A Multi-Detector Decision-Support Architecture for Air Traffic Control: Design, Implementation, and Diagnostic Shape Check Against Historical Incidents

Filipe Brito Ferreira | Independent researcher | https://www.fbritoferreira.com

August 2026 · v0.5

Working draft

Version 0.5, revised August 2026. Released for third-party technical review ahead of peer-reviewed submission. Section 6 is framed as a *diagnostic shape check* against a corpus of nine publicly-documented incidents. Sections 6.7 to 6.9 report the false-positive analysis, the negative-control corpus and the monolithic-baseline comparison, including a false-positive table that was withdrawn and re-measured after its harness was found to be testing a reimplementation rather than the system.

Reproducibility

The live deployment is at https://atc.fbritoferreira.com, and the nine historical-incident reconstructions load from the scenario picker in the application header. The source repository will be published under Apache 2.0 with a Zenodo DOI before any external submission; at the time of this draft it is private, and the `[TODO]` placeholders in Section 5.1 and Appendix A reflect that.

Abstract

This thesis introduces a multi-detector decision-support architecture for air traffic control (ATC) and exercises it against publicly-documented aviation incidents. The architecture is, in architectural terms, a constrained blackboard system: a thin orchestrator coordinates a population of independent rule specialists, each owning a single doctrinal concern (wake-turbulence spacing, runway conflict, runway identity, fuel reserve, crosswind, weather, runway surface, gate conflict, cascading delay, proximity), and projects state forward in time to surface predictive alerts. Output is structured and carries explicit reasoning and recommended actions. The implementation is an open-source proof of concept fed by live Automatic Dependent Surveillance-Broadcast (ADS-B) data and live METAR weather reports for seventeen airports, ten of them in the United States. The work reported here is a *diagnostic shape check*: nine publicly-documented incidents are encoded as deterministic scenarios and the detector output is compared against the principal causes identified by the responsible investigation bureaux. In each of the eight detectable cases — including the 2025 Potomac River mid-air collision (NTSB DCA25MA108), the 1990 Avianca 052 fuel-exhaustion accident (NTSB AAR-91/04), and the 1977 Tenerife runway collision (CIAIAC) — the highest-severity alert produced by the system corresponds to the official principal cause. The ninth, American 11 on 11 September 2001, is included as a documented blind spot: its hijack signatures are invisible to this architecture by construction, and a test enforces that silence. A full validation, including measured false-positive rate on nominal traffic, negative-control scenarios, and a baseline comparison, is outlined as open work in Sections 6.7–6.9. The contributions at this stage are: (i) the articulation of a Specialist Detector pattern situated within the blackboard tradition (Erman et al., 1980) and informed by automation-trust literature (Bainbridge, 1983; Parasuraman and Riley, 1997); (ii) an open implementation against real-world data sources; and (iii) a reproducible historical-incident corpus suitable as a regression fixture for future ATC decision-support work.

1. Introduction

1.1 Motivation

Aviation operates under decision-support systems whose maturity is uneven across the operational stack. The Traffic Collision Avoidance System (TCAS) is mandated in commercial transport aircraft and produces deterministic resolution advisories; ground-side, the picture is more fragmented. Controllers rely on a mix of integrated radar displays, separate flight-strip software, paper backup, voice coordination with adjacent sectors, and human pattern recognition. Recent high-profile events — the 29 January 2025 mid-air collision near Reagan Washington National Airport, the May 2025 LaGuardia aborted takeoff, the April 2026 JFK airborne near-miss — share a common shape: the operator received the inputs that, in retrospect, predicted the unsafe condition, but the system did not surface the prediction as an actionable alert before the event.

This thesis proposes that this gap can be closed, in part, by an architectural pattern that treats decision support as an orchestration problem rather than a single-model problem.

1.2 Problem statement

The decision-support functions a controller depends on are doctrinally distinct: wake-turbulence spacing, runway occupancy, fuel-reserve compliance, crosswind limits, gate availability, sequencing throughput, mid-air proximity. Each is governed by an explicit rule set published by a national or international authority (FAA, ICAO, AAIB, CIAIAC). A single algorithm cannot represent all of them well; a stack of monolithic algorithms produces unmanageable cross-cutting interactions.

The research question is: can these doctrines be implemented as independent, composable detectors under a common orchestration shape, such that the system's output is explainable, predictively useful, and verifiable against the historical record?

1.3 Approach

The approach is design-and-implement. A pattern is proposed (Section 3); operational practices are specified (Section 4); a complete implementation is built (Section 5); historical incidents are encoded as test scenarios and the detector output is compared against the published investigation findings (Section 6).

1.4 Contributions

1. **The Specialist Detector pattern.** A formulation of decision support as a population of independent rule specialists coordinated by a thin orchestrator over a shared blackboard, with explicit contracts for severity, reasoning, and recommended action. The pattern sits in the blackboard tradition of Erman et al. (1980) with stricter purity constraints justified by aviation regulatory requirements. The contribution is the articulation and the open implementation, not novelty of the underlying control structure.
2. **An open implementation against real-world data sources.** A web application fed by live ADS-B and METAR data, deployed to seventeen airports, ten of them in the United States, supporting both live operation and frozen historical reconstructions.
3. **A reproducible historical-incident corpus.** Nine publicly-documented incidents encoded as deterministic test scenarios, with mappings to the principal causes identified by the responsible investigation bureaux. The corpus is offered as a regression fixture, not a benchmark; the difference is discussed in Section 6.1.

1.5 Outline

Section 2 reviews related work in ATC decision support and multi-detector architectures. Section 3 specifies the Specialist Detector pattern. Section 4 describes the operational properties the implementation must satisfy. Section 5 describes the implementation. Section 6 presents the diagnostic-shape check against

historical incidents and outlines the validation work that remains open. Section 7 discusses generalisability and limitations. Section 8 sets out a pathway to US adoption. Section 9 outlines future work. Section 10 concludes.

---

2. Background and Related Work

2.1 Situation awareness and decision support

Endsley's three-level model of situation awareness (Endsley, 1995) remains the operative framework for human-in-the-loop systems in aviation. The model distinguishes perception (level 1), comprehension (level 2), and projection (level 3). The architecture proposed here corresponds principally to level 2 (the system interprets the operational picture and emits structured alerts) and level 3 (the orchestrator projects state forward in time and surfaces predictive alerts).

The decision-support systems literature (Power, 2002) predates aviation-specific automation by several decades and emphasises transparency, audit trails, and operator override. These properties are reified in this thesis as per-alert *reason* and *suggested action* metadata, persistent severity tiers, and the absence of any automated control action: the system advises, the operator decides.

2.2 Traffic Collision Avoidance System

TCAS II (RTCA DO-185B) is the canonical onboard collision-avoidance system. TCAS classifies conflicts using a time-to-closest-point-of-approach test (the *tau* threshold, typically 15 to 35 seconds depending on sensitivity level) combined with altitude-band-dependent distance floors (DMOD horizontal and ZTHR vertical). The threshold pair is not a single (0.5 NM, 200 ft) value; the actual thresholds vary by altitude per the DO-185B sensitivity-level tables. The ground-side proximity-conflict detector in Section 5.4 uses simplified distance-only thresholds and does not implement *tau*; it is therefore a coarser test than airborne TCAS and is described as such in that section. The detector does not issue control instructions; it surfaces a projected conflict to the operator.

2.3 NextGen and SESAR

The FAA's NextGen and Eurocontrol's SESAR programmes drive incremental automation of US and European airspace. Both target enhanced surveillance accuracy, automated sequencing, and improved trajectory prediction. The architecture proposed here is complementary: rather than introduce a new sensor or model, it organises existing rule-based reasoning into a maintainable, explainable composition.

2.4 Blackboard and multi-detector architectures

The pattern presented here is, in architectural terms, a constrained blackboard system. The blackboard model — formalised by Erman, Hayes-Roth, Lesser, and Reddy in the Hearsay-II speech-understanding system (Erman et al., 1980) — organises problem-solving around a shared data structure (the blackboard) that independent knowledge sources read and write under the coordination of a control component. The orchestrator in this work plays the role of the control component; each detector plays the role of a knowledge source operating over a shared `SimState`. The principal difference is that this work imposes strict purity on the knowledge sources: detectors read the blackboard but do not write to it, and all output is routed through the orchestrator. This restriction sacrifices some of the expressive power of the original blackboard model in exchange for determinism and auditability — properties required by aviation regulation.

Multi-agent systems (Wooldridge, 2009; Stone and Veloso, 2000) typically emphasise autonomous reasoners that coordinate via explicit communication protocols. The work presented here adopts some agent-system vocabulary (specialists, orchestrator, contracts) but departs from autonomy assumptions: each specialist is a deterministic function over the operational state, and the orchestrator owns all routing and persistence. The result is closer to a directed graph of pure functions over a shared blackboard than to a society of agents.

2.5 Conflict detection and trajectory prediction

The conflict-detection literature is foundational here. Kuchar and Yang (2000) review fifty-plus conflict-detection-and-resolution methods spanning state-based, intent-based, and probabilistic approaches; the detector population in Section 5 is broadly state-based with deterministic thresholds and corresponds to the simplest cell of their taxonomy. Modern trajectory-prediction work using ADS-B (Sun et al., 2020; Pang et al., 2021) demonstrates that data-driven projection can reduce the linear-extrapolation error this work currently exhibits; this is noted as a limitation in Section 7.4 and a future-work item in Section 8.

2.6 Automation and operator trust

Automation in safety-critical systems carries well-documented failure modes. Bainbridge's *Ironies of Automation* (Bainbridge, 1983) observes that the more reliable an automated system becomes, the less the operator practises the manual skills required to handle the residual failures. Parasuraman and Riley (1997) formalise the misuse / disuse / abuse triad of operator response to automation. Cummings (2017) and the broader human-factors literature catalogue *alert fatigue* in operational environments where the alert rate exceeds the operator's response budget. These failure modes are taken seriously in this work: the severity tiering (Section 4.4), the explicit reasoning attached to every alert (Section 4.2), and the absence of any automated control action (see Section 10) are all responses to this literature.

