DAN WALSH

Senior Software Engineer | Agent reliability - evals, adversarial review, verification

London | hello@danwalsh.co | danwalsh.co | github.com/dang-w | linkedin.com/in/dgwalsh

SUMMARY

Research interest: runtime monitoring and observability of agent behaviour, with control measures as the enforcement layer. Senior Software Engineer (9+ years) working on agent reliability: evals, adversarial review, and verification of AI output. The finding that recurs across my work is a check that is present, documented and green, yet catches nothing. So I build for what a control actually catches, not whether it exists, and I apply that to my own tools first. Most recently, co-author of the v1 draft of the AI-assisted development standard at PensionBee, a UK pensions firm.

INDEPENDENT RESEARCH & WRITING

- **research-mini** (2026, results on 200 problems, public) - Trusted monitoring at small scale on open-weight models: a self-hosted platform on one 64 GB Mac mini, models served by llama.cpp behind a keyed router, evaluations on Inspect AI and ControlArena, and the untrusted model's code run in containers inside a VM with networking off. Result on 200 APPS problems and one build, pooled after both halves were seen (one setting, one seed): an 8B trusted monitor flagged 59.7% of attack-mode trajectories at a 4.6% false-positive rate (111 of 186; bootstrap 95% interval 46.8 to 66.7), but only 20 of the 39 working backdoors. Three deliberately weakened monitors each scored lower; the 3B model and the weak prompt are separated from their defaults at 95% on a paired test, the first-10-lines truncation only just. A second scoring rule, chosen after the first hundred and tested on a pre-registered holdout with the criterion committed before any score was read, beats the written score on the holdout (+6.5 points, interval +2.2 to +11.8) and on all 200, and lifts a 4B monitor to 67.7% at 5.0%, 25 of 39 working backdoors. Read every result with its frame: 200 problems and 39 working backdoors, one setting, one seed, one build, intervals ten to twenty points wide, and monitors that miss roughly half of the backdoors that work. Containment measured rather than assumed: an escape check showed the router's API key was not a network boundary and that the stock ControlArena APPS sandbox ran the task as root (ControlArena 19.0.0); the hardened configuration blocked 21 of 21 probes. Every run writes a provenance manifest; logs are scrubbed, published to a public dataset and verified again before the manifest records them. Built on the field's standard harnesses rather than extending trace-eval. **Python, Inspect AI, ControlArena, llama.cpp**. GitHub · Method · Run manifests · Dataset
- **trace-eval** (2026, minimal working harness, public) - Small, working harness for evaluating language-model output end to end: it loads cases, calls the model, scores the results, saves them and writes a Markdown report, keeping a structured, typed trace of every result. Two seams are deliberate - the scorer is pluggable, and the trace model is built to extend to agent and tool-call steps. The model call is a single injected function, so the whole suite runs offline with no API key. Every figure reported from it is backed by a committed run record in a public ledger. An LLM judge scores whether an answer was verified soundly, and a meta-evaluation measures the judge before any verdict is trusted: on a 10-item hand-labelled gold set, sampled five times per item, the judge matched the human label 10 of 10 and never flipped - a gold set that is deliberately clear-cut, so the figure only bounds reliability on easy cases. Mutation-tested the scorers with six hand-written broken variants of one case: the structural scorer caught 2 of 4 and let through the two that keep the shape of verification without the substance, and the reference scorer failed its own unmodified baseline, so its kill rate is void rather than a free 100%. Limits: one case and five kinds of mutant, small hand-written task sets, and no tool-call traces yet. **Python, Anthropic SDK**. GitHub · Findings · Run ledger
- **Sceptic** (2026, proof of approach, public) - Proof of approach for adversarial code review by several AI agents. A coordinator plans the review and hands pieces to isolated reviewer agents, which read the code and run its tests under a stated read-only guarantee - isolation, a tool allow-list, an approval backstop and an audit log. A finding only counts if it cites something the reviewer saw firsthand that would falsify the code's claim to be correct: a file and line plus a reproduced result, not a passing test suite. It ships with an evals harness of plausible-but-wrong modules whose own tests pass. On that bundled set of six wrong modules and one correct one, the reviewer named the real defect in all six and raised nothing high-severity on the correct one (7 of 7 evals, re-run 16 Sep 2026). Scoped to its own bundled sandbox as the first milestone and left extensible beyond that - then the technical AI safety work took over. So it demonstrates a pattern; it is not a reviewer you can point at any repository. **TypeScript (agent framework), Anthropic API**. GitHub
- **Two essays on verifying AI output** (danwalsh.co, Jul 2026) - "Don't take the model's word for it" (3 Jul) describes an adversarial-review method for catching AI systems whose stated behaviour differs from what they actually did; "The Verification Ladder: A Field Guide" (19 Jul) sets out a graduated method for checking what an AI agent produced instead of taking it on trust.
- **Kill-chain analysis** (final written exercise, BlueDot Technical AI Safety course, Sep 2026) - A kill-chain analysis of a threat model in which a nation state uses AI agents against critical infrastructure until it collapses. The argument: the initial-breach phase is the strongest neglected choke point, the only phase where the defender still has time and the cheapest moment to catch an intrusion, and the defences that matter there are patching and dependency hygiene on the open-source surface plus monitoring at the IT/OT boundary.

