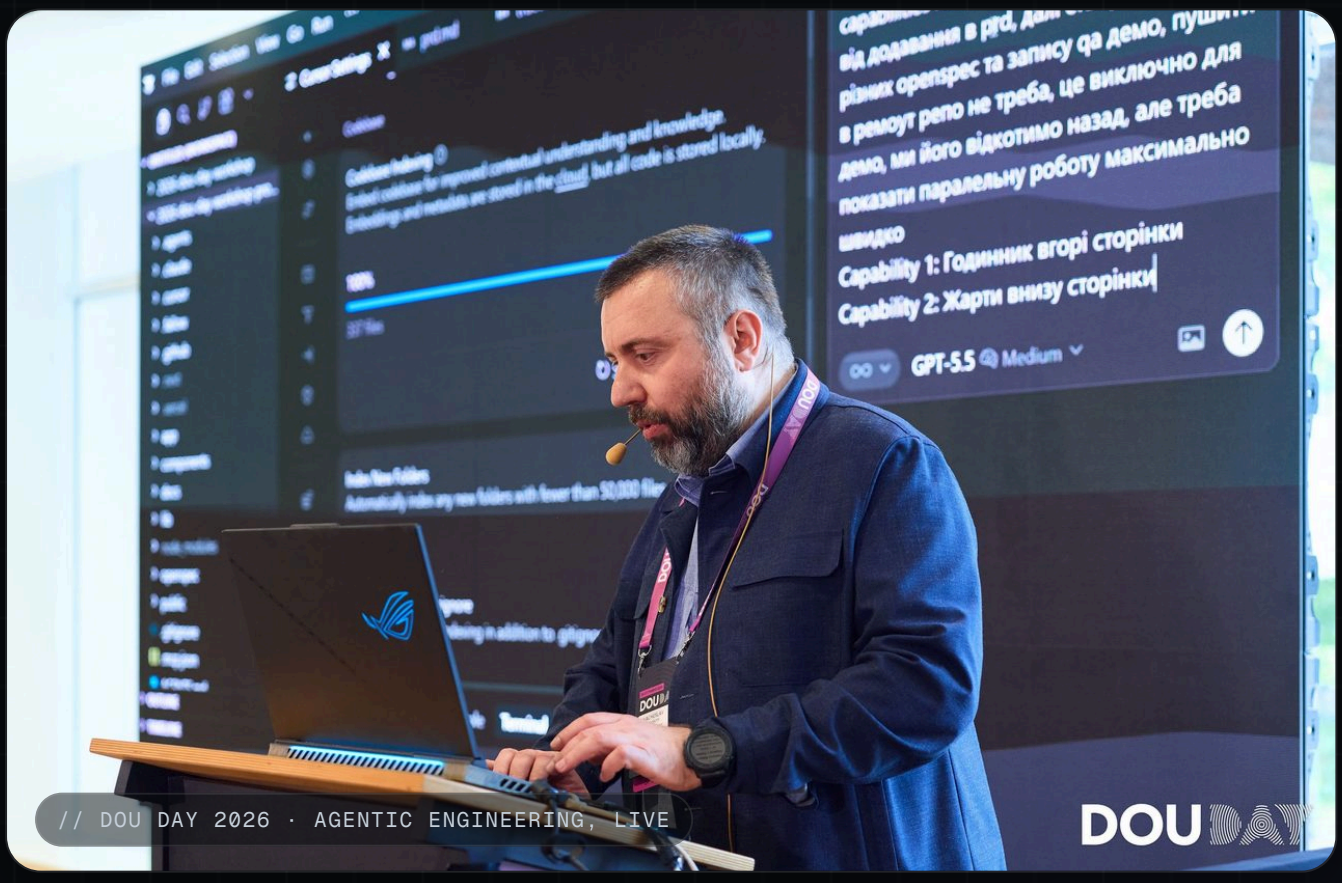
dev-ai.dev // PROPOSAL FOR ENGINEERING LEADERSHIP

Agentic Engineering Implementation

An end-to-end **Stage-Up transformation** for your engineering organization — from ad-hoc AI use to a verified, compounding software factory.



Vyacheslav Koldovskyy, Ph.D.

Founder & Agentic Engineering Architect · educator, speaker & researcher in Agentic Engineering

// 01 — WHAT WE OFFER

Make agentic engineering your team's default — safely, and measured.

dev-ai.dev turns ad-hoc AI use into a verified, repeatable way your teams build software with agents — proven on your own codebase, not slideware.