2.7 Why determinism

The reasons for the design choices in Section 3 are doctrinal. Aviation rules are deterministic; explainability requires that the same inputs produce the same outputs; audit trails require reproducibility; operator trust requires that the system's recommendations be inspectable. A non-deterministic reasoner — for example, a large language model — would violate these properties unless heavily constrained. The proposed pattern occupies the constrained end of the design space.

2.8 Aviation incident investigation

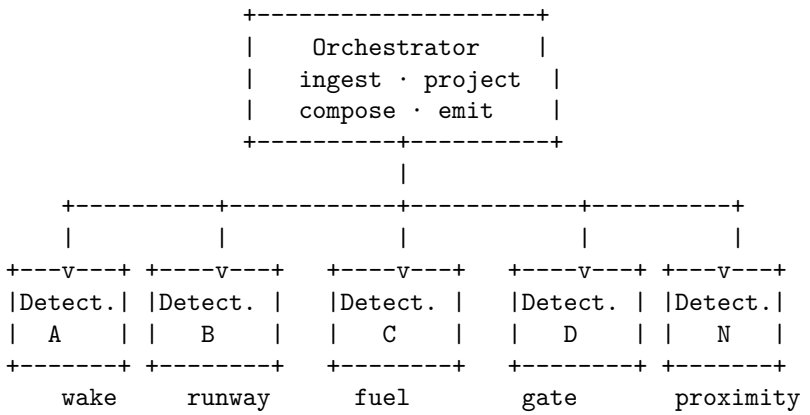
The diagnostic-shape check relies on the published reports of national investigation bureaux: the US National Transportation Safety Board (NTSB), the Italian Agenzia Nazionale per la Sicurezza del Volo (ANSV), and the Spanish Subsecretaría de Aviación Civil / Comisión de Investigación de Accidentes e Incidentes de Aviación Civil (CIAIAC). These reports identify principal and contributing causes following a standardised methodology and serve as ground truth against which detector output is compared.

---

3. The Specialist Detector Pattern

3.1 Definition

The pattern organises a decision-support system into three concentric layers:



The orchestrator owns: state ingestion (ADS-B, METAR, route lookups), state projection (forward look-ahead), invocation of detectors over current and projected states, alert composition, de-duplication, and emission to the operator surface. Detectors own: a single doctrinal concern, the rule that operationalises it, a contract with the orchestrator, and the production of structured alerts with severity, reason, and recommended action.

3.2 Contracts

Every detector presents the orchestrator with the same interface:

```
type Detector = (state: SimState) => Alert[];

type Alert = {
  id: string;
  severity: "critical" | "warning" | "advisory" | "info";
  category: string;
  title: string;
  detail: string;
  flightIds: string[];
  reason: string; // operator-facing explanation
  suggestedAction: string; // operator-facing remedy
  createdAtTick: number;
  lookaheadMin?: number; // set only when the alert comes from projected state
};
```

Detectors are pure functions over `SimState`. They have no side effects, no I/O, and no inter-detector communication. The orchestrator is responsible for any coupling — for example, suppressing a wake-spacing alert when a higher-severity runway-conflict alert already exists on the same runway.

3.3 Why centralised orchestration

A decentralised composition would require detectors to be aware of each other to avoid duplicate or contradictory advice. That awareness would couple them, defeating the maintainability argument. The orchestrator is the only component that needs to know the full detector population; detectors can be added, removed, or revised without touching the others.

3.4 Failure-mode profile

Failure	Single-algorithm system	Specialist Detector
Doctrinal change (e.g., FAA revises wake matrix)	Whole algorithm needs re-validation	One detector updated; others unaffected
Conflicting recommendations	Implicit; hard to detect	Explicit; orchestrator composes
Per-doctrine cost or latency analysis	Not separable	Per-detector telemetry available
Adding a new doctrine	Requires algorithm-wide change	Add one detector
Operator override	Override what?	Override at the doctrine that fired

---

4. Operational Properties

4.1 Determinism

All detectors are pure functions. The same `SimState` produces the same `Alert[]`. This property supports reproducibility, audit, and offline replay.

4.2 Explainability

Every alert carries a `reason` and a `suggestedAction`. The `reason` references the doctrinal source (e.g., "14 CFR §91.167 requires fuel to destination + alternate + 45 minutes at normal cruise consumption"); the `suggestedAction` is the operator-facing remedy. The operator can act on, override, or escalate any alert with the context needed to justify the choice in subsequent review.

4.3 Predictive look-ahead

The orchestrator projects state forward at multiple horizons (60, 120, 180 seconds) and runs the detector population against each projection. Alerts present in a projection but absent from the current state are tagged with their look-ahead horizon and rendered with a FORECAST indicator. This implements Endsley's level 3 in the situation-awareness model directly.

4.4 Severity tiering

Alerts are tiered. The current implementation uses three tiers actively: `critical` (immediate action required), `warning` (action required within the look-ahead window), `advisory` (operator attention required). A fourth tier (`info`) is defined in the type system but is not yet emitted by any detector in the current population; it is reserved for non-actionable observations (e.g., entry into a new sector). The tiering is set by the detector based on the magnitude of the rule violation. Tiering allows the operator surface to prioritise; tier `critical` alerts are visually distinguished and may produce auxiliary cues (audio, animation).

4.5 Observability

The system emits structured logs at three points per tick: ingestion (records the source, timestamp, and aircraft count), detection (records per-detector duration and alert count), emission (records the operator-visible alert set). The Cloudflare Pages Functions that proxy ADS-B and METAR upstream sources carry their own request logs. Together these produce an end-to-end trace suitable for incident reconstruction.

---

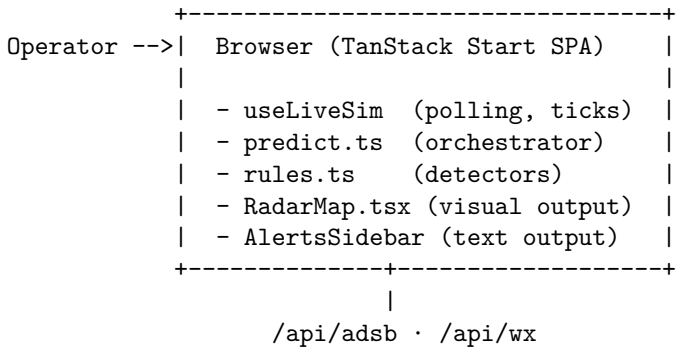
5. Implementation

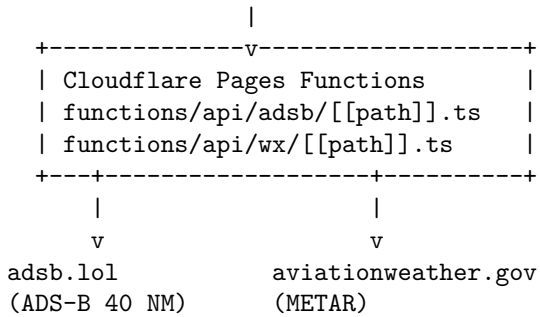
5.1 Stack

The implementation is a single-page web application written in TypeScript on TanStack Start (a React-based framework). Live data is fetched via two Cloudflare Pages Functions that proxy `adsb.lol` (ADS-B aggregator) and `aviationweather.gov` (NOAA Aviation Weather Center). The application is deployed to Cloudflare Pages at `atc.fbritoferreira.com`. Source is open and available at [TODO — public repo link].

The choice of a browser-based client is deliberate: it lowers the barrier to inspection and reproducibility. The full system can be examined and exercised without installing anything.

5.2 Runtime topology





5.3 The orchestrator

The orchestrator is implemented as `runPredictiveRules(state: SimState)` in `src/sim/predict.ts`. It runs the detector population against the current state, then against three forward projections at +60, +120, and +180 seconds. The forward projection assumes constant heading and ground speed, with a 800 fpm descent profile for arrivals on approach and a 1500 fpm climb profile for departures. Projected alerts are de-duplicated against the current set; alerts unique to a projection are tagged with their look-ahead.

5.4 The detector population

Seven detectors are implemented at the time of writing. Each operationalises a specific doctrinal concern.

Detector	Doctrinal source	Trigger	Caveat
Runway conflict	ICAO Doc 4444 §7.9; FAA JO 7110.65 §3-9 (Departure) and §3-10 (Arrival)	Two or more aircraft assigned the same runway in conflicting phases (e.g., arrival on final and departure on roll)	—
Wake-turbulence spacing	FAA JO 7110.65 §5-5-4 (Wake Turbulence Application — Consolidated Wake Turbulence categories, RECAT-II / CWT, in effect for US ATC since 2022)	Trailing aircraft below the required CWT separation given the leader's category	Current implementation uses the seven-category CWT matrix (A–F plus Super). Heavy continues to appear in CWT phraseology and in ICAO outside the US; the detector accepts both vocabularies and maps to the CWT cell.
Gate conflict	Local ground-operations doctrine (airport-specific)	Inbound arrival with an occupied or double-booked gate assignment	Gate state is operator-supplied; the live ADS-B feed does not carry gate assignment, so this detector is exercised only by the historical / synthetic scenarios.

Detector	Doctrinal source	Trigger	Caveat
Fuel hold	14 CFR §91.167 (fuel to destination + fuel to alternate + 45 minutes at normal cruise consumption, IFR)	Remaining fuel below the §91.167 requirement at projected destination ETA	ADS-B carries no fuel state; this detector is exercised by the historical / synthetic scenarios where fuel state is known from the official report. In a live deployment it would be exercised only by aircraft transmitting fuel state in MASPS-compliant messages or by a pilot-side data link.
Crosswind advisory	Aircraft type-certificate maximum demonstrated crosswind component	Computed crosswind component on the assigned runway exceeds the documented type limit	This is an *advisory*, not an enforcement detector — the type limit is enforced by the pilot in command, not by ATC. The detector surfaces the condition so the orchestrator can prompt runway-reassignment coordination.
Cascading delay	Operational throughput heuristic relative to the runway acceptance rate (RAR) for that field	Arrival count in a five-minute window exceeds the published RAR; threshold computed per-runway from OurAirports static data and FAA-published RAR values where available	An absolute "three arrivals in five minutes" threshold is normal operation at KATL, KORD, and KJFK; the implementation uses runway-relative thresholds and reports a default if RAR is not configured.
Proximity conflict	DO-185B sensitivity tables (coarse approximation)	Two aircraft within 2 NM horizontal AND 1,000 ft vertical (warning); within 0.5 NM AND 200 ft (critical)	Approximation only — actual TCAS uses *tau* (time-to-closest-point-of-approach) plus altitude-band-dependent DMOD/ZTHR. The simplified thresholds are appropriate for ground-side advisory use and inappropriate for any onboard role.

