil M. P. van der Aalst. Configurable event correlation for process discovery from object-centric event data. In *2018 IEEE International Conference on Web Services, ICWS 2018, San Francisco, CA, USA, July 2-7, 2018*, pages 203–210, New York City, 2018. IEEE.

[78] Guangming Li, Renata Medeiros de Carvalho, and Wil M. P. van der Aalst. Object-centric behavioral constraint models: a hybrid model for behavioral and data perspectives. In Chih-Cheng Hung and George A. Papadopoulos, editors, *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 48–56. ACM, 2019.

[79] 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 Jan Mendling and Haralambos Mouratidis, editors, *Information Systems in the Big Data Era - CAiSE Forum 2018, Tallinn, Estonia, June 11-15, 2018, Proceedings*, volume 317 of *Lecture Notes in Business Information Processing*, pages 182–199, New York City, 2018. Springer.

[80] Xixi Lu, Marijn Nagelkerke, Dennis van de Wiel, and Dirk Fahland. Discovering interacting artifacts from ERP systems. *IEEE Trans. Serv. Comput.*, 8(6):861–873, 2015.

[81] Giovanni Lugaresi and Andrea Matta. Discovery and digital model generation for manufacturing systems with assembly operations. In *17th IEEE International Conference on Automation Science and Engineering, CASE 2021, Lyon, France, August 23-27, 2021*, pages 752–757, New York City, 2021. IEEE.

[82] R. S. Mans, Nick C. Russell, Wil M. P. van der Aalst, Piet J. M. Bakker, Arnold J. Moleman, and Monique W. M. Jaspers. Proclets in healthcare. *J. Biomed. Informatics*, 43(4):632–649, 2010.

[83] Niels Martin, Marijke Swennen, Benoît Depaire, Mieke Jans, An Caris, and Koen Vanhoof. Batch processing: Definition and event log identification. In Paolo Ceravolo and Stefanie Rinderle-Ma, editors, *Proceedings of the 5th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2015), Vienna, Austria, December 9-11, 2015*, volume 1527 of *CEUR Workshop Proceedings*, pages 137–140. CEUR-WS.org, 2015.

[84] Leyla Moctar Mbaba, Nour Assy, Mohamed Sellami, Walid Gaaloul, and Mohamedade Farouk Nanne. Process mining for artifact-centric blockchain applications. *Simulation Modelling Practice and Theory*, 127:102779, 2023.

[85] Janna Meyer and Josua Reimold. Associative intelligence for object-centric process mining with mpm (extended abstract) 1. 2021.

[86] Leyla Moctar-M'Baba, Nour Assy, Mohamed Sellami, Walid Gaaloul, and Mohamedade Farouk Nanne. Extracting artifact-centric event logs from blockchain applications. In Claudio Agostino Ardagna, Hongyi Bian, Carl K. Chang, Rong N. Chang, Ernesto Damiani, Schahram Dustdar, Jordi Marco, Munindar P. Singh, Ernest Teniente, Robert Ward, Zhongjie Wang, Fatos Xhafa, and Jia Zhang, editors, *IEEE International Conference on Services Computing, SCC 2022, Barcelona, Spain, July 10-16, 2022*, pages 274–283, New York City, 2022. IEEE.

[87] Jorge Munoz-Gama and Josep Carmona. A fresh look at precision in process conformance. In Richard Hull, Jan Mendling, and Stefan Tai, editors, *Business Process Management - 8th International Conference, BPM 2010, Hoboken, NJ, USA, September 13-16, 2010. Proceedings*, volume 6336 of *Lecture Notes in Computer Science*, pages 211–226. Springer, 2010.

[88] Krzysztof Najman and Krystian Zieliński. Outlier detection with the use of isolation forests. In Krzysztof Jajuga, Krzysztof Najman, and Marek Walesiak, editors, *Data Analysis and Classification*, pages 65–79, Cham, 2021. Springer International Publishing.

