## Imperfection patterns in Praeclarus

* Distorted label
  + Contextual
  + Jaro-Winkler
  + Levenshtein
* Polluted label
  + Contextual
* Synonymous label
* Duplicates
  + Within activity
  + Within log
  + Within trace
* Future Entry
* Missing Timestamp
* Precision

## Imperfection patterns not in Praeclarus

* Form Based event capture
* Inadvertent Time Travel
* Unanchored event
* Scattered event
* Elusive case
* Scattered case
* Collateral events

## Detection and Remedies

### Form based event capture (timestamp duplication)

#### Detection:

Search for groups of events with identical case identifier and timestamp values.

Identify 'marker' events by searching for activity names resembling known form field labels.

Check timestamps of 'marker' events for consistency.

Regular occurrences of groups with the same timestamp indicate the pattern.

#### Remedies:

**Simple Aggregation Approach:** (Applicable when it makes sense to represent form information as a single process step.)

* Aggregate events with the same timestamp into one event.
* Create an additional attribute with a complex data structure to capture relevant data.

**Advanced Aggregation Scenarios:**

* Identify situations where timestamp information is recorded in a column different from the 'timestamp' column.
* Partially sequentialize events using information extracted from the alternative column.

**Complex Update Scenarios:**

* Update Recording Variations:- Distinguish between scenarios where updates result in recording all fields versus only changed fields.
* Identification of Changes:-Identify the specific information that has changed during updates.
* Process Step Consideration:-Understand the process steps leading to data item updates.
* Aggregation of Changed Data Items:-Aggregate changed data items into one or more events as needed.

### Inadvertent Time Travel (improbable timestamp in a process sequence)

#### Detection:

1. Identify Temporally Ordered Activities:

Determine pairs of activities with a strict temporal ordering property (e.g., a1 before a2).

1. Spot Deviations in Event Ordering:

Examine cases to identify deviations in the temporal ordering of activities.

Look for cases where misplaced events violate the expected sequential order.

1. Locate Misplaced Events:

Pinpoint the misplaced events within cases where the strict ordering is violated.

Identify pairs of events (e.g., a1 and a2) that are out of sequence.

1. Test Timestamp Adjustment:

Experiment with changing the timestamps of misplaced events based on known rules or constraints.

Aim to adjust timestamps to place the events back into their proper ordering within the overall trace.

1. Verify Chronological Alignment:

Validate whether adjusting timestamps resolves the chronological misalignment.

Ensure that the corrected timestamp sequence aligns with the expected temporal order of activities.

1. Confirm Pattern Existence:

If adjusting timestamps successfully corrects the ordering, consider the existence of the 'Inadvertent Time Travel' pattern in the log.

Confirm that the corrected order reflects the domain's 'ground truth' for the sequential occurrence of activities.

#### Remedies:

1. Define Minimum Ordering Restrictions:

Establish minimum restrictions on the temporal ordering of activities in the log based on domain knowledge.

1. Identify Violating Traces:

Examine each trace in the log to identify cases where the temporal ordering restrictions are violated.

Pinpoint traces with misplaced events that deviate from the expected sequential order.

1. Apply Timestamp Fixes:

Experiment with timestamp adjustments to correct the chronological misalignment of events in violating traces.

Common fixes include:

Adding or subtracting one day from the timestamp.

Flipping timestamp values based on proximity in a standard keyboard layout.

1. Validate Corrected Order:

Verify if the applied timestamp fixes successfully re-order events within the trace, aligning with the expected temporal sequence.

Ensure that the corrected timestamp sequence adheres to the minimum ordering restrictions.

1. Replace Erroneous Timestamps:

If a timestamp fix resolves the ordering violation, replace the original erroneous timestamp with the corrected value.

Use the adjusted timestamp to maintain a proper temporal order for the events within the trace.

1. Confirm Elimination of Pattern:

Validate that the corrected traces no longer violate the minimum ordering restrictions.

Confirm that the repaired timestamps accurately represent the chronological sequence of activities.

### Unanchored event (timestamp in improper format)

#### Detection:

1. Unexpected Process Model Ordering:

Examine process models derived from the event log for unexpected and potentially incorrect ordering of activities.

Anomalies in process flow may indicate issues with the chronological accuracy of events.

1. Analysis of Working/Waiting Times:

Analyze working and/or waiting times extracted from the event log.

Identify instances of unreasonably long or short durations that may signal irregularities in timestamp data.

1. Missing Timestamp Information:

Investigate the presence of missing timestamp information across numerous events in the log.

Absence of timestamp values may indicate parsing issues where the tool fails to interpret and incorporate the timestamps correctly.

1. Timestamp Value Range Checks:

Perform checks on timestamp element values to identify patterns outside the expected range.

Look for elements with values that fall outside the anticipated range or span only part of the expected range (e.g., 'day' values restricted to [1…12]).

1. Identification of Timestamp Element Anomalies:

Scrutinize timestamp elements (e.g., day, month) for anomalies such as values that seem swapped or partially within expected ranges.

