NEXUS-1 · ROOT-CAUSE ENGINE

# FROM FLOOD TO CAUSE

Deterministic, Grounded, and Auditable Root-Cause Analysis for Alarm-Driven Systems

ALARM FLOOD

ONE ORIGIN · CAUSAL COLLAPSE

GRIGORIOS AGATHANGELIDIS

A NEXUS-1 Companion · live browser-based root-cause console

gregory82gr.github.io/Nexus-1-phase-0 · Preview build

CHAPTER 1

# Anatomy of an Alarm Flood

## What you already know

You have watched a dashboard go red. A service degrades, and within moments your monitoring fills with alerts — latency, error rate, queue depth, a dozen dependent services all paging at once. You already know the feeling that follows: the alerts are real, but they are not telling you where to look. The page that fired first was not necessarily the cause, and the service with the most alerts is often just the one with the most neighbours. The flood is genuine information arriving in an order that actively misleads.

A plant control room is the same problem with the volume turned up and the stakes raised. The physics is unfamiliar; the failure shape is not.

## The wall of correct, useless lights

On the night of 28 March 1979, the operators at Three Mile Island Unit 2 faced more than a hundred alarms with no means of telling which mattered. The annunciator panel was, in a narrow sense, working perfectly: every light that lit had a real condition behind it. But a relief valve had stuck open, and the indicator the crew trusted showed that the valve had been *commanded* shut — not that it actually was. The panel reported the instruction, the plant obeyed physics, and for hours the two diverged while the alarms kept screaming in chorus. The accident was not caused by missing information. It was caused by a wall of correct information with no structure to say which fact was the origin and which were echoes.

That distinction — between a signal and its echoes — is the subject of this book. The industry's response to Three Mile Island reshaped control-room design and gave us the human-factors and alarm-management standards we will lean on later (NUREG-0700, then EEMUA 191 and ISA-18.2). But standards manage the flood; they do not, by themselves, find the source inside it. For that we need a method.

## The universal pattern

Strip away the domain and every alarm flood has the same anatomy: a small number of true origins, a much larger number of downstream effects, and a propagation path in time that connects them. Consider the worked incident this book returns to, drawn live from the console: **EVT-2026-0418**, a feedwater disturbance on Unit 1. Fourteen alarms arrive over about four minutes and resolve, on inspection, into one origin and a handful of cascade strands. The origin is a single modest flag — a feedwater control valve, FV-104, with rising actuator latency. The most conspicuous node in the flood — a pump bearing running hot and rough, alarming twice — sits several links downstream of it.

Figure 1.1 shows the same incident ordered two ways. Rank the components by how much they alarmed, as the panel on the left does, and the pump bearing leads while FV-104 — a single low-severity alarm — sits at the very bottom. Rank them by causal origin, as the panel on the right does, and the order inverts: FV-104 rises to the top with the largest causal weight, and the bearing drops to a *proximate* symptom. Same fourteen alarms, two questions, two answers. Only one of the

questions is the one you actually asked.

