Towards Business Process Observability

Avirup Saha
IBM Research
India
avirup.saha2@ibm.com

Prerna Agarwal
IBM Research
India
preragar@in.ibm.com

Sambit Ghosh
IBM Research
India
sambit.ghosh@ibm.com

Neelamadhav Gantayat
IBM Research
India
neelamadhav@in.ibm.com

Renuka Sindhgatta
IBM Research
India
renuka.sr@ibm.com

ABSTRACT

The recent move towards migrating business processes in the cloud often requires organisations to have some parts of their business processes execute on cloud-native systems and others on the organizations' own infrastructure. In many scenarios, there could be multi-cloud configurations with parts of the process run by different cloud providers. Hence, current-day business process executions rely on complex software ecosystems with high levels of cross-dependencies, heterogeneity, and redundancies. This growing complexity amplifies the need to monitor and maintain business process executions near real time to ensure the key performance indicators are met. Instances of system failure require early detection and diagnosis of the problem demanding minimal time to understand its impact on the process execution. We propose the notion of business process observability where the intention is to leverage parameters amenable to external monitoring, to adequately represent the state of a business process. The objective of the proposed approach is to offer an end-to-end observability of business processes provisioned in distributed environments that allows for (1) detection, (2) diagnosis, and (3) actions to address business process execution failures. The paper proposes a solution approach that uses observability data to build a cross-layer topology linking a business process and its underlying software ecosystem and provide detailed diagnostics correlating the business process and software execution failures.

CCS CONCEPTS

• **Applied computing → Business-IT alignment**; *Enterprise computing infrastructures*; *Business process monitoring*.

KEYWORDS

Business Process Observability, Key Performance Indicators, Process Monitoring, Unified Topology

CODS-COMAD 2024, January 04–07, 2024, Bangalore, India
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1634-8/24/01.
https://doi.org/10.1145/3632410.3632435

ACM Reference Format:
Avirup Saha, Prerna Agarwal, Sambit Ghosh, Neelamadhav Gantayat, and Renuka Sindhgatta. 2024. Towards Business Process Observability. In *7th Joint International Conference on Data Science & Management of Data (11th ACM IKDD CODS and 29th COMAD) (CODS-COMAD 2024), January 04–07, 2024, Bangalore, India.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3632410.3632435

1 INTRODUCTION

Today's enterprises are increasingly migrating and provisioning their business processes on the cloud, with parts of the process running on infrastructures and services that might come from different cloud providers, and many other parts running on their premises. As business processes undergo constant changes through transformations, they translate to changes in underlying software environments leading to higher levels of heterogeneity and complexity. Given the increased distribution of the process execution environment, it is a challenge to manage the underlying systems, identify any failures or varying loads, and predict the process execution behavior due to failures.

In order to ensure the quality of service of applications in cloud-native systems, *application observability* has emerged as a critical requirement [19]. The idea of observability comes from control theory and improves visibility into understanding the behavior of a system utilizing telemetry gathered from the system while it is running. Observability provides deeper contextual insight about the system in addition to straightforward *black-box monitoring* that provides an overview of the health of the system [12]. While there has been widespread adoption of application observability, the insights and understanding do not propagate to process monitoring. For example, consider a real-world scenario of a large manufacturer having the customer order management process enabled by multiple applications: the order is created through a hand-held device, the orders are validated by an order processing system on a cloud, and additional steps of sales order creation, shipping of order, and processing of invoice are executed by an enterprise resource planning system (ERP). An outage on the cloud causes the order processing system to fail. While application observability enables the IT team to monitor and detect the failure in the infrastructure, a business owner, using state-of-the-art business process monitoring tools, has no insight into the reason for observing a significant decrease in the creation of orders. Lack of understanding often results in significant cost due to inadequate or delayed actions.

We reason that for improved business process operations in complex, heterogeneous, and distributed environments, business owners should be able to (i) detect, (ii) diagnose, and (iii) react to anomalous events or failures. Application observability encompasses monitoring information that captures system health and querying the information about systems' behavior. Correspondingly, *Business Process Observability* [17] will provide insights into the correctness and performance of the process executions with a goal of minimizing time to insight by capturing the information required at real-time. We propose a solution approach to Business Process Observability by first **detecting** the anomalous events impacting the business process. Here, the design and management of business process telemetry capturing all the information is crucial. Next, the **diagnosis** of the anomalous events is realized by localizing the source of the event and predicting the impact on the business process, using the collected data. Finally, the predicted impact allows for **prioritizing the actions** and ensures there is an optimal set of actions for restoring the system. Hence, we make the following contributions:

- present and analyze the characteristics of the types of data that can capture all the information about the process and the underlying system behavior.
- predict the impact of the anomalous events based on the data captured to prioritize the resolution of the faults.

The paper is organised as follows: A brief overview of previous studies and the reason for process observability is presented in Section 2. The details of our approach to building Business Process Observability are presented in Section 3. The evaluation of the approach and discussions on real-world event logs are presented in Section 4. Finally, we summarise the contributions of our work and outline future work (Section 5).

