



Situation-Aware eXplainability for Business Processes Enabled by Complex Events

Guy Amit, Fabiana Fournier, Lior Limonad^(✉), and Inna Skarbovsky

IBM Research – Haifa, Haifa, Israel
guy.amit@ibm.com, {fabiana,liorli,inna}@il.ibm.com

Abstract. Traditionally, the organizational IT landscape is split between Business Process engines that are developed to handle process execution workloads and Complex Event Processing engines designed to search for event correlations in real time to derive actionable insights. For the benefit of process trustworthiness, this work focuses on combining the two engines, resulting in an enriched form of a process log that serves as an input to recently developed eXplainable Artificial Intelligence frameworks, yielding more adequate explanations for process execution outcomes. A designated methodology and a test scheme were created to systematically implement and evaluate our overall approach and its effectiveness in gaining situation-aware explainability. Specifically, we demonstrate our approach using a dataset populated for an illustrative process example, replaying its trace log against the PROTON CEP engine and feeding the result as an input for the SHAP explainer.

Keywords: eXplainable Artificial Intelligence · Complex Event Processing · Business Processes Management · Situation-Aware eXplainability

1 Introduction and Motivation

Augmented Business Process Management Systems (ABPMSs) [5] are a new generation of business process management systems. The goal of ABPMSs is to empower the execution of business processes with novel AI-based capabilities. One of the main characteristics of ABPMSs is their enhanced trustworthiness, shaped by an ability to explain and reason about processes executions. Finding an adequate explanation is not easy, because it requires understanding the situational conditions in which specific decisions were made during process enactments. Frequently, explanations cannot be derived from “local” inference (i.e., current undergoing task or decision in a business process) but require reasoning about situation-wide contextual conditions relevant to the current step as derived from some actions in the past. The recent manifesto [5] calls for “Situation-Aware eXplainability (SAX)” as one of the most prominent research challenges. SAX entails ongoing tracking of reasoning assumptions and inferential associations between subsequent enactments, as a basis for providing trustful explanations.

Complex Event Processing (CEP) enables situation awareness by applying temporal and contextual reasoning on incoming events to produce higher-level of insights (‘complex events’ or ‘situations’). In this paper, we combine situations derived by a CEP engine with traces of a business process executions and show that the resulting “enriched” event log can produce better situation-aware explanations. Our contribution in this work is the augmentation of the conventional use of BPM and XAI with CEP to achieve more adequate explanations of process execution outcomes. Respectively, our solution enables to hypothesize about any plausible causal situation to be examined for its possible effect on process execution outcomes, both in real-time and in retrospect. Our method is generic and can be applied with any CEP and XAI tools. We henceforth elaborate on the underlying fundamental concepts.

2 Background

A business process (BP) is a collection of tasks that execute in a specific sequence to achieve some business goal [18]. The digital footprint that depicts a single execution of a process as a concrete sequence of activities or events is termed a ‘trace’ [1]. A multi-set of traces is usually referred to as a trace-log or event-log. We hereforth describe some basic concepts related to CEP, XAI, and replaying. **Event Stream Processing (ESP)** or CEP is computing that is performed on streaming data (sequence of events) for the purpose of stream analytics or stream data integration. ESP is typically applied to data as it arrives (data “in motion”). It enables situation awareness and near-real-time responses to threats and opportunities as they emerge, or it stores data streams for use in subsequent applications [4]. The results of ESP computation are complex events. A complex event may be derived from just a few or from millions of base (input) events from one or more event streams. Stream analytic applications provide continuous intelligence to enhance situation awareness, enable sense-and-respond behavior or just inform real-time decisions. Organizations are doing more stream processing because of the need for continuous intelligence and better situation awareness, as well as faster, more precise business decisions [15].

