Explaining Process Behavior: A Declarative Framework for Interpretable Event Data

Christian Dormagen[1], Jonas Amling[2], Stephan Scheele[3] and Ute Schmid[1]

[1] *University of Bamberg, Germany*
[2] *dab: Daten - Analysen & Beratung GmbH, Deggendorf, Germany*
[3] *OTH Regensburg, Germany*

Abstract
Process mining has become a cornerstone for organizations to analyze and optimize their operational workflows. However, as these methods become increasingly complex and data-driven, they often exhibit black-box characteristics, leading to an interpretability gap. This gap undermines stakeholder trust and widespread adoption. We introduce a domain-specific, declarative explanation approach for interpreting complex patterns in event data. We present a semantic event log data model derived from the object-centric event data (OCED) metamodel, which is represented as a formal ontology (relational perspective) and enriched with Declare constraints (temporal perspective). Based on the semantic event log, we use different concept-learning approaches to derive rule-based explanations for domain experts. We demonstrate our method in two real-world case studies: (1) explaining clustering results in a classic event log, and (2) deriving interpretable subprocess explanations from an object-centric dataset. The results show that our approach produces high-fidelity explanations for classical event data but that further research is needed for object-centric data.

Keywords
Process Mining, Explainable AI, Rule-based Explanations, Post-Hoc Explanations, Object-centric Event Data

1. Introduction

Process Mining (PM) has emerged as a field at the intersection of business process management and data science, providing methods for analyzing and improving business processes. By extracting insights from event data generated by Enterprise Resource Planning (ERP) systems, PM enables organizations to gain an actionable understanding of their workflows, provided that these technical insights are communicated in an understandable and trustworthy manner.

Despite this, the field of PM suffers from interpretability problems in three ways: (1) As it is a subfield of Data Science [1], PM is affected by the increasing use of black box methods and their inherent interpretability challenges [2]. Classical statistical methods such as clustering [3] identify behavior patterns while neural networks, for example, predict process outcomes [4, 5]. (2) Even specialized PM methods that are not inherently black boxes can produce uninterpretable results due to the complexity of real-world data. The result is “spaghetti models”, learned process models which are so complex that they are of limited usefulness for furthering human understanding of the process behavior [6]. (3) The field of PM is currently moving towards a more complex, object-centric data model [7, 8]. This more complex data model has the potential to exacerbate the existing process complexity problems which lead to “spaghetti-models”. These issues limit stakeholder trust and hinder the adoption of PM [9].

These challenges highlight the importance of explainability in process mining. Effective explainability ensures that stakeholders trust the results and understand the rationale behind them, thereby increasing the usefulness of process mining methods.

Late-breaking work, Demos and Doctoral Consortium, colocated with The 3rd World Conference on eXplainable Artificial Intelligence: July 09–11, 2025, Istanbul, Turkey
christian.dormagen@uni-bamberg.de (C. Dormagen); jonas.amling@dab-gmbh.de (J. Amling); stephan.scheele@oth-regensburg.de (S. Scheele); ute.schmid@uni-bamberg.de (U. Schmid)
0009-0006-7744-746X (C. Dormagen); 0009-0000-1554-9950 (J. Amling); 0000-0003-0787-3181 (S. Scheele); 0000-0002-1301-0326 (U. Schmid)

