

Hierarchical Anomaly Detection and SHAP-Based Root-Cause Attribution for Robotic Process Automation Workflows

Yanka ALEKSANDROVA*

University of Economics – Varna, Varna, Bulgaria

**Corresponding author, yalexandrova@ue-varna.bg*

Mihail RADEV

University of Economics – Varna, Varna, Bulgaria

radev@ue-varna.bg

Mila GEORGIEVA

Medical University – Varna, Varna, Bulgaria

mila@mu-varna.bg

Desislava KOLEVA

University of Economics – Varna, Varna, Bulgaria

desi_koleva@ue-varna.bg

Abstract. *Nowadays detecting anomalies is very important for automated business processes to proactively mitigate process failures. Useful data for identifying possible causes for irregularities and anomalous behavior is generated by the Robotic Process Automation (RPA) workflows logs. Most of the implementations in this area, however, summarize and analyze generated data at run level which doesn't allow the differentiation of the problems caused by issues with external environment from those associated with the internal process logic. To support a more detailed analysis of the root cause, the paper proposes a hierarchical framework with two tiers for monitoring RPA workflows. Each tier is focused on detecting anomalies at different levels (run-level and process-level) and then the outcome from both tiers is integrated in a mapping layer to more clearly identify the root cause for the anomaly. The proposed framework is validated with a synthetic dataset comprising 10,000 runs generated in a controlled environment to simulate real world scenarios. At Tier 1, the framework implements Isolation Forest and feed-forward Autoencoder to detect anomalies at run-level. Process-level anomalies associated with unusual durations and sequence patterns are detected at tier 2 using Z-score statistics and a Long Short-Term Memory Autoencoder. The mapping layer combines results obtained from both tiers into four diagnostic states - Normal, Environment Stress, Latent Logic Issue and Systemic Failure. To support interpretation of the results, an explanation layer is added using SHAPley additive explanations. The analysis of main drivers for anomalies is used to calculate External Influence Index, which separates the contribution of external factors from those related to internal process-related patterns.*

For injected anomalies, the best models achieved ROC-AUC values close to 0.96. At the process level, Z-score and LSTM-based scores showed moderate but statistically significant agreement, with Pearson's $r = 0.55$ and $p < 0.001$. These results suggest that the two process-level methods capture related but not identical patterns. To further evaluate the framework feasibility, a pilot test on 325 runs of a university process is implemented. The results confirmed that the two-tier approach can reveal consistent anomaly patterns without ground-truth label, while the External Influence Index can determine source of detected anomalies. Results from both experiments suggest that the framework provides a more informative view than a single-level monitoring.

Keywords: Robotic Process Automation, hierarchical anomaly detection, root cause analysis, explainable artificial intelligence, Isolation Forest, Long Short-Term Memory Autoencoder, operational diagnostics.

DOI: 10.2478/picbe-2026-0043

© 2026 Y. Aleksandrova; M. Radev; M. Georgieva; D. Koleva, published by De Gruyter.

This work is licensed under the Creative Commons Attribution 4.0 License.

Introduction

Robotic Process Automation is widely used across different domains such as finance, logistics, and energy systems. With its broader implementations, the RPA deployments become more and more complex. This increases the need for a robust systems that can detect and prevent anomalies and failures of the business process executions (Deng *et al.*, 2024). Problems associated with poorly monitored RPA workflows can cause compliance risks, disruptions, increased remediation costs, which can in turn lead to decreased adoption of process automation projects.

There are many studies focused on proposing approaches to detect anomalies in RPA systems. Essakki and Bagavathi (2025) used TinyML on event logs and deep-learning models for scheduling-data analysis. These studies usually treat anomaly detection as a single-level task. They are mainly focused on detecting unusual cases where the data is aggregated on run-level. This level of summarization doesn't allow us to determine whether the problem is caused by the execution environment, by internal process logic, or by a combination of both groups of factors. Understanding the source of the anomaly is crucial for all areas of robotic and manufacturing systems in order to achieve interpretable diagnostic framework. Such frameworks enable operators to take appropriate remediation actions (Chirayil Nandakumar *et al.*, 2024).

RPA systems naturally generate telemetry at different levels, such as run-level (e.g., queue times, concurrency, throughput) and action-level measures (e.g., step durations, sequences). Within a single execution run data can be generated for measuring both levels performance. To correctly identify problems, we need to take into account both levels, because if only one level is analyzed, different types of problems may be confused. For example, a long execution time can be caused by platform overload, or by one inefficient action inside the process. If the monitoring framework do not provide timely and accurate actionable results about the main drivers for failures, it limits their usefulness for decision-making (Tayeh and Shami, 2021).

To overcome the limitation of anomalies detection only at one level, the present paper proposes a hierarchical anomaly detection and attribution framework for RPA workflows. The framework separates anomaly detection into two complementary and integrated levels. The first tier (Tier 1) implements models that are trained on environmental and structural features, such as queue time, concurrency, payload, connector complexity, step count, and total execution duration. This tier is designed to capture deviations in the overall execution context. The action-level patterns related to internal process logic are analyzed using at the second tier (Tier 2) by implementing statistics at action level and sequence models over action durations. This tier is focused on identifying local irregularities, unusual step duration, and temporal degradation within the workflow.

The two tiers are then integrated through a rule-based mapping layer. Each run execution is assigned to one of four diagnostic states: Normal, Environment Stress, Latent Logic Issue, or Systemic Failure according to the results obtained from Tier 1 and Tier 2 models. The aim of this classification is to distinguish cases in which the anomaly is mainly caused by external factors from cases in which it is more likely related to the internal logic of the process. To support interpretation, surrogate tree models and SHAP-based feature attribution are applied to grouped features. The contributions of the features and their influence rankings are used to calculate External Influence Index which estimates the relative contribution of environmental and process-level factors and supports the attribution of anomalies to infrastructure-related or workflow-related causes. The integration of an explanation layer within the hierarchical framework strengthens the capability to determine the root cause compared to an approach where the explanation is implemented separately from the anomaly detection systems (Renner *et al.*, 2025; Muaz *et al.*, 2025; Landauer *et al.*, 2023a).

