ARCHITECTURE

Bounded Agentic Coordination: A Safety Architecture for AI-Assisted Wildfire Response

"Agentic AI" should make a careful person nervous — especially near aircraft and fire. This paper is about why Inari Watch's system is safe by construction: not because we trust an AI to behave, but because of what the architecture will not let it do.

Start with the worry

A great deal of what is sold as "agentic AI" is a capable model placed in a loop with tools and an open-ended goal. That pattern has produced exactly the surprises a careful person fears: systems that pursue an objective in unintended ways, that are confidently wrong, that can be steered by their own inputs, that behave well in a demo and badly in the world. Put that pattern near aircraft, near fire, near anything kinetic, and nervousness is the correct response. We share it.

This paper is written for the reader who wants to know which kind of company this is before spending more time — the careful kind or the reckless kind. The honest way to answer that is not to reassure you. It is to show you the design, including the parts we have not solved.

Safety as a property of the architecture, not of behavior

There are two ways to make an autonomous system safe. One is to build a capable agent and try to make it behave — through training, evaluation, and the hope that good behavior generalizes to situations you didn't test. The other is to constrain what the system is able to do, structurally, so that safety does not depend on any one component behaving well. We build the second kind.

Bounded agentic coordination is the precise name for it: bounded software components that perceive, communicate, and propose within strict, declared limits, under people who decide. The intelligence in the system is used where it earns its place — perception, characterization, prioritization — and it feeds a coordination layer that is explicit, inspectable, and gated. This is the distinction that matters most, so we will state it plainly: this is not a collection of language models conversing with one another and improvising actions. The coordination layer is a classical, well-understood multi-agent pattern — the Contract Net Protocol — in which narrow agents bid for bounded tasks within policy. Machine learning is confined to where it belongs, and its outputs are treated as fallible inputs to be verified, never as decisions. The system is not a mind we hope is aligned. It is machinery we can reason about.

Many narrow agents, not one capable one

The system is operated by six classes of deliberately narrow software agents — sensor, flight, depot, regional coordination, operator-liaison, and policy. The narrowness is not a limitation we are apologizing

for; it is the safety property. There is no single agent that holds broad authority and a general goal. There is a division of labor in which each agent has a small scope and a hard ceiling on what it may do on its own:

- A **sensor agent** reports anomalies; it cannot task an aircraft or commit any resource.
- A **flight agent** flies an approved mission within its envelope; it cannot release a payload or leave that envelope without human authorization.
- A **depot agent** dispatches verification flights within policy; it cannot authorize a payload-equipped or out-of-envelope mission.
- The **regional coordination agent** — the most capable component — fuses, assesses, and proposes; it must route every kinetic action and every out-of-envelope dispatch to a human, and it cannot alter the policy it runs under.
- An **operator-liaison agent** presents decisions for authorization; it cannot authorize anything on a human's behalf.
- The **policy agent** holds and enforces the rules; it cannot act inside them, and changes to policy are themselves logged.

This is least privilege applied to autonomy. The blast radius of any single agent failing, being wrong, or being manipulated is bounded by design, because no agent has the reach to turn its own error into a consequential action.

The line a machine may not cross

The most important line in the system is the one between what software may do and what only a human may authorize. It is not a setting or an interface convention; it is enforced in the policy layer. Any action that releases a payload, commits a regional resource beyond a set threshold, or steps outside a declared envelope requires explicit, contemporaneous human authorization. Agents propose; humans authorize. So when someone asks whether the system is autonomous, the precise answer is that it is autonomous in how it senses and coordinates, and never in the decision to act. Defaults fail closed: lose the control link and an aircraft climbs and returns — it does not improvise.

A skeptic will immediately raise the real problem, and it is the right one: a human "in the loop" who rubber-stamps whatever the machine proposes is not a safeguard. It is a liability shield wearing the costume of one. This is the failure mode most likely to defeat a design like ours, so we treat it directly. The mitigation lives in how decisions are presented — the operator-liaison agent surfaces a prioritized recommendation with its reasoning and the evidence behind it, and the operator can modify or decline, not merely approve. That reduces automation bias. It does not eliminate it. Eliminating it is a training, staffing, and human-factors problem as much as a software one, and we treat it as a standing obligation rather than a solved checkbox. Anyone who tells you software alone has cured automation bias has not understood the problem.

Authority that is declared, not emergent

The reason those boundaries can be trusted is that they are written down as data, not buried in code or left to runtime improvisation. A shared policy registry declares each agent class's authority limits, escalation triggers, and operational envelopes. Two consequences follow for safety.

First, there is no open-ended objective for an agent to satisfy in a surprising way. An agent has a bounded task and an explicit envelope — which closes off the specification-gaming and reward-hacking failure modes that arise when you hand a capable optimizer a goal and tell it to maximize. Nothing here is maximizing anything; agents execute declared tasks within declared limits.

Second, the rules are inspectable and versioned. An evaluator, an auditor, or a regulator can read exactly what the system was permitted to do, and the policy in force is itself under change control. The agents also do not address one another in free-form language. They exchange typed, signed, versioned messages over a mutually authenticated bus, so one agent cannot talk another into something outside policy. There is simply no channel for the kind of manipulation that open-ended agent chatter invites.

Accountability built in, not bolted on

Every action the system takes is logged against the policy version in effect, and the log is append-only — reconstructable end to end. This answers the question a careful person asks last and means most: when something goes wrong, can you tell what happened, and why? For most autonomous systems the honest answer is "partially." Here it is the design's explicit job to make the answer complete — every proposal, every authorization or refusal, every dispatch, every escalation, recorded against the exact policy that governed it. That serves the obvious operational needs of after-action review, liability, and regulatory inspection. But it is also a safety mechanism in its own right, because a system whose every decision is on the record is a system whose behavior can be audited, challenged, and corrected — rather than guessed at after the fact.

What we don't claim

A safety argument that admits no open risk is not a safety argument; it is marketing. Here is what we do not claim.

We do not claim the machine learning is infallible. Perception produces false positives and false negatives, always will, and that is precisely why a detection is treated as a fallible input — corroborated by a second sensor or a verification flight — and never as ground truth for action. We do not claim to have eliminated automation bias; we have engineered against it and we keep working at it. We do not claim that multi-agent systems can never surprise their builders; bounded envelopes, typed protocols, and the human gate constrain the surprises that could become consequential, but we test for emergent behavior rather than assuming it away. And we do not claim that a demonstration is a deployment. That gap is real and it is why Phase I is recon and verification only — no suppression, nothing kinetic released

from an aircraft — and why authority is earned incrementally: prove the perception, then the coordination, then the human workflow, and only then extend the envelope. The architecture is deliberately built so that adding a capability later means attaching it to the same authorization spine, not loosening it.

A vendor who tells you a safety-critical AI system has no open risks is telling you they have not found them. We would rather show you ours.

The careful kind

The difference between the careful kind of company and the reckless kind is not how capable the AI is. It is whether safety depends on the AI behaving. We have built so that it does not: narrow agents with hard ceilings, authority declared as inspectable data, a structural human gate on anything consequential, and a complete record of everything done. If that is the standard you were hoping to hold a system to, then the next conversation is an evaluation — and we would welcome it.

About Inari Watch

Inari Watch is a veteran-founded company building an integrated system of systems for regional wildfire defense. We unify the sensors, aerial assets, and data feeds a region already depends on into one coordinated capability — connected by AI at the edge and bounded agentic coordination, with humans in command of every consequential decision.