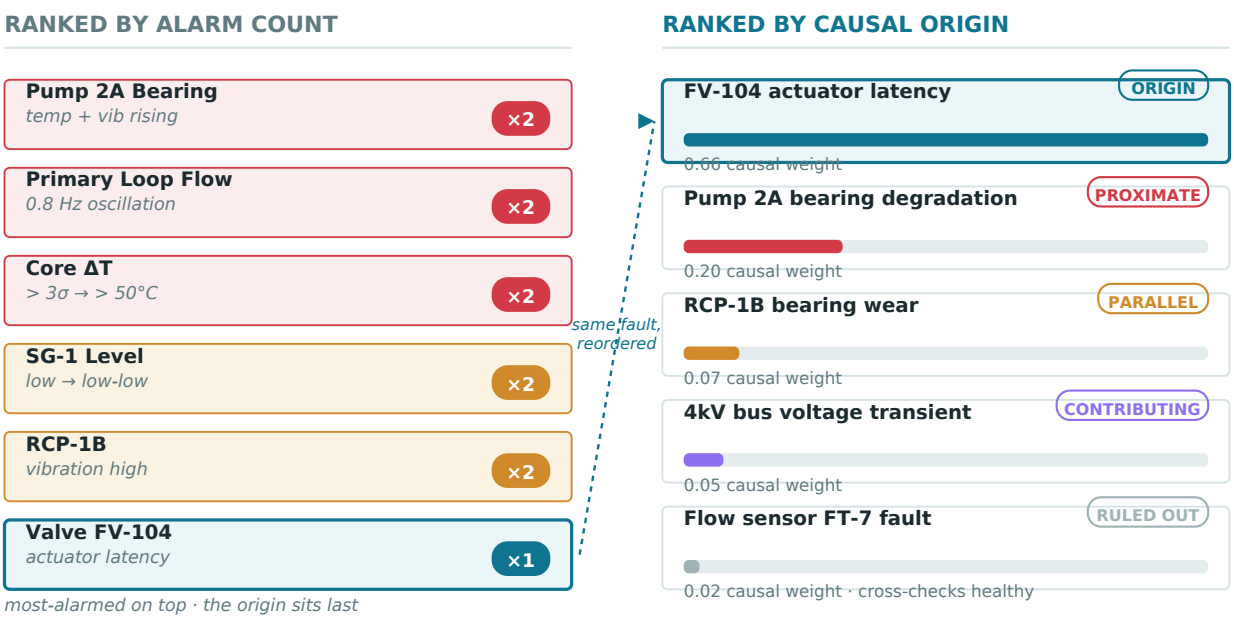


***Figure 1.1*** *— EVT-2026-0418 ranked two ways. By alarm count the pump bearing leads and the valve origin is last; by causal origin FV-104 leads at 0.66 and the bearing falls to a downstream proximate symptom. The causal weights are illustrative, derived from the graph's edge weights — not identified from plant data.*

## Reading the cascade

The raw timeline makes the trap concrete. The first alarm to fire, at 17:07:58, is a primary-loop flow oscillation — a high-severity, attention-grabbing event near the *end* of the causal chain. The valve that started everything does not announce itself until 17:11:04, more than three minutes later, as a single low-priority latency warning sandwiched between a steam-generator level alarm and a voltage dip. Neither *loudest* nor *first* points to it. An alarm fires when a measurement crosses a configured threshold, and a sluggish, high-threshold valve signal crosses late even when the fault behind it came early. Arrival order is not causal order — which is the second reason, after loudness, that ranking the flood by its own surface features fails.

## One origin, five strands

Collapse the flood onto the causal structure and the fourteen alarms fall into one origin and five strands. The **feedwater path** carries the fault forward: FV-104 starves SG-1, whose level drops, which forces FW Pump 2A to overspeed, which loads and heats the pump bearing. The **thermal-hydraulic cascade** runs on from there — loop-flow oscillation, a core temperature-difference excursion, and finally the reactor trip that ends the sequence. A third strand is a **parallel contributor**: RCP-1B carries its own vibration into the loop oscillation, overlapping the cascade without originating it. A fourth is a **data-informed contributor** — a 4kV bus voltage transient that the data layer proposes as a minor stressor on the pump, carried at low weight. The fifth strand is the one most diagnostic tools never show you: a **ruled-out** candidate. Raw correlation flagged Flow Sensor FT-7, the engine tested it, the sensor cross-checked healthy, and the candidate was rejected — and that rejection is recorded, not discarded. A system that cannot show you what it dismissed cannot be audited; this one keeps its refusals in plain sight, a habit we make rigorous in Chapter 6.

**SEE IT IN NEXUS-1**

Open the **Alarm Flood** screen and load EVT-2026-0418. The default view lists the fourteen alarms by arrival; the flow oscillation and the bearing dominate, and FV-104 is easy to miss. Switch the **Ranking** screen between *by count* and *by causal origin* and watch FV-104 travel from the bottom of one list to the top of the other. The console calls this collapse the ISA-18.2 alarm-management payoff: the operator is oriented, not drowning.

## Standards manage the flood; a method finds the source

It is worth being precise about where the alarm-management standards fit. EEMUA 191 and ISA-18.2 give us the discipline of rationalising alarms — suppressing nuisance chatter, assigning priorities, managing floods so a crew is not buried under a hundred simultaneous annunciations. That discipline is necessary, and this book assumes it. But rationalisation organises the flood; it does not diagnose it. Priorities encode how urgently a human should look at an alarm, not whether that alarm is a cause or an echo. The method in the chapters ahead sits on top of a well-managed alarm system and answers the question the standards deliberately leave open: of everything that fired, what started it?

## First sight of the node test

Here, in one sentence, is the whole engine in miniature. Take any alarmed node, treat it as the hypothesised root, walk the causal graph forward from it, and count how many of the other alarms in the flood it can explain. Run that test from FV-104 and it accounts for the bulk of the cascade; run it from the pump bearing and it explains almost nothing, because nothing upstream of a symptom follows from the symptom. The ranking in Figure 1.1 is nothing more than this test performed from every alarmed node and sorted by what each one explains. Chapter 2 builds the graph that makes the walk possible and the timing that keeps it honest.

**HONEST BOUNDARY**

Counting is not always wrong. Alarm-rate and alarm-count metrics are genuinely useful for managing operator load and spotting nuisance alarms, and the standards above are built on them. What this chapter rejects is narrower: using *loudness, count, or arrival order to infer cause.* Those features tell you where the noise is and how to triage human attention; they do not tell you where the trouble started. The two questions are different, and conflating them is the habit this book is written to break.