1 — Assess
We measure where each team stands today — 9 dimensions, stages 0–4 — and agree the targets.

→

2 — Set foundations
Shared rules, tools and guardrails so agents are safe to delegate to.

→

3 — Practice
We rebuild your everyday workflow with the teams: plan → build → test → review.

→

4 — Compound
The setup starts improving itself; every fix becomes a permanent rule.

// WAYS TO WORK WITH US

Assessment
Baseline & roadmap for one team or repo.
FREE · 1 WEEK

Transformation
The full Stage-Up across your teams.
1-3 QUARTERS

Focused track
Fix one gap: verification, harness, governance.
MODULAR

Embedded lead
A fractional Agentic Engineering Architect.
RETAINER

// WHAT YOU GET

- ✓ A clear baseline & roadmap
- ✓ A working AI setup your teams use daily
- ✓ Measured improvement on your own metrics

Start with a FREE one-week assessment →

For engineering leaders & their delivery teams

// 02 – THE SHIFT

From autocomplete to autonomy

In a few short years, AI in the editor went from guessing the next token to running multi-file work on its own. Your developers are already moving up this curve — the only choice is whether your org moves with intent, or by accident.

Autocomplete	Suggestions	Chat	Agents	Autonomous
next token	whole functions	describe → code	multi-file, runs tests	plans & ships PRs

> That's where dev-ai.dev comes in: we move your teams further up this curve on purpose — with the specs, verification and guardrails that make each step safe to take.

SYNTAX → INTENT

The constraint is no longer typing speed — it's the quality of the spec going in and the trust in the verification coming out. Your people shift from writing code to directing, reviewing and verifying it.

AI AMPLIFIES YOUR CULTURE

4× output · ~12% more value · 1.7× issues

Bolt agents onto an unchanged process and output soars while churn, defects and unreviewed merges climb with it. The capability multiplies whatever culture it lands in.

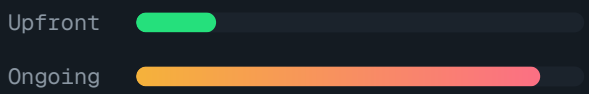
// 02 — THE ECONOMICS

Why ad-hoc “vibe coding” gets expensive

It looks free — a subscription and a prompt. The real cost is hidden in what comes after: trial-and-error token burn, rework on unstructured code, and security flaws found late. Agentic engineering inverts the curve — pay more upfront, far less to run.

“Vibe coding”

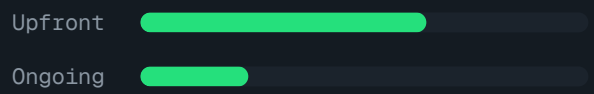
LOW UPFRONT · HIGH ONGOING



Cheap to start — then you pay forever: token burn from re-prompting, debugging unstructured AI code, and expensive late security fixes.

Agentic engineering

HIGH UPFRONT · LOW ONGOING



Invest once in specs, tests, context & guardrails — and the marginal cost of every new feature drops, with model routing trimming token spend.

> **dev-ai.dev gets your teams onto the agentic-engineering curve — we make the upfront investment with you, so the costs below are designed out, not paid forever.**

// HIDDEN COST

Token burn — low first-pass success means re-prompting loops that quietly run up the bill.

// HIDDEN COST

Maintenance tax — unstructured code costs days to reverse-engineer when it breaks months later.

// HIDDEN COST

Late security — a flaw caught in production costs far more than the same flaw caught at design time.

// 03 — OPERATING MODEL

The maturity ladder

You no longer mainly author code — you design the **factory** that produces it. Inside it, every agent is `Model + Harness`: a decent model with a great harness beats a great model with a bad one.

01
Intent & Specification
Is the *why* written down before the agent runs — goals, constraints, definition of done?

02
Context, Memory & Harness
Right context, durable memory, scaffolding — and no rot as the window fills?

03
Loops & Orchestration
Designing systems that prompt agents — or hand-prompting one turn at a time?

04
Verification & Review
Tiered by blast radius, adversarial, evidence-gated, human-owned at merge?

05
Tests & Deterministic Gates
Do real tests and immovable CI carry the weight human reading no longer can?

06
Safety, Security & Guardrails
Dangerous actions blocked by construction — sandboxes, hooks, scoped permissions?

07
Comprehension & Judgment
Does the team still understand what it ships — or is the thinking surrendered?

08
Reusable Capability & the Ratchet
Does every mistake become a permanent rule, skill, or hook — so it compounds?

09
Measurement, Cost & Compounding
Can you measure, trace and budget the factory — and prove it's getting better?

// 03 — THE FIVE STAGES

From improvised to autonomy

Not a generic “ad-hoc → disciplined” scale — the posture shift of the agentic era. We move lagging, load-bearing dimensions up one rung at a time.