[89] Minh Khoi Nguyen, Hanh Nhi Tran, and Ileana Ober. Process mining to discover the global process from its fragments' executions. In Hermann Kaindl, Mike Mannion, and Leszek A. Maciaszek, editors, *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2022, Online Streaming, April 25-26, 2022*, pages 363–370, Setubal, 2022. SCITEPRESS.

[90] Anil Nigam and Nathan S. Caswell. Business artifacts: An approach to operational specification. *IBM Syst. J.*, 42(3):428–445, 2003.

[91] Erik H. J. Nooijen, Boudewijn F. van Dongen, and Dirk Fahland. Automatic discovery of data-centric and artifact-centric processes. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business Information Processing*, pages 316–327, New York City, 2012. Springer.

[92] Chun Ouyang, Marlon Dumas, Arthur H. M. ter Hofstede, and Wil M. P. van der Aalst. From BPMN process models to BPEL web services. In *2006 IEEE International Conference on Web Services (ICWS 2006), 18-22 September 2006, Chicago, Illinois, USA*, pages 285–292. IEEE Computer Society, 2006.

[93] Ana Pajic, Sladjan Babarogic, and Ognjen Pantelic. A domain-specific language for supporting event log extraction from erp systems. In *2018 7th International Conference on Computers Communications and Control (ICCCC)*, pages 12–16, New York City, 2018. IEEE.

[94] Ana Pajic, Sladjan Babarogic, Ognjen Pantelic, and Stefan Krstovic. Towards a domain-specific modeling language for extracting event logs from erp systems. *Applied Sciences*, 11(12):5476, 2021.

[95] Ana Pajic and Dragana Becejski-Vujaklija. Metamodel of the artifact-centric approach to event log extraction from ERP systems. *Int. J. Decis. Support Syst. Technol.*, 8(2):18–28, 2016.

[96] Gyunam Park, Jan Niklas Adams, and Wil M. P. van der Aalst. Opera: Object-centric performance analysis. In Jolita Ralyté, Sharma Chakravarthy, Mukesh Mohania, Manfred A. Jeusfeld, and Kamalakar Karlapalem, editors, *Conceptual Modeling - 41st International Conference, ER 2022, Hyderabad, India, October 17-20, 2022, Proceedings*, volume 13607 of *Lecture Notes in Computer Science*, pages 281–292, New York City, 2022. Springer.

[97] Gyunam Park, Jan Niklas Adams, and Wil M. P. van der Aalst. Conformance checking and performance analysis using object-centric directly-follows graphs. In Andrea Marrella, Manuel Resinas, Mieke Jans, and Michael Rosemann, editors, *Business Process Management Forum - BPM 2024 Forum, Krakow, Poland, September 1-6, 2024, Proceedings*, volume 526 of *Lecture Notes in Business Information Processing*, pages 179–196. Springer, 2024.

[98] Gyunam Park and Wil M. P. van der Aalst. Monitoring constraints in business processes using object-centric constraint graphs. In Marco Montali, Arik Senderovich, and Matthias Weidlich, editors, *Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers*, volume 468 of *Lecture Notes in Business Information Processing*, pages 479–492. Springer, 2022.

[99] Carl Adam Petri. *Kommunikationen mit automaten*. PhD thesis, PhD Thesis, University of Bonn, 1962. English translation: Technical Report ..., 1962.

[100] Anastasiia Pika, Michael Leyer, Moe Thandar Wynn, Colin J. Fidge, Arthur H. M. ter Hofstede, and Wil M. P. van der Aalst. Mining resource profiles from event logs. *ACM Trans. Manag. Inf. Syst.*, 8(1):1:1–1:30, 2017.

[101] Viara Popova and Marlon Dumas. From petri nets to guard-stage-milestone models. In Marcello La Rosa and Pnina Soffer, editors, *Business Process Management Workshops - BPM 2012 International Workshops, Tallinn, Estonia, September 3, 2012. Revised Papers*, volume 132 of *Lecture Notes in Business Information Processing*, pages 340–351, New York City, 2012. Springer.

