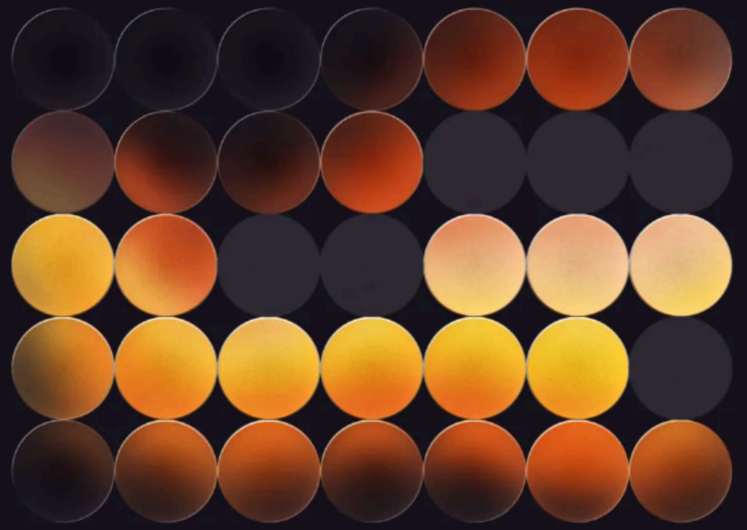
Turbo ML Conf · July 18, 2026

State of AI4SDLC: AI Shifts Delivery Bottlenecks

Coding accelerates first. Delivery follows only when the whole flow changes.



TURBO ML CONF

The bottleneck moves

Key thesis Flow

A local coding speedup creates a queue downstream

Loading diagram...



Part 1

01

Coding speeds up first

AI removes a local constraint and exposes the next stage

AI is no longer an IDE experiment

Context Shift

Manage changed development physics

Mass adoption

default tool

AI writes code, tests, docs, explanations.

Uneven effect

task-dependent

Coding speeds up before delivery.

New question

next constraint

Where will the next constraint land?

How we know what is happening

Research Data

Industry reports and AI4SDLC point the same way

Industry research

- DORA and AI-assisted development reports.
- Cases: Google, Microsoft, GitHub, Meta, Uber.

AI4SDLC research

- 50+ studies from 2023–2025.
- Survey of engineers and leads.

Adoption is high, trust is lower

Signals repeat across studies

Everyday tool

AI is part of the workday.

Uneven productivity

The effect depends on tasks and process maturity.

Delivery lags

Reviews, tests and evals are mandatory.

Coding is faster. Delivery isn't

Key thesis **Bottleneck**

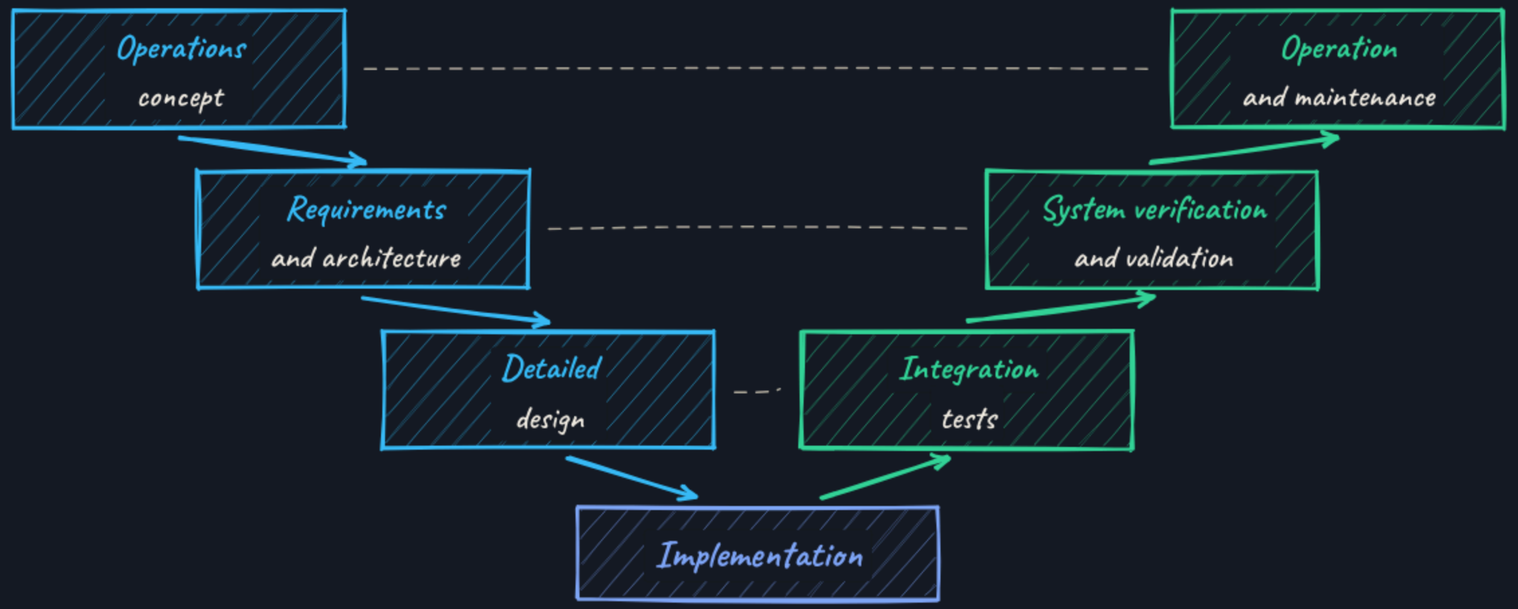
The queue moves to review, tests, and intent