The main research question of the study is whether the combination of run-level and process-level anomaly detection can provide more informative diagnostics than a single-level anomaly detection approach in complex RPA environments. Three hypotheses are formulated. The first one is that combining run-level and process-level anomaly detection can produce more stable and operationally useful diagnostic states than using only one or either of them separately. The second hypothesis is that SHAP-based explanations could be converted into consistent External Influence Index that correctly isolates the underlying mechanism of anomalies, to effectively decouple infrastructure stress and flow logic defects. Third, an interpretable integration and attribution scheme can convert low level telemetry into actionable diagnostics when applied at individual levels. The last hypothesis is that the integration of anomaly detection and attribution is expected to make RPA generated telemetry more useful for monitoring, escalation, and managerial decision-making in RPA governance.

Literature review

Much of the existing research on RPA focuses on adoption, expected productivity gains, and risks. Less attention is given to the post-deployment stage, where companies have to monitor processes executions and manage anomalies and failures. Studies in accounting, finance, and auditing documents show that RPA combined with artificial intelligence can improve transaction processing and support fraud detection. However, anomaly detection is usually discussed as one general function, and not as a process that should distinguish between different levels of execution.

Prucha (2023) identifies self-healing and autonomous bots as an important future direction for RPA systems but also points to the limited ability of current systems to detect and explain failures in an operationally useful way. Reports on various industry-oriented studies confirm the existence of similar problems, related to process monitoring based only on run-level indicators or manual log inspection. Such monitoring can show that a run has failed or exceeded its expected duration, but it often does not explain what the reason for this behavior is - whether the problem comes from platform load, connector delays, workflow logic, or configuration errors.

Several recent studies have explored implementation of different models for anomaly detection within RPA environments. For example, lightweight models have been applied to event logs for detecting anomalous payment requests, while deep-learning models have been used in power-system scheduling environments where abnormal behavior may have serious operational consequences (Prucha, 2023). Statistical process control techniques have also been used to monitor unstable behavior through indicators based on execution time and process outcomes (Essakki and Bagavathi, 2025). These studies show that anomaly detection can be useful in automation contexts. However, they usually analyze anomalies at one level only and mainly report detection performance. They do not fully address how run-level and action-level signals can be combined and explained within the same diagnostic framework (Deng *et al.*, 2024).

Research outside the RPA field also support the discussed findings. In robotics, industrial inspection, and log analysis, anomaly detection models often use autoencoders, virtual autoencoders (VAEs), and Long Short-Term Memory (LSTMs) neural networks to capture temporal or sequential patterns (Shukla et al., 2025; Landauer et al., 2023b; Shi et al., 2024). Studies on log anomaly detection are particularly important because they explore sequence modelling, reconstruction error, and the use of synthetic logs for benchmarking under controlled conditions. This justifies the use of a structured synthetic dataset in the present study, where the anomaly types are injected into the dataset as ground truth labels and then models are evaluated against these targets to determine their predictive power. The aggregation of event logs data at

different levels overcomes the limitation of treating process telemetry only as generic time-series data. Their structure and internal process patterns should be also considered to determine and classify the abnormal state.

A related group of studies focuses on explainable anomaly detection and root-cause analysis. SHAP-based explanations have been used in manufacturing and process industries to interpret fault detection models and to identify which variables contribute most to an anomaly prediction. These approaches are useful, especially when feature attributions are combined with domain knowledge. However, most of them analyze all variables together, without separating environmental and process-level features. They do not explicitly separate environmental factors from process-level factors, and they do not connect explanations to different anomaly states across run and action levels (Cortes *et al.*, 2025; Muaz *et al.*, 2025). This leaves a gap for an RPA-specific framework that uses the natural hierarchy of RPA logs, combines anomaly detection at the run and process levels, and uses SHAP-based explanation to distinguish infrastructure-related anomalies from workflow-logic issues.

Methodology

The proposed framework is shown on Figure 1. It is structured into different tiers starting from data ingestion, followed by parallel anomaly detection at run and process levels, integration, and diagnostic and attribution tiers.

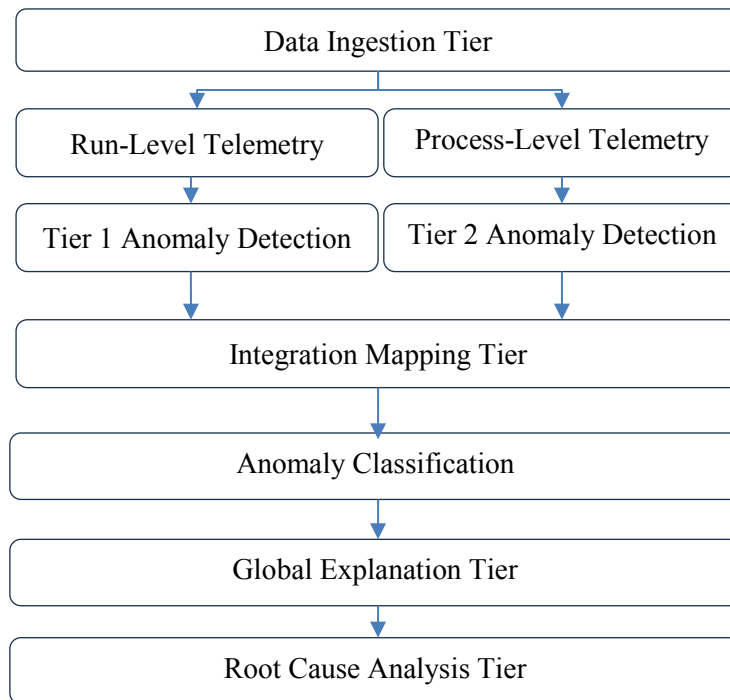


Figure 1. Proposed hierarchical anomaly detection framework

Source: authors' own work.

