PINGUINISTA · SENIORITY ROUNDS

The Sovereign Seniority Manifesto

A manifesto for staying senior, to say the least — aiming not to avoid atrophy but to keep improving as humans, and getting more so, while agents write the code.

LEONEL SALVO · PINGUINISTA.COM · 2026

Agents changed the arithmetic of a working day. A project now moves further in an afternoon than it used to move in a month, and the person who is supposed to understand it is handed the result as a diff (what was before vs what changed). Read that diff the way review tools ask you to and you keep almost none of it. A few months later you are a stranger to your own project: you can prompt it, but you can no longer *reason* about it.

The enemy has a name — **atrophy** — and it is older than this technology. But avoiding decay is a low bar. The same flood of decisions that erodes you if you skim it is, handled right, the richest source of practice a developer has ever had. Any amateur becomes a senior by seeing more cases, with feedback, than they could ever have generated alone. Your agents are that ward. **The goal is not to hold your seniority steady. It is to grow it faster than you could by typing.**

That takes a loop, not a gate: the agent produces, you understand, you object, it answers. Feed, interact, repeat. This document is the stance behind that loop.

To weaponize this flood, we must transition from passive consumption to two core operational mechanics:

- **Epistemic Overdrive** The deliberate act of intercepting the massive volume of agent decisions and converting it into hyper-compressed experience. Instead of skimming the surface, the developer uses the agent to pump years of architectural edge-cases into their brain in a matter of weeks. It is turning information overload into a cognitive supercharger.
- **The Metamorphic Loop** A structural interaction where the goal of the feedback cycle is not just to fix the code, but to evolve the human mind. The code changes, but the developer morphs alongside it, maintaining conceptual control. It forces the system to render architecture not as a linear diff, but as a multi-dimensional topology of assets and variations that the human can mentally map.

We value

Understanding	OVER throughput
Rounds	OVER review
Prediction	OVER reading
Confession	OVER confidence
Adding	OVER hiding
Growth	OVER maintenance
Real work	OVER drills

That is, while there is value in the items on the right, we value the items on the left more. (The form is borrowed, gratefully, from the Agile Manifesto.)

Why growth, not maintenance

Expertise is built by deliberate practice: effortful work on cases at the edge of your ability, with immediate, informative feedback. Memory is built by retrieval, not re-reading — testing yourself beats studying the answer, even without being told whether you were right. And the conditions that make learning feel easy produce the weakest retention; a well-placed difficulty is what makes knowledge stick.

Rounds are those three findings applied to a diff. Every card is a case. The prediction gate is the retrieval — you commit to a hypothesis before you see the answer. The agent's confession and the raw change are the feedback. The card asks you to understand *what was decided and why*, not merely whether it is correct, because judgment is what a senior has and a junior with a linter does not. Do that on every batch and seniority compounds instead of draining, because you are now judging more real decisions per week than a career used to offer.

Principles

01 **Rounds, not review.** Review scans for defects. Rounds keep — and make — the one who knows. The question is never only "is this correct?"; it is "do you understand what was decided, and why?"

02 **Predict before you look.** A card can ask first: *what do you think this broke?* Your answer is completed, never graded. Being wrong in private is the whole point; that is the moment the correct answer gets written into you.

03 **The agent confesses.** Three signals on every card: *I wasn't sure* (what it chose and what it dropped), *you didn't ask for this*, and *this might break* (concrete sentences, not a score). They are credible because none of them flatter the author. `null` means "I never looked"; an empty list means "I looked and found nothing" — and you get to hold that against it.

04 **Augmentation may only add, never hide.** Cards are a layer over the material, never a substitute. No card is the only way to reach a change; drop to Raw and you have everything. Distrust the notes and you have lost nothing.

05 **Didactic by design; gamified, never gamed.** Decks, weight, light, glass, the card that shatters when you finally own it — attention follows form. But there are no points, no streaks, no leaderboard. The one metric is honest: minutes until you have gone over everything. "Not now" is not "done".

06 **Real work, not a gym.** Drills rebuild a skill you lost. Rounds are how you never lose it and keep building it: on the real changes of the real project, the same afternoon, while the decisions are still warm and you still care about the outcome.

07 **Every kind of change, every kind of interaction.** Code, schemas, configuration, images: something produced a lot of material, and you want to remain the person who understood it.

08 **You talk back.** A note on any card reaches the next agent you speak to, and the skill makes it answer. The loop is not "generate, approve". It is "generate, understand, object, regenerate". That is what keeps you *in* the loop rather than *around* it.

Others said it first, for other things

- **Aviation, 1997.** American Airlines' Warren Vanderburgh called them *children of the magenta line*: pilots so used to following the computer's course line that they lost the ability to fly the airplane by hand — and the accidents came when the automation dropped out. His remedy was not less automation; it was knowing when to step down a level, and practising the level you stepped down to.
- **Control rooms, 1983.** Lisanne Bainbridge's *Ironies of Automation*: automate the work and the human who remains supervises a process they no longer practise, so their skill decays precisely until the moment it is needed most.
- **Chess, 1998.** Garry Kasparov's *advanced chess*: a weak human with a machine and a better process beat a strong human with a machine and a worse one. The process is the product.
- **Medicine, always.** Teaching rounds: the attending asks the residents what they see before saying what it is. Prediction before the answer, on a real patient, every morning.
- **Software, 2009.** The Software Craftsmanship Manifesto asked for "not only working software, but also well-crafted software". We add a line: not only shipped code, but *understood* code.
- **Learning science.** Ericsson on deliberate practice (1993), Bjork on desirable difficulties (1994), Roediger and Karpicke on the testing effect (2006).
- **The evidence for code, 2026.** A randomized trial found that engineers who delegated code to an AI understood it measurably less, while those who kept asking *why* kept their

competence; researchers now call the gap *knowledge debt* and *cognitive debt*, and argue that deskilling by AI is structural, not a personal failing.

What we are not

Not a code-review tool: those find what is wrong before merge, and they are good at it. Not a tutor bolted onto an editor: the material is your own project, not a curriculum. Not a productivity tool: it makes you slower on the day, by design, and faster on every day after — because you will still know where things are.

The one-line version

> Your agents write the code. Seniority Rounds is how you stay — and grow — the one who

> understands it. *Stay senior on what your agents are producing.*

Sources

- Vanderburgh's *Children of the Magenta,* and what it taught: AOPA · Air Facts Journal
- Bainbridge, L. (1983). *Ironies of Automation.* Summary
- Kasparov, G. (2010). *The Chess Master and the Computer.* The New York Review of Books
- Ericsson, Krampe & Tesch-Römer (1993), *The role of deliberate practice in the acquisition of expert performance,* revisited in Royal Society Open Science (2019)
- Bjork, R. A. (1994), desirable difficulties: a teacher's summary
- Roediger, H. L. & Karpicke, J. D. (2006). *Test-Enhanced Learning.* Psychological Science
- *Manifesto for Agile Software Development* (2001): https://agilemanifesto.org · *Manifesto for Software Craftsmanship* (2009): https://manifesto.softwarecraftsmanship.org
- The 2026 randomized trial on AI assistance and comprehension, as reported by The Register
- *Agents That Teach: Towards Designing Incidental Learning Back into AI-Assisted Software Development* (2026): https://arxiv.org/pdf/2607.06101
- *AI deskilling is a structural problem*, AI & Society (2025): https://link.springer.com/article/10.1007/s00146-025-02686-z
- *De-skilling, Cognitive Offloading, and Misplaced Responsibilities* (CHI 2025): https://arxiv.org/pdf/2503.03924