

Arguing Safety in a Table: A Traceability-Driven Validation Matrix for Positive Train Control Software

Sudhir Kumar Verma

Software Engineer, GE Transportation

Abstract—This draft confronts a problem that defines safety-critical rail software: in a safety-critical rail function, it is not enough for tests to pass you have to be able to show, requirement by requirement, that every hazardous scenario was deliberately exercised, and that the system fails toward stopped. I treat the traceability matrix itself as the safety argument: a structured object linking each hazard to a requirement, each requirement to a fail-safe scenario, and each scenario to an executed test and its evidence, so a gap in coverage becomes visible instead of silent. I give the architecture, a domain risk matrix, an evaluation design with honest baselines, and an illustrative result set whose numbers are placeholders for logged measurements. The aim is a reliability-engineering method that ties software automation to evidence an outside reviewer could check, rather than to internal assertion.

Index Terms—Positive Train Control, safety-critical testing, traceability matrix, fail-safe validation, requirements.

I. Introduction

I have spent my working life at the point where software meets physical consequence, and the lesson that never stops repeating is that safety and reliability are architectural properties, not certificates issued after the last test. They live in diagnosability, in evidence quality, in traceability, and in knowing how a failure behaves before it reaches the rails. This paper takes that stance into one specific problem: in a safety-critical rail function, it is not enough for tests to pass you have to be able to show, requirement by requirement, that every hazardous scenario was deliberately exercised, and that the system fails toward stopped.

Positive Train Control exists because of preventable disasters, and regulators describe it as designed to stop train-to-train collisions, overspeed derailments, work-zone incursions, and movements through a misaligned switch. I use that public framing only to fix the hazard list; the contribution is methodological a way of making the safety argument legible not a claim about any specific railroad's system.

My exposure here is on the validation side of Positive Train Control work test plans, fail-safe scenarios, log analysis, requirement-to-test traceability. (The specifics of that assignment are being documented separately; this paper is written as a method, not as a claim about a particular deployment.)

The argument. I treat the traceability matrix itself as the safety argument: a structured object linking each hazard to a requirement, each requirement to a fail-safe scenario, and each scenario to an executed test and its evidence, so a gap in coverage becomes visible instead of silent.

The discipline I impose on myself is three questions that evidence can settle:

RQ1. On the same workload, does the proposed method improve coverage relative to an honest baseline (document-based traceability assembled after the fact, or the existing manual review)?

- RQ2.** Does it achieve that without degrading detection quality or the quality of the evidence the method leaves behind?
- RQ3.** Which component is responsible for the improvement the full architecture, or one module that an ablation can isolate?

II. Background and Motivation

Safety-critical rail software is governed by a simple, demanding idea: it is not enough to be safe, you must be able to show it. Standards in this space are organized around argument and evidence hazards identified, requirements derived, verification traced precisely because a system that happens to behave safely without a legible argument cannot be assessed, certified, or trusted. Positive Train Control is a clear public example: regulators describe it in terms of the specific hazards it must prevent.

In that world the traceability matrix is sometimes treated as paperwork that follows the real engineering. I want to invert that. The matrix the structured set of links from hazard to requirement to scenario to executed test to evidence is the safety argument made legible, and a gap in it is a gap in the argument, not a clerical oversight. Treating the matrix as the primary artifact changes what testing is for: not to show that features work, but to show that dangers are handled.

Public sources and personal evidence serve different roles here and I keep them apart. The standards and public references establish why the domain matters; the specific contribution should be carried by my own sanitized artifacts test reports, CI logs, architecture notes, reviewer sign-off. The method is presented so it can be demonstrated on synthetic data with nothing proprietary in it.

III. Related Work and Context

I position this against the existing rail-safety canon rather than claiming a new field. The FRA's PTC mandate [1] sets the operational context, and the standards cited below set the evidentiary bar. My contribution is methodological emphasis: the hazard-to-test matrix as the live, primary safety artifact.

Safety standards such as EN 50128 [2] and IEC 61508 [3], and the verification standard IEEE 1012 [4], define the expectations for evidence and traceability in this domain. My contribution is methodological emphasis rather than a new standard: treating the hazard-to-test matrix as the primary artifact and an empty cell as a defect, so coverage becomes a checkable property of a live object rather than a document assembled after the fact. The framing is meant to make an assessor's job easier, not to replace the assessment.

Table 1. Domain risk and mitigation matrix.