Data Ingestion Tier. Data are ingested from log files and then split into two datasets corresponding to the dual-layer anomaly detection architecture. The first dataset contains run-level telemetry, while the second contains process-level telemetry. Some of the features included in these datasets are:

- **Run-level Features** – captured at the flow’s execution boundary that represent the environmental context, such as *QueueTimeSec* (seconds spent on waiting), *Concurrency* (number of parallel executing actions/runs), *TotalDurationSec* (total duration in seconds of the run flow), *ConnectorComplexityScore* (an estimator of how complex (heavy) is the primary connector used by the flow), *StepCount* (number of steps in the run), *FlowRollingFailRate3* (Rolling average failure rate for the last 3 runs of the same flow), etc.
- **Process-level Features** – captured at the individual action from the run flow, e.g. *ActionDurationSec* (duration in seconds of the action), *PayloadKB* (data size in kilobytes), *StepNumber* (a sequential number of the action within the run), etc.

Separating data at run and process level is very important for determining the anomaly attribution. For example, an anomaly may be visible at run-level, but it may be caused by a systemic failure (e.g. the whole server is shut down), whereas a process-level anomaly can reveal latent logic issues with particular action within the run. This two-tier hierarchy allows for analysis of contributing factors and improves diagnostic by preventing problems on process-level masking systemic issues and vice versa.

Tier 1: Run-Level Anomaly Detection. Tier 1 is designed to identify outliers at execution-wide basis. Two distinct algorithmic approaches are proposed, although the framework suggests that other suitable models for anomaly detection can be also built and compared to evaluate their performance and consensus:

- **Isolation Forest (IF).** This algorithm is selected for its efficacy in high-dimensional space without assuming a normal distribution. IF isolates anomalies by randomly selecting a feature and a split value. Since anomalies are "few and different," they are isolated in fewer steps (shorter path lengths)(Cao *et al.*, 2024).
- **Autoencoder (AE).** A neural network designed to compress and reconstruct input data. A base starting encoder proposed for the experiments is fully connected dense AE, which is well-suited to detecting anomaly in tabular data (Kumar *et al.*, 2024).

Using two models for anomaly detection allows them to combine their strengths. Isolation Forest is performing well in the presence of global outliers, while the Autoencoder is better at identifying subtle, non-linear correlations between environmental features that a tree-based model might not be able to learn.

Tier 2: Process-Level Anomaly Detection. Tier 2 analyzes the internal execution of steps within the flow. On this tier, two algorithms are proposed as a starting point for the process-level anomaly detection:

- **Z-Score Analysis:** This method establishes a statistical baseline for every action type (e.g., "HTTP Request") and then flags action with duration, exceeding the threshold of three standard deviations.
- **LSTM Autoencoder:** The implementation of a Long Short-Term Memory (LSTM) Autoencoder is chosen because the RPA flows are sequential and a model for modeling temporal dependencies of action durations is needed. Such models can identify sequences that are structurally anomalous, even if individual Z-scores appear normal.

Integration Tier. Since run and process data is detailed at different granularity levels, the integration tier is achieved through a relational mapping of the granular process-level action features to global run-level execution envelopes. The mapping is important for ensuring that the anomaly detection framework maintains a holistic view, accounting for both the run and the content (individual actions).

Anomaly classification. Each run execution is classified into one of four hierarchical states based on the presence of Tier 1 (r) and/or Tier 2 (p) anomaly flags:

- Systemic Failure (r and p) –anomalies that are detected at both run and process levels, caused by both environment and logic problems.
- Environment Stress (r and not p) – run-level anomalies occurring in the absence of internal irregularities of the action executions. The anomaly is definitively attributed to environmental factors. This state represent cases where the system is struggling within the execution environment, but the flow logic remains stable.
- Latent Logic Issue (not r and p) – these are failures where no anomaly at run level is detected, but within the process there are specific abnormal actions behavior patterns. Such patterns can be seen often related to data-specific issues or unhandled exceptions. Such states are usually associated to executions where the external environment is not exhibiting problems, but the internal flow steps are causing the overall process failure.
- Normal (not r and not p) – state of normal operation both at run and at process levels.

The interpretation of Tier 1 and Tier 2 models is implemented by surrogate tree based and SHAP explanation models.

Global Explanation Tier. To interpret and explain models on Tier 1 and Tier 2, surrogate tree-based models are trained and SHAP values are calculated to show feature contribution at global level. The run-Level SHAP analysis focuses on identifying global drivers from the external environment. This allows the framework to identify which platform-wide variables are the most triggers for anomaly. At the process level, SHAP is applied to the sequence-based anomalies identified by the LSTM Autoencoder or Z-Score spikes. The focus of process level SHAP analysis is the feature interaction by displaying SHAP Interaction values to determine if an internal anomaly is independent or contingent upon Tier 1 environmental features.

Root Cause Analysis Tier. The final stage of the framework is to evaluate the core reason behind detected anomalies. To do this, an *External Influence Index (EII)* is defined as the ratio of SHAP values attributed to environmental features from the total sum of SHAP values for all features (environmental and internal) and is calculated as shown in (1).

$$EII = \frac{\sum_{i=1}^n |SHAP_{env}|}{\sum_{i=1}^n |SHAP_{env}| + \sum_{i=1}^n |SHAP_{proc}|} \quad (1)$$

where $SHAP_{env}$ are the SHAP values for environmental features, and $SHAP_{proc}$ are the SHAP values for internal process-level features.

