A MERIDIAN AUTONOMY METHOD NOTE

Decision Persistence: Operating a Large Language Model as *a Measurement Instrument.*

meridianautonomy.com

HOW WE ACTUALLY TREAT THE MODEL

We do not treat the model as an agent to be trusted, *but as a component inside an instrumented system.*

An autonomous agent is something you hand a goal and then trust to proceed. A component is something you bound, observe and hold to a fixed standard. What follows is the second thing, never the first, resting on five controls that work together.

Bounded roles	Persistent records	Fixed criteria	Independent	Human release
one task each	append-only	set in advance	can overrule	the only door

Take any one of these away and the system reverts to an agent you are simply trusting. Together they are what let you stand behind the output.

WHAT TEAMS PERSIST, AND WHAT THEY LOSE

Common engineering practice is to persist code and data. *But decisions often evaporate.*

Teams using large language models keep the code, in version control, with its history and its reviews, and they keep the data. The decisions are the part they let go, scattered across chat logs and half remembered, reconstructed days later from someone's recollection of what the model said on Tuesday.

In any serious engineering practice the decisions are the asset, and the code is only the current expression of the latest one. Lose the code and a competent team can rebuild it, but lose the reasoning, the ruling, the threshold you agreed and why you agreed it, and you have lost the part that was genuinely hard to produce.

INVERT WHAT IS DURABLE AND WHAT IS DERIVED

Once the decision becomes the durable layer, *the code and data are free to express it.*

CONVENTIONAL

Decisions
transient, scattered, lost

Code
persisted, version-controlled

Data
persisted

DECISION-FIRST

Decisions
durable, append-only record

Code
expresses the latest decision

This is no licence for careless code. Good engineering still matters. The point is simply that the reasoning is what cannot be regenerated, so the reasoning is what we keep.

STATELESSNESS AS THE SAFEGUARD

What most people distrust is that the model remembers nothing. *Under discipline, that is exactly what makes it safe.*

The usual objection is that a model is stateless. It remembers nothing from one session to the next, and it may contradict on Wednesday what it told you on Monday.

Turn that around. A worker that holds nothing in its own head, and must read every fact and prior ruling from the record before it acts, cannot quietly drift. It has no private place left to drift in.

BOUNDED ROLES, COMMUNICATING ONLY THROUGH THE RECORD

No window designs, builds, checks or approves itself. *Every role has a mandate and operates at a different altitude.*

Roles never paste long context between windows; they communicate only through the record. A reviewer independent of the author catches what self-review never will.

A PERSISTENT RECORD AS THE AUDIT TRAIL

Nothing is ever edited or deleted. *Information and decisions are only superseded.*

Every ruling, finding and verdict, with the reasoning behind it, is written to a shared append-only store and can be looked up later. A decision is only ever replaced by a new entry that names the one it supersedes. This is version-control thinking applied to judgement. The history is the audit trail: you can walk back through the chain that produced any output, including the decisions later overturned, and see what overturned them and on what evidence.

RECORDS BANKED

1,292

Decision and finding entries on the shared record.

CARRYING LINEAGE

935

Linked to a parent or the entry they supersede.

EDITED OR DELETED

0

Append-only by construction. The history is the trail.

THE DISCIPLINE THAT PROTECTS EVERY PRIOR PASS

A failed check is a diagnosis. *Failing a pre-registered test is never permission to move the bar.*

A criterion may be amended only if the world shows it measured the wrong thing, and even then the amendment may correct its validity only. It may never make the bar easier to clear.

The decisive test is direction-neutrality: would I make this exact change if the system were passing? If the answer is no, then it is not a correction but a loosening, and it does not happen.

Three guards hold it. The amended criterion must still be able to fail, the old one is kept on the record and never overwritten, and someone other than the author confirms that the change is not a quiet relaxation. Schedule pressure and product urgency are named explicitly as things that do not count as reasons.

WORK FLOWS ONE WAY; A HUMAN HOLDS THE DOOR

Models are allowed to draft, prove and argue. *Responsibility for production stays with a person.*

No model deploys on its own, and no chat ships to production by itself. That single separation is what keeps a stray action, or an over-confident assistant, from ever reaching the live system.

THE OVERHEAD, STATED PLAINLY

None of this is free. *The upfront cost of coordination buys reconstructability and speed.*

It carries real coordination overhead. Separate roles, a written record, independent confirmation, and a human release are all slower, change by change, than letting one capable assistant run end to end.

It is the wrong choice for a throwaway script or a quick exploration. It earns its keep only when the cost of being quietly wrong is high, which is exactly the regulated, high-stakes setting where these tools are otherwise hardest to trust. The overhead buys an output you can stand behind and reconstruct, rather than one you merely hope is right.

There is a second return we did not expect. Because every role works from the same durable record, the work parallelises rather than stalling while context is rebuilt. In our own practice the discipline that makes the work auditable has also made it faster, turning weeks of careful change into days, without trading away quality.

THE PATTERN, DISTILLED

Four things *worth remembering.*

— **Persist decisions, not just code.** Make the ruling and its reasoning the durable, append-only artefact, and let the code be the current expression of it.