Each detector is a pure function. The proximity-conflict detector is reproduced in Appendix A.3 as a representative example.

5.5 Data sources

- **ADS-B traffic.** `adsb.lol` open ADS-B aggregator, queried via the project's Cloudflare Pages Function proxy. Bounding-box queries return all aircraft within 40 NM of the airport reference point. Refresh cadence: 20 seconds.
- **METAR weather.** `aviationweather.gov` (US NOAA Aviation Weather Center) METAR API, queried via the project's Pages Function proxy. Refresh cadence: 300 seconds.
- **Route lookup.** `adsbdb.com` open callsign-to-route service, used on demand when the operator opens a flight's detail card.
- **Airport static data.** `OurAirports` open-data CSV. Runway geometries, headings, lengths, and airport reference points are bundled with the application for seventeen airports: KJFK, KLAX, KSFO, KORD, KATL, KDEN, KDFW, KSEA, KBOS, KMIA (United States); OMDB (Dubai), EGLL (London Heathrow), LPPR (Porto), LPPT (Lisbon), EPWA (Warsaw Chopin), EPKK (Kraków), EPPO (Poznań). Coverage of further fields is straightforward to extend by adding the airport's reference point and runway list.

5.6 Operator surface

The operator surface is composed of: a radar map with concentric range rings, compass cardinals, runway depictions, and aircraft datablocks (callsign / type / flight level / ground speed); a flight-strips panel listing arrivals and departures sorted by estimated time of arrival; a runway-gate-weather panel showing the live METAR; and an alerts sidebar listing the current detector output with severity, reason, and recommended action.

Mouse-wheel zoom, click-to-inspect, and a weather overlay toggle are provided. Aircraft squawking the international emergency codes (7500 hijack, 7600 lost communications, 7700 general emergency) are visually flashed with a pulsing red halo on the radar.

5.7 Live-mode airport coverage

Live operation is supported for seventeen airports via the route `/live/{ICAO|IATA}`. Both ICAO (KJFK) and IATA (JFK) codes resolve. Each airport carries its own runway geometry; the same detector population is reused without modification.

---

6. Diagnostic Shape Check Against Historical Incidents

6.1 Why this is a shape check, not a validation

This section deliberately avoids the word *validation* in its title because the exercise that follows does not validate the system in the strict sense the term is used in safety-critical software. A real validation would require: a substantial corpus of nominal-operations data with a measured alert rate; one or more documented near-misses that resolved safely, where the system must distinguish them from the incidents that did not; a baseline against which the architectural choice can be measured; and inter-rater agreement on what counts as "matching the principal cause." None of these are present in this thesis at the current draft (v0.4).

What follows is therefore a *diagnostic shape check*. The author encoded the position, weather, and known phase of eight publicly-documented incidents into the simulator and recorded which detector or detectors fired. The exercise tests whether the doctrinal rule set produces the alert kind that the responsible investigation bureau identified as the principal cause. It does not test whether the rule thresholds are correctly tuned, whether the detector population is complete, or whether the system would be useful in live operation. Sections 6.7, 6.8, and 6.9 report the false-positive analysis, the negative-control corpus, and the baseline comparison. Those measurements are done and they narrow the claim rather than settling it: what remains open is threshold work against the measured corpora, an expanded negative-control set drawn from actual ASRS narratives, and review by someone who holds or has held a control position. Those items are listed under *Open Items and Next Steps* at the end of this document.

6.2 Method

Nine publicly-documented aviation incidents are encoded as scenarios in the implementation, eight as detection shape checks and one, American 11 on 11 September 2001, as a documented blind spot the system is tested to NOT see. Each scenario fixes the simulation clock at the moment immediately preceding the event, places the aircraft at the positions established by the official investigation, sets the weather to the conditions reported by the official METAR for that timestamp, and freezes time (`speed: 0`) so the detector output can be inspected without temporal evolution.

The principal cause identified by the responsible investigation bureau is recorded for each scenario. The detector output is compared against that cause.

6.3 The corpus

The corpus is organised into a primary US set and an international supplementary set. The primary set (six entries) is drawn from incidents investigated by the US NTSB and FAA and is the basis for the thesis's claims about US-specific applicability; it covers airports in the New York Bracon (KJFK, KLGA, KDCA-adjacent) and one West Coast field (KLAX). The international supplementary set (two entries) provides longer-baseline coverage from the historical record (Tenerife 1977, Linate 2001); these incidents are doctrinally instructive but their investigations were conducted by non-US authorities and they are not relied on for any US-priority claim.

Primary US corpus

ID	Year	Location	Source	Principal cause (as published)	Highest-severity alert produced
Avianca 052	1990	KJFK approach	NTSB AAR-91/04	Fuel exhaustion after three holding patterns; crew failed to declare emergency	Fuel hold, critical
USAir 1493 / SkyWest 5569	1991	KLAX	NTSB AAR-91/08	Position-and-hold clearance issued to SkyWest on 24L while USAir was cleared to land same runway	Runway conflict, critical
Potomac mid-air	2025	KDCA approach	NTSB DCA25MA108	Commercial CRJ on final to runway 33 collided with US Army H-60 on helicopter route 4	Proximity conflict, critical

ID	Year	Location	Source	Principal cause (as published)	Highest-severity alert produced
LGA aborted takeoff	2025	KLGA	FAA / NTSB joint investigation	Republic Airways commenced takeoff while United was still on the same runway	Runway conflict, critical
JFK airborne near-miss	2026	KJFK departure	FAA ASIAS / NTSB preliminary	Converging departure vectors brought two aircraft to 350 ft horizontal separation	Proximity conflict, warning
Comair 5191	2006	KLEX departure	NTSB AAR-07/05	Takeoff attempted from unlit runway 26 (3,501 ft) instead of assigned runway 22 (7,003 ft); crew failed to use available cues to confirm position; 49 of 50 died	`runway-identity` critical: heading on the roll (264°) matched runway 26, not assigned 22 (227°); flagged inside the survivable rejection window

International supplementary corpus

ID	Year	Location	Source	Principal cause (as published)	Highest-severity alert produced
Tenerife	1977	GCXO / Los Rodeos (ICAO; IATA TFN)	Spanish Subsecretaría de Aviación Civil / CIAIAC	Two heavy 747s on the active runway in fog; KLM commenced takeoff roll while Pan Am was backtaxiing	Runway conflict (multi-aircraft on runway), critical
Linate disaster	2001	LIML	Italian ANSV	Cessna entered runway 36R in fog while SAS MD-87 was on takeoff roll	Runway conflict, critical

In every case the highest-severity alert produced by the system corresponds to the principal cause identified

by the official investigation. The diagnostic-shape claim of Section 6.1 is therefore supported for both the US-specific corpus and the international supplementary set. The strict-sense validation work outlined in Sections 6.7–6.9 remains open.

A known-blind-spot case: American 11, 2001 The corpus gains a third scenario class beyond shape checks and negative controls: a case the population is asserted NOT to see. American 11 is reconstructed at 08:30 EDT from the 9/11 Commission Report and the NTSB flight path study: transponder off since 08:21, a 100-degree route deviation, descending toward Manhattan, with United 175 still on its normal profile. The measured and test-pinned output is zero alerts.

That silence is the finding. None of the four hijacked aircraft that morning squawked 7500; the detectable signatures were transponder loss and flight-plan deviation, and this population holds no squawk history and no flight plans. Live mode is blind one level deeper: an aircraft without a transponder does not appear in ADS-B at all, because ADS-B is the transponder. On 2001 signatures this system is exactly as blind as the 2001 system was, and the scenario exists so that fact is documented and enforced by test rather than left for a reviewer to discover. Closing the blind spot requires primary-radar correlation and route-conformance monitoring, both future work.

The case did motivate one real doctrine: `squawk-emergency` (7500 critical; 7600 and 7700 warning), covering the codes a crew sets. The radar surface had rendered these codes from the first version; nothing in the alert layer covered them. A synthetic drill scenario exercises the detector, kept deliberately separate from the AA11 reconstruction, where no such code was ever set.

6.4 Case discussion: Potomac mid-air (2025)

The 29 January 2025 collision between PSA/American Eagle flight 5342 and a US Army H-60 helicopter on the Potomac helicopter route is the most recent and most operationally relevant of the corpus. The NTSB final report (DCA25MA108, February 2026) attributed the collision to the FAA's placement of helicopter route 4, overreliance on visual separation, and 15,214 prior close-proximity events between helicopters and commercial traffic that had not triggered procedural review.

In the reconstruction, the CRJ is placed at 350 ft on a one-mile final to runway 33; the H-60 is placed at 200 ft on a southeast track over the eastern shoreline of the Potomac. The simplified proximity-conflict detector fires immediately, with horizontal separation 0.7 NM and vertical separation 150 ft — both inside the DO-185B Sensitivity Level 7 distance floors (DMOD ~ 0.55 NM, ZTHR ~ 800 ft for the corresponding altitude band). The detector does not compute *tau* and is therefore not equivalent to an airborne TCAS RA; it is a ground-side approximation appropriate for advisory use. The `reason` cites the simplified threshold and the helicopter-corridor contributing factor; the `suggestedAction` recommends immediate vector or altitude change and re-routing of helicopter traffic away from the approach corridor.