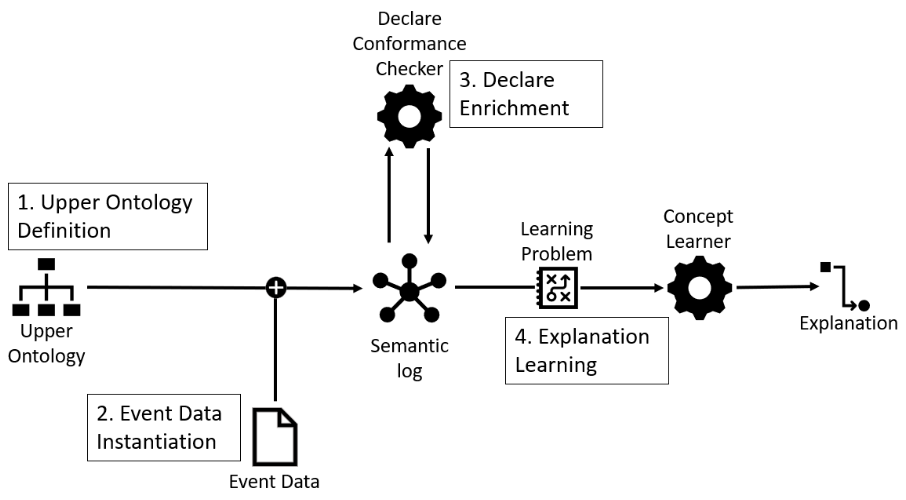

Figure 1: High-level overview of the method

Despite these needs, a significant gap remains: an explainer tailored specifically to explaining and differentiating subprocesses within event data, including object-centric event data, is lacking. Such an explainer could be used in combination with classical process models, that can become highly complex, and should be able to concisely explain what separates subprocesses within the data. In addition, the unique characteristics of PM need to be addressed. Temporal dependencies between events, which constitute a step in a process, are at the core of process understanding. For instance, the temporal dependency between receiving and closing an invoice must be expressible in an invoice handling process. Using process mining-specific languages to solve this need, such an explainer could bridge the technical-business divide and help overcome the interpretability problem.

Our contributions are the definition of a domain ontology for process mining and its application in an event data explainer. In Section 2 we will introduce related work and in Section 3 some preliminaries. Then in Section 4 we introduce the architecture of our explainer via the four steps shown in Figure 1. Section 5 presents preliminary results on two use-cases. And finally, in Section 6 we discuss our results, including its limitations and an outlook on future work.

2. Related Work & Preliminaries

2.1. Declarative Process Mining

Declarative process mining refers to techniques based on process models specified in declarative, constraint-based languages. Early work on declarative PM was based on $\mathcal{SCIFF}$ integrity constraints (ICs) [10, 11, 12]. Later declarative approaches moved away from $\mathcal{SCIFF}$ and towards Declare [13], which is based on Temporal Linear Logic with Past Operators over Finite Traces (PLTL$_f$).

The first discovery method [14] for Declare constraints thus used automata, an approach which was later made more efficient [15] and more meaningful via the exclusion of vacuous (misleading) constraints [16, 17] and discovery approaches [18, 19]. Notably, many approaches increase the expressivity of Declare constraints, for instance, by including the data perspective [16] or by introducing disjunction [20, 17].

2.2. Explainable Process Mining

Particularly in the field of predictive process monitoring (predicting results for incomplete processes), various xAI approaches, such as Shapley values[21] or LIME[22] have been applied to PM. Beyond predictive process monitoring, xAI has been used to explain concept drift in event data [23], introduces methods to increase trust in suggestions provided by process-aware recommender systems [24] and is used to group events into cases based on a process model and missing case identifiers [25].