If EII is greater than 0.7, the root cause can be attributed to external infrastructure stress factors. If EII has lower values, e.g. below 0.3, then the root cause can be explained to the flow logic as internal failure. These thresholds were chosen in order to make the framework more conservative and to improve the classification of borderline cases. During practical implementation, practitioners can tune these thresholds to match the perceived risk tolerance according to implemented escalation and remediation policies.

By using force and decision plots, a diagnostic root cause analysis on individual failed runs can be implemented to determine and further analyze the influence of external and internal factors and to visualize the specific feature contributions.

Results and discussions

The framework is validated with a synthetic dataset consisting of 10000 runs. The dataset is generated with controlled parameters to simulate different project runs. Most of the runs don't have

any irregularities or anomalies, while some of the runs are programmed to represent infrastructure stress, latent logic defects, and systemic failures. To evaluate unsupervised machine learning models, an anomaly label is injected into the runs to allow for benchmarking against known ground-truth labels.

Approximately 21.6% of runs contain at least one form of anomaly (see Table 3), distributed across different categories such as internal process logic abnormalities, systemic failures, and environmental stress. The anomaly injection was done both on run-level and process-level to support measuring the capabilities of the models to detect abnormal patterns in the presence of heterogeneous failure modes. All practical experiments are implemented in Python using packages like keras, scikit-learn, shap, etc.

The dataset includes indicators that can be observed in production (queue times, concurrency, payload size, connectors, step counts, durations, rolling failure rates), so the same ingestion and feature engineering steps can be applied with minimal changes to actual operational logs. Data is distributed into two linked tables - a run-level dataset (*runs_df*) and an action-level dataset (*actions_df*).

At Tier 1, a run-level matrix with seven environmental and aggregate features is generated containing *QueueTimeSec*, *PayloadKB*, *Concurrency*, *ConnectorComplexityScore*, *StepCount*, *TotalDurationSec*, and *FlowRollingFailRate3*.

Two unsupervised anomaly models are trained, and scores are added to the dataset:

- **Isolation Forest (IF).** The contamination percent is set at 5% meaning that a small ratio of anomalies is expected which is more aligned with real world RPA scenarios. The chosen value also prevents the model to flag more normal runs as anomalous. The absolute value of the anomaly score generated by the model is converted into a positive anomaly score, and based on the value a binary anomaly label (*IF_Anomaly*) is derived from the prediction.
- **Feed-forward AE with ReLU activations and linear output** was trained for 15 epochs with batch size 64 and MSE loss. Reconstruction errors per run is computed and used as *AE_Score*. Higher reconstruction errors indicate that the model had more difficulty reproducing the original input, which can be explained with anomalous behavior. To identify anomalies, a threshold was set at the 95th percentile. Runs with reconstruction errors higher than the threshold were labeled as anomalies. The value of the threshold is selected because it provides a reasonable balance between detecting irregularities and reducing the number of false positives. It is also consistent with the assumption that anomalies account for small proportion of all runs in real RPA environments.

The run-level anomaly scores are evaluated against three labels: *InjectedAnomalyFlag* (feature capturing the modification of the run execution during synthetic data generation), *SLAOverrunFlag*, and *FailureFlag* (failed status of the run flow) using AUC-ROC (see Table 1).

Table 1. Tier 1 Model evaluation with AUC

Label	Isolation Forest	AE
InjectedAnomalyFlag	0.9571	0.9025
SLAOverrunFlag	0.5787	0.5173
FailureFlag	0.6511	0.6287

Source: authors' own research.

As evidenced by the results from Table 1, the consistently high value of AUC above 0.9 for both models indicates that the models are able to correctly identify anomalies. IF is better performing with $AUC \approx 0.96$, which makes this model the primary anomaly detector at this level. The ability of the models to correctly classify SLA overruns is relatively moderate ($AUC \approx 0.52-0.65$), which is expected to some extent. Such outcomes can be attributed to both environmental issues and internal logic, so Tier 1 alone cannot fully detect them. This finding also justifies the need for another layer (Tier 2) to correctly identify anomalous runs.

Tier 2 – Process-level anomaly detection. The Tier 2 data is extracted from *actions_df*, which contains detailed metrics per every action from the run such as *ActionDurationSec*, *ActionName*, *RunID*, and *ActionIndex*. A statistical baseline for each action type is computed by grouping the data by *ActionName* and calculating the mean and standard deviation of *ActionDurationSec* for each group. The difference from typical duration of the same action type is modeled using a z-score. The anomaly flag (*ZAnomaly*) is set to 1 if z-score is greater than 3. At the run level, the process layer is aggregated with *MaxZScore*, calculated as $\max(|Z|)$ per *RunID*. A new feature is added to the dataset - *HasZAnomaly* - indicating if any action in the run has a Z-score anomaly.

To capture temporal structure we train a Sequential LSTM AE on fixed-length sequences of action durations per run. Actions are sorted by *RunID* and *ActionIndex*, then padded or truncated to maximum 10 actions per run. Reconstruction error (*LSTM_Score*) per run is calculated as mean squared error over sequence and time. The error is used as a basis to define an anomaly flag (*LSTM_Anomaly*) at the 95th percentile of reconstruction error. This threshold is chosen to provide ensure methodological consistency across tiers. Tier 2 scores are evaluated against the same labels as Tier 1 and the results are shown in Table 2.

Table 2. Tier 2 Model evaluation with AUC

Label	Z-Score	LSTM
InjectedAnomalyFlag	0.9472	0.9643
SLAOverrunFlag	0.7834	0.7114
FailureFlag	0.5862	0.5423

Source: authors' own research.