2 BACKGROUND

2.1 Related Work

Process monitoring is one of the crucial phases of a business process management life cycle [9] where event data from the executing process is used to analyze the process performance, detect bottlenecks, and identify deviations. Predictive process monitoring has emerged as a sub-field that forecasts the behavior of an ongoing business process case such as its outcome, completion time and sequence of future activities [10]. Predicting the outcome of the process execution can allow process owners to take relevant measures and prevent undesired outcomes. Many machine learning techniques have been applied for predicting process behavior [6, 21, 23]. Recent studies have also focused on prescriptive process monitoring techniques to recommend actions during the execution of a process in order to optimize its performance [13]. The monitoring approaches currently limit the analysis to process execution data. In this work, we also consider the underlying software environment that enables the process executions and aim to provide near real-time insights by considering additional data from process and system environments.

Application observability has gained importance with the application going through several changes with distributed deployment through containerisation and use of microservices architectures [19]. Recent work emphasises the need to design and manage large volumes of the telemetry data to support application observability in complex environments [12, 18]. Supporting multi-level observability by collecting telemetry from both the application and the infrastructure levels has been investigated for optimal allocation of infrastructure resources [15]. In this work, we focus on providing multi-level observability by considering business process and application telemetry.

2.2 Preliminaries

The fundamental goal of observability is timely insight into the state of business process executions. The basic premise is to collect data that provides these insights [19]. Currently, IT systems are instrumented to support application observability by emitting four types of data: (i) numeric data, (ii) structured information, (iii) unstructured logs that developers instrument, and (iv) a sequence of the execution path of a request. These are referred to as Metrics, Events, Logs, and Traces (MELT). We first discuss the data and its characteristics in the context of a business process:

- *Metrics:* Numeric measurements of the business process that indicate performance or availability at a specific point in time. Hence, a metric would include a numeric value, the timestamp when the metric was computed, and additional meta-data. Metrics are used in two ways: i) generate alerts on unexpected values representing the state, or ii) analytics currently used for visualization on dashboards. Business key performance indicators (KPIs) representing the health of the process and counts of activities completed that are captured by process mining tools are common examples of metrics. In the application domain, a common set of metrics computed by application monitoring tools are referred to as *Golden signals*: latency, error, saturation, and traffic [5]. These signals help understand the application performance, the current demand, problems or errors, and the capacity of the application.
- *Events:* They are structured data emitted at run time. Events have a finite set of possible values [12]. In the context of business processes, business events have been well defined. An event, as defined in one of the related business process management literature [23], is a tuple $(a, c, t, (d_1, v_1), (d_2, v_2) \ldots (d_m, v_m))$, where a is the activity name, c is the case it belongs to, t typically indicates the timestamp of completion of the activity, and $((d_1, v_1), (d_2, v_2) \ldots (d_m, v_m))$ is the set of attributes and their values that are defined by a domain expert. In application observability, events may be considered as a structured type of log [19]. An event can be defined as a tuple (e, s, t_s, t_e) containing the event identifier e, the infrastructure, platform, or application component s the event emanates from, the start of the event t_s, and the end of the event t_e. The application event can have additional information such as the function or operation of the service being called or the type of event.
- *Logs:* They refer to the collection of semi-structured or unstructured strings providing run-time information. They contain highly granular information with rich context of executions. Logs are often used to identify any unexpected

behaviour. In the application context, it often refers to developers emitting a stack trace or any information that is suitable to provide the status of the application. In business processes, (activity) logs are additional unstructured information such as a mail exchange or conversation summary stored along with the activity [22]. While logs are extensively used for localizing and understanding unexpected behavior in application monitoring, they are less commonly used in the context of process monitoring. *Event logs* that are a collection of traces are used in business process monitoring [23].

- *Traces:* Represent a series of events encoding the end-to-end execution path. Although tracing in application observability has focused on distributed systems in a microservices environment, the concept of a trace is widely prevalent in business process monitoring. A trace is a sequence of events that belong to a case $C = \{e_1, e_2, \ldots, e_n\}$, where $\forall i, j \in [1 \ldots n]$ $e_i.c = e_j.c$. Understanding the entire case life cycle makes it possible to pinpoint the source of bottlenecks such as increased latency or resource utilization.

3 APPROACH: BUSINESS PROCESS OBSERVABILITY

We outline the approach to Business Process Observability in Figure 1. The approach has two phases: the offline phase (or the train phase) where the historical MELT data is used to arrive at trained models and the online phase where the inference is done at run time with the MELT data to detect and diagnose faults in the process execution. The first step in the offline phase uses the traces captured by application and business process monitoring tools as input to connect the components involved in the application with the business process activities. In the microservices architecture, components refer to services (hence, used interchangeably). We call this cross-layer topology a *unified topology*. The unified topology would be useful to identify the business process activities that are linked to, and hence impacted by, a faulty service or an infrastructure component. Next, we train a suffix prediction model using the business process traces that would take in a partially running case as input and predict the future sequence of activities. Further, the metrics and events from the process and application are transformed to a time series. The time-series data is used to discover correlations between application metrics/events and business metrics/events. Finally, a time series forecasting model is trained to support forecasting of business metrics and KPIs based on the application metrics and events.

The online phase at run time has the application events and metrics which capture an anomaly or a failure in an application component. The unified topology is used to determine the business activities that rely on the application component for execution. These activities are marked as ‘impacted activities’. We refer to *impact* as business activity or application system component determined to be affected, and therefore worthy of inspection [3]. The suffix prediction model takes all the partially executing business traces and predicts the suffixes, allowing for identifying the cases that are likely to have the ‘impacted activity’ in their suffixes. The time series metric correlation model identifies the business KPIs that are impacted due to an application event or a metric change. The time series KPI forecasting model quantifies the impact by forecasting the change in business KPIs and metrics. The diagnosis of the impacted cases, the impacted KPIs, and forecasted KPIs enables assessment and prioritization of the impact of an application event or metric change.