[102] Viara Popova and Marlon Dumas. Discovering unbounded synchronization conditions in artifact-centric process models. In Niels Lohmann, Minseok Song, and Petia Wohed, editors, *Business Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers*, volume 171 of *Lecture Notes in Business Information Processing*, pages 28–40, New York City, 2013. Springer.

[103] Viara Popova, Dirk Fahland, and Marlon Dumas. Artifact lifecycle discovery. *Int. J. Cooperative Inf. Syst.*, 24(1):1550001:1–1550001:44, 2015.

[104] Mahsa Pourbafrani, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Supporting automatic system dynamics model generation for simulation in the context of process mining. In Witold Abramowicz and Gary Klein, editors, *Business Information Systems - 23rd International Conference, BIS 2020, Colorado Springs, CO, USA, June 8-10, 2020, Proceedings*, volume 389 of *Lecture Notes in Business Information Processing*, pages 249–263. Springer, 2020.

[105] Shaya Pourmirza, Remco M. Dijkman, and Paul Grefen. Correlation miner: Mining business process models and event correlations without case identifiers. *Int. J. Cooperative Inf. Syst.*, 26(2):1742002:1–1742002:32, 2017.

[106] Lihi Raichelson and Pnina Soffer. Merging event logs with many to many relationships. In Fabiana Fournier and Jan Mendling, editors, *Business Process Management Workshops - BPM 2014 International Workshops, Eindhoven, The Netherlands, September 7-8, 2014, Revised Papers*, volume 202 of *Lecture Notes in Business Information Processing*, pages 330–341, New York City, 2014. Springer.

[107] Lihi Raichelson, Pnina Soffer, and Eric Verbeek. Merging event logs: Combining granularity levels for process flow analysis. *Inf. Syst.*, 71:211–227, 2017.

[108] Adrian Rebmann, Jana-Rebecca Rehse, and Han van der Aa. Uncovering object-centric data in classical event logs for the automated transformation from XES to OCEL. In Claudio Di Ciccio, Remco M. Dijkman, Adela del-Río-Ortega, and Stefanie Rinderle-Ma, editors, *Business Process Management - 20th International Conference, BPM 2022, Münster, Germany, September 11-16, 2022, Proceedings*, volume 13420 of *Lecture Notes in Computer Science*, pages 379–396, New York City, 2022. Springer.

[109] Anne Rozinat and Wil M. P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In Christoph Bussler and Armin Haller, editors, *Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005, Revised Selected Papers*, volume 3812, pages 163–176, 2005.

[110] G Schuh, A Gützlaff, S Schmitz, C Kuhn, and N Klapper. A methodology to apply process mining in end-to-end order processing of manufacturing companies. In *Recent Advances in Manufacturing Engineering and Processes*, pages 127–137. Springer, New York City, 2022.

[111] Daniel Schuster, Sebastiaan J. van Zelst, and Wil M. P. van der Aalst. Alignment approximation for process trees. In Sander J. J. Leemans and Henrik Leopold, editors, *Process Mining Workshops - ICPM 2020 International Workshops, Padua, Italy, October 5-8, 2020, Revised Selected Papers*, volume 406 of *Lecture Notes in Business Information Processing*, pages 247–259. Springer, 2020.

[112] Sebastian Steinau, Kevin Andrews, and Manfred Reichert. A modeling tool for philharmonicflows objects and lifecycle processes. In Robert Clarisó, Henrik Leopold, Jan Mendling, Wil M. P. van der Aalst, Akhil Kumar, Brian T. Pentland, and Mathias Weske, editors, *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*, volume 1920 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[113] Florian Stertz, Juergen Mangler, and Stefanie Rinderle-Ma. Temporal conformance checking at runtime based on time-infused process models. *CoRR*, abs/2008.07262, 2020.

[114] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016.

[115] Wil M. P. van der Aalst. Object-centric process mining: Dealing with divergence and convergence in event data. In Peter Csaba Ölveczky and Gwen Salaün, editors, *Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings*, volume 11724 of *Lecture Notes in Computer Science*, pages 3–25. Springer, 2019.