S0
Improvised
Copy-paste from a chat window. No shared setup, no spec. Trusted because it “looks right.”

S1
Assisted
Autocomplete & chat daily. Real speed-ups, but every step hand-driven and held in one head.

S2
Delegated
Agent-sized tasks against a written spec, context curated, each result verified. Tests ride along.

S3
Orchestrated
Loops & harnesses. A maker writes, a different checker verifies, merges evidence-gated. Human on the loop.

S4
Autonomous Agents
The factory improves itself. Every failure becomes a rule. Evals guard prompts like tests guard code.

> A team is only as mature as its weakest load-bearing dimension. Stage 4 on tooling but Stage 1 on verification is a Stage 1 risk — the weakest gate is the one that fails.

// 04 — THE ENGAGEMENT

End-to-end Stage-Up transformation

Not a workshop series with a certificate at the end — a measurable stage transition across all nine dimensions, in four phases, each gated by explicit exit criteria.

Phase 0

Diagnose & Baseline

starting rung

Score every team and key individuals; set the measurement baseline; audit repositories and the SDLC for AI-readiness. Delivered as a maturity profile, a metric baseline and a sequenced roadmap — available as a standalone engagement.

Phase 1

Foundations

→ Stage 2 · Delegated

The governance & security baseline, the first shared harness (root AGENTS.md, a starter skill/command library, vetted MCP), and role-differentiated literacy. Exit: teams scope agent-sized tasks against a written spec, verify each result, and ship tests with the change.

Phase 2

Practice

→ Stage 3 · Orchestrated

Rebuild the real workflow on live artifacts — plan → implement → test → review; stand up the verification system; redesign the SDLC phases. Exit: maker/checker separation is standard, review is evidence-gated and human-owned at merge, parallel work scaled to review capacity.

Phase 3

Autonomous agents

→ Stage 4 · Autonomous Agents

Engineer self-improving loops with guardrails; install the ratchet; guard prompts & skills with eval suites; stand up the measurement flywheel and the internal community. Exit: fewer repeat mistakes, rising eval pass rates, stable quality as volume grows.

// 05 — WORK TRACKS

Twelve capability tracks

Sequenced across the phases — or run modularly. We work across all twelve, or the subset that targets your biggest gaps — agreed with you after the assessment.

Adoption assessment
Maturity scoring + the metric baseline that anchors the program.

All 9

Educational program
Role-differentiated workshops — theory plus practice on real work.

Dim 7 · all

Competence management
Matrices & interview/appraisal mapped to the maturity stages.

Dim 7, 8

Team implementation
Hands-on enablement on live artifacts and reusable skills.

Dim 1-3

SDLC redesign
Re-shape each phase around intent: spec, AI-aware review, gates.

Dim 1,4,5

Harness & platform
The factory as shared infrastructure: AGENTS.md, skills, MCP, hooks, evals.

Dim 2, 8

Verification system
Tiered review, heterogeneous AI reviewers, red/green TDD, evals.

Dim 4, 5

Governance & security
AI Use Policy, data classification, BYOT, MCP vetting, sandboxing.

Dim 6

Loop & orchestration
Loop design, maker/checker, fleet scaling, guarded overnight loops.

Dim 3, 8

Economics / FinOps
Token economy, CapEx→OpEx / TCO modeling, model routing, budgets.

Dim 9

Community & cases
Center of Excellence, champions, retrospectives, demo days.

Dim 7, 8

Managed sustainment
Quarterly re-assessment, harness upkeep, a fractional AI-eng lead.

All 9 · opt.