In our work, we use the PROactive Technology ONline¹ (PROTON) tool as our CEP engine. PROTON follows the terminology and semantics presented in [6]. Its programming model is based on the notion of an Event Processing Network (EPN). An EPN comprises a collection of event processing agents (EPAs), event producers, events, and event consumers. The network describes the flow of events originating at event producers and flowing through various event processing agents to eventually reach event consumers. An EPA is a software module that processes input events and looks for matches between these events, using an event processing pattern or some other kind of matching criterion. An event pattern is a template specifying one or more combinations of events. Given any collection of events, if it’s possible to find one or more subsets of those events that match a particular pattern, it is said as satisfying the pattern. We denote

¹ PROTON open source (Apache v2 licence): <https://github.com/ishkin/Proton>.

situations as the complex events emitted by a CEP engine. A PROTON CEP application consists of a JSON file that defines the EPN that is matched against a streaming of events in real-time to derive the defined situations.

eXplainable AI - Recent advancements in Machine-Learning (ML) [2, 11, 17] have been achieved with increase in the complexity of models that require external explanation frameworks, namely XAI. Such frameworks are predominately developed for post-hoc interpretations of ML models [2, 11]. Context-wise, they can be divided into global, local, and hybrid explanations [2, 8, 13]. Global explanations attempt to explain the ML model’s internal logic, local explanations try to explain the ML model’s prediction for a single input instance, and hybrid approaches vary (e.g., explaining the ML model’s internal logic for a subspace of the input space). This paper adds to a series of recent efforts [3, 16] that focus on exploiting XAI frameworks that are compatible with tabular data for the interpretation of BP execution results. We use process logs as the main data input and train surrogate ML models with this data to represent real-world business processes. As such, ML model faithfulness to the real BP may be lacking, capturing only parts of the holistic situations in which decisions were made. We show how with the use of CEP, the data input for the explainer could be augmented with situation relevant enrichments that result with more adequate explanations. The most commonly used ML-model-agnostic post-hoc local XAI frameworks, compatible with tabular data, are LIME [14] and SHAP [10]. Both rely on sampling data points by way of feature perturbations for derivation of feature importance around the examined sample. To assess the effectiveness of our method, we used SHAP, mostly due to its ability to accommodate inter-feature dependencies. The approach for process explainability is based on the training of a decision-tree to associate process execution variables with process outcomes. Such training uses historical process execution logs that are enriched by the CEP. An XAI tool is then employed to explain individual process execution cases by ranking the importance of the process variables with respect to specific outcomes of interest.

Replaying Process Logs - The terms ‘play in’, ‘play out’ and ‘replay’ were originally used in the work of Harel [9], and later adopted by van der Aalst [1]. ‘Play out’ typically relates to conventional use of a BPM engine, and ‘play in’ refers to the core notion of process mining. Most relevant to our work, the notion of process ‘replay’ combines the process event-log and the model as input for purposes such as checking for execution conformance, predictions, and diagnostics. In the context of process explainability, replay describes best the operation in which historical traces are replayed for the elicitation of richer, situation relevant events that may have previously promoted to internal process decisions but have not been originally persisted by the BPM engine. The replay module enables simulating the process execution as input to the CEP engine as if events were occurring in real-time. More concisely, the events in the process log are streamlined according to the original BP model sequencing as an input to the CEP engine, having the latter derive situations that enrich the original event log as an input to the explainer. This could be employed either in real-time or in a simulated mode, for better explainability derived in retrospect.

3 Methodology

3.1 Types of Events

We identify three types of situations as a function of the source of the events:

- **Events source is internal to the BP.** The situations are a combination of process execution events along with the CEP defined patterns. These situations are tightly coupled with the BP execution.
- **Events source is external to the BP.** These situations are not coupled with the BP execution. We note that although the event is external to the BP it might influence its execution and the outcomes.
- **Events source is a combination of internal and external circumstances.** These situations are bi-coupled to external as well as internal to the BP and are typically characterized by transient/ad-hoc events. The BP model does not change but the (temporal) alteration of its flow affects the possible explanation given to a specific execution instance.