3.1 Unified Topology of Business Process and Applications

The traces emitted from business process monitoring and application monitoring are used to discover a *Unified Topology.* This unified topology enables the mapping of the business process activities and application services that enable the execution of the activities. We use temporal correlation based on Apriori algorithm [2] to discover the cross-layer topological mapping and the motivation is as follows: *if a business event is a result of one or more application events, then they should exist in the temporal proximity of that business event.* An event is associated with either a business activity or an application service. The idea is to identify the frequency based weighted mapping of application services and business process activities. The historical event log containing traces is used to discover this weighted mapping of business process activities and application services as follows:

(1) The timestamp information of each event in a business process trace is first used to mine the correlated application traces using temporal proximity. The temporal proximity is determined with the help of domain knowledge and could constitute certain factors such as latency to record these traces from applications and business processes in a database.
(2) During discovery of the topological mapping, a single event of a business process activity is insufficient to determine the correlated application traces. Therefore, all events corresponding to a business process activity are used to filter the set of application traces in temporal proximity and to obtain the set of application services invoked in these traces (itemsets).
Formally, let n be the number of business process events corresponding to a business process activity a in the event log and let $\sigma_e = \{s_1, s_2, ..., s_m\}$ be the set of application services corresponding to the application events in the temporal proximity of business event e. The overall set of services $\Sigma_a = \bigcup_{i=1}^{n} \sigma_{e_i}$ is computed as the union of σ_{e_i} for all n business process events e_i corresponding to a business process activity a.
(3) To determine the weight $w_{s_i a}$ of the temporal correlation between application service $s_i \in \Sigma_a$ and business activity a, the frequency $f_{s_i a}$ of application service s_i in the temporal proximity of the business process activity a is used to compute the fraction of business process events in whose σ set, the application service s_i is present (frequency of each item in the itemset), i.e.,
$$w_{s_i a} = \frac{f_{s_i a}}{\sum_{i=1}^{m} f_{s_i a}}$$
(4) Let the set of business process activities be $A = \{a_1, a_2, ..., a_k\}$. The above steps are followed for each

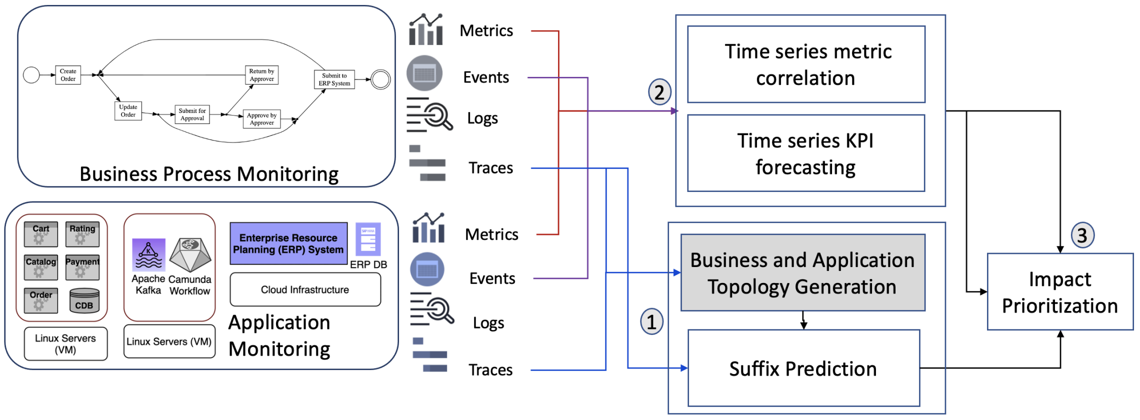

Figure 1: Overall approach towards observability for business processes

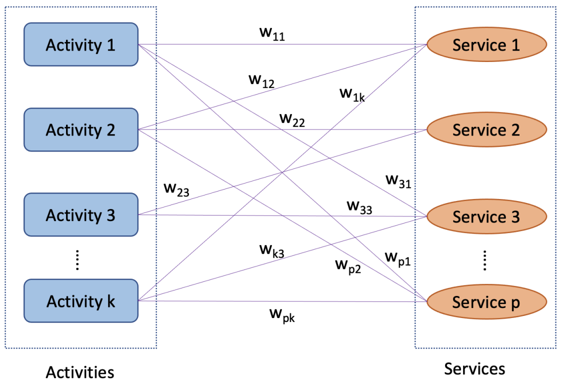

Figure 2: Business and Application Unified Topology Structure

business process activity $a_i \in A$ to obtain Σ_{a_i} and hence the weighted topological mapping.

The topological map will be used to identify the set of business process activities that are linked to an application service at runtime. As shown in the Figure 2, for the unified topology U, nodes V represent the business process activities (on the left) and application services (on the right). The edges E depict the weights of the topological map learned from the historical traces depicting the relevance of each cross-layer mapping in the unified topology. Hence, Business and Application Unified Topology U is a weighted un-directed graph i.e., $U = G(V, E)$. Given a failure event and the application service s_f associated with it, the unified topology can be used to identify the activities linked to the service (i.e. which depend on the service for their execution). The linkage can be determined by considering the weight of the edge (higher than a given threshold) connecting the service and the activity. The set of activities A_f linked to the service s_f are said to be *impacted* by the failure event.