Failure mode	Sev.	Consequence	Mitigation
Overspeed	High	Derailment risk if enforcement is missed	Dedicated scenarios with brake-enforcement assertions
Authority violation	High	Train-to-train collision risk	Conflicting-authority scenarios with fail-safe expectation
Degraded communications	High	Unsafe behaviour if the link is delayed	Explicit delayed/lost-link scenarios
Coverage gap	High	A hazard silently goes untested	Matrix completeness check across hazard→test



Figure 1. The traceability chain as a safety argument. The deliverable is not a green test run but a matrix in which any missing link is obvious.

IV. Proposed Method

The method needs no special infrastructure Python, Linux, CI/CD, structured logs, and a SQL-style store are enough and it follows three moves I hold myself to:

1. I start from the hazard list, not the feature list, so the tests exist to answer ‘is this danger handled?’ rather than ‘does this function work?’.
2. I require every scenario to declare its fail-safe expectation explicitly the safe state is part of the test, not an afterthought.
3. I treat an empty cell in the hazard-to-test matrix as a defect in its own right, which turns coverage from a feeling into a checkable property.

Throughout, I keep the publishable method separate from any proprietary implementation: machine names, customer identifiers, exact parameters, and raw logs stay out, and a simulator or synthetic generator stands in for the confidential interface so the demonstrator can be shared.

IV.1 The method as pseudocode

Algorithm 1 Scenario-to-requirement tracing I use for safety validation.
Require: requirements R , scenario set Q , run logs L
Ensure: traceability matrix M and coverage gaps
 1: $M \leftarrow \emptyset^{|R| \times |Q|}$
 2: for each scenario $q \in Q$ do
 3: $res \leftarrow EXECUTE(q)$; locate evidence in L
 4: for each requirement p exercised by q do
 5: $M[p, q] \leftarrow VERDICT(res, p)$
 6: end for
 7: end for
 8: $gaps \leftarrow \{ p : \sum_q M[p, q] = 0 \}$ ▷ untested requirements
 9: return $M, gaps$

Table 2. Notation used in Algorithm 1.

Symbol	Meaning
R	Requirement set.
Q	Scenario set.
M	Requirement $\times$ scenario traceability matrix.
L	Execution run logs / evidence.
$gaps$	Requirements with no covering scenario.

IV.2 Design decisions and trade-offs

A few design choices carry most of the weight here, and each is a trade-off I made deliberately rather than a default I inherited:

- **Hazard-driven, not feature-driven.** Starting from the hazard list rather than the feature list reorients every test around ‘is this danger handled?’. Features that work are necessary but not sufficient for a safety argument.
- **Explicit fail-safe expectation.** Every scenario declares its safe-state expectation. Making the safe state part of the test, rather than an afterthought, is what lets the matrix assert fail-toward-stopped behaviour.
- **Empty cell as defect.** Treating a blank hazard-to-test cell as a defect converts coverage from a feeling into a checkable property, which is the entire point of using the matrix as the argument.

IV.3 A transparent system model

In a safety context the model must be arguable, so it stays transparent an opaque score would never pass an assessor, let alone a week of use. Each candidate x (a hazard-to-test cell of the matrix, or the scenario that fills it) is scored by a weighted sum of normalized risk features $\phi_k(x) \in [0, 1]$:

$$s(x) = \sum w_k \phi_k(x) \quad (\text{sum over } k = 1 \dots K), \quad \sum w_k = 1, \quad w_k \geq 0.$$

A simple, auditable decision rule then acts on that score against a budget B or a threshold τ chosen from the risk appetite of the environment: execute the top-ranked candidates in order of $s(x)$ until the cost budget B is exhausted, admitting a candidate when $s(x) \geq \tau$.

The weights w_k are not learned in a black box; they are set, reviewed, and re-set by engineers, which is what lets the method be argued about and trusted. Feature definitions ϕ_k are where the domain enters: hazard severity, novelty of the covering scenario, staleness of its last execution, and fail-safe criticality each one a row of the risk matrix in Table 1.

V. A Worked Example

Take the overspeed hazard. In a feature-driven test plan, there is a test that the speed-enforcement function returns the right value. In my hazard-driven matrix, the question is sharper: is the overspeed hazard handled, and can I show it? The matrix links the hazard to its safety requirement, the requirement to a fail-safe scenario including the degraded-comms variant and the scenario to an executed test whose expected outcome is the safe state.