3.2 Approach

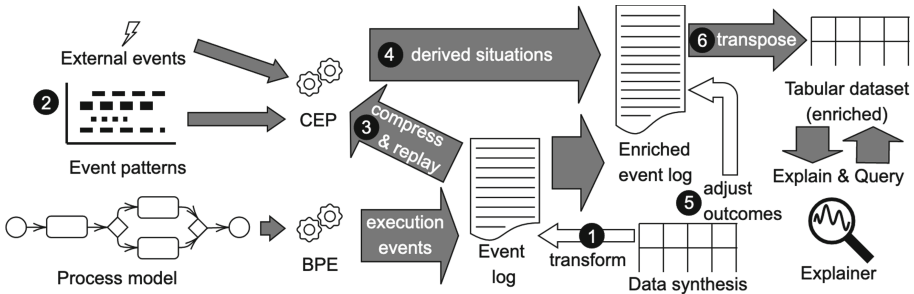


Fig. 1. Sequence of our approach: (1) transform, (2) implement, (3) compress & replay, (4) run CEP, (5) adjust, and (6) transform.

Our methodology includes the following steps (Fig. 1):

1. **Transformation of the tabular dataset into an event log** - The tabular dataset contains rows indicating different states in the BP for each case, with task execution and completion times denoted respectively. Given these times, each row is transformed into a series of timestamped events, where each event is assigned the original case-id and time corresponding to the original row.
2. **Implementation of the CEP application** - This stage unfolds the definition of the event patterns and situations to be detected by the CEP engine. In the case of PROTON, this CEP application specification is implemented as a JSON file consisting of the EPN for the specific application.
3. **Compression of time windows for expedited replay of the event log** - CEP applications are meant to run and trigger situations in real-time. However, when testing a CEP application, we cannot “wait” for events to

happen at their original occurrence times and therefore aim at replaying the events in shorter time windows. For this, we apply PROTON’s simulation tool called Proton EventT Injection and Time comprESSION (see footnote 1) (PETITE) that compresses the original times into shorter intervals. The result is a compressed event log that serves as input file for the CEP application and a new JSON file with “compressed” times.

4. **Running of the CEP application** - Replay of the event log resulted from step 3 as input to the CEP engine. The outcome of this execution is an “enriched” event log, containing the original events interwoven with the situations detected by the CEP engine (timed events).
5. **Adjustment for situation effects** - According to each situation, decision variables (e.g., acceptance) are modified and post-situation events are trimmed from the enriched file.
6. **Transformation of the enriched log into an enriched tabular input for the explainer** - The enriched log is transformed back to the original table format, where additional columns are added that represent features of the new events and situations discovered.

Steps 1–6 apply when replaying the log in retrospect. In the case of running in real-time, only steps 2, 4, and 6 are required. Steps 1 and 5 are strictly associated with the case of data synthesis in which data is generated for testing purposes.

4 Illustrative Example

Figure 2 depicts a loan application model using BPMN [7] notation. Each loan application goes through a set of predefined set of activities (e.g., verify amount and credit check) and decisions junctions (e.g., amount \geq 1000) resulting in either acceptance or rejection of the loan application. For simplicity, we use a process example ending with a binary decision. However, our approach is applicable for any multi-class decision outcome explainability, occurring at any point during process execution. Instantiation of the process can be encoded in a tabular form as depicted in Table 1. Each process instance includes a column for its initiation time, for each decision variable (e.g., amount), and for each task, encoded in a Boolean column to depict whether it was executed or not and a second column denoting its completion time in minutes.