Recognize patterns like 'day' values indicating month/day interchange.

1. Cross-Validation of Indicators:

Cross-validate multiple indicators, including unexpected process models, abnormal working/waiting times, missing timestamps, and element anomalies.

A convergence of these indicators strengthens the detection of the unanchored event pattern.

#### Remedies:

1. Prevent Misinterpretation by Tools:

Ensure tools used for importing the event log do not misinterpret timestamp information without issuing warnings.

Verify that the tools correctly parse and handle timestamp formats.

1. Modify Timestamp Fields:

Add special characters (e.g., asterisks) before and after timestamp values.

This modification aims to disable the built-in timestamp interpretation mechanism in tools that may cause inadvertent misinterpretation.

1. String Manipulation Techniques:

Apply string manipulation techniques, such as find and replace, to reformat the modified timestamp strings into the correct timestamp format.

1. Reformat Timestamp Values:

Employ appropriate string manipulation operations to transform modified timestamp strings into the desired timestamp format.

Ensure the final timestamp format is consistent and compatible with the expected format for subsequent process mining analyses.

1. Quality Assurance Checks:

Conduct quality assurance checks to verify the accuracy of the reformatting process.

Confirm that timestamps are correctly interpreted and reflect the chronological order of events.

1. Documentation and Tool Configuration:

Document the steps taken for timestamp modification and reformatting.

Configure tools to recognize and correctly interpret the modified timestamp fields during future imports.

### Scattered event (event which can further be derived into new events)

#### Detection:

1. Identify Columns:

Identify guiding and target columns within the event log.

1. Recognize Marker Values:

Identify marker values within the guiding column(s).

1. Understand Information Extraction:

Understand how information for new events can be extracted from the guiding column based on marker values.

1. Manual Inspection:

Manually inspect attribute values within guiding columns to identify patterns and hidden information.

1. Domain Expertise:

Seek assistance from domain experts to recognize encoding mechanisms and patterns within attribute values.

1. Consistency Check:

Check for consistency and recurring structures in attribute values indicative of scattered event patterns.

#### Remedy:

. Given the variety of manners in which this pattern may manifest itself, a generic solution to fix this issue is unlikely.

Nevertheless, once the location of the information from which new events can be re-constructed is known, it is possible to develop a tool to automate the creation of the new events.

### Elusive Case (missing an explicit case id)

#### Detection:

Detecting the 'Elusive Case' pattern involves a trial-and-error approach to finding a common attribute that can serve as a case identifier. Here are the brief steps on how to detect the Elusive Case pattern:

1. Attribute Tagging:

Randomly select one or more attributes from the event log and tag them as potential 'case identifier' attribute(s).

1. Process Model Discovery:

Load the event log, considering the tagged attribute(s) as the case identifier, into a process mining tool.

1. Model Evaluation:

Examine the resulting process model to check if it captures a complete and meaningful process with reasonable temporal dependencies between events.

1. Iterative Testing:

If the first attempt does not yield a satisfactory model, repeat the process by tagging different attributes until a suitable case identifier is found.

1. Evaluate Model Quality:

Assess the quality of the process models obtained with different attribute tags. Look for models that represent the expected process structure.

1. Identification of an Elusive Case:

If no attribute, when tagged, produces an acceptable process model with meaningful temporal dependencies, it suggests the presence of the 'Elusive Case' pattern.

#### Remedy:

1. Define Case Identification Criteria:

Establish criteria for identifying cases, defining what constitutes a unique case. This may involve considering multiple attributes or external information.

1. Utilize External Information:

Leverage information from external sources that can provide insights into the grouping of events into meaningful cases. This could be data from associated tables, databases, or supplementary records.

1. Correlation with External Data:

Correlate the events in the event log with the external information, using common identifiers or attributes to link events to their respective cases. For example, match GPS events with journey information from a journey planner table.

1. Mapping Events to Cases:

Based on the correlation results, map each event to its corresponding case identifier. This involves assigning the correct case identifier to events according to the established criteria.

1. Validation and Quality Check:

Validate the mapped cases to ensure accuracy and consistency. Perform quality checks to confirm that the identified cases align with the expected process structure.

1. Update Event Log:

Update the event log with the correct case identifiers assigned to each event. This involves modifying or adding attributes in the event log to reflect the repaired case structure.

1. Reprocess for Process Mining:

After repairing the case identification, reprocess the event log using process mining tools to generate accurate process models that reflect the identified cases.

### Scattered case (missing key process steps)

#### Detection:

1. Define Expected Activity List:

Define the expected list of activities that should be recorded in the event log based on the known process or workflow. Include all relevant predecessor and successor activities for each main activity.

1. Compare Activity Lists:

Compare the list of unique activity names recorded in the event log against the expected list of activities. Identify any activities that are missing or not recorded in the log.

1. Check Predecessor and Successor Activities:

Examine each identified activity in the event log and check for the presence of expected predecessor and successor activities. Verify that the recorded sequence of activities aligns with the expected process flow.