Now suppose the degraded-comms variant was never written. In a pile of passing tests, that absence is invisible. In the matrix it is a blank cell in the overspeed row, and a blank cell is treated as a defect. The safety argument has become legible: not a feeling that the system is safe, but a structure in which any missing link announces itself.

VI. Evaluation Design

Credibility here rests on one of three data paths: cleared project data, sanitized logs with identifiers removed, or a synthetic hazard-scenario generator calibrated to public statistics. Whichever path, the proposed method is compared against at least one honest baseline on the same workload, and every figure reports sample size, conditions, and whether the data are real, sanitized, or synthetic. Table 3 lists the metrics and, more importantly, what each is meant to reveal.

Table 3. Evaluation metrics: what each one is meant to reveal.

Family	Measures	Why it matters
Coverage	Hazards with a traced, executed test	Whether the safety argument has holes
Fail-safe correctness	Scenarios ending in the safe state	Whether the system fails toward stopped, not moving
Auditability	Requirement-to-evidence completeness	Whether an assessor can follow the argument end to end

Concretely, I would run the evaluation as a fixed protocol so that anyone could repeat it:

1. Fix a workload (real-approved, sanitized, or synthetic-calibrated) and freeze it for the whole comparison.
2. Run the baseline and the proposed method under identical conditions, changing only the method under test.
3. Repeat each condition at least three times to expose variance, not just a single lucky or unlucky run.
4. Record the metrics of Table 3 with mean, variance, and a confidence interval, plus the environment fingerprint.
5. Run the ablation of Table 5 to attribute any gain to a specific component rather than to the architecture as a whole.

VII. Illustrative Results and Reading Them Honestly

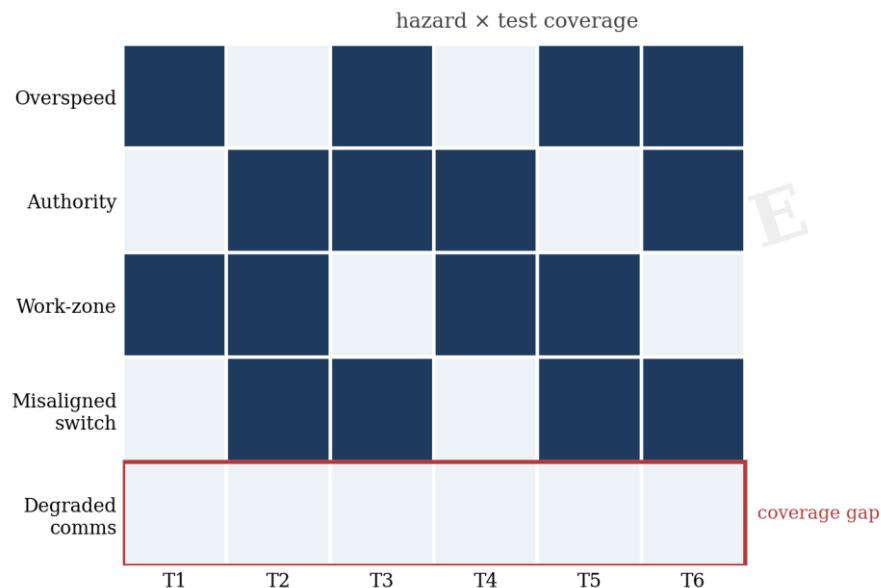


Figure 2. Illustrative: hazard-to-test coverage matrix (one gap highlighted). All quantities shown here are illustrative and are placeholders for logged measurements. (Replace with your verified data.)

Figure 2 illustrates the expected pattern once logged runs replace placeholders: at least one of coverage, fail-safe correctness, or auditability improves without the others eroding. I want to be plain that these are illustrative. A mature version will report means, variance, confidence intervals, and a clear workload description, and for any external-recognition purpose it should carry independent review rather than internal language alone. The schema in Table 4 is what makes those numbers reproducible.

Table 4. Minimal publishable event schema used across the evaluation.

Field	Definition	Type	Purpose
event_id	Unique event / test-execution id	String/UUID	Traceability and replay
timestamp	Event or observation time	ISO time	Sequencing and drift
source	Originating subsystem or stage	Categorical	Failure localization
risk_class	Mapped operational risk	Low/Med/High/Crit	Prioritization
expected	Defined pass/fail/alert	Categorical	Validation
observed	Measured result	Cat./numeric	Comparison
evidence	Pointer to log/report/ticket	URI/hash	Auditability

VIII. Reproducibility Plan

I would start with a lightweight reference prototype, not a production integration: read a bounded input stream, normalize it to the canonical schema, apply the checks, and emit an evidence bundle (input summary, rules run, pass/fail, timing, anomalies, trace ids, reviewer notes). Proprietary interfaces are swapped for stand-ins a scenario

simulator for the train-control interface, a synthetic hazard log for the requirements source, a plain SQL store for the matrix so a small, GitHub-safe demonstrator can reproduce the results without any employer-owned data.