Table 1. An example of an instance in the tabular dataset

case ID	credit_score	risk	done_receive_loan	done_verify_ammount	done_credit_check	done_risk_assessment	done_skilled_agent	done_novice_agent	done.accept
TVQR	1173.08	397.78	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE
...									
arrival_time	post_received_time	post_verify_amount_time	post_credit_check_time	post_risk_assessment_time	post_skilled_agent_review_time	post_novice_agent_review_time	post_decision_time		
48	54	59	71	NaN	87	NaN	92		

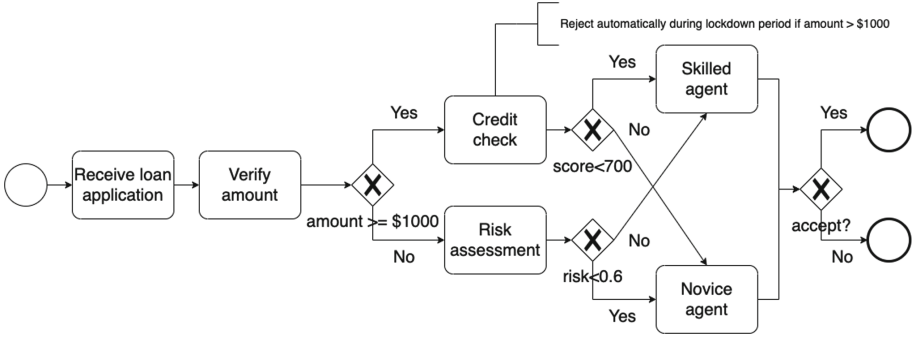


Fig. 2. Loan application BP model in BPMN, with the potential deviation during lockdown informally denoted by an annotation.

4.1 Applying the Methodology in Our Example

Henceforth, we exemplify the above methodology in our illustrative example:

1. **Transformation of the tabular dataset into an event log.** For example, for a case ID=TVQR (Table 1) with `done_credit_check = TRUE` and `post_credit_check = 71` (mins), an event `CreditCheck` with “case ID = TVQR, ... , OccurrenceTime = timestamp of streaming initiation + 71 mins” was added to the event log. Similarly, events with the same case ID were added for: `Arrival`, `Received`, `VerifyAmount`, `SkilledAgentReview`, and `Decision`.
2. **Implementation of the CEP application** - Let’s assume we would like to derive the following complex events or situations:
 - Derive a situation *AgentOverflow* that causes the rejection of new loan applications when more than 4 applications pile-up on an agent’s table. The rationale is that when there is a large workload, the tendency is to reject to speed up the process while “being on the safe side”.
 - Derive a situation *DecisionTimeMoreThanTwoDays* that determines an application as rejected if it stays in the system more than 2 days. Again, the rationale is that want to eliminate bottlenecks while not taking the risk of accepting a somewhat risky application.
 - Derive a situation *LockDownNewGuideline* that concludes an application as rejected if it was submitted during Covid-19 lock-down period with an amount greater than \$1000.

We also have two input events that are part of the *LockDownNewGuideline* situation: *LockDownInitiator* and *LockDownTerminator* that are employed by the CEP engine to denote the beginning and end of a lockdown period. The *AgentOverflow* and *DecisionTimeMoreThanTwoDays* situations are examples of situations in which the event source is the BP in hand as they are derived only as a result of BP instance’s state. On the other hand, the *LockDownNewGuideline* situation is an example of a situation in which the event sources are a combination of internal (e.g., the amount of the requested loan)

and external to the BP (e.g., the lockdown). An EPN with the event pattern definitions in a JSON file specifying these three situations was created.

3. **Compression of time windows for expedited replay of the event log.** The event log produced in step 1 spans four days, the same as the time horizon of the original dataset. By applying the PETITE simulator, we compressed the 4 days elapsed time into 5 min.
4. **Running of the CEP application** - PROTON was applied using the JSON file and the event log resulting from step 3. The output is an enriched log that includes the input events as well as the situations derived by the CEP engine.
5. **Adjustment for situation effects** - We tweaked the log from ‘acceptance’ to ‘rejection’ where the situation was affecting the process result. Trimming was not required here, since all situations related to final process outcomes.
6. **Transformation of the enriched log to tabular input for the explainer** - The additional columns contain new features with time in min from first instance initiation. For example, in the case of the detection of the DecisionTimeMoreThanTwoDays situation, a non-null value representing the occurrence time is added to the TwoDaysWithoutDecision column. Concretely, the value of these new features is either NaN to denote the non-occurrence of the situation, or a value depicting the time from process initiation. In our case, we further decoded the enrichment into a Boolean, translating a NaN value to ‘false’ and a time value to ‘True’.