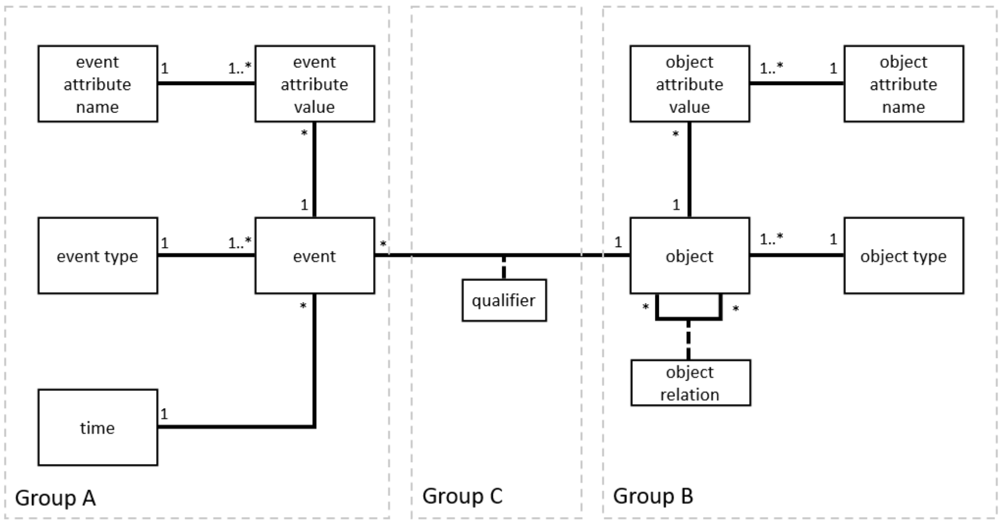

Figure 2: The OCED metamodel in simplified Unified Modeling Language (UML) notation, based on [26].

2.3. Event Data

ERP systems record organizational workflows through *events*, each representing a step in a process (e.g., a payment) occurring at some time. Events have types, referred to as *activities*. Events relate to *objects* such as persons, documents, or units, which may also relate to each other. Both events and objects can have *attributes*. To analyze event data one or more *case concept*s are identified, which are objects representing a type of process (e.g., an invoice for an invoice handling process). The ordered sequences of events related to instances of a case concept form *traces*, representing instances of a process. Traces have a *variant*, which is the ordered sequence of activities for the events.

Classical event data flattens ERP records to a single table based on one case concept. This discards object relations and can distort process behavior via the introduction of artifacts [7].

Object-centric event data (OCED) extends classical data by preserving the relational structure of ERP systems. It supports multiple case concepts enabling process interactions (e.g., an invoice process triggering a payment process), while avoiding artifacts from flattening and including more semantic knowledge. We base our ontology on the OCED metamodel [26], which includes: (A) events with types and attributes, which are equivalent to classical event data, (B) typed objects with attributes and interrelations, and (C) qualified event-object relations such as create or modify. It is shown in Figure 2.

3. Explainer Architecture

3.1. Upper Ontology Definition

We formalize our ontology in the Web Ontology Language (OWL) profile OWL2DL, which is defined based on the Description Logic (DL) $\mathcal{SROIQ}(\mathcal{D})$. For the formal definitions, we refer to [27].

For our upper ontology we can directly capture the concepts and relations in the OCED metamodel via the DL classes `Event`, `Time`, `Attribute`, and `Object` and the properties `event_to_object_relation`, `object_to_object_relation`, `observed_at` (a `Event` is observed at exactly one `Time`), `has_attribute` (a `Event` or `Object` has any number of `Attributes`) and `has_value` (a `Attribute` has exactly one data value).

3.2. Event Data Instantiation

We can then take any classical or object-centric event log based on our upper ontology to derive a process-specific ontology. For this, we introduce all the types (Event types, object types, attributes names) in the data as subclasses of their respective concepts (`Event`, `Object` and `Attribute`). We do the same

Table 1
Some Declare constraint templates and their informal semantics

Template	Informal Semantics
Presence(x)	At least one activity x must occur.
Absence(x)	No activity x can occur.
Response(x, y)	If activity x occurs, then activity y must occur afterwards.
Precedence(x, y)	If activity y occurs, then activity x must occur beforehand.
ChainResponse(x, y)	If activity x occurs, then there must immediately be an activity y afterwards.
ChainPrecedence(x, y)	If activity y occurs, then there must be an activity x immediately preceding it.

for qualifiers, specializing `event_to_object_relation` and `object_to_object_relation` with new properties for each qualifier. Whenever we create a property we also create its inverse. Then, we instantiate all individuals, relations and data values in accordance with the data.

3.3. Declare Enrichment

In declarative process mining, the constraint language Declare defines process models as conjunctions of constraints over activities. Typically, Declare constraints specify temporal dependencies between activities, grounded in Linear Temporal Logic over finite traces (LTL$_f$) [28]. For formal semantics, we refer to [29]. For an informal overview of some Declare constraints, see Table 1.