1. Evaluate Activity Frequencies:

Assess the frequencies of each activity, its predecessor, and successor activities in the log. Ensure that the frequencies match the expected relationships. A significant discrepancy in frequencies may indicate partial trace incompleteness.

1. Explore Multiple Logs:

Explore additional event logs or data sources that may contain information about the entire process. Look for logs that provide a comprehensive view of activities, including their sequencing.

1. Analyze Coherence:

Analyze the coherence and logical flow of activities within individual cases in the log. Confirm that the recorded activities make sense in the context of the process, and identify any gaps or discontinuities.

1. Use Expected Activity Patterns:

Utilize known patterns of activity sequences in the process. Check if the recorded sequences match these expected patterns, and identify deviations that may indicate missing activities.

1. Quantitative Discrepancy Analysis:

Quantitatively analyze discrepancies in activity frequencies, especially for key activities and their expected predecessors and successors. Establish thresholds for acceptable deviations to flag potential trace incompleteness.

#### Remedy:

1. Identify Unique Identifiers:

Determine if there are unique identifiers (e.g., case identifiers) in the various source logs that can be used for record linkage.

1. Merge Using Common Identifiers:

If all data sources share exactly the same case identifiers, merge the source logs into one event log directly.

1. Utilize Global Unique Identifier:

When different case identifiers are used, but a global unique identifier exists (e.g., 'Master Record Number'), employ it as the merge key. Set the case identifier as either the global unique identifier or one of the existing case identifiers.

1. Apply Record Linkage Techniques:

If no unique identifier exists, employ standard record linkage techniques to match events across different logs. Techniques may include those outlined by Newcombe, formalized by Fellegi and Sunter, or other merge-purge techniques.

1. Follow Record Linkage Process:

Implement the record linkage and de-duplication process, considering factors such as similarity metrics, thresholds, and blocking methods.

1. Verify Merged Events:

Verify the correctness of the merged events by checking if events from various sources are properly attributed to cases and exhibit meaningful correlations.

1. Ensure Data Consistency:

Ensure consistency in the merged data, especially regarding case identifiers and any additional attributes critical for linking events.

1. Handle No Unique Identifier Scenario:

If no unique identifier is available, carefully perform record linkage using relevant attributes, taking into account potential variations in the data.

1. Refer to Existing Techniques:

Refer to established record linkage and de-duplication techniques for guidance, considering literature such as Newcombe's and Fellegi-Sunter's methods.

### Collateral events

#### Detection:

1. Examine Timestamps:

Analyze the timestamps of activities in the event log. Look for groups of activities with timestamps that are very close to each other, typically within seconds.

1. Identify Similar Labels:

Identify activities with very similar labels or logical consequences from one another within the identified timestamp groups. Activities in the same timestamp group should be logically related.

1. Engage Domain Experts:

Collaborate with domain experts who are familiar with the functionality of the systems generating the event log and have a comprehensive understanding of the overall process steps. Their insights are crucial for distinguishing meaningful process steps from overlapping activities.

1. Evaluate Logical Flow:

Assess the logical flow of activities within the timestamp groups. Determine whether the events represent distinct process steps or if they are part of a broader, interrelated set of activities.

1. Consider System Functionality:

Take into account the functionality of the systems from which the event log is extracted. Understand how system interactions and processes may lead to multiple events occurring in close temporal proximity.

1. Look for Notification Patterns:

Specifically, watch for patterns related to notifications, emails, or system-generated alerts that may result in a cluster of events within a short time frame.

1. Check High Event Count:

Be alert to an unusually high number of events within most traces in the event log. A significantly high event count, especially exceeding 100 events per trace, could indicate the presence of overlapping activities.

#### Remedy:

1. Develop a Knowledge Base:

Create a knowledge base that records information about activity names that, when occurring together within a short time period, should be merged into a single activity.

1. Include Timestamp Information:

Specify in the knowledge base the name of the merged activity and determine the timestamp that should be used (e.g., earliest or latest timestamp). This ensures consistency in merging overlapping activities.

1. Involve Process and System Analysts:

Engage process analysts and system analysts in the development of the knowledge base. Leverage their understanding of how process steps are created by the system and identify the set of collateral events triggered by the system.

1. Collaborate with Domain Experts:

Collaborate with domain experts who have a comprehensive understanding of the process steps and can contribute valuable insights into the logical grouping of activities.

1. Capture System-Generated Events:

Ensure that the knowledge base captures information about system-generated events, especially those triggered as collateral events within a short time period.

1. Specify Merged Activity Names:

Clearly specify in the knowledge base the name of the merged activity that should result from overlapping activities. This contributes to reducing the total number of unique activities in the event log.

1. Define Timestamp Handling:

Define rules within the knowledge base for handling timestamps, such as selecting the earliest or latest timestamp for the merged activity. Consistent timestamp handling is crucial for maintaining chronological accuracy.