5 Evaluation

Conforming to the tabular form in Table 1, we populated a dataset with 1000 instances with decision variables drawn at random from Gaussian distributions and completion times drawn from Poisson distributions. Task execution columns were computed based on the values of the decision variables as entailed from process flow logic. As an input for training of the explainer ML model (i.e., SKLearn’s [12] decision tree classifier), we split the data at random into training (800 records) and test (200 records) sets. Our set of test cases for each of the three situations included the following variations, where for each situation 4–5 instances were instantiated:

- T1 Loan acceptance toggled into rejection - a set of test instances in which prior to the enrichment, the original process execution decision was loan ‘acceptance’, and where the decision was altered into loan ‘rejection’ as a result of the newly incurred situation. For any of these test instances, our expectation was to have the explainer identify the corresponding enriched situation manifest itself as a top importance feature in the explanation.
- T2 Rejection overridden by a ‘new’ rejection reason - a set of test instances in which the original process execution decision was loan ‘rejection’, and for which the decision remained a ‘rejection’ as a result of the newly incurred situation, and where the new situation becomes the prominent reason for the rejection instead of the original one. For any of these test instances, our expectation was to have the explainer identify the corresponding situation manifest itself as a top importance feature in the explanation.

- T3 Rejection that persists itself - specific to the *LockDown* situation, some of its corresponding test instances reflected process execution scenarios in which the original decision to reject the loan was not affected by the occurrence of the *Lockdown* situation. For these particular instances, although the result was a ‘rejection’, we did not expect the explainer to identify the corresponding situation manifest itself as a top importance feature in the explanation.
- T4 Contrasting set - with respect to each type of enriched situation and our overall dataset, we identified the subset of instances in which the situation did not apply. Our examination here was to ensure that none of these instances happens to be incorrectly explained by an enriched situation.

The loan rejection toggled into acceptance and loan acceptance that remains acceptance variations were dropped since they are symmetric to the above.

We present here the global model explanation of the enrichment, followed by detailed examination of the local model explanations with respect to the test cases, applied to corresponding situation instances. Our code is available at: <https://github.com/IBM/SAX/tree/main/AI4BPM>.

6 Results

Global distribution of feature importance is illustrated in Table 2. The newly added features play a role in about 5% of the explanations, reflective of the proportion of instances that were tweaked to test for the effects of the newly introduced situations. Furthermore, model accuracy in predicting eventual process outcomes has also been fully mitigated by the enrichment.

Table 2. Global explainability with and without situational enrichment

Without enrichment	With enrichment
Model accuracy (DT): 0.97	Model accuracy (DT): 1.00
0. amount : 0.4826136779898476	0. risk : 0.7714530282992648
1. risk : 0.3426102769869956	1. credit_score : 0.15177506322470122
2. credit_score : 0.17477604502315672	2. AgentOverflow : 0.04241708399719542
	3. LockDownTerminated : 0.019477216063815347
	4. TwoDaysWithoutDecision : 0.014877608415023355

For each of the enriched situations, we elaborate in a corresponding table example cases that highlight test scenario results. Each table includes local feature importance in a descending order for test cases T1–T3 without and with the enrichment, and a collective force-plot with all non incurred situation instances that are graphically stacked horizontally corresponding to test case T4.

AgentOverflow situation: A sample of local test results for this situation is shown in Table 3. With respect to [T1], case id ‘FSAN’ was manually tweaked from originally being accepted into being rejected due to overflow. As listed, prior to the

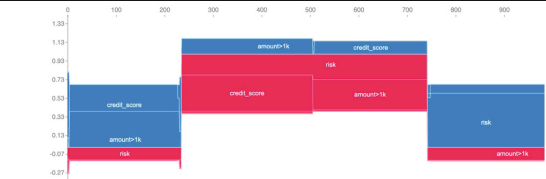
enrichment, the original explanation incorrectly included ‘risk’ as the top importance feature in the explanation for the rejection, while after the enrichment, the most important feature was correctly recognized as ‘agent-overflow’.

