Predictive Process Monitoring via Automated Planning

Angelo Casciani[1], Alessandra Giovannetti[1], Silvia Macagnano[1], Andrea Marrella[1], Mario Luca Bernardi[2], Marta Cimitile[3], and Fabrizio Maria Maggi[4]

[1] Sapienza University of Rome, Rome, Italy,
lastname@diag.uniroma1.it
[2] University of Sannio, Benevento, Italy,
bernardi@unisannio.it
[3] Unitelma Sapienza University, Rome, Italy,
marta.cimitile@unitelma.it
[4] Free University of Bozen-Bolzano, Bolzano, Italy,
maggi@inf.unibz.it

Abstract. Predictive Process Monitoring (PPM) forecasts the future behavior of ongoing business process executions to support proactive decision-making. Although state-of-the-art deep learning approaches achieve high predictive accuracy, they often act as black boxes, without explaining the execution steps that lead to a given outcome, thus limiting their applicability in scenarios requiring transparent what-if analysis. In this paper, we propose a planning-based framework for *situation prediction*, which anticipates not only whether a specific outcome is achievable but also the exact sequence of activities and data updates required to reach it. We introduce an automated pipeline that transforms event logs into PDDL planning tasks by combining process discovery with decision mining, allowing classical planners to generate compliant, goal-oriented predictions. Experiments on real-world logs show that our framework provides high solvability and predictive performance while offering transparency.

Keywords: Automated Planning, PDDL, Predictive Process Monitoring

1 Introduction and Motivation

Predictive Process Monitoring (PPM) has emerged as one of the most impactful subfields of process mining [27]. By analyzing historical *event logs* recorded by information systems, PPM techniques generate predictive models that can be queried during the ongoing execution of a business process (BP). These models aim to forecast future behavior, support proactive decision-making, and mitigate in advance operational risks.

Consider a healthcare scenario in which a patient p_1 enters the hospital with suspected sepsis [12]. After the prioritization in the emergency room (*ER Sepsis Triage*) and blood tests to measure infection markers like C-reactive protein (*CRP*), doctors must decide on the next steps, such as administering intravenous antibiotics (*IV Antibiotics*) or hospitalizing p_1 in the normal care ward (*Admission NC*). By analyzing historical data from patients with similar clinical profiles, they can anticipate whether antibiotics will be prescribed or a full admission will be required. Unlike reactive BP monitoring, PPM allows for anticipating issues, and, if a high risk is predicted, doctors can proactively change the treatment path before further complications arise. Given

a partial execution prefix $\alpha = \langle ER\ Sepsis\ Triage,\ CRP,\ Leucocytes \rangle$, PPM supports: (*i*) *outcome* predictions, which focus on boolean, categorical, or numerical outcomes (e.g., "*Will* p_1 *undergo antibiotics?*"); and (*ii*) *suffix* predictions, forecasting upcoming activities (e.g., "*Given* α*, will* p_1 *be admitted to the normal care and be discharged?*", resulting in the suffix $\beta = \langle IV\ Antibiotics,\ Admission\ NC,\ Release\ A \rangle$).

A major challenge in PPM, whose state-of-the-art technique are mainly machine learning-based, is *limited transparency* [5]. Outcome methods predict *only if* a result is achievable, but *not the sequence that produces it*, obscuring the reasoning in high-stakes scenarios. Conversely, suffix methods rely on black-box sequence matching without the causal rationale (e.g., specific clinical data) required to reach a goal, making them unsuitable for transparent what-if analyses.

In this visionary paper, we propose a paradigm shift for PPM. We argue that, to achieve transparent and actionable what-if analysis, PPM should not be treated solely as a statistical forecasting problem, but can be framed as an *Automated Planning* (AP) task based on the notion of *situation prediction*. Drawing upon the field of *reasoning about actions* [19], we define the concept of *situation* as a future state determined by the sequence of actions leading to it. In this direction, AP offers the ideal abstraction for predicting the causal effects of actions, e.g, returning a plan π that shows *how* p_1 reaches *Admission NC* given his current *CRP* levels.