- More WIP before review
- Tests absorb more changes
- Weak intent scales the error
- Verification becomes the constraint

Engineers steer the whole process

Loading diagram...

The V-model links requirements and verification





Part 2

02

The constraint becomes systemic

The platform expands throughput without losing control

Scale dictates the requirements

Context Scale

10,000+ engineers need a platform

Scale

10,000+

Thousands of services, repositories, and stacks.

Security

fintech

Data, secrets, and PII require a control perimeter.

Economics

unit economics

Token and infra costs are company-level.

North Star: three axes

Metrics North Star

Each axis needs baseline, guardrails, and owner

Speed

speed

Lead time and cycle time to production.

Quality

quality

Defects, incidents, rollbacks, security findings.

Economics

economics

Token, infra, and human-time costs.

IDP lives; GUI fades

Agent-first IDP GUI → agents

Control stays platform-side

To agent

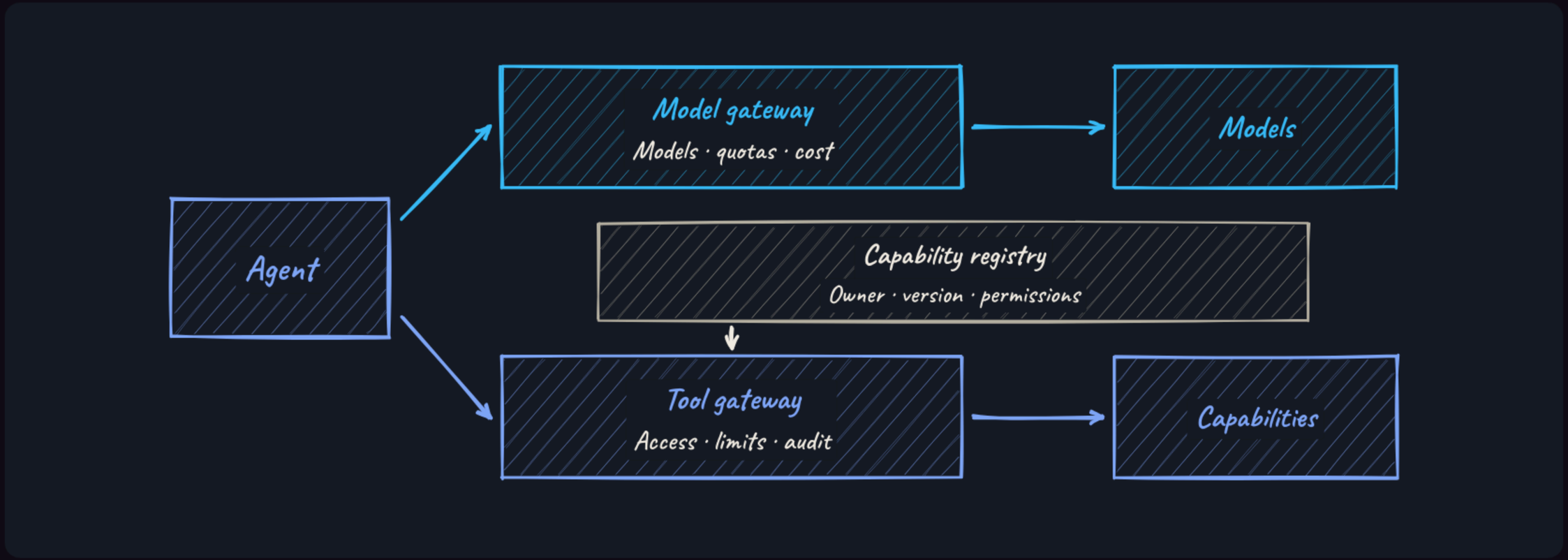
- User intent and next action.
- IDE, wiki, CI/CD, observability.
- Natural language for capabilities.

