Event-Driven Process Mining for Failure Cascade Detection in Vehicle Telemetry Systems

Abstract

Modern motorsports teams generate gigabytes of telemetry data per race, yet analysis of this data is still predominantly a manual operation. In this project, we propose the application of process mining techniques from the business intelligence world for discovery and analysis of discrete system events and states on vehicle telemetry data. We present a methodology of event classification using threshold and signal process algorithms. These events are organized into temporal traces using a time-window approach centered on an event of interest. To construct the event graphs, we utilize the PM4PY library to construct Directly-Follows Graphs (DFGs) that visualize event relationships, transition probabilities, and timing characteristics.

1 Background Information

1.1 The Challenge of Telemetry Analysis

Motorsports teams generate gigabytes of telemetry data per session. Diagnosing failures in this environment requires expert knowledge of system behavior. When a failure occurs, engineers must manually trace backward through the data, identifying which events preceded the failure and hypothesizing causal relationships.

1.2 Process Mining: A Solution from Business Intelligence

Process mining is a family of techniques developed by a Dutch computer scientist by the name of Wil van der Aalst for the analysis of business and manufacturing workflows. This project addresses that gap by introducing a methodology to transform continuous telemetry data into discrete system events suitable for process mining and analysis.

1.3 Target Use Cases

This methodology targets several use cases for event analysis.

Fault Tree Analysis identify how failures cascade across a system by observing how minor events escalate into major failures. Process mining reveals the statistical patterns of failure propagation across many instances.

Event Tree Analysis to discover which system events are related to each other, enabling engineers to understand the forward consequences of specific conditions or actions.

Variant Analysis to identify the nominal sequence of events during normal operation and contrast it with non-nominal or defect paths, highlighting where behavior diverges from expectations.

2 Implementation Approach

In the event classification stage, continuous sensor telemetry is classified into discrete events using an `EventExtractor` class. The extractor supports threshold-based detection to detect when a sensor value crosses a defined limit, combined condition detection, and local extrema detection using `scipy` to identify peaks and valleys in sensor data.

The `CaseGenerator` class implements a temporal trace to draw a time-window around an event of interest (ex: "bumpstop hit" or "full throttle" event), constructing a local trace.

Event discovery can be performed using the PM4PY library. Directly-Follows Graphs (DFG) are generated, showing the order of events, the average timing between events, and the transition counts between states. Variant analysis can also be performed to identify all unique event sequences across cases.

3 Data Collection and Analysis

This project uses the UCONN FSAE 2016 Endurance dataset, publicly available on HuggingFace. The dataset contains telemetry from a Formula SAE racecar during an endurance event, recorded by an AiM data acquisition system.

4 Schedule and Milestones

Phase 1: Foundation (Week 1-2, 1/23 - 2/5)

- Deliverable: Literature Research (due 1/30/26)

Phase 2: Core Implementation (Week 3-5, 2/6 - 2/26)

- Deliverable: Project Progress and Preliminary Analysis (due 2/20/26)

Phase 3: Analysis and Documentation (Week 6-7, 2/27 - 3/6)

- Deliverable: Project Presentation (due 3/5/26)
- Deliverable: Project Report (due 3/6/26)

5 Challenges & Risk Mitigation

Threshold selection risks generating events from noise or missing genuine events from overly conservative limits. We mitigate this through sensitivity analysis across threshold ranges.

Time-window sizing: windows too small miss relevant context, while windows too large introduce noise and blur causal relationships. We address this by testing multiple window sizes and comparing the resulting process models.

Interpretability challenges emerge when many event types create overly complex DFGs that are difficult to analyze.