With respect to [T2], case id ‘BROG’, which originally concluded with a loan rejection decision, was modified in the dataset to still be rejected, but in its modified form was tweaked to ensure the reason for the rejection was an overflow situation. As listed, the original incorrect explanation with ‘amount’ as the top feature has been properly replaced by ‘agent-overflow’ as the top importance feature in the enriched explanation.

[T3] doesn’t apply in the case of an agent overflow situation. This is since whenever an overflow event occurs, it is inevitable for the decision to entail an immediate rejection, overriding whatever was the original reason for the rejection, as already covered by [T2]. An exception for a situation that is relevant for test case [T3] is elaborated in the context of the Lockdown Guidelines situation.

With respect to [T4], we examined all contrasting instances associated with the non occurrence of an overflow situation to ensure none happens to incorrectly include in its explanation the ‘agent-overflow’ feature. Horizontal stacking with all relevant instances was plotted as illustrated. As desired, no instance in this set was identified to include the ‘agent-overflow’ feature in its explanation.

Table 3. AgentOverflow situation - test results

Test case	Case ID	Non enriched	Enriched
T1: ‘accept’ to ‘reject’	FSAN	risk-, credit-score-, amount+	✓ agent-overflow-, risk+, credit-score-
T2: overridden ‘reject’	BROG	amount-, credit-score-, risk+	✓ agent-overflow-, credit-score-, risk-
T3: persisted ‘reject’	na		
T4: contrasting set			
		✓ An example plot we created to verify no instance in the contrasting set includes ‘agent-overflow’ in its explanation.	

Decision Time More Than Two Days Situation: As with the previous agent-overflow situation, local test results for the situation of decision time greater than two days are shown in Table 4. Case id ‘YMOJ’ was tweaked to match the scenario of [T1]. Correspondingly, the top importance feature correctly changed from ‘risk’ to ‘two-days-without-decision’ as a result of the enrichment. Adjustment of case id ‘MJKQ’ for test [T2] resulted with proper alteration from ‘credit-score’ to ‘two-days-without-decision’ as desired. [T3] was skipped for the same reason as above. [T4] presented no undesired side effects in the explanation of any of the contrasting set instances for the situation inspected.

Table 4. Decision time > 2 days situation - test results

Test case	Case ID	Non enriched	Enriched
T1: ‘accept’ to ‘reject’	YMOJ	risk-, credit-score-, amount+	✓ two-days-without-decision- , risk+, credit-score-, agent-overflow+
T2: overridden ‘reject’	MJKQ	credit-score-, amount-, risk+	✓ two-days-without-decision- , is-credit-, credit-score-, risk+, agent-overflow+
T3: persisted ‘reject’	na		
T4: contrasting set	✓ No instance resulted with ‘two-days-without-decision’ in its explanation		

Lockdown Guideline Situation: We repeated the same tests for the Lockdown guideline situation as shown in Table 5. As with the two other situations, [T1] demonstrated a remedy in the explanation due to the enrichment for case id ‘YMOJ’. With respect to [T2], the complexity of the situation lets us examine two event encoding nuances: implicit and explicit encoding of the guideline as an interaction between the external occurrence of a lockdown and loan amount being greater than \$1000. For the former implicit case, only ‘LockdownInitiated’ and ‘LockdownTerminated’ events were included in the enrichment examination. For technical simplicity, both events were transformed in the dataset from their original timestamped form into a single Boolean variable that was either ‘true’ in any instance that occurred between the two timestamps, or otherwise was ‘false’ (named ‘lockdown-terminated’). Our aim was to test the ability of the explainer to reveal interactions among features. In our case, between the process external occurrence of a lockdown and the process internal amount being greater than \$1000. First run of [T2] failed to recognize ‘amount’ and ‘lockdown-terminated’ as mutual top-importance features in the explanation. We realized that the explainer might be incapable of handling an interaction between a numeric variable (i.e., ‘amount’) and a boolean variable (‘lockdown-terminated’), and particularly discovering the conditional split of amount around \$1000. To test this, we added another Boolean variable ‘amount>1K’ to explicitly denote this split and re run [T2]. As illustrated in Table 5, with the additional variable, the SHAP explainer was able to correctly recognize the interaction as the top two features.