Selecting Declare Constraints We use several approaches from declarative process mining to determine which Declare constraints to instantiate. First, we compute itemset-support [30] to exclude rarely co-occurring activities from binary constraints, and then we apply a threshold to constraint-support [30] to retain only frequently satisfied constraints. Finally, we filter out vacuously satisfied [17] constraints, which are logically correct but can be misleading and thus are less interpretable. To compute the metrics we use established declarative conformance checker. The results are then instantiated into our ontology. Since the OCED metamodel lacks explicit traces, we represent each Declare constraint as a subclass of the case concept (e.g., Presence(payment) as a subclass of `Invoice`). It represents all invoice traces where the constraint holds. As classical event logs lack objects, we introduce abstract case concept instances based on case IDs.

3.4. Explanation Learning

With the inclusion of Declare constraints our ontology is complete. We can now extract concept-learning problems and use concept-learning algorithms to derive explanations for patterns within the data. Informally, a concept learning problem is: Given a knowledge base $\mathcal{K}$ and sets of positive (E^+) and negative (E^-) examples of a target concept, find a class expression C that best fit the examples, balancing coverage of positive instances and exclusion of negative ones. We use our ontology as background knowledge and identify positive and negative examples based on the use case.

Our method is agnostic to the concept learner used. We employ two CELOE [31] and the EvoLearner [32]. CELOE finds less expressive concepts, excluding data values. Hence, it is suitable when focusing only on temporal dependencies and object relations. EvoLearner offers greater expressiveness via including data values in its concepts, but potentially generates longer concept expressions.

In contrast to classical process discovery techniques, which produces process models that try to encompass the whole process, our explanations identify the core behavior that separates two sets of traces within one process from each other.

Restricting the Background Knowledge We introduce two settings of restrictions on background knowledge, which enable different types of explanations. First is the Declare-only setting. It limits the background knowledge to Declare constraints, simplifying explanations to the temporal perspective

Table 2
Results for the first use case, by cluster. The table shows the number of traces and variants, along with the metrics: accuracy, F1-score, precision, recall, and the syntactic length of the learned explanations.

Cluster	Traces	Variants	Acc.	F1	Prec.	Rec.	Len.
Noise	1210	130	-	-	-	-	-
0	46383	2	1.00	1.00	1.00	1.00	1
1	20385	1	1.00	1.00	1.00	1.00	7
2	3587	6	1.00	1.00	1.00	1.00	5
3	3160	41	0.99	0.97	0.95	1.00	9
4	17041	36	1.00	1.00	1.00	1.00	7
5	2122	14	1.00	1.00	1.00	1.00	3
6	56482	1	1.00	1.00	1.00	1.00	7

and produces explanations which can be interpreted as Declare models with atomic negation. Second is the unrestricted setting. It includes the temporal perspective via Declare constraints and the remaining relational knowledge encoded in the ontology. The explanations learned in this setting are an embedding of Declare into Description Logic.

4. Preliminary Results

We present preliminary results on two use cases: a classical event log (Declare-only setting) and an object-centric one (unrestricted setting). We evaluate explanation fidelity using standard classification metrics (Accuracy, F1-score) and use the syntactic length of the explanation as a complexity proxy.

4.1. Use Case 1: Road Traffic Fine Management Process (RTFMP)

We analyze clusters derived from the RTFMP log using density-based clustering on boolean activity presence vectors. Since only the presence or absence of activities is used for the clustering, we are only interested in the temporal perspective. Hence, we employ the CELOE algorithm together with Declare-only background restrictions. The RTFMP dataset captures the process of managing traffic fines, which includes issuing fines, notifying recipients, and resolving payments or appeals. Fines that remain unpaid are escalated to credit collection through several distinct escalation and appeal paths.