3.2 Suffix prediction

Given a prefix, or a partly-complete trace, suffix prediction aims to predict the most likely continuation of that prefix. The goal is to identify cases that have activities $a \in A_f$ (or the impacted activities) in their suffix. One approach is to train a machine learning model to predict the next event given an event prefix [6]. The suffix is predicted by recursively feeding the predicted output event as input and obtaining the suffix till the End of Case [EOC] is reached. The code for this model is available online[1]. Another approach is to train a model that predicts the entire suffix [21]. The code for this model is also available online[2]. Further, existing work on suffix prediction can be categorized as activity suffix prediction and attribute suffix prediction [16]: i)*Activity suffix*: predicts the sequence of activities given an event prefix. ii)*Attribute suffix*: predict the sequence of activities and attributes given an event prefix. Here, approaches that recursively predict activity and use other attributes of the event as inputs also need to predict the sequence of attributes in the predicted events.

While we are interested in predicting the sequence of activities, we observe that the attributes of an event often carry the necessary information related to the sequence of steps a case takes in future. For example, in a order process, the order amount may be an important attribute that could be used to determine if an order needs approval or if it can directly be processed. We extend the LSTM-based model by Camargo et al [6] to support attribute suffix prediction. Figure 3 illustrates the architecture that has the input layer for each categorical attribute $(d^1, \ldots, d^p)$ and activity (a). The categorical attributes have an embedding layer. The prefixes of length k of categorical attribute embedding and activity embedding are concatenated and passed to the first layer of the LSTM.

The architecture supports numeric case level attributes (such as loan amount, order quantity) as input to the model. These numeric attributes (d^c) and the time duration (t) are concatenated and passed through another layer of LSTM. The output is passed through a second layer of the LSTM and a dense layer for each activity, attribute, and timestamp prediction. The output predicts the next

[1]https://github.com/AdaptiveBProcess/GenerativeLSTM
[2]https://github.com/farbodtaymouri/MLMME

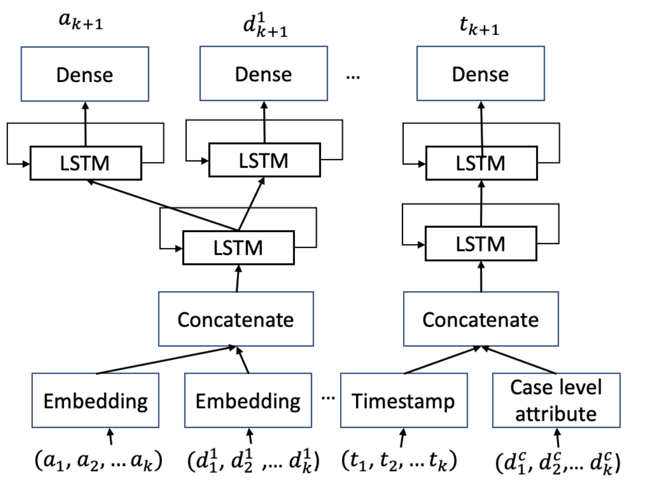

Figure 3: Attribute Suffix Prediction

activity, attribute and timestamp $(a_{k+1}, d^1_{k+1}, \dots, d^p_{k+1}, t_{k+1})$. These values are recursively provided as input to the model along with the case level attributes, to predict the suffix till the end of case is reached.

3.3 Time series metric causal discovery

The business events along with the application events are transformed into a time series. First, we define the sampling time or the time window size $w \in \mathbb{R}^+_0$, for which we want to transform the data. Next, the business and application events are segmented into subsequent time intervals of size w. Finally, numerical features are computed from the business and application events in each segment. Multiple features can be extracted from business process events for different perspectives of process mining [1]. Examples of features include execution frequency of activities (control-flow), duration of activities (performance), aggregation of event attributes such as *processed order amount* (data), or number of active business users/resources (resource). We then have a time series $T_f = \{t_1, t_3, \dots, t_n\}$ consisting of n values for each feature f. Multiple features from the business process and application events results in a multivariate time series $D = \{T_1, T_2, \dots T_f\}$. These time series are from both business and application events. Similarly, the time series can be derived from metrics and KPIs as they are pre-computed numeric values with feature functions on the business and application events. The metrics provided by the monitoring tools are transformed into a time series for the time intervals by an aggregation function such as sum, mean or median based on each metric. Thus, business KPIs are pre-defined metrics computed on business events, and application events are transformed into a time series.

Once the transformed time series of the business and application events and metrics is available, the next step is to determine causal relationships between them. We identify causal relationships between application events or application metrics and the business KPIs and events using causal models on time series data. Different algorithms may be suited for this task based on characteristics of the time series data. Granger Causality [11] is based on linear regression and tests a weak form of causality: the extent to which the past values of one time series is predictive of another time series. Impulse Response Analysis (IRA) [14] is another linear model that is suited to identifying causal relationships when there is a sudden change in the time series. IRA is suitable when the application events are binary events that are triggered on some threshold (an event triggered when the utilization of a server ≥ 90%). Additionally, there have been recent algorithms for capturing long range nonlinear Granger causal relationships [20]. However, these algorithms did not perform well on our dataset, presumably because the dataset is small. Hence we do not report results for them. The code for Granger Causality and IRA is available online[3].