Similarly to the models from Tier 1, both Tier 2 models demonstrate high predictive performance on *InjectedAnomalyFlag* and *SLA overruns* ($AUC > 0.94$ for injected anomalies and $AUC \geq 0.71$ for SLA). These results reveal strong capability to determine internal performance degradation and logic-induced failures. Performance for *FailureFlag* is still moderate with $AUC \approx 0.54-0.59$, contributed to the multi-causal and latent characteristics of failures, where such failures are rarely demonstrated through notable duration increases or unusual patterns. Several recommendations can be given to further improve the performance of models on *FailureFlag* prediction, such as adding more features presenting short-term historical flow execution data except for *FlowRollingFailRate3*. Capturing more detailed error-related traces (e.g. error codes) and connector diagnostics indicators may also contribute to detecting failures better.

The Tier 2 consensus mechanism is assessed with Pearson correlation between *MaxZScore* and *LSTMScore*. The correlation coefficient is 0.5453 ($p < 0.001$), which shows that both process-level models tend to identify similar runs when there are strong deviations in the internal

performance. The relatively lower value of the coefficient suggests some independence between models. They can be viewed as complimentary, not redundant anomalies detectors.

Integration mapping tier. At this tier, anomalies detected at action-level are summarized at run level by taking the maximum anomaly magnitude across process metrics per run. With this approach, the model can capture the strongest detected deviation within given process executions as a run-level. This produces a combined Tier 2 feature space with variables capturing the presence of action level anomalies (*MaxZScore*, *HasZAnomaly*, *LSTM_Score*, *LSTM_Anomaly*). Using binary run-level (*r*) and process-level (*p*) flags for Tier 1 and Tier 2 anomalies, a boolean diagnostic matrix is constructed, where each run is assigned to one of four categories, namely Normal (not *r* and not *p*), Environment Stress (*r* and not *p*), Latent Logic Issue (not *r* and *p*) and Systemic Failure (*r* and *p*). The resulting distribution is shown on Table 3.

Table 3. Diagnostic anomaly matrix

Anomaly Category	Number of runs	Relative share
Normal	7843	78,4%
Latent Logic Issue	1657	16,6%
Systemic Failure	355	3,6%
Environment Stress	145	1,5%
Total	10000	100,0%

Source: authors' own research.

Results evident from Table 3 show that most executions are normal, while 21,6% exhibit some form of anomaly with most of these being internal logic issues, rather than environmental problems. Systemic failures are relatively rarely observed, representing $\approx 3.6\%$ of all runs. The smallest group of anomalies is formed by pure Environment Stress (Tier 1 only anomalies) with a relative share of about 1.5%. These results are expected given the controlled design of the synthetic dataset, where injected anomalies and SLA overruns are mainly concentrated at the process level. The clear separation between Environment Stress and Latent Logic Issue categories suggests that the two-tier based anomaly detection framework provides meaningful localization of anomalies. The different tiers help distinguish different sources of abnormal behavior rather than just repeating the same signals.

To examine misclassification patterns in more detail, a ground truth anomaly category label is defined and compared with the predicted anomaly labels. This comparison makes it possible to assess whether the model correctly distinguishes between different types of anomalies or whether some categories are systematically confused. The confusion matrix is shown on Table 4.

Table 4. Confusion matrix

		Predicted			
		Normal	Latent Logic Issue	Systemic Failure	Environment Stress
True	Normal	7587	154	29	73
	Latent Logic Issue	173	1348	96	40
	Systemic Failure	36	47	238	34
	Environment Stress	41	11	7	86

Source: authors' own research.

As shown by the confusion matrix, the integration layer tends to be conservative and reports fewer borderline anomalies than over-detecting them. Most normal runs are correctly classified as such with only a small number being incorrectly classified as Latent Logic Issue or System Failure. Latent Logic Issue runs are mostly classified correctly, but a small number of these cases are predicted as normal. This suggests that some internal process degradations may not produce strong anomaly signals and therefore are difficult to detect. Most Systemic Failure process runs are classified correctly as such, although notable cases fall under the category of "Latent Logic Issue" or "Environment Stress". This can be explained with situations where only part of the failure pattern is captured by the available models. The class with the most noise is the Environment Stress with more than 50% of the cases classified correctly, but there are a lot of cases which are mistakenly classified as Normal. This can be contributed to the fact that temporary environmental stress may not always exceed the anomaly threshold at both tiers. In such cases there can be difficulties to correctly classify weaker stress patterns, which can be mistakenly confused with normal executions.

Global Explanation Tier. Surrogate Random Forest models are trained to interpret the logic of the Isolation Forest predictive model (Tier 1) and LSTM model (Tier 2). SHAP Tree Explainers are then constructed on these surrogates and SHAP values are computed over the scaled data. Comparison of the SHAP summary plots is shown in Figure 2.

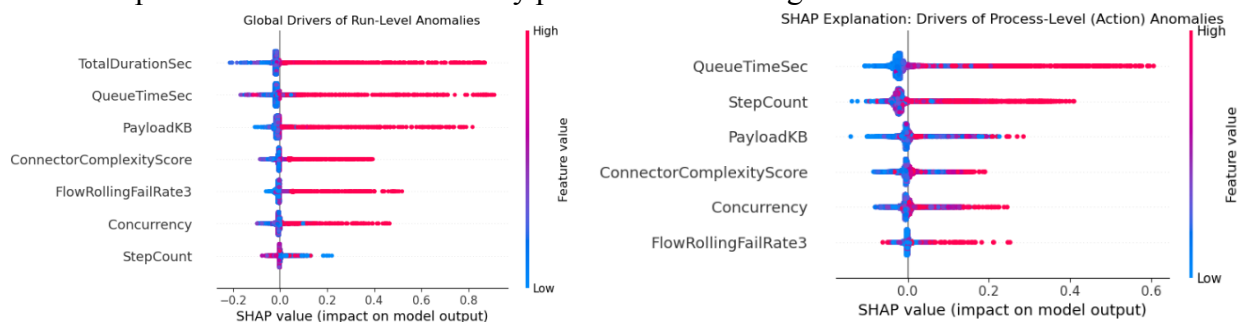


Figure 2. SHAP values for Run-Level (left) and Process-Level (right) anomalies

Source: authors' own research.