For the latter explicit case, a ‘LockdownAndLargeAmount’ event was derived by the CEP engine and was explicitly added as a Boolean variable in the dataset to mark corresponding instances. As with the previous two situations, SHAP was able to recognize such an explicit encoding as a top importance feature.

The uniqueness of the Lockdown guideline also allowed us to instantiate the scenario of [T3] in which the external occurring of a lockdown may not force a rejection i.e., when the amount is smaller than \$1000, reflecting a situation where the original reason for the loan rejection should persist. As demonstrated by case id ‘LJLP’ in the results, in such a case, the ‘lockdown-terminated’ feature was recognized as part of the explanation, but not as a top importance feature, and also ‘amount>1K’ was detected as affecting the model towards loan acceptance.

Table 5. Lockdown guideline situation - test results

Test case	Case ID	Non enriched	Enriched
T1: 'accept' to 'reject'	YVDN	amount-, risk+, credit-score-	✓ lockdown-and-large-amount- , risk+, credit-score-, agent-overflow+
T2: overridden 'reject'	NDZY	credit-score-, amount-, risk+	✓ amount>1K- , lockdown-terminated- , credit-score-, risk+, agent-overflow+
T3: persisted 'reject'	LJLP	risk-, amount+, credit-score-	risk-, ✓ amount>1K+ , lockdown-terminated- , credit-score-, agent-overflow+
T4: contrasting set	✓ No instance resulted with 'lockdown-and-large-amount' in its explanation		

Collective Summary of Results: A handful of cases was adjusted to repeat the tests in each situation. An overall summary of all test results is listed in Table 6. Our results conclusively show that in all three situations, the enrichment of the dataset promoted to a perfect feature correctness in the explanations, without any loss in the correctness of all other instances that were not affected by the newly incurred situations. For the first two situations, the enrichment also promoted the ML model accuracy in predicting process decisions.

Table 6. Summary of test cases: # - num of test cases; T-EXP - % of feature correctness in explanation; ACC - % of correct process result predictions; F-EXP - % of affected cases with false explanation. Noteworthy improvements marked in bold.

Test:	T1: 'accept' to 'reject'					T2: 'reject' overridden					T3: 'reject' persisted					T4: affected
	Not enriched		Enriched			Not enriched		Enriched			Not enriched		Enriched			Enriched
Situation	#	T-EXP	ACC	T-EXP	ACC	#	T-EXP	ACC	T-EXP	ACC	#	T-EXP	ACC	T-EXP	ACC	F-EXP
Agent- overflow	5	0%	60%	100%	100%	5	0%	100%	100%	100%	na	na	na	na	na	0%
Decision > 2 days	4	0%	50%	100%	100%	4	0%	100%	100%	100%	na	na	na	na	na	0%
Lockdown guideline	4	0%	100%	100%	100%	4	0%	100%	100%	100%	6	100%	100%	100%	100%	0%

7 Conclusions and Future Research

The enrichment of process events logs with situations derived by a CEP engine, demonstrates that temporal contextual information can be leveraged to improve the adequacy of explanations given for process execution instances. As also demonstrated, with some tweaking of process outcomes, such enrichment can also be employed in retrospect. The effect of the enrichment manifests itself not just in properly adjusting the importance of factors that correctly correspond to the outcome, but also promotes the accuracy of the surrogate ML model.