Time series metric causal discovery uses the event and the metric data as input. The output of the model is an attributed graph $G = (N, E)$, where N is a set of nodes representing the business KPIs or metrics or application event features, and E indicates the causal relationships between business and application time series. Each node $n \in N$ has a set of attributes X such as the name of the event or metric, and the underlying business or application component associated with it. Similarly the edge stores the lag length of the causal relationship between two nodes. Lag length is the time difference between an event in the time series of one node and its impact in the time series of the other node.

3.4 Time series KPI forecasting

The time series data is used to train a variety of statistical, machine learning, and deep learning multi-step forecasting models for forecasting the future values of KPIs in presence of application events at multiple horizons based on the maximum lag length of the edges in the attributed graph. The goal of training the model is to quantify the impact of application events such as failure or high latency on the business metrics or KPIs. Classical time series forecasting methods (such as ARIMA, Holt-Winters, etc.) are univariate methods and can only forecast based on the historical values for one time-series. In contrast, we are interested in forecasting multiple time-series related to each other (business KPIs and application events). Hence, we use multivariate time series which consist of more than one time-dependent variable and each variable depends not only on its past values but also has some dependency on other variables. We evaluated various multivariate time series (MTS) forecasting models that consist of statistical, machine learning, and deep learning models:

- **VAR**, short for Vector Autoregression [14], is a statistical model for MTS forecasting. VAR is often used in applications where the variables of interest are linearly related to each other and is a good choice for representing and predicting the behaviour of multivariate time series. The linear relationships make it interpretable. In addition to time series analysis and prediction, the VAR model is additionally utilized for causality inference of the multivariate time series. The model implementation available from the *statsmodels*

[3]https://github.com/statsmodels/statsmodels/blob/main/statsmodels/tsa/vector_ar/var_model.py

library is interpretable and offers in-built support for statistical tests such as Granger Causality and Impulse Response analysis. Code is available online[4].
- **GRU**[5] is a simple Recurrent Network-based model for MTS forecasting using a GRU [8] layer followed by an output layer. Code is available online[6].
- **AGCRN** [4] is a GCN based method for Multivariate Time Series forecasting, developed especially for the traffic forecasting problem. It automatically infers the location graph from the time series data, which makes it interpretable. Code is available online[7].
- **N-HiTS** [7] is a recent neural time series forecasting model which focuses on accuracy and computational complexity, especially in the context of long-horizon forecasting. The model uses multi-rate sampling of the input time series and multi-scale synthesis resulting in a hierarchical construction of forecast. The long-horizon forecast could be particularly useful in business KPI forecasting as there can be daily and weekly computed KPIs which may have an impact due to some system failure, after a significant time lag. Code is available online[8].

In our final implementation, we chose VAR as our forecasting model because of the highest PCC score, as detailed in Section 4.4.

3.5 Impact Prioritization

The analysis and insights provided by the unified topology, impacted cases, impacted business metrics and KPIs, and the quantum of impact using forecasting can be used to prioritize the resolution of failures or anomalies. Our current approach provides the information to support the decision of a domain expert: a business owner or a service reliability engineer (SRE).

4 EXPERIMENTAL EVALUATION

For evaluation, we have a synthetic dataset from a sample order approval process (Figure 4 (a)) that is deployed in an environment representative of a real-world environment (Figure 4 (b)) on multiple cloud clusters and virtual machines. The business process involves various steps including order updation, order approval, and order acceptance. The data generation pipeline, which involves proprietary code developed by our company, incorporates sufficient randomness while generating the event logs. It is designed to mimic real world data as much as possible, while allowing us some control over major hyperparameters such as the average expected time for an activity and so on. We capture the events, traces, and the KPIs from a process monitoring tool[9]. Similarly, we have the application traces and events that include failure events and anomalies captured by an application monitoring tool [10]. Table 1 captures the MELT data captured for the application. The KPI *weekly_avg_order_cycle_time* measures the average cycle time of all orders in the system on a weekly basis. The *count_** KPIs measure the throughput of the corresponding process steps (activities). The metrics are transformed into a time series containing 31968 data points with 5-minute time intervals. There are 20086 business events and the event log has 4200 traces. There are 723 IT events, encoded as binary signals for the time series data.

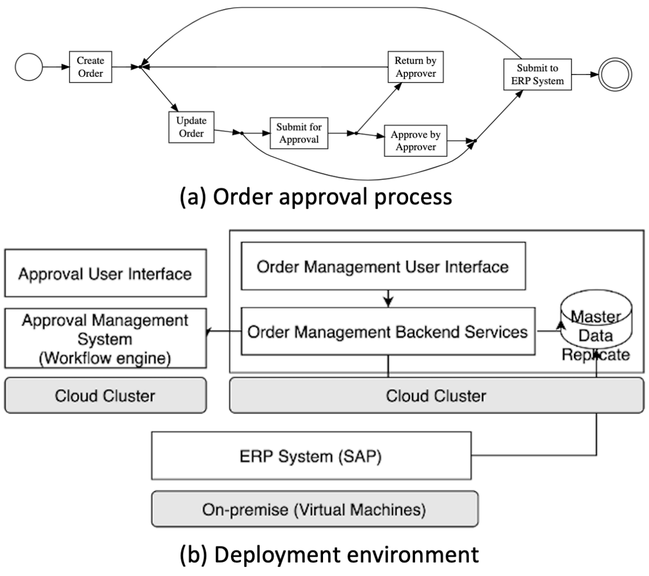
Create Order
Return by Approver
Submit to ERP System
Update Order
Submit for Approval
Approve by Approver
(a) Order approval process
Approval User Interface
Order Management User Interface
Approval Management System (Workflow engine)
Order Management Backend Services
Master Data Replicate
Cloud Cluster
Cloud Cluster
ERP System (SAP)
On-premise (Virtual Machines)
(b) Deployment environment