The left plot indicates that at run-level, the main factors contributing to anomaly detection are *TotalDurationSec*, *QueueTimeSec*, *PayloadKB* and *ConnectorComplexityScore*. These features are closely related to environmental stress and execution load, which supports the role of Tier 1 as a detector of external environmental induced failures.

The most influential factors at process level (right plot on fig.2) are *QueueTimeSec*, *StepCount*, *PayloadKB*, *ConnectorComplexityScore*, and *Concurrency*, suggesting that anomalies at this layer are due to execution delays and flow complexity. The detection of common drivers for both tiers can be explained with the fact that both tiers capture common underlying mechanisms. Total duration is associated with queueing delays and cumulative processing times of all actions within a run flow, which respectively are influenced by the size of a payload.

Overlap between tiers can be observed when anomaly labels at both levels are based on similar types of deviations (very long runs, extreme waiting time, unusually large payloads). Some overlap can also be caused by the multicollinearity among features. For example, payload size, number of steps, concurrency, queueing, and duration are often correlated and as a result SHAP contributions may be distributed across several features that represent different aspects of the same bottleneck.

The main difference between the two tiers is in their relative focus. Run-level models place greater emphasis on overall duration while process-level models are more oriented to queue time, step level complexity, and local timing variation. This separation, however, cannot be defined as absolute. Overall runtime metrics can still influence process-level detection, because both levels are influenced by the same execution environment. This means that some overlap is expected, but at the same time both models provide different capabilities to identify the root cause of the anomalies.

Another important aspect that can be explored is the interaction between external conditions and internal process behavior. This can be demonstrated by using SHAP interaction plots. One such plot showing the interaction between external feature *Concurrency* and the internal *PayloadKB* is shown in Figure 3.

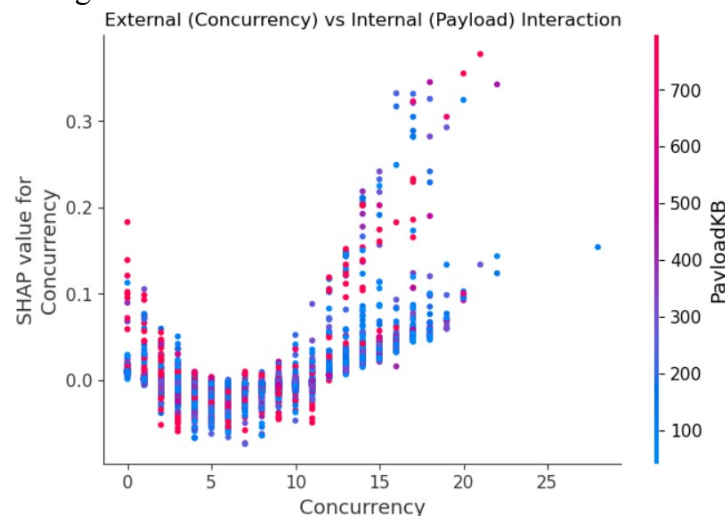


Figure 3. SHAP interaction plot for Concurrency (external) and PayloadKB (internal)

Source: authors' own work.

When the concurrency is less than 3, SHAP values are close or slightly below zero. This suggests that concurrency alone doesn't significantly increase the anomaly score. In the mid-range (4-10) the SHAP values become positive, showing that concurrency begins to increase the anomaly risk. At high concurrency levels (>15-20+) SHAP values become strongly positive. This validates concurrency as an important driver for anomalies. The color patterns also describe the interaction with *PayloadKB* which is an internal feature. Runs with large payloads (red points) tend to be placed at the higher end of SHAP values compared to smaller payloads (blue points). The variations of the vertical placement of SHAP values at the same concurrency levels indicate that when concurrency is high, larger payloads intensify the anomaly effect. This implies that an internal workload factor (payload size) and an external load factor (concurrency) work together. While high concurrency alone increases risk, high concurrency in combination with a large payload creates the strongest push toward anomalous behavior.

Root Cause Analysis Tier. To analyze and diagnose specific runs SHAP force and decision plots are built. A decision plot for RUNSYN-0004714 is shown on Figure 4.

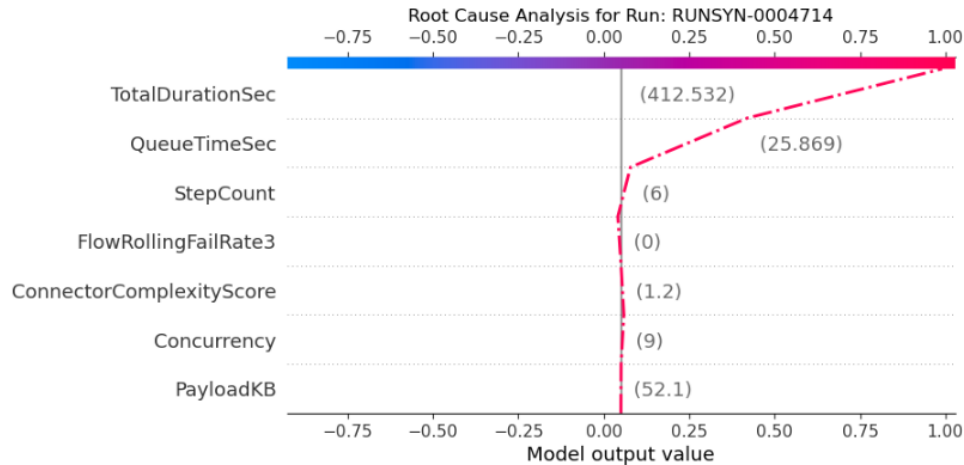


Figure 4. Decision plot for a single run

Source: authors' own work.