// 06 — MEASUREMENT

We prove it on your numbers — not someone else's

We baseline these on your codebase in week one, agree targets with you, and re-measure on a cadence. The classics are **DORA** metrics your leadership already trusts — we add the AI lens. The promise: **speed up without quality regressing.**

Speed & flow	faster delivery
Lead time for change (commit → prod)	↓ lower
Review / PR cycle time	↓ lower
Deployment frequency	↑ higher

Quality & reliability	GUARDRAIL
Change-failure rate	↓ lower
Defect escape rate (bugs in prod)	↓ lower
Mean time to restore (MTTR)	↓ lower
Code churn / rework	↓ lower

AI leverage & verification	the new workflow
AI-assisted change share	↑ higher
First-pass acceptance (no major rework)	↑ higher
Median PR size	↓ smaller
Zero-review merges on critical paths	→ 0

Economics & compounding	cheaper, durable
Cost per delivered feature	↓ lower
Time saved / engineer / week	↑ higher
Reusable skills & reuse rate	↑ higher
Repeat-mistake rate	↓ lower

> Quality sits next to speed on purpose: the gain is only real if change-fail rate and defects hold or improve as you ship faster.

// 07 — GOVERNANCE, SAFETY & SECURITY

Adoption is gated by trust

A governance foundation that keeps AI usable in daily work while protecting client confidentiality, IP, personal data and system security — modeled on a policy already deployed across an enterprise delivery org.

- **A tool policy built for you.** We help define which AI tools are approved — and under what conditions — from your security, client and compliance requirements: bring-your-own-tool where it fits, locked down where it doesn't, with any provider training on your data switched off.
- **Data classification as the core mechanic.** Concrete categories for what may and may not enter a tool; assume anything entered may leave unless authorized; data minimization — summarize, redact, isolate.
- **Per-engagement classification.** Every engagement labeled AI Allowed, AI Allowed with Restrictions, or AI Prohibited. Stricter client rules always win.
- **Technical guardrails, not just policy.** Secrets kept out of context; scoped tool/MCP permissions; provenance-vetted, version-pinned MCP; sandboxing & injection defenses; system-level blocks on destructive actions.
- **Accountability stays human.** AI output clears the same review, testing & security scanning as any code, with extra control on auth, crypto, IaC, data-access & compliance paths; authorship attributed, usage auditable.

// 08 — PROOF

Selected case studies

Anonymized from real engagements; client and product names replaced with code names, tools described generically.

Project Atlas

2.5× cost ↓

Skeptical team, cost pressure on AI work. Validated code-quality standards turned skepticism into adoption and a 2.5× development-cost reduction.

Project Beacon

~$200/mo

Large regulated US platform, 60k+ indexed files. Full-stack bugs fixed in under a minute; a month-long custom report now delivered in hours.

Project Orbit

93% AI

Greenfield product built AI-friendly. ~93% of code AI-generated team-wide; ~1,400 integration tests; the plan/implement/test/review workflow demoed live.

Harbor & Summit

BA enablement

Beyond developers: business analysts now generate requirements grounded in the actual codebase — implementation-aware specs from the start.

Project Cobalt • solo greenfield

~30 hrs → MVP

A full MVP — 36 requirements, ~395 files — built in ~28–32 engineer-hours via a spec-driven, fully AI-assisted workflow with a complete QA proof pack.

> As output scales, the constraint moves from writing code to reviewing it and resolving external dependencies — which is exactly what this engagement builds.

// 09 — ABOUT & HOW WE START

Built and delivered by a practitioner

Vyacheslav Koldovskyy, Ph.D. — Founder & Agentic Engineering Architect at dev-ai.dev; sole author and driver of a company-wide agentic-engineering program: 100+ hands-on workshops, competency frameworks, an AI Use Policy, maturity checklists, and hands-on enablement of 10+ delivery teams.

// workshops & corporate training

SoftServe · Raiffeisen Bank · Ukrainian Delivery Center · InventorSoft · Intelliarts · Langate · MR Lab · KeenEthics

// speaking & academia

DOU Day 2025 & 2026 · fwdays AI Summit 2026 · IT Step University (Assoc. Prof., Lead of the Agentic AI Research Center) · American University (Agentic Engineering course author).

// HOW WE START

- **1 · Diagnostic & Baseline** — free, ~1 week. Maturity profile, metric baseline, sequenced roadmap. Useful on its own.
- **2 · Agree the Stage-Up target** — teams, load-bearing dimensions, target stage, pace, and the governance guardrails.
- **3 · Run the phased transformation** — a fixed cadence of sessions, embedded team time, and a regular metrics readout.

We need: a sponsor in engineering leadership, access to representative repos & teams, and sign-off on tool-use guardrails.

Start on day one.

› dev-ai.dev

Begin with the diagnostic →

// sources — external 2026 figures from public research on agentic engineering & the AI-driven SDLC (Osmani; the "New SDLC" whitepaper; GitClear, Faros AI, CodeRabbit, METR). Program results from an anonymized enterprise enablement initiative. Detailed sources on request.