Object-Centric Process Mining (and More) Using a Graph-Based Approach With PromG*

Ava Swevels, Eva L. Klijn and Dirk Fahland

Eindhoven University of Technology, The Netherlands

Abstract

PromG is an extensible Python library for managing and enriching object-centric event data (OCED) and for developing object-centric process mining (OCPM) techniques. It does so by using Event Knowledge Graphs, which model process-related concepts as a property graph in a Neo4j database. The library automatically generates Cypher queries to transform, enhance, and manipulate object-centric event data, giving analysts a straightforward way to explore and analyze object-centric processes. To enable others to develop OCPM techniques, the library is available as a Python package on PyPi and has been tested with real-life examples.

Keywords

Object-Centric Process Mining, Object-Centric Event Data, Event Knowledge Graphs, Neo4j

1. Introduction

Analysis of real-life processes with multiple interrelated objects has revealed the limitations of traditional case-centric process mining techniques [1, 2, 3]. As a result, classical process mining techniques such as *control-flow discovery* and *conformance checking* must be adapted, and new techniques must be developed addressing the multi-object interactions of the process. These techniques are collectively referred to as *object-centric process mining* (OCPM) [4]. Some techniques have already been proposed by academia [5, 6, 7, 8, 9] and process mining vendors (notably MyInvenio/IBM and Celonis).

However, an open-source ecosystem that enables development and application of OCPM in the broader process mining community has yet to form. It should offer extensible, easy-to-use functionality for (1) managing *object-centric event data* (OCED), e.g., import, storage, preprocessing, export, (2) exploring OCED from various angles, (3) routine analysis of OCED, e.g., discovery, performance, and (4) one-off analysis specific to a particular use case.

Toward this goal, we developed the open-source Python library PromG which uses the Neo4j graph DB system to *store data and analysis in a multi-layered knowledge graph.* PromG implements a recent community proposal for standard OCED[1] and provides standard functionality for importing, managing, and analyzing OCED (by automatically generating queries against Neo4j). Additionally, it allows users to script custom OCPM analyses and implement newly

ICPM 2023 Doctoral Consortium and Tool Demonstration Track

*The research underlying this paper was supported by AutoTwin EU GA n. 101092021

a.j.e.swevels@tue.nl (Ava Swevels); e.l.klijn@tue.nl (Eva L. Klijn); d.fahland@tue.nl (Dirk Fahland)

[1] cf. ICPM 2022, https://icpmconference.org/2022/program/xes-symposium/

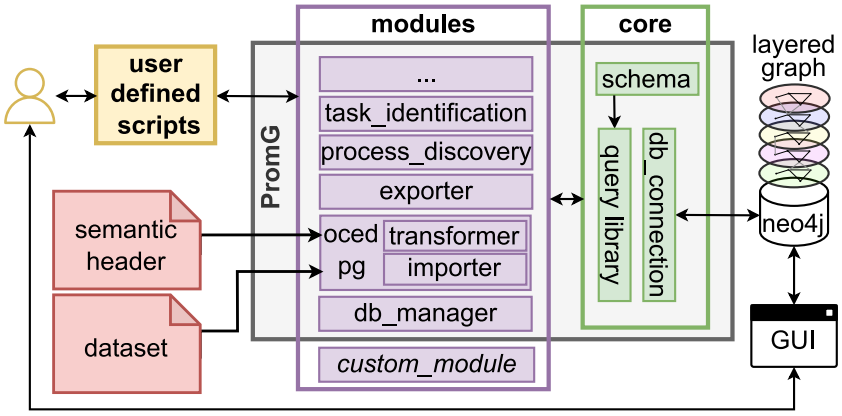

Figure 1: PromG architecture

developed OCPM techniques. By leveraging on industrial GUIs for Neo4j, we relieve analysts and researchers of engineering efforts for interactively querying, exploring, and visualizing OCED.

2. Overview and Design

PromG is a Python library that realizes OCPM by using a Neo4j graph database as data store. Its architecture is illustrated in Fig. 1. The Neo4j database stores OCED and process mining analysis results in multiple *layers* of an Event Knowledge Graph [3], a specific labeled property graph, that describes (qualified) relations between events, objects, relations, and their attributes (over time).

PromG translates process mining tasks into Cypher queries that are run against the Neo4j instance. It consists of *modules* that capture the logic to store and analyze the data, and *core* functionalities that provide a query library, a database connection to the Neo4j instance and the data schema. The latter is implemented in the *core*, as Neo4j (or any graph database) lacks a schema implementation.

Users can build a process mining analysis using existing modules. Additionally, since the data is stored in a Neo4j instance, it can be accessed through Cypher queries and industrial GUIs, allowing further processing, exploration, and analysis to be built on top of PromG. Therefore, we provide users with a template to create their own modules that interact with the core features, thus enabling them to realize their own OCPM analysis techniques.