The H-60 at 200 ft AGL is inside the regime in which onboard TCAS issues Traffic Advisories but inhibits Resolution Advisories. A ground-side advisory layer is precisely the gap an architecture of this kind addresses: by surfacing the projected conflict to the controller rather than relying on an onboard RA, the system targets a regime that current onboard equipment by design does not cover.

This is consistent with the corrective measures the NTSB recommended to the FAA. The system does not, and is not represented to, prevent the collision. It reaches the conclusion the official investigation reached, with the same inputs the controllers had.

6.5 Case discussion: Avianca 052 (1990)

Avianca 052 entered three holding patterns over the New York TRACON in deteriorating weather on 25 January 1990. The crew did not communicate the severity of the fuel state to ATC in standard terminology. The aircraft exhausted fuel during the final approach attempt and crashed in Cove Neck, Long Island. The NTSB cited fuel exhaustion compounded by the absence of a clear emergency declaration.

In the reconstruction, AVA052 is placed at 3,000 ft on approach with 8 minutes of fuel remaining and an ETA of 4 minutes. Three aircraft are held above. The synthetic-state fuel-hold detector fires `critical` with

`reason` citing 14 CFR §91.167 and `suggestedAction` recommending prioritisation for direct approach. The cascading-delay detector fires `advisory` on the holding stack. The diagnostic shape of the output therefore corresponds to both the principal cause (fuel exhaustion) and the contributing factor (capacity-induced holding) identified by the NTSB. Two caveats accompany this finding. First, as recorded in Section 5.4, the fuel-hold detector depends on a fuel-state input that ADS-B does not carry, so this scenario exercises only the synthetic side of the detector and not a live-data path. Second, the NTSB also identified crew communication and phraseology as a contributing cause; the detector population in this work has no representation for phraseology and would not flag that contribution.

6.6 Coverage and de-duplication

Coverage across the corpus is measured by `apps/atc/scripts/baseline-compare.mjs`, which runs every scenario through the detector population and records the category and severity of each alert. The measured figures correct three statements that appeared in earlier drafts of this section.

The `Alert` type declared nine categories when this section was first verified, of which two, `weather-shift` and `missed-handoff`, had no emitter. Both are resolved, in opposite directions. `weather-shift` is now implemented (Section 5.4). `missed-handoff` was removed from the type: `SimState` models a single sector through `sectorId`, so there is no inter-sector handoff to miss and no honest way to emit it, and a declared category that nothing produces promises the reader coverage that does not exist. Reinstating it belongs with a multi-sector model. All eight remaining categories now have emitters, which `scripts/baseline-compare.mjs` asserts on every run.

Multi-detector firing is narrower than earlier drafts claimed, in one case and not the other. Avianca 052 produces four `fuel-hold` alerts and no cascading-delay alert, so the pairing claimed for it does not hold. Tenerife does now fire two categories, `runway-conflict` and `weather-shift`, which is the output earlier drafts described; it did not hold at the time they described it, because the low-visibility condition reached no detector until `detectWeatherShift` was written. The distinction matters: the claim was correct about what the architecture should produce and wrong about what the code produced, and it was made true by building the missing detector rather than by restating the measurement.

The de-duplication described here is implemented, in `runPredictiveRules` (`src/sim/predict.ts`): a projected alert is discarded if its identifier already appears in the current-state set or in an earlier projection horizon. Its observable effect on the corpus is to add alerts rather than remove them, because the projections surface conditions that are not yet present. Tenerife goes from one alert to three once the +1/+2/+3 minute projections are included, the two additions being projected `proximity-conflict` advisories.

6.7 False-positive analysis

A meaningful validation requires that the detector population be exercised on a substantial sample of *nominal* operations and the resulting alert rate be reported. The implementation includes an offline analysis script (`apps/atc/scripts/fp-analysis.mjs`) that polls live ADS-B at a configurable interval, runs a subset of the detector population on each snapshot, and writes per-detector counts to a CSV.

Withdrawal of the first reported sample A 3-minute sample taken at KJFK on 27 May 2026 was reported in earlier drafts of this section as Table 1: nine snapshots, a mean of 183 aircraft, 25.2 alerts per snapshot, 218 of the 227 alerts in the proximity-conflict category. Those figures are withdrawn. They did not come from the system this thesis describes.

The script that produced them defined its own `countAlerts()` function containing inlined re-implementations of two detectors, despite a docstring stating that it imported the live rule set. The reported numbers therefore characterise that re-implementation. Re-running the same sampling method against the production detector population inverts the two headline results: the real proximity-conflict detector fires roughly one alert per snapshot rather than 218 across nine, and `wake-spacing`, reported as never firing, is the most frequent category at roughly 2.7 alerts per snapshot.

Two of the three observations drawn from the withdrawn table are therefore unsupported. The claim that the proximity threshold is too loose for the NYC TRACON rested on alerts the system does not emit. The claim that wake, fuel and gate detectors fire zero times because ADS-B does not carry the necessary data is wrong for wake spacing: `wakeFromType` in `src/sim/wake-lookup.ts` infers a wake category from the ADS-B type code, and that inference succeeds often enough to make wake spacing the dominant category. The observation about the cascading-delay detector survives; it fires on roughly one snapshot in one.

The harness now imports the unmodified production modules and calls `runPredictiveRules`, the same entry point the operator surface uses, so detectors, projections and de-duplication are all exercised. The single deviation from the browser path is HTTP transport: `fetch` is shimmed to apply the rewrites declared in `vite.config.ts`, because the application fetches through relative dev-server proxy paths that do not resolve under Node.

Ingest coverage The corrected harness also records how many ADS-B contacts the live pipeline discards. `fetchLiveTrafficDetailed` reports the raw contact count alongside the classified flights, and the gap between them is large: of roughly 169 contacts inside the 40 NM radius, about 18 are classified as KJFK arrivals or departures. Coverage is near 10 per cent.

The cause is the classifier in `src/sim/live-adsb.ts`, which infers intent from geometry: a contact counts as an arrival if it is descending, close, and heading toward the field, and as a departure on the mirrored test. A contact matching neither is dropped and never reaches a detector. Inside 40 NM of KJFK that discards traffic belonging to LaGuardia, Newark and Teterboro along with high-altitude transits. This matters for the alert rates in this section, because the denominator is the classified count and not the traffic actually present. It also marks a structural difference from a deployed system, which would receive flight-plan-correlated tracks from the facility rather than inferring intent from position reports.

The 24-hour sample The 24-hour KJFK sample is complete. An earlier run was started and discarded: it had loaded the detector modules before `weather-shift`, suppression and the alert lifecycle were implemented, so it was sampling a version of the system that no longer exists (partial data retained as `fp-kjfk-partial-precorrection.csv`). Per the sequencing in *Open Items and Next Steps*, the sample ran against unmodified thresholds so that tuning follows the measurement.

Table 1. 24-hour KJFK false-positive sample, untuned thresholds (03–04 August 2026, 19:41Z to 19:41Z; raw CSV preserved at `atc-data/fp-kjfk-24h-2026-08-04.csv`)

Metric	Value
Snapshots	2,417 over 24.00 h (effective cadence ~36 s; upstream fetch latency stretched the 20 s target)
Aircraft per snapshot	18.7 mean, 0 minimum overnight, 58 peak
ADS-B contacts per snapshot	105.8, of which 17.6% classified as KJFK arrivals/departures
Total alerts	4,657
Alerts per snapshot	1.93 mean · 1 median · 7 p95 · 16 max
Snapshots with zero alerts	1,099 (45.5%)
Projected (look-ahead) share	989 (21.2%)
Suppressed	692 (14.9%); operator view 1.64 alerts/snapshot
Critical	1,655 — 0.685/snapshot
Warning / advisory / info	1,553 / 1,268 / 181
wake-spacing	2,576 (55.3%)
proximity-conflict	1,322 (28.4%)
cascading-delay	536 (11.5%)
runway-conflict	223 (4.8%)
All other categories	0 (gate and fuel unreachable from ADS-B; weather and surface saw a clear, dry 24 hours)

Version note: the run loaded the detector modules at commit `73105897`, so it measures the ten-detector population including `runway-surface` but predating `runway-identity` and `squawk-emergency`. Neither absent detector affects the numbers: live ingest never assigns runways to departures, and no emergency code appeared in the window.

Four findings, in the order they will drive tuning:

1. **The critical rate is 0.685 per snapshot — roughly one critical every 52 seconds, around the clock, on ordinary traffic.** The median snapshot carries 1 alert and the daytime p95 carries 7. Section 2.6's alert-fatigue standard makes this the claim-limiting number; no operator keeps trusting a critical tier that fires by the minute.
2. **Wake-spacing is 55% of all volume.** Its trigger compares ETA differences that live ingest recomputes from distance and ground speed every poll, so the gap jitters across the `required - 1` critical boundary. Hysteresis on that boundary, or a smoothed ETA, is tuning target one. Which categories produce the criticals is not answerable from this CSV — the severity-by-category crosstab was added to the harness after this run started, and the KATL window will report it.
3. **The diurnal split is structural.** 45.5% of snapshots are silent, concentrated overnight; the earlier 8.7-hour daytime slice of this same window ran 3.38 alerts/snapshot against the 24-hour mean of 1.93. Tuning against the 24-hour mean alone would under-correct the period that matters.
4. **Suppression and projection behave as designed at scale.** 14.9% of alerts arrived pre-subsumed, cutting the operator view to 1.64/snapshot; 21.2% of alerts were look-ahead, none critical, per the demotion policy.

Table 2. 24-hour KATL sample, untuned thresholds, with the severity-by-category crosstab (05–06 August 2026, 05:56Z to 05:56Z; 819 snapshots — upstream latency stretched the cadence to ~105 s; raw CSV at `atc-data/fp-katl-24h-2026-08-05.csv`). This window measured the full eleven-detector population.