Results Table 2 shows the results for the first use case, excluding the noise cluster from our evaluation. We can see that all clusters observe perfect or near-perfect fidelity, with accuracy, precision, recall, and F1-score all at or very close to 1.00. This indicates that the generated rules are highly effective in covering the positive traces while excluding negatives. The complexity of the explanations is lowest for cluster 0 with a length of 1 and highest for cluster 3 with a length of 9.

Figure 3 shows the explanation for cluster 3. It contains two distinct behaviors: First, a fine is created, followed by an appeal to the prefecture, but no results of this appeal are received. Second, a fine is created and appealed to the prefecture. However, further escalation, such as an appeal to a judge or involvement of credit collection, is not included in the trace, nor is payment from the offender. An interpretation of this explanation is that it describes appeals that have not yet been resolved or were successful and require no payment.

```
Response(create_fine, send_appeal_to_prefecture)
  ∧ Absence(receive_result_appeal_from_prefecture),
Response(create_fine, send_appeal_to_prefecture)
  ∧ Absence(appeal_to_judge)
  ∧ Absence(payment)
  ∧ Absence(send_for_credit_collection)
```

Figure 3: The explanation learned for cluster 3, presented as two disjunct Declare models

Table 3
Results for the second use case, by subprocess and concept learner. The table again shows the number of traces and variants, along with the metrics: accuracy, F1-score, precision, recall, and the syntactic length of the learned explanations.

Subprocess	Traces	Variants	Learner	Acc.	F1	Prec.	Rec.	Len.
1	1770	884	Evo	0.98	0.76	1.00	0.62	4
			CELOE	0.60	0.12	0.07	0.58	1
2	30215	927	Evo	0.90	0.94	0.98	0.89	4
			CELOE	0.90	0.94	0.98	0.89	1
3	21	9	Evo	1.00	0.47	0.31	0.95	6
			CELOE	0.98	0.05	0.03	0.81	2
4	4844	73	Evo	0.85	0.06	0.24	0.03	7
			CELOE	0.78	0.52	0.36	0.93	2

4.2. Use Case 2: Business Process Intelligence Challenge 2019 (BPIC19)

We evaluate explanations for subprocess types in the BPIC19 dataset using both CELOE and EvoLearner in the unrestricted setting. The dataset, which was initially released as a classical event log and then manually transformed into an object-centric one, poses challenges due to noise, imbalance, and high variant complexity. It is derived from the purchase order process of a large multinational and includes four annotated subprocesses with distinct behaviors, which we attempt to differentiate with our explanations.

Results EvoLearner generally achieves higher fidelity, though with more complex explanations (avg. length 5.25 vs. 1.5). Both learners struggle on subprocesses 3 and 4, potentially due to label imbalance and noise. Subprocess 2 is best explained by both learners (Acc. 0.90, F1 0.94), with CELOE producing a smaller yet accurate explanation. Figure 4 shows an explanation with good fidelity. The explanation for subprocess one shows that we do indeed get a combination of the temporal perspective via a Declare constraint, requiring that no `clear_invoice` follows directly after a `record_goods_receipt` and a relational perspective via quantification over relations in Description Logic, requiring that the recording of a service sheet must modify the purchase order item.

Subprocess 1, EvoLearner:
```
NotChainResponse(record_goods_receipt, clear_invoice)
  ⊓ (∃ inverse_modify.record_service_entry_sheet)
```

Figure 4: A explanation including both the temporal and relational perspective for use case 2.

5. Discussion

We presented an ontology-based architecture to explain process behavior using concept learners and Declare constraints, supporting classical and object-centric event data. Through restrictions on the ontology, our method can incorporate temporal constraints alone (Declare-only setting) or both temporal and relational perspectives (Unrestricted setting). Applied to a classical log, we achieved good fidelity in explaining clustering results in the declare-only setting. Initial results are promising but mixed for the unrestricted setting on object-centric data. The explanations include a temporal and relational perspective, but fidelity was low for some subprocesses.