AP is a branch of Artificial Intelligence (AI) that addresses the problem of generating a course of actions (i.e., a *plan*) to achieve a goal, given a description of the domain of interest and an initial state. By automatically transforming discovered BP models and decision-mining rules into standard PDDL (Planning Domain Definition Language) models [8], we can leverage off-the-shelf classical planners to synthesize future BP behaviors. Unlike neural networks, a classical planner does not guess the most likely next token, but it formally proves whether a specific target situation is reachable from the current state and returns the executable plan (i.e., the suffix) required to achieve it. We note that, in recent years, the planning community has produced a wide range of classical planners incorporating highly effective search heuristics (i.e., capable of scaling to large problems), which have been applied to tackle challenging tasks across multiple areas of Computer Science [14, 13].

Through this paper, we outline the conceptual framework required to enable planning-based PPM, provide preliminary evidence of its feasibility using standard event logs, and lay out a roadmap for future research.

2 Background

PPM relies on historical data captured in *event logs*, usually represented in the IEEE *eXtensible Event Stream* (XES) standard format[1]. A log is a collection of *traces*, where each trace represents a single execution instance of a BP. Each event within a trace records an executed activity, its timestamp, and a payload of data attributes.

To formally reason about both the sequence of activities and the evolution of data payloads, we conceptualize the BP as a *Data Petri Net* (DPN) [23]. Unlike industry-standard languages like BPMN (lacking a rigorous execution semantics) or traditional Petri nets that model only the control-flow (i.e., the ordering and concurrency of

[1] IEEE Standard for XES: `https://xes-standard.org/`

activities via places, transitions, and token markings), DPNs provide an unambiguous semantics and explicitly include the data perspective, enriching transitions with *guard conditions* over data variables (thus, a transition can only fire if the control-flow marking enables it and the current data state satisfies its guard conditions). This aspect makes DPNs the ideal foundation for our *situation prediction*: the PPM task of determining not just *if* an ongoing instance (i.e., a trace prefix) will reach a specific target outcome, but exactly *what sequence* of control-flow steps and data updates is required to achieve it.

3 Related Work

PPM techniques are broadly categorized into model-based and machine learning approaches [6]. Traditional model-based methods (e.g., stochastic Petri nets) [20, 28] mainly focus on time or simple suffix estimation, lacking flexible support for goal-driven predictions under complex data constraints. Conversely, deep learning (DL) models [18, 26] dominate current literature due to their high accuracy but act as opaque black boxes. While recent efforts employ Large Language Models (LLMs) [16] or neuro-symbolic approaches [15] to improve interpretability, they remain data-hungry and struggle to guarantee compliance.

To address the black-box nature of DL, recent research in AI has explored the usage of LLMs that, relying on techniques like chain-of-thought prompting, can generate intermediate "reasoning traces" or step-by-step explanations alongside their predictions. However, recent literature in AP [4, 10] highlighted that generating intermediate tokens should not be anthropomorphized as transparent reasoning but rather considered as probabilistic artifacts. Indeed, they documented a severe disconnect between the generated explanation and the final outcome, frequently lacking *faithfulness*: models can generate plausible and coherent explanations that are logically invalid or hallucinate constraints.

While it can be argued that LLMs offer flexibility in handling unstructured contexts, and that hallucinations can be mitigated by tuning generation parameters (e.g., lowering the temperature), in critical compliance-driven BPs, merely reducing the probability of an error is insufficient. This is an important limitation for PPM: as LLMs do not natively compute reachability over a defined state space, even a low-temperature model risks hallucinating a sequence of activities that violates core BP control-flow rules or data constraints. This reasoning flaw motivates our vision: true transparency and actionable predictions cannot be achieved through statistical text generation or likelihood estimations. Instead, it requires the formal synthesis of plans that are mathematically proven to be executable and compliant. To achieve this, we propose a *pure planning-based paradigm*. By automatically translating event logs into PDDL via process discovery and decision mining, we use classical planners to synthesize goal-oriented predictions, and our approach, differently from DL-based methods, our approach requires no training and guarantees that every predicted sequence is valid with respect to the process structural and data constraints.