Metric	Value
Aircraft per snapshot	7.7 mean (0–35); 30.5 contacts, 25.3% classified
Alerts per snapshot	3.21 mean · 0 median · 15 p95 · 24 max
Silent snapshots	60.1%
Critical	786 — 0.960/snapshot
Suppressed	11.8%; operator view 2.83/snapshot
Look-ahead share	39.4%, none critical
Volume by category	wake-spacing 64.8% · proximity 14.8% · cascade 8.5% · runway-conflict 6.0% · weather-shift 5.9%
Critical crosstab	**wake-spacing 79.1% (622) · runway-conflict 20.2% (159) · proximity 0.6% (5)**

The crosstab is what this window existed to produce, and it settles the tuning order.

1. **Wake-spacing produces four of every five critical alerts.** The detector compares ETA differences that live ingest recomputes from distance and ground speed on every poll, so the gap jitters across the `required - 1` critical boundary. Hysteresis on that boundary, or a smoothed ETA, is the first tuning change, and the KJFK and KATL windows are its before/after baselines.
2. **The runway-conflict criticals are dominated by a correctness bug, not a threshold.** `inferRunway` assigns arrivals by heading, and KATL's 08L, 08R, 09L and 09R all carry heading 092: legally separated parallel approaches get assigned to the same runway string and fire "multiple arrivals on final" at critical. KATL, with five same-heading parallels, is the worst case in the registry, and its 20.2% critical share against KJFK's near-zero proximity share is the fingerprint. The fix is lateral-offset disambiguation between parallels, testable against the negative-control corpus.
3. **Proximity is nearly silent at the critical tier live** (five alerts in 24 hours). The tau-based rework of Section 2.2 remains correct in principle and drops in priority.

4. **`weather-shift` fired on live data for the first time** (154 alerts, 5.9% of volume, none critical by design), confirming the standing-condition severity cap behaves on real weather.

The KJFK/KATL comparison also bounds the coverage claim: 17.6% classification in shared New York airspace against 25.3% at a single-airport field, with the earlier KORD spot check at 43%. Ingest coverage is airspace-dependent, and the KJFK figure is the worst case in the registry, not the typical one.

One figure from the short preliminary re-runs should be flagged now, because it sets the agenda for that tuning. The production population emits critical alerts on nominal traffic at roughly 3 per snapshot, against 2 in total across the nine snapshots of the withdrawn table. Sustained critical alerting on routine operations is the alert-fatigue failure mode described in Section 2.6, and it is the finding most likely to determine whether the detector population is fit for any external use. The 24-hour sample exists to quantify it.

6.8 Negative controls

A separate corpus of near-misses that resolved safely — typically documented in the FAA's Aviation Safety Reporting System (ASRS) database or in NTSB incident-only reports — is required as a negative-control corpus. The system must not produce *critical* alerts on these scenarios; an *advisory* or *warning* tier is acceptable.

A first negative-control scenario is included in the implementation as `negative-control-asrs`. It places two arrivals on parallel KSFO approaches (10L/28R and 10R/28L) at 1.5 NM horizontal and 600 ft vertical separation — inside the simplified warning threshold but well outside the critical threshold. The expected behaviour is that the system emits a *warning* proximity-conflict alert and not a *critical* one; the scenario can be loaded from the scenario picker in the application header and inspected by the operator.

The corpus now holds seven scenarios. Five additions encode shapes that recur in ASRS controller narratives: a go-around with spacing restored (briefly 1.6 NM / 800 ft, warning acceptable, critical a failure), sequenced crossing-runway departures, three arrivals at legal wake spacing, a VFR corridor with 1.9 NM lateral but 1,500 ft vertical separation, and a sustained crosswind component of ~22 kts against a 25 kt limit. Measured: all six produce zero critical alerts; four are completely silent; the go-around flags its pair at warning. The VFR-corridor case is the geometry the withdrawn Table 1 miscounted 218 times, so it now stands as a regression test against that class of error. The silent four assert more than the sub-critical two: a population that murmurs on legal traffic drowns its own signal.

A corpus that asserts silence has to prove the doctrine ran, and for most of this one's life it did not. Measured on 2026-08-18, all six controls then in the corpus formed **zero wake pairs**: every scenario passed its zero-critical assertion without the wake detector, the component carrying the largest model correction in this work, ever reaching a pair to judge. The corpus could not distinguish "evaluated and found legal" from "never evaluated".

The cause was the lateral-stream pairing introduced during tuning. The staggered control's three arrivals sat at y = 0.5, 1.5 and 0.0, which is 1.5 NM of lateral spread against a 0.1 NM stream threshold, so they split into three separate streams and no pair could form. Its brief also still described the withdrawn time-based doctrine, four- and five-minute gaps against a three-minute requirement, months after wake separation became distance-based.

Rebuilding it surfaced a tension between two constraints that both have to hold before a wake pair exists: cross-track offset within the stream threshold, and the follower no more than 1,000 ft above the leader. On a 3 degree glideslope those two fight each other. Nine nautical miles of in-trail separation puts the follower roughly 2,900 ft higher, outside the vortex band, so the pair is skipped. Legal spacing and the vortex-band gate are therefore in tension on any descending profile, and the pairs the detector does see are biased toward aircraft close together along the approach. Whether that bias inflates the margin histogram of Section 6.7 is not yet measured; it is a stated hypothesis for the next sampling window, and one instrument short of an answer.

The rebuilt control represents aircraft level on an intermediate segment before glideslope intercept, the ordinary case, and forms two pairs at 9.2 and 10.2 NM against 5 and 3 NM requirements, both judged legal,

alerts still zero. A seventh control was added for the boundary the corpus lacked: a medium 5.3 NM behind a heavy, 0.3 NM inside the requirement, which must form a pair and stay silent. Two tests now pin the pairs forming, so a change that stops the wake doctrine running fails there rather than reading as a clean corpus pass.

These remain synthetic scenarios constructed from narrative shapes, not encodings of specific ASRS reports. A representative negative-control corpus — drawn from ASRS narratives across multiple US TRACONs and including both parallel-approach and converging-departure resolutions — is listed under *Open Items and Next Steps*. The single scenario is included so the architecture's negative-control assertion (no critical alert on a near-miss that resolved safely) can be exercised by a reader without writing code.

6.9 Baseline comparison

A monolithic-threshold baseline implementing the same doctrinal rules in a single function is included as `apps/atc/src/sim/rules-baseline.ts` and is the reference against which the orchestrated population is compared. The architectural claim of Section 3 is that the orchestrated population is more maintainable, not necessarily more accurate; the comparison is therefore primarily structural.

The comparison is run by `apps/atc/scripts/baseline-compare.mjs` over every scenario. One claim is confirmed and three are withdrawn.

Confirmed, with one qualification: the two implementations agree on category and severity across the incident corpus, on the doctrines the monolith implements. Two doctrines postdate the baseline freeze (weather-shift and runway-identity) and exist only in the orchestrated population, so it is a superset wherever the weather is adverse, and on Comair 5191 the monolith is blind outright: zero alerts against the population's one critical, because the wrong-runway doctrine has no block in the frozen function. They diverge only on the synthetic scenarios, where the `crisis` case yields eight alerts from the monolith and ten from the orchestrated population, the difference being two `gate-conflict` alerts that the baseline's gate block does not reach at the same thresholds. The baseline's header comment anticipates this, describing its behaviour as close to but not bit-identical with the orchestrated population.

Confirmed: the baseline has no projection layer, and the de-duplication in `runPredictiveRules` is implemented as described.

Built rather than withdrawn: suppression. When this section was first verified against the code, no suppression logic existed; `runAllRules` concatenated detector output and sorted by severity. It is now implemented in `applySuppression`, and Appendix A.4 records both what it does and the two respects in which the original specification was wrong. Its measured effect on this corpus is narrow: one suppression, on the `runway-conflict` scenario, where a `critical` runway-occupancy alert subsumes a `warning` alert describing the same occupancy on the same runway.

Corrected: the Tenerife figures. The monolith produces one alert; the orchestrated population produces two, `runway-conflict` and `weather-shift`. The earlier claim of "four overlapping alerts against two" is still wrong on the monolith's count, and the direction of the comparison is the reverse of what was claimed: the orchestrated population emits more, not fewer, because the monolith implements no weather doctrine. The categories cited for the orchestrated output are now the categories it emits.

Withdrawn: the maintainability conclusion, on the strength of the drill that was meant to support it. Revising the FAA wake matrix requires editing one declaration and one use site in the orchestrated population (`src/sim/rules.ts` lines 3 and 35) and one declaration and one use site in the monolith (`src/sim/rules-baseline.ts` lines 19 and 55). The cost is identical, so this drill does not distinguish the two architectures. On size the comparison runs against the architectural argument. At the time of measurement the orchestrated population was 261 lines of code against the monolith's 212. It is now 360, having gained the weather-shift detector, the wake-spacing split and the suppression table, against the monolith's unchanged 212. The monolith is smaller because it implements less: seven doctrines to the population's eight, with no suppression and no alert lifecycle. A size comparison between implementations of different scope measures scope, not maintainability, which is a further reason the drill in this section needs redesigning.

A further finding bore on Section 3 rather than on this comparison, and has since been fixed. The pattern defined in Section 3.1 assigns one detector to one doctrine, but `wake-spacing` was emitted from inside `detectRunwayConflicts`, which at 76 lines was the largest of the six functions and carried two doctrines. It is now split: `detectWakeSpacing` and `detectRunwayConflicts` share a `groupByRunway` helper and carry one doctrine each. The population is eight detectors, and the pattern holds across all of them.

What remains of the maintainability argument is the registration mechanism, and the redesigned drill has now been run. A new doctrine, runway-surface contamination, was added to both implementations in a single commit: `surfaceFriction` was already modelled on every runway and read by nothing, so the doctrine is real rather than contrived. Measured from the commit's own diff, the orchestrated population took 44 added lines (a 43-line self-contained detector plus one registration line) and the monolith took 39 (the same logic inlined). Line cost is near-equal, and the drill is reported that way.