— **Make statelessness work for you.** A worker that must verify everything from the record, and remembers nothing privately, is more auditable than one that relies on memory.

— **Never move the bar to pass.** A failed check is a reason to fix the work. Amend a criterion only if you would make the same change while passing, and only ever toward more validity.

— **Keep one human at the door.** Models can draft, prove and debate. A person remains responsible for release.

SEPARATION OF DUTIES, APPLIED TO OURSELVES

Seven seats, *each at a different altitude.*

SEAT	MANDATE — WHAT IT DOES	WHY THE SEAT EXISTS
Operator	Sets direction, holds release authority	A person, not a model, opens the door to production
Integrator	Seam-checks work against the criteria	Clears or stops; keeps the brief coherent
Architect	Owns the structure and canonical patterns	Decides how a thing is built; holds one release key
Builder	Implements against the spec	Produces the work; never fakes a pass
Instrument	Scores against the fixed criteria	Produces the evidence the result rests on
Validator	Independently confirms the score reproduces	The measure is checked by a seat that did not make it
Red team	Attacks the result before it stands	The last challenge; byte-level approval to release

No seat reviews its own work, and a failed check is a diagnosis, never licence to move the bar. The seats are bounded model roles, separate instances with separate mandates, so the separation costs coordination, not headcount. This is how Meridian operates; we offer it as our own discipline, not as a prescription for others.

A SINGLE RELEASE, THROUGH THE SEVEN SEATS

One change, *from first draft to production.*

A release candidate enters the record and moves through the seats in order. No seat reviews its own work. Each step writes an entry that names the one before it, so the whole path is reconstructable. What follows is a real chain, abstracted: the names of the artefact and its internals are removed, the structure is exactly as it ran.

STEP	SEAT	WHAT HAPPENS	EFFECT ON THE RECORD
01	**Builder**	Implements the change against the spec and produces the candidate	writes entry, status: for_review
02	**Instrument**	Scores the candidate against the fixed, pre-registered criteria	writes entry, carries the evidence
03	**Validator**	Independently reproduces the score; divergence would halt the release	writes entry, confirms or stops
04	**Red team**	Attacks the candidate at byte level before it is allowed to stand	writes entry, status: approved
05	**Integrator**	Seam-checks the whole chain against the criteria; clears or stops	writes entry, status: cleared
06	**Architect**	Confirms it fits the canonical structure; holds one release key	co-signs the release
07	**Operator**	The person turns the second key and moves it to production	writes entry, status: released

When the candidate failed a check earlier in this very chain, the Builder refused to substitute a passing stand-in. The failure was recorded as a diagnosis and the bar held. That entry is still in the record, superseded but never deleted.

WHAT ONE RECORD ENTRY CONTAINS

Every entry is a small, *addressable, permanent object.*

An entry is not free text in a log. It carries a fixed set of fields, and the parent link is what turns a pile of entries into a chain you can walk. A role does not read the whole record: it fetches by key, follows the parent lineage of the thing it is working on, filters by category, and searches the content for the rest. Retrieval, not memory, is how a stateless role finds the prior ruling that binds it.

FIELD	WHAT IT HOLDS
key	A stable, human-readable identifier for the entry
title	What the entry decides or finds, in one line
category	Which kind of work: build, review, methodology, coordination
status	Where it stands: for_review, cleared, approved, released, superseded
parent	The entry this one answers, links to or supersedes
content	The full reasoning, written to be read later
created	When it was written; entries are never edited after

A decision is never overwritten. When replaced, the new entry names the one it supersedes and both remain, so the history is itself the audit trail.

EU-RESIDENT, DISTRIBUTED, RECOVERABLE

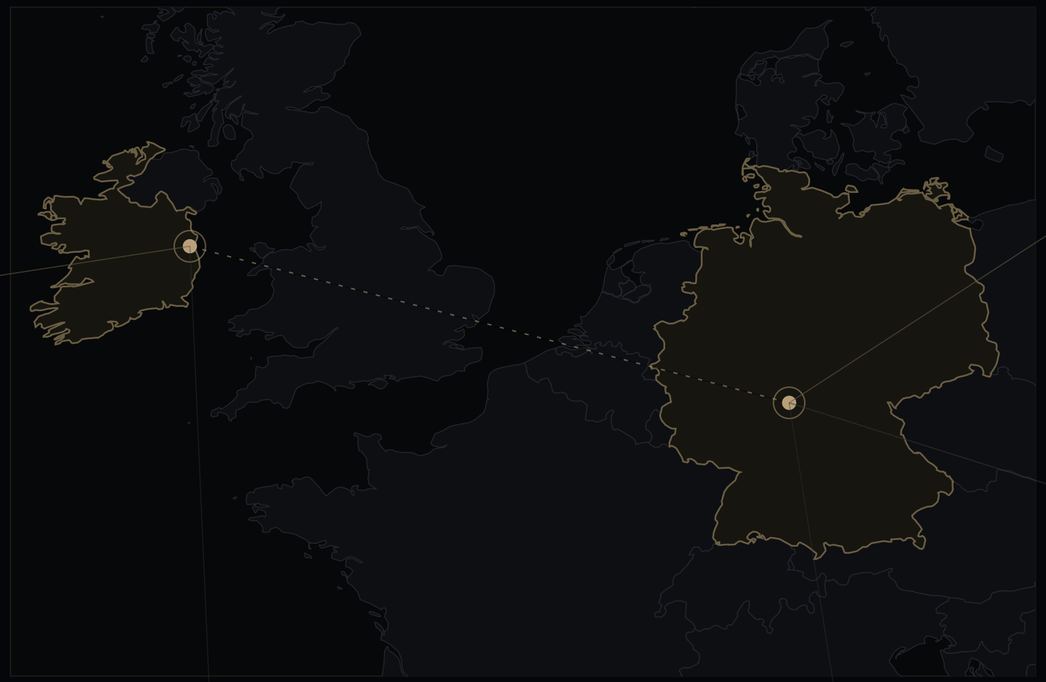
The infrastructure runs *inside the EU.*