Table 5. Planned ablations and the effect each is expected to expose.

Ablation	What is removed	Expected effect
Remove live matrix (use documents)	Traceability assembled at audit time	Empty cells found late or never
Remove fail-safe scenario class	Test nominal behaviour only	'Fails toward stopped' goes unverified
Remove empty-cell-as-defect rule	Gaps are tolerated	Coverage debt accumulates silently
Remove execution evidence links	Matrix asserts, not proves	An assessor cannot verify a row
Remove baseline comparison	Report proposed only	Weakens the empirical argument

Table 6. From this draft to a submission: the work that remains.

Step	Minimum action	Artifact
Dataset	Generate or sanitize 500–5,000 hazard-scenario executions with trace links	CSV/JSON dataset with schema notes
Baseline	Document the existing manual or rule-based process	Baseline runtime and error report
Prototype	Build the proposed pipeline without proprietary code	GitHub-safe demonstrator
Evaluation	Run ≥ 3 repeated trials per workload	Metrics with mean, variance, CI
Review	Have one domain expert review assumptions and threats	Reviewer comments and revision log

IX. Deployment and Integration

In a real programme this lives alongside the requirements and hazard baseline rather than in a separate test repository, because the matrix is only as trustworthy as its links to those documents. Each scenario references the requirement and hazard it covers, and the executed test writes its result and evidence back into the same matrix, so the artifact stays live rather than becoming a stale spreadsheet.

The completeness check every hazard row reaching an executed test runs as an automated gate, so a coverage gap blocks a claim of readiness the same way a failing test would. None of this replaces an independent safety assessor; it is meant to give that assessor a legible argument to evaluate, which is the most useful thing engineering can hand them.

X. Why This Sits in My Line of Work

The question comes straight from my working history, and the point of the paper is to make that history checkable. I have spent years at the seam of software and operational infrastructure automation, diagnostics, integration,

debugging across coupled systems but a safety paper must stand on its bounded experiment, not on the author. My record only argues that the question matters.

XI. Threats to Validity

This is deliberately a method paper: it shows how to structure and audit a safety argument, but it makes no claim about the completeness of any real PTC deployment, which would require the actual requirements baseline and assessor review. More broadly, the results here are modeled with illustrative data, so the work should not be presented as empirical until a real or faithfully simulated dataset is documented; generalization across environments with different failure costs needs at least two workload profiles; and confidentiality means I publish methods, metrics, and anonymized patterns while keeping proprietary detail out. It is worth stating plainly what would falsify the central claim: if, on logged data with honest baselines, the method failed to improve any of coverage, fail-safe correctness, or auditability without a compensating loss elsewhere, the contribution would not hold, and I would rather discover that in an ablation than paper over it. An unfalsifiable safety argument is a liability, not a contribution this one is specified to be checkable and therefore refutable.

XII. Broader Applicability and Future Work

The matrix-as-argument approach generalizes to any domain with explicit hazards and a need for assessable evidence medical devices, avionics, industrial safety functions. The discipline of treating an empty hazard-to-test cell as a defect is domain-independent, and it is the part most worth carrying elsewhere, because it turns a comfortable feeling of coverage into something a skeptic can check.

I want to be explicit about scope as future work rather than a present claim: this paper shows how to structure and audit a safety argument, but it makes no assertion about the completeness of any particular deployment. Demonstrating that would require a real requirements-and-hazard baseline and independent assessor review, which is exactly the next step and deliberately outside what a method paper should claim on its own.

XIII. Conclusion and Next Steps

What stands between this draft and a submission is a checklist, deliberately short: sanitized dataset, baseline and prototype, repeated trials with variance, the ablation, independent review. The matrix-first method above was designed so that checklist is executable.

References

- [1] U.S. Federal Railroad Administration, "Positive Train Control (PTC)," 2018. [Online]. Available: <https://railroads.dot.gov/>
- [2] CENELEC EN 50128, *Railway applications Software for railway control and protection systems*, 2011.
- [3] IEC 61508, *Functional safety of E/E/PE safety-related systems*, 2010.