[116] Wil M. P. van der Aalst. Federated process mining: Exploiting event data across organizational boundaries. In Nimanthi L. Atukorala, Carl K. Chang, Ernesto Damiani, Min Fu, George Spanoudakis, Mudhakar Srivatsa, Zhongjie Wang, and Jia Zhang, editors, *IEEE International Conference on Smart Data Services, SMDS 2021, Chicago, IL, USA, September 5-10, 2021*, pages 1–7, New York City, 2021. IEEE.

[117] Wil M. P. van der Aalst, Paulo Barthelmess, Clarence A. Ellis, and Jacques Wainer. Workflow modeling using proclets. In Opher Etzion and Peter Scheuermann, editors, *Cooperative Information Systems, 7th International Conference, CoopIS 2000, Eilat, Israel, September 6-8, 2000, Proceedings*, volume 1901 of *Lecture Notes in Computer Science*, pages 198–209. Springer, 2000.

[118] Wil M. P. van der Aalst, Paulo Barthelmess, Clarence A. Ellis, and Jacques Wainer. Proclets: A framework for lightweight interacting workflow processes. *Int. J. Cooperative Inf. Syst.*, 10(4):443–481, 2001.

[119] Wil M. P. van der Aalst and Alessandro Berti. Discovering object-centric petri nets. *Fundam. Informaticae*, 175(1-4):1–40, 2020.

[120] Wil M. P. van der Aalst, H. T. de Beer, and Boudewijn F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In Robert Meersman, Zahir Tari, Mohand-Said Hacid, John Mylopoulos, Barbara Pernici, Özalp Babaoglu, Hans-Arno Jacobsen, Joseph P. Loyall, Michael Kifer, and Stefano Spaccapietra, editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE, OTM Confederated International Conferences CoopIS, DOA, and ODBASE 2005, Agia Napa, Cyprus, October 31 - November 4, 2005, Proceedings, Part I*, volume 3760 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2005.

[121] Wil M. P. van der Aalst, Hajo A. Reijers, and Minseok Song. Discovering social networks from event logs. *Comput. Support. Cooperative Work.*, 14(6):549–593, 2005.

[122] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.*, 16(9):1128–1142, 2004.

[123] Jan Martijn E. M. van der Werf, Andrey Rivkin, Artem Polyvyanyy, and Marco Montali. Data and process resonance - identifier soundness for models of information systems. In Luca Bernardinello and Laure Petrucci, editors, *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*, volume 13288 of *Lecture Notes in Computer Science*, pages 369–392, New York City, 2022. Springer.

[124] Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, and L. Wen. Process mining: Overview and outlook of petri net discovery algorithms. *Trans. Petri Nets Other Model. Concurr.*, 2:225–242, 2009.

[125] Boudewijn F. van Dongen and Wil M. P. van der Aalst. A meta model for process mining data. In Michele Missikoff and Antonio De Nicola, editors, *EMOI - INTEROP'05, Enterprise Modelling and Ontologies for Interoperability, Proceedings of the Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability, Co-located with CAiSE'05 Conference, Porto (Portugal), 13th-14th June 2005*, volume 160 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.

[126] Maikel L. van Eck, Natalia Sidorova, and Wil M. P. van der Aalst. Multi-instance mining: Discovering synchronisation in artifact-centric processes. In Florian Daniel, Quan Z. Sheng, and Hamid Motahari, editors, *Business Process Management Workshops - BPM 2018 International Workshops, Sydney, NSW, Australia, September 9-14, 2018, Revised Papers*, volume 342 of *Lecture Notes in Business Information Processing*, pages 18–30, New York City, 2018. Springer.

[127] Maikel L. van Eck, Natalia Sidorova, and Wil M. P. van der Aalst. Guided interaction exploration and performance analysis in artifact-centric process models. *Bus. Inf. Syst. Eng.*, 61(6):649–663, 2019.

[128] Sebastiaan J. van Zelst and Yukun Cao. A generic framework for attribute-driven hierarchical trace clustering. In Adela del-Río-Ortega, Henrik Leopold, and Flávia Maria Santoro, editors, *Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers*, volume 397 of *Lecture Notes in Business Information Processing*, pages 308–320. Springer, 2020.