4 A Planning-Based PPM Framework

To realize our vision of transparent PPM, we propose a framework to reduce the situation predictions to a planning task. By automatically encoding the BP control-flow and data constraints into PDDL, we can rely on off-the-shelf classical planners to

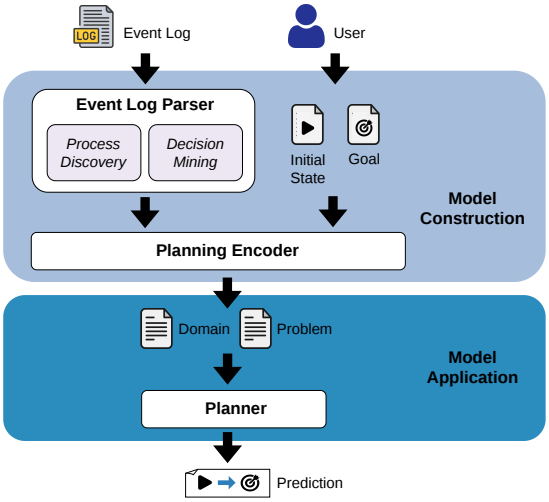

Fig. 1. Framework architecture. The XES log is converted into PDDL domain and problem files with initial states and goals, enabling the planner to generate situation predictions.

synthesize sound predictions. A *classical planning task* in PDDL is specified through: a *domain*, defining the structure of the environment, and a *problem*, which describes a specific instance. A PDDL domain includes a set of *predicates*, specifying the properties and relations that can hold, and a set of *actions* Ω. An action schema $a \in \Omega$ specifies *parameters*, *preconditions* for action executability, and deterministic *effects* describing how it changes the world state. A PDDL problem complements the domain by specifying an *initial state* and a *goal condition*. The planner's objective is to find a *plan* that transitions the initial state into one satisfying the goal. In this work, we focus on *classical planning tasks* in PDDL 1.2. and Action Description Language (ADL) [17].

As illustrated in Figure 1, our PPM approach includes two stages: the translation of a XES event log into PDDL (i.e., *model construction*) and the situation prediction performed using a classical planner (i.e., *model application*).

Model Construction. In the initial phase, the event log is processed by the Parser to capture distinct BP perspectives. First, to extract the control-flow rules, our framework supports a suite of process discovery algorithms (i.e., Inductive, Heuristics, ILP, and Alpha miners [1]). Second, we apply decision mining [21] at the model choice points to get data constraints. Specifically, for every decision point, we train a decision tree (inherently interpretable) to predict the next activity based on the available data attributes (e.g., predicting that a patient undergoes the sepsis triage if their "*CRP*" level is high). This approach naturally discretizes continuous numerical attributes into meaningful intervals (e.g., "*Leucocytes*" ≤ 12.5).

To operationalize the extracted information, the Parser produces a DPN, and the Encoder first translates its control- and data-flow into a PDDL domain.

– **Control-flow.** The DPN structure is modeled using two predicates: `(completed ?a - activity)` to track the execution history, and `(enabled ?a - activity)` to indicate that a transition is structurally executable according to the marking.

- **Data-flow.** Classical planners do not natively handle continuous numeric values. We discretize them into intervals based on the discovered decision thresholds (e.g., a guard representing $[n1,n2)$). These intervals are encoded as PDDL *constants* (e.g., `gte_n1_lt_n2`) and assigned via predicates like `(num-attr ?v - num-type)`.
- **Split Actions.** To handle activities that generate or modify data (e.g., a blood test), we adopt a *split action* encoding. We generate mutually exclusive PDDL actions for each possible data outcome (e.g., `exec_test_lte_n1` and `exec_test_gte_n1_lt_n2`). These actions share the same control-flow precondition (e.g., `(enabled test)`) but yield different data effects. The planner leverages this input-determinism to select the split action required to satisfy downstream goals.
- **Decision Rules.** The rules discovered by the decision miner are strictly enforced as *preconditions* on subsequent actions. For instance, if `next_act1` is only permitted when the test result is lower than $n1$, its PDDL definition will require both `(enabled next_act1)` and `(num-attr lte_n1)`. This mathematically guarantees that the planner cannot take branches that violate the business data rules.

Then, for each predictive query, a PDDL problem file is generated:

- **Initial State.** The `:init` reflects the state of the ongoing BP instance. We replay the observed prefix on the DPN to determine the `completed` and `enabled` activities, and extract the current values of data attributes from the prefix's last event.
- **Goal State.** The user specifies the desired target situation in the `:goal` section. The latter can be a combination of control-flow (e.g., `(completed next_act1)`) and data conditions (e.g., `(num-attr gte_n1_lt_n2)`).