3. Functionalities Available

While PromG is designed to be easily extended with additional features, we discuss the current capabilities along the currently available layers, allowing users to take advantage of the tool immediately.

(a) Raw Records to OCED Event Layer. The *OCED-PG* module enables the automatic import

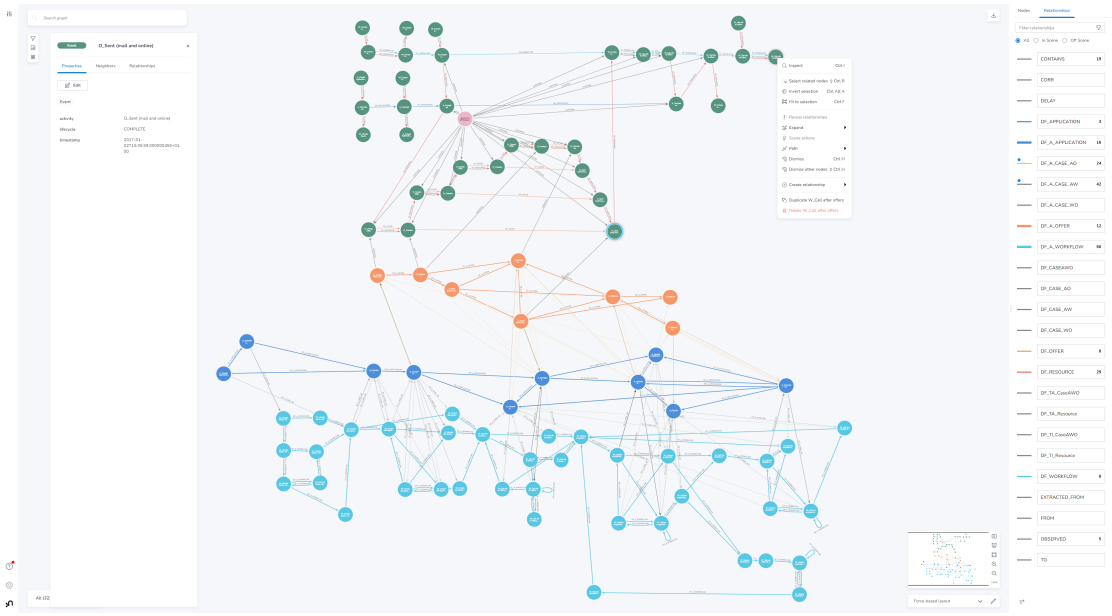

Figure 2: Multi-layered event knowledge graph of BPIC'17 generated by PromG and explored in Neo4j Bloom.

of legacy data records as nodes in *raw record layer* (at the "bottom" of the graph). Based on a user-provided *semantic header* (a JSON document describing the data's domain semantics), *OCED-PG* generates queries that automatically transform the raw record nodes into nodes of related events, objects, and attribute forming the domain-level *event layer* in OCED format [10]; each node of the event layer is linked to the nodes of the raw record layer it originates from.

(b) Object-path inference. Per object chosen by the user, *OCED-PG* infers the *directly-follows (df) path* of events per object (enhancing the event layer), resulting in a partial order over all events that can be analyzed [3].

(c) Event Layer to Process Model layer. The *process discovery module* enables the automated discovery of object-centric process models as multi-object DFGs [3]. The user specifies activity features and objects (or relations) for which the model should be discovered, PromG generates queries that aggregate event nodes and df-relations of the event layer into activity nodes and flow relations per object together – forming a *process model layer*. Each activity node is linked to the event nodes in the event layer it models.

(d) Task Layer. PromG supports OCED analysis beyond classical OCPM use cases. The *task identification module* infers df-paths per resource, uses these to detects sub-graphs where a resource continuously worked on related objects. Queries then abstract the entire event layer into a *task layer* by aggregating sub-graphs into task execution nodes (linked to the underlying events); giving insights into how actors collaborate across executions [11]. Fig. 2 visualizes the interconnected layers on BPIC'17: a task instance node (purple) linked to the underlying event nodes (green) along their DF-paths, and how (some) events link to the multi-object DFG (blue/orange nodes) of BPIC'17.

(e) Custom Modules. We provide a template for users to create their own module that generates queries against Neo4j, enabling user to create custom routine and one-off analyses that enrich existing layers or introduce new layers. Through the template architecture, routine analysis modules can be included in PromG facilitating open-source contributions.

4. Installation, Usage, and Maturity

The PromG library is hosted on PyPi[2] and open-source[3] with example analyses, a demo video and documentation. PromG can be used in any Python project as long as a Neo4j instance[4] (with the APOC plugin[5] installed) is available. PromG provides example projects for constructing EKGs of 5 public real-life event logs of different sizes (BPIC14, BPIC15, BPIC16, BPIC17, BPIC19). Graph construction is a one-time operation that depends on the number of relationships to construct [12, Tab.4]. Improving PromG query performance is planned future work.

