📄 8.1 – Event Definition & States

Purpose

In earlier phases, anomalies were detected and printed to the console, but not treated as actual system “events”. Phase 8 introduces a standard event model that makes faults **portable**, **loggable**, and **actionable** both on-device and off-device.

Event Model

An event is defined as:

```
Event = {
  level,
  harness,
  rule_name,
  message,
  ecu_a_v,
  ecu_b_v,
  dcdc_v,
  ts
}
```

Attributes Explained

Field	Meaning
`level`	Severity state (EARLY, ALERT)
`harness`	Component classification (A, B, C)
`rule_name`	Name of triggering logic rule
`message`	Human-readable description
`ecu_a_v`	Measured ECU_A supply voltage
`ecu_b_v`	Measured ECU_B supply voltage

`dcdc_v`	DCDC reference voltage
`ts`	Timestamp (UTC ISO-8601)

Event Severity States

Two functional states were introduced:

EARLY State

- Indicates **deviation or drift**
- Signals that something is **changing abnormally**
- No immediate functional failure
- Useful for **predictive maintenance** (RUL estimation later)

Triggered by rules such as:

```
HARNESS_A_DRIFT
HARNESS_B_DRIFT
HARNESS_BALANCE_ERROR
```

ALERT State

- Indicates **functional failure**
- Signal out-of-spec relative to DCDC reference
- Corresponds to **fault insertion** demos
- Localizes failure to:

```
HARNESS_A
HARNESS_B
HARNESS_C (shared)
```

Triggered by rules such as:

```
HARNESS_A_LOW_VS_DCDC
HARNESS_B_LOW_VS_DCDC
HARNESS_C_BOTH_LOW_VS_DCDC
```

Why Two States?

This mirrors industrial predictive stacks:

```
EARLY  → predictive
ALERT  → corrective
```

EARLY provides:

- time margin
- failure ordering
- degradation trajectory

ALERT provides:

- root cause
- severity
- fault classification

Together, they allow modeling:

- degradation → failure → recovery