Figure 4: Order approval process and deployment environment

The following sections present the results where, first the *impacted activities* are detected using the *unified topology*. Next, the impacted cases are identified by predicting the suffix of the ongoing cases to mark the cases that could have an *impacted activity* in future. Additionally, impacted business KPIs and metrics are then identified by correlating the business and IT metrics and events.

4.1 Topology generation

The unified topology (subset) obtained for the order approval process is shown in Figure 5. The event log and application traces are collected for a period of two weeks and the unified topology is generated. As shown in the figure, when any anomaly or failure occurs in the system, this weighted topology helps in identifying the the set of *impacted activities* with the help of the impacted application component(s). A default threshold value $\theta = 0.75$ is used to obtain the the set of *impacted activities.* However, the value of θ can be varied to obtain a dynamic unified topology based on the relevance. For illustration, consider a sample failure: *Camunda is down* (see Figure 5), the service *bpm-service* corresponds to the BPM service i.e., Camunda. Using the topology map, the services impacted will be *approve_by_approver, return_by_approver, submit_for_approval* and *submit_to_sap* having confidence values 1.0, 0.97, 0.875, and 0.25 respectively. For $\theta = 0.75$, the resultant

[4]https://github.com/statsmodels/statsmodels/blob/main/statsmodels/tsa/vector_ar/var_model.py
[5]https://pytorch-forecasting.readthedocs.io/en/stable/api/pytorch_forecasting.models.rnn.RecurrentNetwork.html
[6]https://github.com/jdb78/pytorch-forecasting/blob/master/pytorch_forecasting/models/rnn/
[7]https://github.com/LeiBAI/AGCRN
[8]https://github.com/jdb78/pytorch-forecasting/blob/master/pytorch_forecasting/models/nhits/
[9]https://www.ibm.com/products/process-mining
[10]https://www.instana.com/

	Metrics	Events	Traces
Business Process	weekly_avg_order_cycle_time, count_approve_by_approver, count_create_order, count_return_by_approver, count_submit_for_approval, count_submit_to_sap, count_update_order	approve_by_approver, create_order, return_by_approver, submit_for_approval, submit_to_sap, update_order	event log
Application	orm_application_latency, orm_application_errors, orm_application_calls	approval_system_not_available, unable_to_connect_to_sap	application traces

Table 1: Metrics, Events, and Traces for the business process.

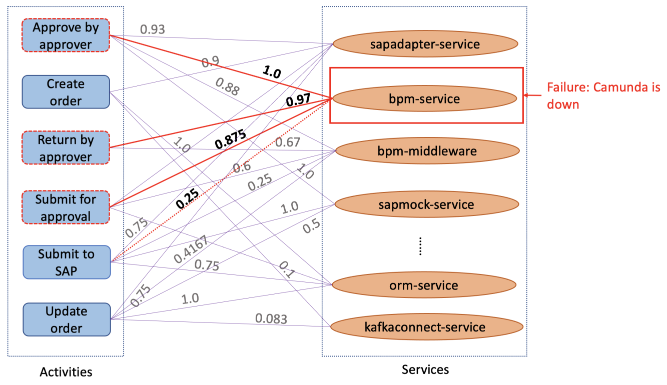

Figure 5: Unified Topology of Order Approval Process (Failure in bpm-service results in Camunda going down. The *impacted activities* obtained are approve_by_approver, return_by_approver, submit_for_approval for $\theta = 0.75$)

set of *impacted activities* will only be *{approve_by_approver, return_by_approver, submit_for_approval}* and *submit_to_sap* gets eliminated due to confidence value being lower than θ.

4.2 Suffix prediction

We train the modified attribute suffix prediction model as detailed in Section 3.2. A widely used evaluation metric for suffix prediction uses the Damerau–Levenshtein (DL) algorithm which measures the distance between sequences in terms of the number of editions necessary for one string to be equal to another. The DL distance is measured between the predicted suffix and the ground-truth suffix. The normalized inverse of DL distance measures the similarity. In Table 2, DL similarity is used to compare our model with two activity suffix prediction models [6, 21] that do not consider any data attributes. On this dataset, the decision paths are influenced by the data attributes and hence our model performs better.

4.3 Time series metric causal discovery

With reference to metrics and events captured in Table 1 and the corresponding process in Figure 4 (a), we have the causal relationships as shown in Figure 6. Here, *approval_system_not_available* affects the KPIs *weekly_avg_order_cycle_time*, *count_submit_for_approval*,

Table 2: DL similarity between predicted and actual activity suffixes

Method	DL Similarity
Attribute suffix prediction (ours)	**0.805**
Activity suffix prediction [6]	0.713
Activity suffix prediction [21]	0.317

Table 3: Business and IT metric correlation.