Explanation fidelity for our object-centric use case is limited partly due to the quality of the OCED dataset. Additionally, our current approach requires identifiable negative examples and has only been evaluated using basic fidelity and complexity metrics. While we argue that explanations are understandable, this has not yet been validated, and future user studies are needed.

Future work will evaluate object-centric data more exhaustively. We want to look at explaining more complex clustering approaches, e.g. clustering based on Declare constraints, or an ontology embedding. We aim to improve interpretability, potentially using verbalization techniques. Furthermore, we also plan to integrate richer domain knowledge such as organizational structures or business rules to improve explanations.

Acknowledgments

Funded by the Bavarian collaborative research program (BayVFP) under project KIGA (DIK0313/02).

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT-4o and Grammarly in order to: Grammar and spelling check, paraphrase and reword, improve writing style, drafting content. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] W. van der Aalst, Process mining: Overview and opportunities, ACM Transactions on Management Information Systems 3 (2012) 1–17. doi:10.1145/2229156.2229157.

[2] V. Hassija, V. Chamola, A. Mahapatra, A. Singal, D. Goel, K. Huang, S. Scardapane, I. Spinelli, M. Mahmud, A. Hussain, Interpreting black-box models: A review on explainable artificial intelligence, Cognitive Computation 16 (2023) 45–74. doi:10.1007/s12559-023-10179-8.

[3] M. Song, C. W. Günther, W. M. P. van der Aalst, Trace Clustering in Process Mining, Springer Berlin Heidelberg, 2009, pp. 109–120. doi:10.1007/978-3-642-00328-8_11.

[4] J. Theis, W. L. Galanter, A. D. Boyd, H. Darabi, Improving the in-hospital mortality prediction of diabetes icu patients using a process mining/deep learning architecture, IEEE Journal of Biomedical and Health Informatics 26 (2022) 388–399. doi:10.1109/jbhi.2021.3092969.

[5] H. Weytjens, J. De Weerdt, Process Outcome Prediction: CNN vs. LSTM (with Attention), Springer International Publishing, 2020, pp. 321–333. doi:10.1007/978-3-030-66498-5_24.

[6] G. M. Veiga, D. R. Ferreira, Understanding Spaghetti Models with Sequence Clustering for ProM, Springer Berlin Heidelberg, 2010, pp. 92–103. doi:10.1007/978-3-642-12186-9_10.

[7] W. M. P. van der Aalst, Object-centric process mining: Dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods, Springer International Publishing, 2019, pp. 3–25. doi:10.1007/978-3-030-30446-1_1.

[8] A. Berti, I. Koren, J. N. Adams, G. Park, B. Knopp, N. Graves, M. Rafiei, L. Liß, L. T. G. Unterberg, Y. Zhang, C. Schwanen, M. Pegoraro, W. M. P. van der Aalst, Ocel (object-centric event log) 2.0 specification, 2024. doi:10.48550/ARXIV.2403.01975.

[9] A. Pery, M. Rafiei, M. Simon, W. M. P. van der Aalst, Trustworthy Artificial Intelligence and Process Mining: Challenges and Opportunities, Springer International Publishing, 2022, pp. 395–407. doi:10.1007/978-3-030-98581-3_29.

[10] E. Lamma, P. Mello, M. Montali, F. Riguzzi, S. Storari, Inducing declarative logic-based models from labeled traces, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 344–359. doi:10.1007/978-3-540-75183-0_25.

[11] F. Chesani, E. Lamma, P. Mello, M. Montali, F. Riguzzi, S. Storari, Exploiting Inductive Logic Programming Techniques for Declarative Process Mining, Springer Berlin Heidelberg, 2009, pp. 278–295. doi:10.1007/978-3-642-00899-3_16.

[12] E. Bellodi, F. Riguzzi, E. Lamma, Probabilistic Declarative Process Mining, Springer Berlin Heidelberg, 2010, pp. 292–303. doi:10.1007/978-3-642-15280-1_28.