Model Application. This phase requires the PDDL domain and problem for the *situation prediction*. The planner finds a solution that, starting from the initial execution prefix (which may also be empty), achieves the user-defined goal situation (e.g., *"patient discharged with leukocyte level smaller or equal than x"*). A classical planner optimizes for plan length by assuming unit costs, so it will return the shortest valid sequence of activities that transforms the initial into the goal state. Since the PDDL domain encodes the discovered DPN, the planner can not hallucinate solutions. If a solution is found, it necessarily corresponds to an executable and transparent sequence of activities and data updates that leads to the specified situation.

Implementation For the proof-of-concept, we used the discovery algorithms provided by the PM4Py library [3] and we implemented both the Parser and the Encoder in Python. For the prediction module, i.e., the planner, we employed Fast Downward [9].

5 Illustrative Example and Preliminary Evaluation

To concretely illustrate our vision, we examine an example from the *Sepsis Cases* event log [12], which records the trajectories of patients with suspected sepsis in a hospital and presents an unstructured control-flow with multiple payload attributes that make it relevant to test the planner's predictive capabilities over data conditions.

Consider a scenario where a patient has just been admitted to the emergency room, triaged, and has undergone initial blood tests. The current trace prefix is: $\langle$*ER Registration, ER Triage, ER Sepsis Triage, CRP*$\rangle$. The clinician wants to know:

"What is the required sequence of clinical steps to ensure this patient is safely discharged, given that their current CRP level is severely high (i.e., >100 mg/L)?"

Consider predicting a path for a registered patient, hypothesizing a scenario where he exhibits infection markers (e.g., high CRP) and reaches the sepsis triage. The initial state reflects the registration completion, while the goal enforces the target situation.

If we feed this prefix and data payload into a Long Short-Term Memory (LSTM) or Transformer-based PPM model trained on this log, it will output a probability distribution. It might predict that the next most likely activity is "Leucocytes test" (e.g., 60% probability) and that the ultimate probability of a safe discharge is low (e.g., 20%). However, the model provides no guidance on *how* to improve those odds or *what exact clinical protocol* must be followed to achieve the discharge, and if we ask an LLM to explain the neural network's prediction, it may hallucinate a plausible-sounding explanations that do not conform to the hospital's protocols.

In our planning framework, this query is formulated as a PDDL problem where:

- the *initial state* sets the DPN marking to the state immediately following the `CRP test`, and the data predicate is set to `(crp_level gte_147)`.
- the *goal* is set to `(completed Release_A)` (i.e., the standard discharge).

When invoked, the planner explores the state space defined by the discovered DPN. Rather than a probabilistic guess, and based on the decision-mining result that high CRP levels route patients toward specific interventions before discharge, it synthesizes the following sound plan, providing clinicians with a transparent, actionable, and compliant path: $\pi = \langle$`exec_Leucocytes_test`, `exec_IV_Antibiotics` (triggered because $CRP > 147$), `exec_Admission_IC`, `exec_Release_A`$\rangle$.

5.1 Preliminary Evaluation

To assess the computational feasibility and predictive ability of our framework, we conducted a preliminary evaluation using well-established event logs: (*i*) *Sepsis* [12]; (*ii*) *BPI Challenge* (BPIC) *2012 W*, representing the work items of the whole dataset [7]; and (*iii*) *BPIC 2013*, in *closed problems* (CP) and *incidents* (I) versions [24, 25]. Following standard PPM protocols, we split the logs chronologically (80% for DPN discovery, 20% for testing). To test the planner, we generated incremental trace prefixes, i.e., we iteratively provided the first k events of each test trace as the observed prefix and tasked the planner with predicting the remaining sequence. We plan to adopt cross-validation techniques in future extensions to consolidate the validity of our findings further. Table 1 summarizes the results, using the Inductive Miner [11] and an optimal search strategy (i.e., A* with a blind heuristic). Note that *outcome prediction* was evaluated exclusively on the *Sepsis* log because it contains a richer set of data attributes (for clinical test results) helpful to define target data-state goals.

The preliminary outcomes validate our vision in four directions:

- Formulating PPM problems as planning tasks is highly efficient. Across all tested logs, grounding the PDDL domain and executing the state-space search took negligible time. The planner experienced zero timeouts, and when it failed to find a path, it was mainly due to the structural unsolvability of the discovered DPN.
- The situation prediction results demonstrated that the planner consistently finds the shortest valid sequence (i.e., the "*happy path*") to target situations (i.e., conjunctions

Table 1. The preliminary results reflect the three supported predictive tasks: *situation prediction* (in terms of structural reachability, grounding overhead GT, plan length difference ΔL, and search time ST), *suffix prediction* (sequence similarity to ground-truth traces), and *outcome prediction* (solvability and similarity, evaluated *only* on the Sepsis log).