Method	Precision	Recall	F1-score
Granger Causality	1.0	0.14	0.25
Impulse Response Analysis	1.0	0.57	0.73

count_approve_by_approver, and *count_return_by_approver*. This is because *approval_system_not_available* impacts all the steps involving the approval steps as well the overall KPI *weekly_avg_order_cycle_time*. The metric of the event *unable_to_connect_to_sap* impacts the KPIs *weekly_avg_order_cycle_time*, *count_update_order* and *count_submit_to_sap*. The event *unable_to_connect_to_sap* would affect *count_submit_to_sap*, since the SAP system is down, as well as the overall KPI *weekly_avg_order_cycle_time*. The orders are often rerouted from the SAP system to the *Update Order* step. Hence, if the SAP system is down, there is a corresponding drop in the number of orders entering the Update Order step, which is reflected in the KPI count_update_order. The precision, recall, and F1 scores obtained using these methods are shown in Table 3. Here IRA obtains the best results in terms of all metrics and we show its predicted causal relationships in Figure 6 as the blue edges labelled "IRA". This is because, the IT events are primarily errors that result in sudden impulses that IRA is suited to support. IRA is unable to find the correlation between the IT events and the KPI *weekly_avg_order_cycle_time* as the change in the KPI is not significant, as well as *count_return_by_approver* since the event is sparse.

4.4 Time series KPI forecasting

We discuss the overall results obtained by using various multivariate time series (MTS) forecasting techniques for forecasting the future

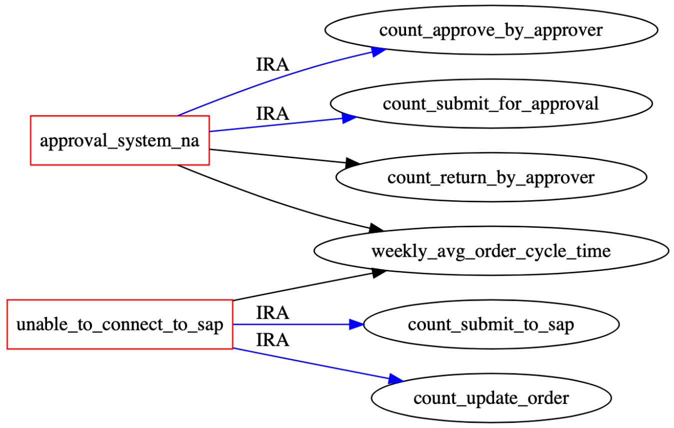

Figure 6: Ground truth causal relationships and those detected using IRA (marked in blue and labelled "IRA".)

Table 4: KPI forecasting results for the order approval process

Model	MAE	RMSE	MAPE	PCC
VAR	1.07	**2.74**	0.14	**0.2830**
AGCRN	1.16	3.15	0.17	0.2371
N-HiTS	**0.98**	2.93	**0.12**	0.2035
GRU	1.25	3.82	0.16	0.1768

values of KPIs in presence of an IT event at multiple horizons based on a fixed lag k. The dataset is divided into train, validation, and test sets in the ratio of 6:2:2. This split is commonly used in the literature on multivariate time series forecasting [4, 24]. The models are trained to take as input a sequence of length 24 (i.e. they have a lag order of 24) and generate a sequence of length 24 as output (i.e. they have a maximum forecast horizon of 24). Hence, we forecast for up to 2 hours of future KPI values. We used a set of standard metrics for evaluation: Mean Absolute Error (MAE), Root Mean-Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Pearson Correlation Coefficient (PCC)[11].

We posit that in addition to the commonly used error metrics (MAE, MAPE, and RMSE), we need to consider PCC to evaluate whether the models can provide early indications of KPI changes in response to IT events. We present a scenario of a synthetically generated KPI (blue line) in Figure 7, where the forecasting models are VAR and DecoderMLP (another deep learning-based model that performed poorly and hence is not used). For this data the MAE of VAR (MAE=12.98) is higher than that of DecoderMLP (MAE=10.63). However, Figure 7 clearly shows that VAR predicts the peaks and the change in trend better when compared to DecoderMLP that forecasts an average KPI value. Hence, use of PCC along with other metrics is useful in this context where it is necessary to detect the changes in the trend of a KPI.

The evaluation of various MTS models (Table 4) indicates that N-HiTS performs the best in terms of MAE, followed by VAR. In terms of PCC, VAR is the best model. Hence we conclude that for

[11]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html

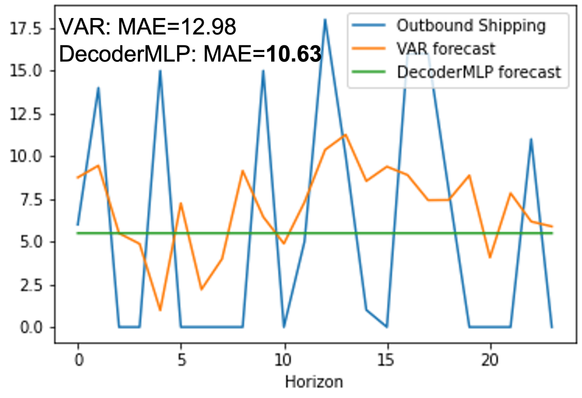

Figure 7: Example predicted and actual values of a synthetic KPI sample, where the prediction is made by VAR and DecoderMLP

this dataset, VAR is sufficient for our purpose even though it does not have the best MAE (but the MAE is still comparable to the best).