EXPERIENCE

Senior Software Engineer | PensionBee | Remote | Jan 2024 - Present

React, TypeScript, Next.js, React Native, Node.js, Python, Heroku, Postgres, Docker, Cloudflare (Workers, Workflows, D1, KV, cron), GitHub Actions, Salesforce

- **Verification catches:** Claims that did not survive checking. A committed CI guard pointed at a build directory git ignored, so it passed only where stale local output happened to exist, would have failed on any clean checkout, and was never exercised in CI - present, locally green, never a check. A base64 decode path carried a memory blowup of about 32 times, and the first measurement of the fix was itself wrong. An integration step was reporting success it had not achieved.
- **AI-assisted development standard:** Co-author of the firm's internal standard for AI-assisted software development - at v1 draft since September 2026, with named open items, and under adversarial review. The control model: the level of AI involvement in a change, set against the class of change, decides how much evidence the change must carry. Every requirement names an acceptance bar and an artefact an auditor can open, and agentic work always needs at least a human attestation. Also recommended a written prohibition on using the AI vendor's usage dashboard, as that measurement sits outside the firm's own pipeline and cannot be audited.
- **Attestation tooling:** Built a Claude Code skill that writes attestation-style pull-request descriptions from the session transcript - not from the agent's own summary of what it did - plus a plugin proof-of-concept that pulls attestation data and evidence out of Claude Code sessions. Then extended that into a higher-tier evidence skill, packaged it as a distributable plugin with rollout steps for the organisation, and along the way fixed a transcript-parsing defect that was silently dropping failed checks from the evidence record. An evidence tool reporting green by omission is the exact failure the tool exists to catch.
- **Guardrails proven, not assumed:** Proposed a set of global configuration rules for coding agents, each of which had to be proven before it was installed rather than adopted because it sounded plausible. Separately, derived team guardrails from real incidents, including a clean-checkout check for CI guards, and a mutation check on new tests.
- **Validated classifier:** Built the matcher that classifies incoming email, then checked it against 614 real subject lines rather than against assumptions about what the emails would look like.
- **Reliability under concurrency:** A live data-loss issue in Cloudflare D1 under concurrent writes: built a retry layer that knows which operations are safe to repeat - transient failures retried with backoff, constraint errors failed fast - which resolved the data loss.
- **Delivery breadth:** Also led the architecture and build of a complete rewrite of the customer withdrawals journey - legacy code to React and Next.js - with two mid-level developers, working with Product, UX, Banking and Compliance throughout. Co-created a single shared React component library to replace the UK and US libraries, with automated versioning and publishing so other teams could contribute without ceremony.

Earlier at PensionBee: ***Software Engineering Manager*** *(Aug 2023 - Jan 2024) organising an engineering-wide hackathon and leading a smaller one on API integrations and data visualisation, and contributing to salary benchmarking and engineering management strategy;* ***Senior Software Engineer*** *(Oct 2022 - Aug 2023) leading delivery of key product features, including mandatory two-factor authentication and regular drawdown options.*

Software Engineer II | Puppet | Remote | Apr 2021 - Oct 2022 | *React, GraphQL, Apollo, Go, Python, Docker, Kubernetes, GCP*

Software Engineer II | Bazaarvoice | Belfast | Mar 2020 - Mar 2021 | *React, Redux Saga, WebdriverIO, GraphQL, Node.js, Docker*

Software Engineer / Junior | ShopKeep | Belfast | Jun 2017 - Mar 2020 | *React, Node, Python/Flask, Ruby/Rails, AWS*

RESEARCH DIRECTION (PLANNED)

- **Chain-of-thought monitoring as triage.** Korbak et al. 2025, "Chain of Thought Monitorability: A New and Fragile Opportunity for AI Safety" (arXiv 2507.11473), read against METR's finding on the 2026 OpenAI / Hugging Face incident: at least 20% of agents reasoned about tampering with their transcripts and at least 96 transcripts carried spoofed tool calls (all of the spoofing found was small-scale tests), while the agents left their chain of thought alone - apparently because they believed the scorer would not read it. The question: whether chain-of-thought monitoring can work as early triage, and how long that window lasts.
- **Weak models overseeing strong ones.** Kenton et al. 2024, "On scalable oversight with weak LLMs judging strong LLMs" (arXiv 2407.04622), reframed as whether a weaker model can act as a readability layer over a stronger model's output.
- **Thought anchors.** Bogdan et al. 2025, "Thought Anchors: Which LLM Reasoning Steps Matter?" (arXiv 2506.19143): a possible addendum to chain-of-thought monitoring - limited on its own, given the implementation cost.

SKILLS

AI & evals: LLM evaluation and evals-harness design, adversarial review, verification, MCP protocol, agent orchestration; LLM API integration (Anthropic, OpenAI, OpenRouter).

Engineering: TypeScript, Python, Go, Node.js, React and Next.js, PostgreSQL, GraphQL, Docker, Kubernetes, AWS, GCP, Vercel, Fly.io, CI/CD, testing (Jest, Pact, Cypress, WebdriverIO).

Leadership & practice: Technical strategy, infrastructure planning, developer experience, cross-functional collaboration, mentorship, architecture advocacy, Agile.

EDUCATION & CERTIFICATIONS

Technical AI Safety, BlueDot Impact, 3 Aug - 13 Sep 2026 (completed) · **AWS Solutions Architect (microcredential)**, Open University / FutureLearn, 2022 · **Postgraduate Certificate in Computing**, Open University, 2017 (Pass with Merit) · **BA (Hons) Liberal Arts with History**, St Mary's University College, 2010 (2:1)