Our contribution is not only in showing the effect of situational enrichment in attaining better explainability, but also in providing a core taxonomy for the different types of situational events, an overall methodological approach on how to realize the enrichment, and a corresponding test scheme. All these elements have been instantiated with respect to the illustrative example using PROTON and

SHAP, highlighting some concrete caveats, such as the need to include Boolean features for meaningful partitioning of quantitative ones to benefit from SHAP’s capacity to identify feature inter-dependencies. Tasks duration as originally captured in timestamps may also unveil impactful interpretations that we haven’t considered in this work, but could be an interesting direction to explore next.

The transposing of a process log into a tabular representation in which all process execution attributes are flattened into a single immutable row may be criticized for having a naive view of a process as a single processing step. We acknowledge that, in reality, a process can better be seen as a sequence of subsequent real-time choices with casual relationships among its steps, some derived from an “in-flight” incomplete view about overall process state. Our approach can also be applied with respect to any sub-partitioning over the set of process variables and with respect to any intermediary execution result as the target variable for the explanation.

We foresee two fundamental directions that remain open for future work. First is designing for an even tighter integration between the XAI framework and the BP engine, one in which insights inferred by the former serve as feedback for the conditional unfolding of the process execution in real-time. This may be attained by extending process flow logic with execution decisions that rely on the evolving ‘explainable state’ of the process’ instance.

A second and probably the most challenging direction is the development of new aids to infer concrete situational enrichment that may be missing in a given process. Reduced ML model accuracy may denote that some situational condition is still missing. However, to date, it is mainly the responsibility of a domain expert to specify such situations. Future work may also attend to the effort of automatic identification of which concrete situations may be missing.

References

1. van der Aalst, W.: *Process Mining: Data Science in Action*. Springer, Heidelberg (2016). <https://doi.org/10.1007/978-3-662-49851-4>
2. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE Access* **6**, 52138–52160 (2018)
3. Amit, G., Fournier, F., Gur, S., Limonad, L.: Model-informed LIME extension for business process explainability. In: *PMAI@IJCAI*. Vienna (2022)
4. Benoit, L., et al.: Hype cycle for the internet of things (2021)
5. Dumas, M., et al.: Augmented business process management systems: a research manifesto. *arXiv preprint arXiv:2201.12855* (2022)
6. Etzion, O., Niblett, P.: *Event Processing in Action*. Manning (2010)
7. Grosskopf, A., Decker, G., Weske, M.: *The Process: Business Process Modeling Using BPMN*. Meghan-Kiffer Press (2009)
8. Guidotti, R., et al.: A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**(5), 1–42 (2018)
9. Harel, D., Marelly, R.: *Come, Let’s Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, Heidelberg (2003). <https://doi.org/10.1007/978-3-642-19029-2>

10. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **30** (2017)
11. Meske, C., Bunde, E., Schneider, J., Gersch, M.: Explainable artificial intelligence: objectives, stakeholders, and future research opportunities. *Inf. Syst. Manag.* **39**(1), 53–63 (2022)
12. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011)
13. Rehse, Jana-Rebecca., Mehdiyev, Nijat, Fettke, Peter: Towards explainable process predictions for Industry 4.0 in the DFKI-smart-Lego-factory. *KI - Künstliche Intelligenz* **33**(2), 181–187 (2019). <https://doi.org/10.1007/s13218-019-00586-1>
14. Ribeiro, M.T., Singh, S., Guestrin, C.: “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*, pp. 1135–1144 (2016)
15. Schulte, W.R., et al.: Market guide for event stream processing (2022)
16. Upadhyay, S., Isahagian, V., Muthusamy, V., Rizk, Y.: Extending LIME for business process automation. *arXiv preprint arXiv:2108.04371* (2021)
17. Verma, S., Lahiri, A., Dickerson, J.P., Lee, S.I.: Pitfalls of explainable ML: an industry perspective. *arXiv preprint arXiv:2106.07758* (2021)
18. Weske, M.: *Business process management architectures*. In: *Business Process Management*. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-662-59432-2_8