IRELAND

Database — Production

Database — Clone

primary record, append-only



GERMANY · COMPUTE

Server 1 | Server 2 | Server 3 | Server 4

GERMANY · BACKUP

Backup 1 · Backup 2

independent, verified recovery

EDGE LAYER · GLOBAL

Zero-trust public surface · origin never exposed · EU data stays in the EU

No client data is retained, and EU data never leaves the EU.

HOW THE DISCIPLINE CORRESPONDS TO THE LAW

Each control answers *a named requirement.*

The method is not built to a framework, but it lines up with one. Each control is set against the article it speaks to under the EU AI Act regime for high-risk systems and under DORA. This is how we read the correspondence, not a claim of conformity or endorsement.

MERIDIAN CONTROL	EU AI ACT (HIGH-RISK)	DORA
Human release gate Only a person can release	Art. 14 — Human oversight	DORA Art. 5 — Governance, clear roles, ultimate responsibility
Anti-loosening gate A failed check never moves the bar	Art. 17 — Quality management system	DORA — ICT change management (RTS 2024/1774)
Append-only record Superseded, never deleted; with lineage	Art. 12 — Record-keeping; Art. 19 — Logs	DORA Art. 6 — ICT risk framework; documentation
Independent validation Confirmed by a seat that did not make it	Art. 15 — Accuracy and robustness	DORA — three-lines-of-defence control
Red-team challenge Attacked before it is allowed to stand	Art. 15 — Robustness and cybersecurity	DORA Art. 24–27 — Resilience testing
Bounded roles No seat checks its own work	Art. 9 — Risk management system	DORA Art. 5 — Segregation of ICT functions

This is our own analysis, offered to make the method legible to a compliance reader. It is not legal advice, and it does not assert certification under either regime.

ABOUT MERIDIAN AUTONOMY

The infrastructure to see *AI-governance exposure across finance.*

Financial institutions are deploying AI agents faster than they can govern them, and the duty to govern is now law under DORA and the EU AI Act. We measure that exposure from public evidence, and do not assert it.

WHAT WE PROVIDE

- Topology diagnostic
- Semantic search
- Benchmarking and research
- The MAR® algorithmic rating
- Continuous intelligence

WHO IT IS BUILT FOR

- Regulators
- Institutions
- Risk underwriters
- Vendors and advisors

WHY IT IS DIFFERENT

- Independent: the issuer does not pay
- External: no systems access, no client data
- Published methodology, proprietary IP

INSTITUTIONS	AI AGENTS MAPPED	GOVERNANCE EDGES	ON ONE VENDOR
543	95,876	626,390	66.9%

WILLIAM M. COLLINS · FOUNDER & CEO · BILL@MERIDIANAUTONOMY.COM · MERIDIANAUTONOMY.COM

DATED, CITABLE, EXTERNALLY AVAILABLE

The method rests on *published research.*

All papers authored by William M. Collins and available via SSRN.

01 ***The Stationary Sea, Part 2: The Long and Winding Road***
Substrate Identification Reproducibility, Two-Lane Canonicalisation, and the Multi-Institution Empirical Validation of the Hundreds-Not-Thousands Counter-Prior
SSRN: 10.2139/ssrn.6813081 · 2026

02 ***Annex 1a to The Stationary Sea, Part 1***
Canonicalisation Methodology, Variance Decomposition, and the Development Sequence
SSRN: 10.2139/ssrn.6813018 · 2026

03 ***The Stationary Sea, Part 1: Substrate Construction***
Measurement Instrument Validation for External Assessment of AI Governance in Regulated Financial Institutions
SSRN: 10.2139/ssrn.6675603 · 2026

04 ***Systemic Governance Risk in AI Agent Networks***
Cross-Institutional Contagion and Topological Vulnerability
SSRN: 10.2139/ssrn.6535599 · 2026

05 ***Credit Ratings Cannot Measure What They Cannot See***
Zero Correlation, Within-Band Dispersion, and the Structural Limits of Financial Risk Assessment
SSRN: 10.2139/ssrn.6524438 · 2026

06 ***The Coase Inversion***
Topological Governance of Autonomous AI Agents in Regulated Financial Services
SSRN: 10.2139/ssrn.6470098 · 2026