Event Log	Situation				Suffix	Outcome	
	Solv. (%)	GT (s)	ΔL	ST (s)	DL Sim. (%)	Solv. (%)	DL Sim. (%)
BPI12 A	100.0	0.06	-0.67	0.00	95.21	–	–
BPI12 W	31.0	0.30	-1.27	0.00	25.95	–	–
BPI13 CP	100.0	0.01	-1.53	0.00	90.45	–	–
BPI13 I	48.8	0.02	-0.69	0.00	95.09	–	–
Sepsis Cases	100.0	0.03	-1.60	0.01	95.94	100.0	91.4

of control-flow and data conditions). While this idealized trajectory may underestimate real-world inefficiencies, it shifts decision support from forecasting suboptimal human behaviors to prescribing the most efficient and compliant interventions.

– Regarding suffix prediction, the framework achieved high normalized Damerau-Levenshtein (DL) similarity to the expected historical traces in logs from which a high-fitness DPN could be reliably discovered. However, in *BPI12 W*, the similarity dropped to $\sim 26\%$, highlighting that while DL models attempt to mimic noisy behaviors, the planner synthesizes the ideal compliant path, reporting what should happen according to the discovered rules, rather than guessing what a human might do.
– When tasked with an outcome prediction in the sepsis log, i.e., querying the planner to find plans leading to a target data-flow state, the planner achieved perfect solvability, consistently finding plans that achieved the goal state.

6 Conclusion and Research Roadmap

In this paper, we introduced a PPM framework that formulates prediction as a classical AP task by integrating process discovery and decision mining. This enables *situation predictions*: transparent forecasts that determine not only if an outcome is achievable, but the exact activity sequence required to reach it. The preliminary outcomes demonstrate that formulating PPM as an AP problem is not only conceptually sound but computationally feasible. The main advantage of our planning-based approach is *transparency*. Indeed, any resulting plan can be verified via model checking [2], ensuring trustworthy decisions in high-stakes domains like healthcare. However, realizing the full potential of this vision requires addressing the following research challenges (RCs).

RC1: Beyond the Strong Completeness Assumption. Our framework, like many PPM model-based approaches, operates under the *strong completeness assumption*, assuming that the discovered DPN captures all valid behaviors. However, real-world logs are noisy, and BP executions may deviate. If a deep learning model encounters a deviating trace, it simply predicts the next most likely event or outcome. In contrast, a planner will fail to ground the initial state if the trace cannot be replayed on the DPN.

To address this matter, we plan to integrate *conformance checking* techniques into the framework. When an ongoing trace deviates from the model, an alignment algorithm could be dynamically invoked to map the noisy prefix to the closest valid marking in the DPN, which acts as the planning initial state. This integration

enables the framework to robustly handle deviations without sacrificing the soundness guarantee that the prediction remains compliant with the encoded rules.

RC2: Temporal Prescriptions. While the current approach predicts the plan to reach a goal, time is an important missing dimension for real-world PPM applicability, as stakeholders often need to know not just *what* to do, but also *when* to do it (i.e., complementing the situation prediction with remaining time prediction [28]).

Our idea is to extend the framework from classical to *temporal* planning [8], to encode in PDDL also the duration of activities and the waiting times between them mined from the event log. A temporal planner could then synthesize schedules that not only sound sequences of events but also optimize for time, moving from predictive to actionable prescriptive insights, e.g., "*to ensure a safe discharge within 48 hours, antibiotics must be administered immediately, and a new blood test must conclude within 4 hours*".

RC3: Embedding Real Stochasticity. Our framework manages data uncertainty via "input-determinism", creating mutually exclusive, deterministic action variants for different data outcomes. While this enables classical planning and the exploration of what-if scenarios, it does not truly account for real-world BP execution stochasticity.

To tackle this, we aim to explore *probabilistic* planning (e.g., using Markov decision processes in RDDL [22]). By mining control- and data-flow probabilities, probabilistic planners produce general *policies* (rather than linear plans) that guide practitioners along the optimal path, independent of specific events or outcomes.

RC4: Operational Integration and Decision Support. We envision integrating the framework into operational BP management systems as a continuous prescriptive engine: as event streams are ingested, the planner can synthesize *intervention plans*.