4.4.1 Threats to Validity. The evaluation comes with a threat to *external validity*. External validity concerns the generalization of the results from this study. We have considered one application representative of real world deployment. This can be addressed by conducting further experiments on other such deployed processes. *Internal validity* is established for a study if it is free from systematic errors and biases. Our study used existing monitoring tools to capture MELT data and throughout this period, the deployment configurations remained constant. Thus, the extent of this threat to validity is limited. For *construct validity*, the experiments are based on data observed during the period of the study. While the study may not cover all process execution scenarios, the transformation and evaluation is based on standard methods.

Reproducibility: The data used in this paper is available at https://github.com/BizITObs/BizITObservabilityData. With the exception of attribute suffix prediction, most of the code is available online in different public repositories.

5 CONCLUSION AND FUTURE WORK

In this work, we developed an approach to utilize both business and application information that allows correlating the information for detection and diagnosis of failures in complex business process execution environments. Our approach provides mechanisms that allow for understanding the process executions as opposed to black-box monitoring of events. We present multiple techniques that utilize the business and application data for correlation, forecasting, and impact identification. In future, we would want to evaluate the approach in multiple complex deployments of business processes. Additionally, this work does not use unstructured logs from applications and business process task executions. We would extend the work to consider the wealth of data existing in application and process logs to provide additional insights for business process observability.

REFERENCES

[1] Jan Niklas Adams, Sebastiaan J. van Zelst, Thomas Rose, and Wil M. P. van der Aalst. 2023. Explainable concept drift in process mining. *Inf. Syst.* 114 (2023), 102177.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In *Procs of the 20th Intl. Conf. VLDB (VLDB '94)*. 487–499.

[3] Robert S. Arnold and Shawn A. Bohner. 1993. Impact Analysis - Towards a Framework for Comparison. In *Procs. of the ICSM 1993*. IEEE Computer Society, 292–301.

[4] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems* 33 (2020), 17804–17815.

[5] Betsy Beyer, Chris Jones, Jennifer Petoff, and Niall Richard Murphy. 2016. *Site Reliability Engineering: How Google Runs Production Systems.* http://landing.google.com/sre/book.html

[6] Manuel Camargo, Marlon Dumas, and Oscar González-Rojas. 2019. Learning accurate LSTM models of business processes. In *BPM 2019, Procs.* 286–302.

[7] Cristian Challu, Kin G Olivares, Boris N Oreshkin, Federico Garza, Max Mergenthaler, and Artur Dubrawski. 2022. N-hits: Neural hierarchical interpolation for time series forecasting. *arXiv preprint arXiv:2201.12886* (2022).

[8] Kyunghyun Cho, Bart Van Merriënboer, et al. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[9] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. 2018. *Fundamentals of Business Process Management, Second Edition.* Springer.

[10] Chiara Di Francescomarino. 2019. Predictive Business Process Monitoring. In *Encyclopedia of Big Data Technologies*, Sherif Sakr and Albert Y. Zomaya (Eds.). Springer.

[11] Clive WJ Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: J of the Econometric Society* (1969), 424–438.

[12] Suman Karumuri, Franco Solleza, Stan Zdonik, and Nesime Tatbul. 2020. Towards Observability Data Management at Scale. *SIGMOD Rec.* 49, 4 (2020), 18–23.

[13] Kateryna Kubrak, Fredrik Milani, Alexander Nolte, and Marlon Dumas. 2022. Prescriptive process monitoring: *Quo vadis*? *PeerJ Comput. Sci.* 8 (2022), e1097.

[14] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis.* Springer Science & Business Media.

[15] Rodolfo Picoreti, Alexandre Pereira do Carmo, Felippe Mendonca de Queiroz, Anilton Salles Garcia, Raquel Frizera Vassallo, and Dimitra Simeonidou. 2018. Multilevel observability in cloud orchestration. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 776–784.

[16] Efrén Rama-Maneiro, Juan Carlos Vidal, and Manuel Lama. 2023. Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. *IEEE Trans. Serv. Comput.* 16, 1 (2023), 739–756.

[17] SAP. 2022. What is Business Process Observability and why does it matter? https://blogs.sap.com/2022/09/16/what-is-business-process-observability-and-why-does-it-matter/.

[18] Mario Scrocca, Riccardo Tommasini, et al. 2020. The Kaiju project: enabling event-driven observability. In *14th ACM DEBS 2020*. 85–96.

[19] C. Sridharan. 2018. *Distributed Systems Observability: A Guide to Building Robust Systems.* O'Reilly Media.

[20] Alex Tank, Ian Covert, Nicholas J. Foti, Ali Shojaie, and Emily B. Fox. 2022. Neural Granger Causality. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 8 (2022), 4267–4279.

[21] Farbod Taymouri, Marcello La Rosa, et al. 2021. A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences. In *Procs. of SDM 2021*. 522–530.

[22] Irene Teinemaa, Marlon Dumas, et al. 2016. Predictive Business Process Monitoring with Structured and Unstructured Data. In *BPM. Procs.*, Vol. 9850. 401–417.

[23] Irene Teinemaa, Marlon Dumas, et al. 2019. Outcome-Oriented Predictive Process Monitoring: Review and Benchmark. *ACM Trans. Knowl. Discov. Data* 13, 2 (2019), 17:1–17:57.

[24] Zonghan Wu, Shirui Pan, et al. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD*. 753–763.