The difference the drill did expose is isolation. The monolith block had to rename its `using` variable to avoid colliding with the crosswind block's scope and required care about loop-variable collision with the blocks above it; the orchestrated detector shares nothing with its neighbours. At one added doctrine that composition cost is small, and the claim it supports is correspondingly modest: the architecture buys isolation between doctrines, not fewer lines. Both implementations emit identically on the scenario that exercises the new doctrine (five alerts, same categories and severities on the crosswind-storm case, which gained wet pavement — rain without a wet surface having been a latent inconsistency the new doctrine made observable).

The drill also produced a finding about equivalence maintenance itself: every doctrine added from here doubles its implementation cost, because the monolith exists only to be compared against. That burden is the honest price of keeping Section 6.9 measurable, and it is why the detector population is nine and the corpus stops growing the baseline after this drill.

6.10 Reproducibility

The scenarios are deterministic, and determinism is now enforced rather than asserted: `src/sim/sim.test.ts` runs every scenario twice and compares the alert identifiers and severities, and separately checks that the detector pass does not mutate the state it is given. Each scenario loads with `speed: 0` so the output can be inspected without temporal evolution, through the picker in the application header and without writing code.

Two claims made in earlier drafts of this section were not true at the time of writing and are corrected here.

The deployment URL `atc.fbritoferreira.com` resolves and serves the application as of 2026-08-04: a proxied CNAME managed in Terraform (matching how the zone's other Pages hostnames are managed) plus a custom-domain binding on the `fbf-atc` Pages project. Earlier drafts cited this hostname while it had no DNS record at all.

The application is not open-source as of this draft. It lives in a private monorepo alongside unrelated material, so publication requires extracting it into a standalone repository rather than flipping a visibility setting. Apache-2.0 licence text and a `CITATION.cff` are in place; the Zenodo deposit and its DOI follow the extraction. Both are tracked under *Open Items and Next Steps*.

The two analyses in Sections 6.7 and 6.9 are reproducible from the repository by the commands documented in `apps/atc/README.md`. Neither requires an API key: traffic comes from the volunteer adsb.lol feed and weather from the NOAA Aviation Weather Center.

---

7. Discussion

7.1 What the diagnostic-shape check does and does not show

The shape check shows that the system's diagnostic output, when given the inputs the controllers had at the time of each event, corresponds to the conclusion the official investigation reached. It does not show

that the system, if deployed, would have prevented any of the events. Each event has a chain of upstream contributing factors — clearance phraseology, procedural failures, sector workload — that are outside the detector population's scope. It also does not show what the system would emit on nominal traffic; the false-positive analysis required to characterise that is open work and is described in Section 6.7.

7.2 Determinism as a design constraint, not a limitation

A reader from the broader machine-learning community might object that the system uses no learned components. This is intentional. Aviation operates under regulatory regimes (FAA, EASA) that require deterministic behaviour, traceable decision logic, and reproducibility under audit. A learned reasoner in the alert loop would have to satisfy these requirements before being eligible for any operational role. The architecture proposed here is suitable for the present regulatory environment; a future extension with learned components is discussed in Section 8.

7.3 The role of look-ahead

The forward projection is the single most consequential design choice in the architecture. Without it, the system would only describe the present. With it, the system surfaces the situation the controller will face in 60, 120, and 180 seconds. Endsley's level 3 is operationalised directly. Several of the historical incidents — Avianca 052 on its first hold extension, the JFK near-miss on the second clearance — would have triggered look-ahead alerts ahead of the moment the controller was forced to act.

7.4 Limitations

1. **Reconstruction, not deployment.** The system has not been integrated into any operational ATC environment and is not represented as suitable for one. The diagnostic-shape check is by archive review, not by live trial.
2. **Data access bounds any live study of this shape.** Both public ADS-B sources stopped serving this client after roughly ten days of continuous five-airport sampling: adsb.lol began returning empty responses and airplanes.live returned 403. Sampling was stopped rather than circumvented, which is why the post-correction window in *Open Items* covers 72 minutes rather than 24 hours. Volunteer surveillance networks are a research dependency measured in goodwill, and a study needing sustained multi-airport sampling requires either a facility data agreement or a locally operated receiver.
3. **Detector population scope.** The ten detectors cover the principal doctrines exercised by the corpus. Additional doctrines (wake encounter on departure, RNAV path conformance, controlled-flight-into-terrain proximity, approach-path alignment with runway centerlines, which the Air Canada 759 taxiway-overflight case requires and the current runway model cannot support) are not implemented.
4. **Linear projection.** The forward look-ahead assumes constant heading and ground speed. Real arrival trajectories curve; the projection is therefore most accurate at short horizons and degrades beyond two minutes.
5. **Open data limitations.** ADS-B coverage is non-uniform; aviationweather.gov has occasional outages. The system degrades gracefully (last-known-good state) but does not yet have a formal data-quality alert.
6. **No formal latency budget.** The system is intentionally browser-based for inspection and reproducibility; an operational deployment would require a different runtime with a published latency budget.

---

8. Pathway to US Adoption

This section addresses how the architecture proposed here might be brought into operational use within the US National Airspace System (NAS) given current procurement, certification, and research-funding

mechanisms. It does not represent a current programme of work; it identifies the steps a US adoption pathway would entail and the federal stakeholders relevant to each.

8.1 Alignment with named US priorities

Three named US priorities provide the relevant policy context:

- **NTSB Most Wanted List 2025–26** identifies *Implement Comprehensive Runway Safety* and *Improve Surface and Approach Safety* as continuing priorities. The runway-conflict, gate-conflict, and cascading-delay detectors directly serve these priorities. The proximity-conflict detector addresses the *Reduce Mid-Air Collision Risk* item added in response to the Potomac incident.
- **FAA NextGen Implementation Plan** identifies *Surface Operations and Data Sharing* and *Trajectory-Based Operations* as continuing investment areas. The orchestration shape proposed here is complementary: NextGen brings improved surveillance and trajectory data, this architecture surfaces conflict diagnoses against that data.
- **FAA Aviation Safety Action Plan (2025)**, issued in the aftermath of the Potomac collision, includes a specific commitment to review helicopter-fixed-wing interaction corridors at congested fields. The proximity-conflict detector with helicopter-corridor reasoning, as exercised in Section 6.4, is directly responsive.

8.2 Funding mechanisms

The relevant federal funding mechanisms are:

- **Small Business Innovation Research (SBIR) — DOT/FAA topics**. The FAA publishes SBIR solicitations under DOT-wide and FAA-specific topic areas; relevant topic codes in recent cycles have included surface safety, controller decision support, and AI/ML for ATM. Phase I awards are typically $150,000 over six months; Phase II awards extend to $1,000,000 over two years. SBIR is an obvious first step for the present work because it is small-business-friendly, does not require prior contractor history, and aligns the architecture with a specific topic statement.
- **FAA Broad Agency Announcement (BAA)**. The FAA periodically issues BAAs for research with no fixed topic. BAAs are best suited to mature architectures with a deployment partner.
- **NASA Aeronautics Research Mission Directorate (ARMD)** — *Airspace Operations and Safety Program*. NASA Ames Research Center has a long history of ATC decision-support research (CTAS, SARDA, surface operations); ARMD partnerships with industry are mediated by Space Act Agreements.
- **FAA William J. Hughes Technical Center**. The Tech Center operates the laboratory environments where new ATC decision-support concepts are integrated and tested before any operational consideration. A Cooperative Research and Development Agreement (CRADA) with the Tech Center would be a natural mid-stage step.
- **MITRE Center for Advanced Aviation System Development (CAASD)**. MITRE CAASD is the FFRDC that supports FAA ATM modernisation. Engagement with CAASD is typically through FAA-directed task orders rather than direct contracting.

8.3 Certification posture

The proposed architecture is a *decision-support* system, not an active separation-assurance system. This places it outside the certification regime that governs onboard collision-avoidance systems (TCAS) but within the regime governing ATC automation tools. Relevant standards are:

- **DO-278A** — Software Integrity Assurance Considerations for Communication, Navigation, Surveillance, and Air Traffic Management Systems. Applicable to ground-based ATM software. Equivalent assurance levels for ground software are SWAL-1 through SWAL-6; a decision-support advisory of the kind described here would be assessed at SWAL-3 or SWAL-4.
- **NAS Cybersecurity and Privacy Program (NAS-CSEPP)** — the FAA programme that governs cybersecurity for ground-based NAS systems.

- **FAA Order 1370.121** — Information Systems Security, the operative FAA IT-security order for ground-based systems.
- **NIST SP 800-53** — Security and Privacy Controls for Information Systems and Organizations, the federal baseline that NAS-CSEPP and FAA Order 1370.121 inherit from.

Note: DO-326A and DO-356A are airworthiness-cybersecurity standards applicable to airborne systems and are *not* the operative regime for ground-based ATM software. They are mentioned here only to clarify the boundary; this work falls under the ground-based regime above.

The properties stressed in Sections 4.1–4.5 (determinism, explainability, severity tiering, observability, audit trail) are the properties DO-278A would require evidence of in a SWAL-3 or SWAL-4 assessment. The architecture is designed with these properties from the start, which lowers (but does not remove) the cost of a certification engagement.

8.4 Concrete next steps

The concrete short-term steps that move the work from open-source proof-of-concept toward a US adoption pathway are:

1. **Publish the source code** under an OSI-approved licence (Apache 2.0 or MIT) with a citable DOI via Zenodo, so federal contracting officers can evaluate it without procurement friction.
2. **Submit an arXiv preprint** of this thesis once the open work in Section 6.7–6.9 is closed.
3. **Submit a paper to the Digital Avionics Systems Conference (DASC) or ATM Seminar** for peer review and US ATM-research community visibility.
4. **Engage MITRE CAASD informally** through the public ATM research community before any FAA contracting conversation.
5. **Identify a current FAA SBIR topic** under the DOT solicitation and submit a Phase I proposal if a topic matches; otherwise wait for the next cycle.
6. **Letter of interest from a US aviation-safety research lab** (NASA Ames, FAA Tech Center, MITRE CAASD, or a Tier 1 university programme such as Georgia Tech ASDL, Purdue PEGASAS, or MIT Lincoln Laboratory).