This SHAP decision plot on Figure 4 shows how each feature cumulatively pushes the surrogate model from the baseline output toward a final high anomaly score. *TotalDurationSec* is the main driver, moving the output in the direction of an anomalous prediction. Strong environmental stress is indicated by a very high queue time, which further increases the anomaly score. The other factors show relatively smaller influence on the output.

The plot explains why the run was flagged as anomalous by the Tier-1 surrogate. The main reasons are almost entirely due to extreme *TotalDurationSec* and *QueueTimeSec* values. The root cause can be thus attributed to infrastructure-driven performance issue – the flow run and waited a long time even though recent failure rate and connector complexity didn't suggest anomaly.

The obtained results on the realistic synthetic dataset demonstrate that the hierarchical architecture is both conceptually sound and empirically effective. Tier 1 captures environmental infrastructure anomalies, while Tier 2 is focused on internal performance. The correlation between Z-score and LSTM scores for Tier 2 models indicates a useful agreement between models that detect different anomaly modes. The SHAP explanation layer with External Influence Index adds intuitive interpretable diagnostic on the main drivers and root cause analysis on run-level and process-level anomalies. To evaluate whether the chosen thresholds are robust, additional experiments with different thresholds for defining EII were conducted. The experiments tested combinations 0.25/0.75 and 0.5/0.65. The relative share of runs in each category changed, but the classified runs in both categories changed by less than 4%, indicating that the boundaries are relatively stable. Different experiments are also recommended to further tune the threshold levels and compare the results.

To further test the reasonability of the proposed framework data about 325 runs of an automated process in the university environment was extracted. The process is automated in the Power Automate environment and is focused on automating the process of submission and evaluation of scientific projects proposals. The proposals are submitted in a Moodle dedicated course. New submission notification is triggered by the platform, and the scientific committee is notified. After project metadata like project title, authors, program call, etc. is captured, a unique submission ID is assigned, and the documents are checked for validity and consistency. The validated submissions are downloaded from Moodle and stored in Share Point folder. Main evaluation features are then extracted like scientific filed, expected outcome, project duration, budget amount, etc. Pre-evaluation

compliance check is run, and the reviewers are notified by email with direct links to Share Point. After the reviewers' assignments, parallel review stages are executed to wait for all required reviews with conditional splits for deadline handling. After all reviews are completed, aggregation steps are executed to calculate the overall project score and the propose the final decision on process submission. Runs use standard connectors to Moodle, SharePoint/OneDrive and track run-level and action-level features like step index, action duration, etc.

The pilot implementation was conducted on 325 process runs using the same run and process level models but without known ground truth labels. At Tier 1 evaluation of the models is focused on the stability and consistency of the anomaly patterns, rather than AUC. At Tier 1, the IF and AE showed broadly similar anomaly scores. The Pearson correlation coefficient was 0.68 and was statistically significant ($p < 0.001$). This result confirms that both models detect related anomalous patterns, although they differ in the way they detect irregularities.

At Tier 2, Z-score statistics and LSTM models highlighted anomalies primarily related with `wait_for_reviews` and `aggregate_reviews` segments, with approximately 14% of runs exhibiting at least one strong process-level deviation. Consistent with the previous findings, agreement between Z-score and LSTM anomaly indices remained moderate with Pearson correlation coefficient of 0.5782 ($p < 0.001$). This indicates that models detect different anomaly patterns and can be used as complimentary detectors. When the outputs from Tier 1 and Tier 2 were combined, most runs were classified as Normal (73.3%), 13.6% as Environment Stress, 8% as Latent Logic Issue and 5.1% as Systematic Failure.

The empirical results indicate that the proposed framework provides an end-to-end pipeline. It starts with feature engineering, then applies unsupervised anomaly detection on both run and process levels and uses explanation layer to perform root-cause analysis for distinguishing external and internal drivers. The framework is model-agnostic and hierarchical. Tier 1 captures any run-level features, while Tier 2 captures any per-action metrics, and the integration relies only on binomial anomaly indicators, which makes it robust to different platforms, algorithms and log schema variants. The use of SHAP explanation models over two groups of features (environment and process-level) decouples the anomaly attribution from anomaly detection, which allows for the implementation of different unsupervised anomaly detection models without changing the explanation and EII implementation.

The suggested hierarchical design shows better diagnostic differentiation between environmental and process-related anomalies than other RPA anomaly detection approaches that function at a single granularity level (e.g., TinyML-based or deep-learning models). The validation of the framework confirms the significant advantages of employing structured anomaly attribution in combination with interpretation tier. Unlike existing studies (e.g. Deng, 2024; Essakki & Bagavathi, 2025) where the model is primarily focused on detection performance, our approach combines SHAP-based explanations and hierarchical classification. This way gives meaningful information instead of mere independent anomaly detection indices. This allows you to be more granular than traditional monolithic anomaly scoring frameworks since you can assess the anomalies (Prucha, 2023).

The framework is tested on a desktop computer (Intel i3-8130U CPU at 2.2 GHz, 12 GB RAM, 2 GB Nvidia GPU) to measure the computational costs on 10000 runs. The training of tier 1 models (IF and AE) completed in the order of 3-4 minutes and the inference for a new run below second. Training the LSTM AE on Tier 2 required 5-6 minutes. The most computation cost is associated with the generating of SHAP-based explanations, however the total overhead remained modest, typically under 10 minutes for the whole dataset and per-run scoring in the second's range.

These measurements indicate that, even on modest laptop-class hardware, the hierarchical framework can be retrained periodically in batch mode and executed online for continuous monitoring.

Conclusion

This practical validation of the proposed framework both on synthetic and on real case data from RPA system confirms that the integration of two tiers for anomaly detection at run and process levels enhances the capability to correctly detect and attribute the root-cause for process failures. The hierarchical characteristic of the framework integrating not only detection tiers but also explanation tier enables the organizations to move from detection only environment into actionable and interpretable diagnostic RPA systems. The External Influence Index allows for deeper understanding of the underlying anomaly drivers which in turn enables the operators to take the appropriate remediation and prevention policies.