PromG's approach and queries have been used in developing custom analyses in multiple industrial case studies in baggage handling systems [13], semiconductor [14] and ship manufacturing [15], and configuration management [16] with consistently positive feedback that the graph-based approach enables insights and analytics not obtainable previously. Incorporating relevant analysis functions into PromG is planned future work.

5. Comparison to Related Software

Next to closed-source implementations of OCPM, only the open-source Python library OCPA [9] addresses the same objective as PromG. OCPA currently offers more analytics functionality than PromG, and serves as "backbone" for the GUI-based analysis tools OCPM [7] and OCπ [8].

PromG's strengths lie in the multi-layered Event Knowledge Graph (EKG) within a standardized data store (Neo4j): the EKG implements standard OCED with domain semantics; the extensible layers persist analysis results linked to the source data (see Fig. 2); Neo4j's query language Cypher and GUIs enables advanced, interactive data exploration and visualization crucial for OCPM analysis.

6. Conclusion

PromG is an open-source Python library designed to manage and explore OCED and to perform OCPM analyses. Although its current functionality is limited compared to some academic counterparts, PromG's architecture prioritizes ease of extension and future development, positioning it as a valuable tool in the growing field of OCED and OCPM.

Particularly, PromG's multi-layered knowledge graph promotes the development of a number of extensions: next to realizing further OCPM capabilities [9, 8] an inference engine for inferring missing or latent information [14] building on an integration of event data with system design

[2] https://pypi.org/project/promg/
[3] https://github.com/PromG-dev
[4] https://neo4j.com/product/neo4j-graph-database/
[5] https://neo4j.com/labs/apoc/

and context data [13]; analysis of actor behavior and organizational routines [11, 15]; and detecting emergent behavior and its propagation across cases [17].

References

[1] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: SEFM 2019, volume 11724 of *LNCS*, Springer, 2019, pp. 3–25.

[2] M. Dumas, F. Fournier, L. Limonad, et al., AI-augmented business process management systems: A research manifesto, ACM Trans. Manag. Inf. Syst. 14 (2023) 11:1–11:19.

[3] D. Fahland, Process mining over multiple behavioral dimensions with event knowledge graphs, in: Process Mining Handbook, volume 448, Springer, 2022, pp. 274–319.

[4] W. M. P. van der Aalst, Twin transitions powered by event data - using object-centric process mining to make processes digital and sustainable, in: ATAED 2023, volume 3424 of *CEUR-WS.org*, 2023.

[5] X. Lu, M. Nagelkerke, D. van de Wiel, D. Fahland, Discovering interacting artifacts from ERP systems, IEEE Trans. Serv. Comput. 8 (2015) 861–873.

[6] W. M. P. van der Aalst, A. Berti, Discovering object-centric petri nets, Fundam. Informaticae 175 (2020) 1–40.

[7] A. Berti, W. M. van der Aalst, Oc-pm: analyzing object-centric event logs and process models, International Journal on Software Tools for Technology Transfer 25 (2022) 1 – 17.

[8] J. N. Adams, W. M. van der Aalst, Ocπ: Object-centric process insights, in: Applications and Theory of Petri Nets, 2022.

[9] ocpa: A python library for object-centric process analysis, Software Impacts 14 (2022) 100438.

[10] A. Swevels, D. Fahland, M. Montali, Implementing object-centric event data models in event knowledge graphs, in: Process Mining Workshops. ICPM 2023, Lecture Notes in Business Information Processing, 2023. Accepted, to appear.

[11] E. L. Klijn, F. Mannhardt, D. Fahland, Classifying and detecting task executions and routines in processes using event graphs, in: BPM'21 Forum, volume 427 of *LNBIP*, Springer, 2021, pp. 212–229.

[12] S. Esser, D. Fahland, Multi-dimensional event data in graph databases, Journal on Data Semantics (2021).

[13] V. Chu, Using event knowledge graphs to model multi-dimensional dynamics in a baggage handling system, 2022.

[14] A. Swevels, R. Dijkman, D. Fahland, Inferring missing entity identifiers from context using event knowledge graphs, in: BPM 2023, volume 14159 of *LNCS*, 2023.

[15] Y. Wang, Event graph model discovery for waiting time and workflow analysis in damen's process, 2022.

[16] K. Marangoz, Capturing multi-dimensional dynamics in a configuration management process through event knowledge graphs, 2023.

[17] B. Bakullari, J. van Thoor, D. Fahland, W. M. P. van der Aalst, The interplay between high-level problems and the process instances that give rise to them, in: BPM 2023 Forum, volume 490 of *LNBIP*, 2023, pp. 145–162.