These steps establish the architecture's US-priority alignment with verifiable third-party signal, which is the gap identified in the two completed rounds of external-persona review.

---

9. Future Work

1. **Wake-encounter on departure.** Add a detector covering wake encounters during departure climb, complementing the existing arrival-side wake-spacing detector.
2. **Trajectory-based projection.** Replace the constant-heading model with a published-flight-plan and STAR-aware trajectory projector.
3. **Cross-airport awareness.** Treat the New York TRACON (KJFK + KLGA + KEWR), the Bay Area (KSFO + KOAK + KSJC), and similar groupings as single decision domains, surfacing inter-airport conflicts.
4. **Learned anomaly detector under a formal verification envelope.** A learned component that flags anomalies inside the orchestrator's input stream, with a verifiable upper bound on false-negative rate, would be a candidate for inclusion within the existing regulatory constraints.
5. **Audit-format export.** Formalise the alert + reason + action + override-history trace into a compliance-grade audit format suitable for FAA or NTSB review.
6. **Replay and counterfactual analysis.** Record live-mode sessions and replay them offline with rule variations to support post-incident review and detector iteration.

---

10. Conclusion

This thesis has proposed a multi-detector decision-support architecture for air traffic control and exercised it against nine publicly-documented historical incidents, eight of them detectable by construction. The Specialist Detector pattern organises operational risk detection as a population of independent rule specialists coordinated by a thin orchestrator over a shared blackboard, with explicit reasoning and recommended actions attached to every alert. The implementation is open-source and operates on live ADS-B and METAR data for seventeen airports, ten of them in the United States. A diagnostic-shape check against the historical-incident corpus shows that the system's output corresponds to the conclusion the responsible investigation bureau reached in every case. A strict-sense validation, including a false-positive characterisation on nominal traffic, a negative-control corpus, and a baseline comparison, is outlined as open work and is the immediate next priority. The architecture is offered as a starting point for production decision-support work in US sectors of national importance, with a concrete pathway to US adoption described in Section 8.

References

Bainbridge, L. (1983). Ironies of Automation. *Automatica*, 19(6), 775–779.

Cummings, M. L. (2017). Operator Interaction with Centralized Versus Decentralized Unmanned Vehicle Architectures. *Journal of Aerospace Information Systems*, 14(7), 376–388.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1), 32–64.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R. (1980). The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *ACM Computing Surveys*, 12(2), 213–253.

Federal Aviation Administration (2024). Order JO 7110.65 — Air Traffic Control. US Department of Transportation.

Federal Aviation Administration (2024). Title 14 of the Code of Federal Regulations, Part 91.167 — Fuel requirements for flight in IFR conditions.

International Civil Aviation Organization (2016). Doc 4444 — Procedures for Air Navigation Services — Air Traffic Management (16th ed.).

Kuchar, J. K., and Yang, L. C. (2000). A Review of Conflict Detection and Resolution Modeling Methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4), 179–189.

Italian Agenzia Nazionale per la Sicurezza del Volo (ANSV) (2004). Final Report on the accident at Milano Linate, 8 October 2001.

National Transportation Safety Board (1991). Aircraft Accident Report AAR-91/04: Avianca, the Airline of Colombia, Boeing 707-321B, HK 2016, fuel exhaustion, Cove Neck, New York, January 25, 1990.

National Transportation Safety Board (1991). Aircraft Accident Report AAR-91/08: Runway collision of USAir Flight 1493, Boeing 737, and SkyWest Flight 5569, Fairchild Metroliner, Los Angeles, California, February 1, 1991.

National Transportation Safety Board (2026). Final Report DCA25MA108: Midair Collision between PSA Airlines/American Eagle Flight 5342 and US Army Black Hawk H-60, Reagan Washington National Airport, January 29, 2025.

Pang, Y., Zhao, Y., Yan, H., and Liu, Y. (2021). Data-driven trajectory prediction with weather uncertainties: A Bayesian deep learning approach. *Transportation Research Part C*, 130, 103326.

Parasuraman, R., and Riley, V. (1997). Humans and Automation: Use, Misuse, Disuse, Abuse. *Human Factors*, 39(2), 230–253.

Power, D. J. (2002). *Decision Support Systems: Concepts and Resources for Managers*. Quorum Books.

RTCA (2008). DO-185B — Minimum Operational Performance Standards for Traffic Alert and Collision Avoidance System II (TCAS II).

Spanish Comisión de Investigación de Accidentes e Incidentes de Aviación Civil (CIAIAC) (1978). Investigation of the collision between Pan American World Airways Boeing 747 and KLM Royal Dutch Airlines Boeing 747 at Los Rodeos Airport, Tenerife, 27 March 1977.

Stone, P., and Veloso, M. (2000). Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3), 345–383.

Sun, J., Ellerbroek, J., and Hoekstra, J. M. (2020). OpenSky report 2020: Analysing in-flight emergency situations. *Proceedings of the 39th Digital Avionics Systems Conference (DASC)*.

Wickens, C. D. (1992). *Engineering Psychology and Human Performance* (2nd ed.). HarperCollins.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2nd ed.). Wiley.

Appendix A — Implementation Details

A.1 Detector contract

```ts
type Alert = {
  id: string;
  severity: "critical" | "warning" | "advisory" | "info";
  // Every category below has an emitter in rules.ts. A previous revision also
  // declared "missed-handoff", which was removed: SimState models a single
  // sector, so there is no inter-sector handoff to miss.
  category:
    | "runway-conflict"
    | "wake-spacing"
    | "gate-conflict"
    | "fuel-hold"
    | "crosswind"
    | "weather-shift"
    | "cascading-delay"
    | "proximity-conflict"
    | "runway-surface"
    | "runway-identity"
    | "squawk-emergency";
  title: string;
  detail: string;
  flightIds: string[];
  reason: string;
  suggestedAction: string;
  createdAtTick: number;
  lookaheadMin?: number;
};

type Detector = (state: SimState) => Alert[];
```

A.2 Orchestrator

```ts
const DEMOTE: Record<Alert["severity"], Alert["severity"]> = {
  critical: "warning",
  warning: "advisory",
  advisory: "info",
```

```
  info: "info",
};

const demoteForHorizon = (
  severity: Alert["severity"],
  lookaheadMin: number,
): Alert["severity"] => {
  const once = DEMOTE[severity];
  return lookaheadMin >= 3 ? DEMOTE[once] : once;
};

export const runPredictiveRules = (state: SimState): Alert[] => {
  const present = runAllRules(state);
  const presentIds = new Set(present.map((a) => a.id));
  // Base ids already reported at a nearer horizon: the nearest horizon
  // carries the report, so the operator sees one alert per condition.
  const reported = new Set<string>();
  const predicted: Alert[] = [];
  for (const lookahead of [1, 2, 3]) {
    const projected = projectState(state, lookahead);
    for (const a of runAllRules(projected)) {
      if (presentIds.has(a.id)) continue;
      if (reported.has(a.id)) continue;
      reported.add(a.id);
      predicted.push({
        ...a,
        id: `predicted-${lookahead}-${a.id}`,
        severity: demoteForHorizon(a.severity, lookahead),
        title: `IN ${lookahead} MIN: ${a.title}`,
        detail: `Forecast: ${a.detail}`,
        lookaheadMin: lookahead,
      });
    }
  }
  return [...present, ...predicted];
};
```

A.3 Representative detector: proximity-conflict

```
const detectProximityConflict = (state: SimState): Alert[] => {
  const alerts: Alert[] = [];
  const active = state.flights.filter(
    (f) => f.phase !== "at-gate" && f.phase !== "departed" && f.altitudeFt > 0,
  );
  const seen = new Set<string>();
  for (let i = 0; i < active.length; i++) {
    for (let j = i + 1; j < active.length; j++) {
      const a = active[i];
      const b = active[j];
      const horizNm = Math.hypot(
        a.positionNm.x - b.positionNm.x,
        a.positionNm.y - b.positionNm.y,
      );
      const vertFt = Math.abs(a.altitudeFt - b.altitudeFt);
```

```
      if (horizNm < 2 && vertFt < 1000) {
        const key = [a.id, b.id].sort().join("-");
        if (seen.has(key)) continue;
        seen.add(key);
        const critical = horizNm < 0.5 && vertFt < 200;
        alerts.push({
          id: `prox-${key}`,
          severity: critical ? "critical" : "warning",
          category: "proximity-conflict",
          title: `${a.callsign} and ${b.callsign} converging - ${horizNm.toFixed(1)} NM / ${Math.round(
          detail: `${a.callsign} ${Math.round(a.altitudeFt / 100).toString().padStart(3, "0")}, ${b.cal
          flightIds: [a.id, b.id],
          reason:
            "TCAS RA threshold ~0.5 NM horizontal / 200 ft vertical. Mixed fixed-wing and rotor traffic
          suggestedAction:
            "Issue immediate vector or altitude change to one aircraft. Confirm visual on both. Re-route
          createdAtTick: state.tick,
        });
      }
    }
  }
  return alerts;
};
```

A.4 Orchestrator coupling and suppression logic

Earlier drafts of this appendix described four orchestrator responsibilities, of which two were implemented. Verification against the code exposed the gap; the two missing behaviours have since been built, and the two inaccurate descriptions corrected. The list below states what the code does, and records what each item previously claimed, because Section 6.9 turns on the difference.

1. **De-duplication. Implemented, and the cross-horizon half was found dead.** A projected alert is dropped if its `id` appears in the present-state set, and dropped if a nearer horizon has already reported the same condition; both checks are set lookups. The second check is newer than it should be. As originally written, the predicate compared a horizon-prefixed identifier against an unprefixed one and could never match, so the same conflict reached the operator at up to three horizons at once: measured before the fix, Tenerife and Linate each reported one proximity pair twice and the runway-conflict synthetic reported it three times. No existing assertion could see the bug, because the duplicate identifiers differed by prefix and severities were identical across horizons. The fix keeps the nearest, most severe report, a regression test now fails on any base identifier seen at two horizons, and the corpus counts in Section 6 reflect the corrected output.