Results from the practical experiments demonstrate that the hierarchical design leads to excellent performance on detecting global run-level anomalies (AUC up to 0.96 at Tier 1) while keeping recall to internal process-level degradation at Tier 2. The moderate agreement between models on both tiers are statistically significant which confirm their complementary rather than redundant detection behaviour. With the addition of explanation tier based on SHAP-based explanations, the framework transforms multi-level telemetry into a well-structured action-oriented diagnostic knowledge.

The effect of the proposed methodology can be viewed not only in the methodological dimension. It also outlines a practical approach to integrate anomaly detection results into monitoring dashboards and remediation workflows which has a great potential in governance-oriented RPA management practices as well.

Despite the results obtained, several limitations must be acknowledged. The empirical validation is performed on a structured synthetic dataset, which, although designed to simulate real-world RPA telemetry, cannot fully capture the variability and noise of production systems. The injected anomalies also might not reflect all forms of failure. These limitations suggest that additional validation over heterogeneous real-world RPA deployments and operational conditions is also required.

As a pilot test experiment and as an attempt to overcome the mentioned above limitations, the framework was validated on 325 runs of university process for automating scientific project submissions. Results obtained were similar to the results of implementation on synthetic data. Future research may extend this validation to larger real-world production logs collected from different RPA environments. Another future research direction is oriented towards the exploration of more richer attribution metrics to calculate the External Influence Index, which is expected to provide more detailed and actionable insight on anomaly drivers. A promising future research area is related to the integration of predictive modelling and reinforcement learning to achieve proactive anomaly prevention and remediation. This would help the transition from reactive diagnostics to a more adaptive anticipatory approach.

Acknowledgment

This research is financed by the project “PNI-KS24-03-ISAUPM - Integrated system for automating university projects management”.

References

- Cao, Y., Cao, Y., Xiang, H., Zhang, Y., Zhu, Y., & Ting, K. (2024). Anomaly Detection Based on Isolation Mechanisms: A Survey. *Mach. Intell. Res.* 22, 849–865 doi: 10.1007/s11633-025-1554-4.
- Chirayil Nandakumar, S., Mitchell, D., Suphi Erden, M., Flynn, D., & Lim, T. (2024). Anomaly Detection Methods in Autonomous Robotic Missions, *Sensors (Basel, Switzerland)*. Multidisciplinary Digital Publishing Institute (MDPI), 24(4), p. 1330. doi: 10.3390/s24041330.
- Deng, B., Wang, D., Li, H., Wang, Z., Zhang, L. & Shu, Y. (2024). Deep Learning Enabled Scheduling System Data Anomaly Analysis in Robotic Process Automation, *International Conference on Distributed Systems, Computer Networks and Cybersecurity, ICDSCNC 2024*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/ICDSCNC62492.2024.10939815.
- Essakki, P. & Bagavathi, R.(2025). Lightweight TinyML-based Anomaly Detection for Intelligent Robotic Process Automation Workflows, *Proceedings of the 4th International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2025*. Institute of Electrical and Electronics Engineers Inc., pp. 93–97. doi: 10.1109/ICIMIA67127.2025.11200680.
- Kumar, V., Srivastava, V., Mahjabin, S., Pal, A., Kluttermann, S. & Muller, E. (2024). Autoencoder Optimization for Anomaly Detection: A Comparative Study with Shallow Algorithms, *Proceedings of the International Joint Conference on Neural Networks*. Institute of Electrical and Electronics Engineers Inc. doi: 10.1109/IJCNN60899.2024.10650057.
- Landauer, M., Onder, S., Skopik, F. & Wurzenberger, M. (2023a). Deep learning for anomaly detection in log data: A survey, *Machine Learning with Applications*. Elsevier, 12(1), p. 100470. doi: 10.1016/j.mlwa.2023.100470.
- Muaz, M., Sajid, S., Schulze, T., Liu, C., Klasen, N., & Drescher, B. (2025). Explainable AI for correct root cause analysis of product quality in injection moulding, *Journal of Manufacturing Processes*. Elsevier, 145, pp. 371–380. doi: 10.1016/j.jmapro.2025.03.114.
- Prucha, P. (2023). Towards Discovering Erratic Behavior in Robotic Process Automation with Statistical Process Control, *Information Systems and e-Business Management*. Springer Science and Business Media Deutschland GmbH, 22(4), pp. 741–758. <http://arxiv.org/abs/2305.00205>
- Renner, J., Maestre, R., Ochoa, I. & Madrid, B. (2025). REX: Causal discovery based on machine learning and explainability techniques. doi: 10.1016/j.patcog.2025.112491.
- Santos, P., Rocha, M., & Krohling, A. (2025). Combining SHAP and Causal Analysis for Interpretable Fault Detection in Industrial Processes. <https://arxiv.org/pdf/2510.23817v1>
- Shi, Y., Zhang, N., Song, X., Li, H. & Zhu, Q. (2024). Novel approach for industrial process anomaly detection based on process mining, *Journal of Process Control*. Elsevier, 136, p. 103165. doi: 10.1016/j.jprocont.2024.103165.
- Shukla, V., Shukla, A., Surya, S., & Shukla, S. (2025). A systematic survey: role of deep learning-based image anomaly detection in industrial inspection contexts, *Frontiers in Robotics and AI*. Frontiers Media SA, 12, p. 1554196. doi: 10.3389/frobt.2025.1554196.
- Tayeh, T. & Shami, A. (2021). Anomaly Detection in Smart Manufacturing with an Application Focus on Robotic Finishing Systems: A Review. <https://arXiv:2107.05053>