[129] Sebastiaan J. van Zelst and Sander J. J. Leemans. Translating workflow nets to process trees: An algorithmic approach. *Algorithms*, 13(11):279, 2020.

[130] Borja Vázquez-Barreiros, Manuel Mucientes, and Manuel Lama. Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Inf. Sci.*, 294:315–333, 2015.

[131] H. M. W. Verbeek. The log skeleton visualizer in prom 6.9. *Int. J. Softw. Tools Technol. Transf.*, 24(4):549–561, 2022.

[132] H. M. W. Verbeek, Twan Basten, and Wil M. P. van der Aalst. Diagnosing workflow processes using woflan. *Comput. J.*, 44(4):246–279, 2001.

[133] Lotte Vugs, Maarten van Asseldonk, and Niek van Son. Konekti: A data preparation platform for process mining (extended abstract). In Marwan Hassani, Agnes Koschmider, Marco Comuzzi, Fabrizio Maria Maggi, and Luise Pufahl, editors, *Proceedings of the ICPM Doctoral Consortium and Demo Track 2022 co-located with 4th International Conference on Process Mining (ICPM 2022), Bolzano, Italy, October, 2022*, volume 3299 of *CEUR Workshop Proceedings*, pages 104–107. CEUR-WS.org, 2022.

[134] Julian Weber, Gyunam Park, Majid Rafiei, and Wil M. P. van der Aalst. Interactive process identification and selection from SAP ERP (extended abstract). In Marwan Hassani, Agnes Koschmider, Marco Comuzzi, Fabrizio Maria Maggi, and Luise Pufahl, editors, *Proceedings of the ICPM Doctoral Consortium and Demo Track 2022 co-located with 4th International Conference on Process Mining (ICPM 2022), Bolzano, Italy, October, 2022*, volume 3299 of *CEUR Workshop Proceedings*, pages 61–64. CEUR-WS.org, 2022.

[135] A. J. M. M. Weijters and Joel Tiago S. Ribeiro. Flexible Heuristics Miner (FHM). In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France*, pages 310–317. IEEE, 2011.

[136] Jing Xiong, Guohui Xiao, Tahir Emre Kalayci, Marco Montali, Zhenzhen Gu, and Diego Calvanese. Extraction of object-centric event logs through virtual knowledge graphs (extended abstract). In Ofer Arieli, Martin Homola, Jean Christoph Jung, and Marie-Laure Mugnier, editors, *Proceedings of the 35th International Workshop on Description Logics (DL 2022) co-located with Federated Logic Conference (FLoC 2022), Haifa, Israel, August 7th to 10th, 2022*, volume 3263 of *CEUR Workshop Proceedings*, Aachen, 2022. CEUR-WS.org.

[137] Jing Xiong, Guohui Xiao, Tahir Emre Kalayci, Marco Montali, Zhenzhen Gu, and Diego Calvanese. A virtual knowledge graph based approach for object-centric event logs extraction. In Marco Montali, Arik Senderovich, and Matthias Weidlich, editors, *Process Mining Workshops - ICPM 2022 International Workshops, Bozen-Bolzano, Italy, October 23-28, 2022, Revised Selected Papers*, volume 468 of *Lecture Notes in Business Information Processing*, pages 466–478. Springer, 2022.

[138] Baoxin Xiu, Guangming Li, and Yidan Li. Discovery of object-centric behavioral constraint models with noise. *IEEE Access*, 10:88769–88786, 2022.

[139] Yang Xu, Qi Lin, and Martin Q Zhao. Merging event logs for process mining with hybrid artificial immune algorithm. In *Proceedings of the International Conference on Data Science (ICDATA)*, page 10. The Steering Committee of The World Congress in Computer Science, Computer ..., 2016.

[140] Jing Yang, Chun Ouyang, Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Yang Yu. Orgmining 2.0: A novel framework for organizational model mining from event logs. *CoRR*, abs/2011.12445, 2020.