[13] M. Pesic, H. Schonenberg, W. M. van der Aalst, DECLARE: Full support for loosely-structured processes, in: 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE, 2007. doi:10.1109/edoc.2007.14.

[14] F. M. Maggi, A. J. Mooij, W. M. van der Aalst, User-guided discovery of declarative process models, in: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2011. doi:10.1109/cidm.2011.5949297.

[15] F. M. Maggi, C. Di Ciccio, C. Di Francescomarino, T. Kala, Parallel algorithms for the automated discovery of declarative process models, Information Systems 74 (2018) 136–152. doi:10.1016/j.is.2017.12.002.

[16] F. M. Maggi, R. P. J. C. Bose, W. M. P. van der Aalst, A Knowledge-Based Integrated Approach for Discovering and Repairing Declare Maps, Springer Berlin Heidelberg, 2013, pp. 433–448. doi:10.1007/978-3-642-38709-8_28.

[17] C. Di Ciccio, F. M. Maggi, M. Montali, J. Mendling, On the relevance of a business constraint to an event log, Information Systems 78 (2018) 144–161. doi:10.1016/j.is.2018.01.011.

[18] C. Di Ciccio, M. Mecella, A two-step fast algorithm for the automated discovery of declarative workflows, in: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2013. doi:10.1109/cidm.2013.6597228.

[19] C. D. Ciccio, M. Mecella, On the discovery of declarative control flows for artful processes, ACM Transactions on Management Information Systems 5 (2015) 1–37. doi:10.1145/2629447.

[20] C. Di Ciccio, F. M. Maggi, M. Montali, J. Mendling, Ensuring Model Consistency in Declarative Process Discovery, Springer International Publishing, 2015, pp. 144–159. doi:10.1007/978-3-319-23063-4_9.

[21] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, 2017. doi:10.48550/ARXIV.1705.07874.

[22] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, ACM, 2016, pp. 1135–1144. doi:10.1145/2939672.2939778.

[23] J. N. Adams, S. J. van Zelst, T. Rose, W. M. van der Aalst, Explainable concept drift in process mining, Information Systems 114 (2023) 102177. doi:10.1016/j.is.2023.102177.

[24] A. Padella, M. de Leoni, O. Dogan, R. Galanti, Explainable process prescriptive analytics, in: 2022 4th International Conference on Process Mining (ICPM), IEEE, 2022. doi:10.1109/icpm57379.2022.9980535.

[25] D. Bayomie, K. Revoredo, C. D. Ciccio, J. Mendling, Improving accuracy and explainability in event-case correlation via rule mining, in: 2022 4th International Conference on Process Mining (ICPM), IEEE, 2022. doi:10.1109/icpm57379.2022.9980684.

[26] OCED Standard - IEEE Task Force on Process Mining — tf-pm.org, https://www.tf-pm.org/resources/oced-standard, ???? [Accessed 18-05-2024].

[27] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2007. doi:10.1017/cbo9780511711787.

[28] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 854–860.

[29] C. D. Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: Lecture Notes in Business Information Processing, Springer International Publishing, 2022, pp. 108–152. doi:10.1007/978-3-031-08848-3_4.

[30] F. M. Maggi, R. P. J. C. Bose, W. M. P. van der Aalst, Efficient discovery of understandable declarative process models from event logs, in: Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer, 2012, pp. 270–285. doi:10.1007/978-3-642-31095-9_18.

[31] J. Lehmann, S. Auer, L. Bühmann, S. Tramp, Class expression learning for ontology engineering, Journal of Web Semantics 9 (2011) 71–81. doi:10.1016/j.websem.2011.01.001.

[32] S. Heindorf, L. Blübaum, N. Düsterhus, T. Werner, V. N. Golani, C. Demir, A.-C. Ngonga Ngomo, Evolearner: Learning description logics with evolutionary algorithms, in: Proceedings of the ACM Web Conference 2022, WWW '22, ACM, 2022, pp. 818–828. doi:10.1145/3485447.3511925.