Future research will explore interactive interfaces where domain experts can tweak data parameters on the fly to visualize alternative compliant paths, enabling a sound human-in-the-loop approach to decision support.

Reproducibility. The appendix, source code, and instructions for reproducibility are available at: `https://github.com/angelo-casciani/xes2plan`.

Acknowledgments. This work was supported by the Sapienza FOND-AIBPM, PRIN 2022 MOTOWN, and NRRP MUR PE0000013-FAIR projects. A. Casciani conducted this work while enrolled in the National Doctorate in AI run by Sapienza (CUP B53C23003500006, Mission 4 NRRP Generic Research Scholarship, Component 1).

References

1. Augusto, A., et al.: Automated discovery of process models from event logs: Review and benchmark. IEEE Trans. Knowl. Data Eng. **31**(4), 686–705 (2019)
2. Baier, C., Katoen, J.P.: Principles of model checking. MIT press (2008)
3. Berti, A., et al.: Pm4py: A process mining library for python. Softw. Impacts **17** (2023)
4. Bhambri, S., et al.: Interpretable traces, unexpected outcomes: Investigating the disconnect in trace-based knowledge distillation. CoRR **abs/2505.13792** (2025)
5. Ceravolo, P., Comuzzi, M., De Weerdt, et al.: Predictive process monitoring: concepts, challenges, and future research directions. Process Science **1**(1), 1–22 (2024)
6. Di Francescomarino, C., Ghidini, C.: Predictive process monitoring. In: Process Mining Handbook, pp. 320–346. Springer (2022)

7. van Dongen, B.: Bpi challenge 2012 (2012). https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f, https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204/1
8. Fox, M., Long, D.: PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. JAIR **20** (2003)
9. Helmert, M.: The fast downward planning system. J. Artif. Intell. Res. **26**, 191–246 (2006)
10. Kambhampati, S., et al.: Stop anthropomorphizing intermediate tokens as reasoning/thinking traces! CoRR **abs/2504.09762** (2025)
11. Leemans, S.J.J., et al.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM 2013 Int. Workshops (2013)
12. Mannhardt, F., Blinde, D.: Analyzing the trajectories of patients with sepsis using process mining. In: 8th Int. WS on Enter. Mod. and Inf. Sys. Arch. (EMISA). pp. 72–80 (2017)
13. Marrella, A.: Automated planning for business process management. J. Data Semant. **8**(2) (2019)
14. Marrella, A., Mecella, M.: Continuous planning for solving business process adaptivity. In: BPMDS 2025 (2011)
15. Mezini, A., Umili, E., Donadello, I., Maggi, F.M., Mancanelli, M., Patrizi, F.: Neuro-Symbolic Predictive Process Monitoring. CoRR **abs/2509.00834** (2025)
16. Pasquadibisceglie, V., Appice, A., Malerba, D.: LUPIN: A LLM Approach for Activity Suffix Prediction in Business Process Event Logs. In: ICPM'24. IEEE (2024)
17. Pednault, E.P.D.: ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In: KR'89. pp. 324–332 (1989)
18. Rama-Maneiro, E., Monteagudo-Lago, P., Vidal, J.C., Lama, M.: Encoder-decoder model for suffix prediction in predictive monitoring. arXiv:CoRR/2211.16106 (2022)
19. Reiter, R.: Knowledge in action. MIT press (2001)
20. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: ICSOC'13'. pp. 389–403 (2013)
21. Rozinat, A., van der Aalst, W.M.P.: Decision Mining in ProM. In: BPM'06 (2006)
22. Sanner, S., et al.: Relational dynamic influence diagram language (rddl): Language description. Unpublished ms. Australian National University **32**(27), 12 (2010)
23. Sibertin-Blanc, C.: High level petri nets with data structure. In: 6th European Workshop on Application and Theory of Petri nets. pp. 141–168 (1985)
24. Steeman, W.: Bpi challenge 2013, closed problems (2013). https://doi.org/10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11
25. Steeman, W.: Bpi challenge 2013, incidents (2013). https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee, https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_incidents/12693914/1
26. Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks. In: CAiSE'17. pp. 477–492. Springer (2017)
27. van der Aalst, W.M.P.: Process mining: A 360 degree overview. In: Process Mining Handbook, LNBIP, vol. 448, pp. 3–34. Springer (2022)
28. van der Aalst, W.M., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information systems **36**(2), 450–475 (2011)