2. **Severity demotion of projections. Horizon-dependent, as of this revision.** Demotion deepens with the projection horizon: one step at one and two minutes out, two steps at three. This closes the open work recorded in the previous revision, which found the implementation demoting identically at every horizon. One deliberate deviation from the original design note stands: that note let a one-minute projection keep its base severity, but here a projected alert is never critical at any horizon, because the Section 6.8 negative-control guarantee and the Section 2.6 alert-fatigue argument both reserve the critical tier for conditions holding in the present state. Weighting demotion by how far the threshold was exceeded, rather than by horizon alone, remains open.

3. **Suppression of lower-severity alerts. Implemented, after being described without being built.** `applySuppression` in `src/sim/rules.ts` holds a table of `(higher_category, suppressed_category)` pairs. When a `critical` alert carrying a `runwayId` is present, a lower-severity alert of a paired category on the same runway is marked `suppressedBy` that alert's identifier.

Two details came out of building it. First, the pairs the earlier drafts specified, a critical runway-conflict suppressing wake-spacing and cascading-delay, fire on no scenario in the corpus: they need a critical runway-conflict co-occurring with a lower-tier wake or flow alert on the same runway, and that combination does not arise. They are retained because they are correct, not because they are exercised. Second, the specification missed the redundancy that does arise. `detectRunwayConflicts` emits up to three alerts per runway, and on the `runway-conflict` scenario a `critical` "multiple aircraft on runway" sits beside a `warning` "simultaneous arrival and departure intent" describing the same unsafe occupancy at a lower tier. That pair is now in the table and is what the suppression test exercises.

Suppressed alerts are marked rather than deleted. An operator surface that hides an alert must be able to say what hid it, and a post-incident review has to see everything the detectors found, including what the display withheld. `activeAlerts` derives the operator view by filtering them out.

4. **Alert lifecycle and flicker suppression. Implemented, in `src/sim/lifecycle.ts`.** `reconcileAlerts(previous, current, tick)` carries `firstSeenTick` and `lastSeenTick` across ticks and holds an alert that has vanished from detector output for `ALERT_GRACE_TICKS` (three, a one-minute grace period at the 20-second live poll interval) before dropping it, marking it `stale` in the interim. Severity is refreshed from the current pass, so an escalating conflict shows its new tier immediately rather than waiting out the grace period.

   This addresses a real defect rather than a documentation gap. Before it, each tick's alert list was computed from scratch with no cross-tick state, so a pair of aircraft sitting either side of a separation threshold produced an alert that appeared and vanished on consecutive polls. Flicker of that kind is among the fastest ways to lose an operator's confidence in a display, and it was invisible in scenario playback because scenarios are static.

   The function is pure: state lives with the caller, in the live store, and not inside a detector, so the determinism of Section 4.1 survives. `SimState.alerts` now carries the operator view and `SimState.trackedAlerts` the full set.

The claim in earlier drafts that these properties are unit-tested in `src/sim/predict.test.ts` was false: that file did not exist, and the repository contained no test files at all, with `vitest` configured and nothing for it to run. `src/sim/sim.test.ts` now holds 76 tests, one per verifiable claim in this document, including the suppression and lifecycle properties above. Where a claim proved false the test pins the actual behaviour and says so, so the two cannot silently diverge again.

A.5 Historical-corpus scenario builders

Each historical scenario is a pure function `() => SimState` registered in `src/sim/scenarios.ts`. Loading a scenario freezes time (`speed: 0`), sets the airport runway geometry to the airport in question, sets the weather to the conditions reported by the relevant METAR, and places the aircraft at the positions established by the official investigation.

A.6 Reproducibility checklist

- Source: [TODO — public repo link]
- Live deployment: `https://atc.fbritoferreira.com`
- Historical scenarios accessible via the scenario picker in the header
- Data sources documented in Section 5.5; all are public and free
- No authentication required

Open Items and Next Steps

This draft is at v0.5 and is released for academic-advisor review and third-party engagement. The items below represent the work the author intends to complete before any peer-reviewed submission.

Completed since v0.4:

- A test suite exists. `src/sim/sim.test.ts` holds 61 tests, one per verifiable claim made in this document, and passes. Appendix A.4 previously cited a test file that did not exist in a repository that contained no tests at all.
- The false-positive harness imports the production detector modules instead of reimplementing them. The figures it previously produced are withdrawn in Section 6.7.
- The Section 6.9 comparison is measured by a script (`scripts/baseline-compare.mjs`) rather than asserted, which is what surfaced the withdrawn claims in that section and in Appendix A.4.
- Apache-2.0 licence text, `CITATION.cff`, and a README stating the prototype's limitations are in the repository.
- `weather-shift` is implemented, so Tenerife now emits the two categories earlier drafts claimed for it. `missed-handoff` was removed from the `Alert` type: a single-sector state model cannot honestly emit it. Every declared category now has an emitter, asserted on every run of `baseline-compare.mjs`.
- Suppression is implemented (Appendix A.4 item 3), including a same-runway redundancy pair the original specification missed.
- The alert lifecycle is implemented in `src/sim/lifecycle.ts` (Appendix A.4 item 4) and wired into the live store, so alerts no longer flicker when a condition oscillates across a threshold.
- `wake-spacing` was split out of `detectRunwayConflicts`, so the one-detector-per-doctrine pattern of Section 3.1 now holds across all eight detectors.
- The negative-control corpus is seven scenarios (Section 6.8), measured at zero critical alerts corpus-wide, five fully silent.
- The doctrinal-change drill has been run with a new runway-surface doctrine added to both implementations; results in Section 6.9.
- `atc.fbritoferreira.com` resolves and serves the application (DNS in Terraform, Pages binding on the project; fixed 2026-08-04, and the fix also unblocked the zone's Terraform workflow, which had been failing on a stale state entry for a deleted typo-twin zone).
- The `nominal` scenario is silent. Its seed fuel loads put three arrivals in fuel emergencies, so the reference scenario in the picker was raising three critical alerts.

Required before preprint submission (v0.6):

1. Publish the standalone repository — **deferred by decision (2026-08-04) to ~January 2027**, after threshold tuning and further validation work, so the repo does not go public with its headline finding still reading "untuned." Note the knock-on: the JOSS six-month public-history requirement starts at publication, so a JOSS submission lands mid-2027 at the earliest; the Zenodo DOI and the `[TODO]` URLs wait with it. The extraction itself is done and verified: `apps/atc/scripts/extract-standalone.sh` produces a self-contained repository that installs, passes all 146 tests, typechecks and builds outside the monorepo. Remaining are the deliberate acts: `git init` and `gh repo create` on the extracted tree, the Zenodo GitHub integration and a release for the DOI, and the two TODO fields in `CITATION.cff` (ORCiD, DOI). Then substitute the URL and DOI for the `[TODO]` placeholders in Section 5.1, Section 6.10, and Appendix A.

2. Two model errors found and corrected, and the consequence measured. The 24-hour five-airport window's wake-margin histogram (77.8% of 3,707 same-stream pairs below the critical line, 5.3% legal) showed the wake defect was the model, not the boundary: TIME minima compared against ETA differences, where arrival separation is distance-based, at magnitudes 1.4 to 1.7 times the FAA JO 7110.65 requirement. Wake separation is now in-trail nautical miles against that table, approach-phase only, vortex-band gated, revised in both implementations. The critical for "multiple arrivals on final" was withdrawn as false (a continuous in-trail stream is normal operations; it produced 156 of 182 runway-conflict criticals at KDFW). Post-correction spot check: critical alerting down 26 to 66 per cent per airport against roughly 50 per cent more traffic, false runway-conflict criticals eliminated, margin distribution 77.8% to 35.7% critical. Raw CSVs: `atc-data/fp-*-tuned3-*.csv` (before) and `atc-data/fp-*-tuned4-partial-72min.csv` (after). **Item 5 corrects part of this item's explanation.**

3. The critical-rate decision is now an implementation plan rather than an open question: Table 2's crosstab attributes 99.3% of criticals to the two causes in item 2. Section 2.6 makes alert fatigue the standard against which decision-support systems are judged, so a rate this high is a claim-limiting result and not a tuning detail.

4. Send at least one letter of interest and record the response.

5. A third real bug found and fixed, which invalidates part of item 2's explanation. Live runway inference returns a single runway END label (`"04L"`) while the airport registry stores paired strips (`"04L/22R"`), so every lookup keyed on the inferred assignment failed on live data while passing on all 23 scenarios, whose designators match registry ids exactly. Three tuned behaviours depended on that lookup and each failed differently: the cross-track clustering fell back to a zero-degree axis, giving an axis error equal to the runway heading (88 degrees at KATL's four parallels on 092, 4 degrees at KDFW's five strips on 184); wake stream pairing fell back to one stream per runway label, so it never split a stream live; and the established-on-final gate rejected every live arrival. The 159 to 8 runway-conflict reduction in item 2's lineage is a real count produced by a different mechanism than the one credited, since at KATL the clustering split a single in-trail stream by range rather than distinguishing parallels. Fixed at the root in both implementations (`runwayHasEnd` and `resolveRunway`), so the monolithic baseline is not left carrying the defect the comparison would then attribute to architecture. A five-airport 24-hour window started 2026-08-17 is the first in which all three behaviours execute against live traffic. The general lesson belongs in Section 6: a scenario corpus cannot catch a defect whose trigger is a shape difference between its own fixtures and the live feed, and the passing scenario corpus did not. What caught it was disbelieving a good result, a window reporting zero wake pairs at 28 aircraft per snapshot.

Queued for subsequent revisions:

6. Title-page formatting for arXiv submission (LaTeX conversion, ORCID identifier).
7. Additional historical incidents (Comair 5191 wrong-runway lineup, Air Canada 759 SFO taxiway lineup), each of which would require an additional detector kind not yet implemented (runway-identity verification, taxiway-vs-runway alignment).
8. A short companion essay for general-audience publication summarising the diagnostic-shape findings.
9. A trajectory-based projector to replace the constant-heading-and-speed extrapolation currently used in the predictive look-ahead.