Platform

- Domain logic, policies, state changes.
- Limits, quotas, cost, and blast radius.
- GUI: settings, investigations, approvals.

Article: ↗ tellmeabout.tech

Gateways and registry serve different needs



From API to capability map

Platform product | Capability map

Design around workflows, not endpoints

Intent map

why

Deploy, observe, secure, govern, approve, remediate.

Machine contracts

how

Tool schema, errors, side effects, cost, policy.

Registry

who owns

Owner, version, scopes, SLA/SLO, evals.

Trust levels: read -> act

Trust model `Read → act`

Autonomy is granted per capability

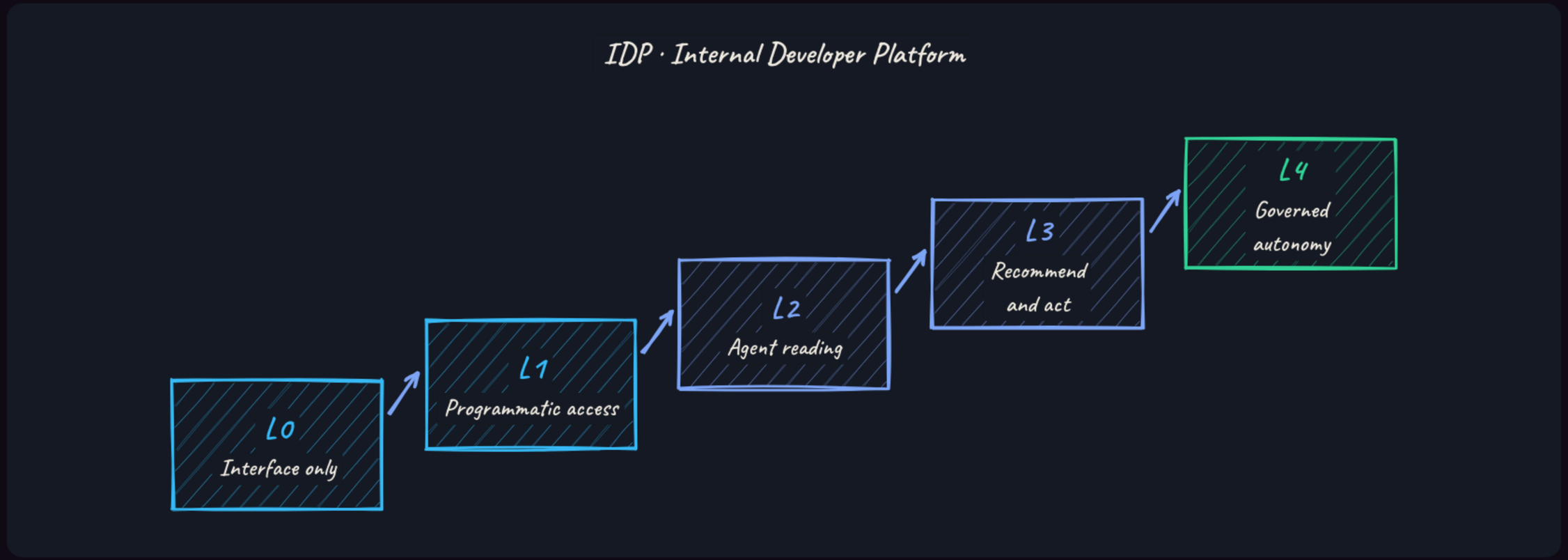
Growing autonomy

- Read: identity, quotas, telemetry.
- Recommend: plan, diff, risk, approval.

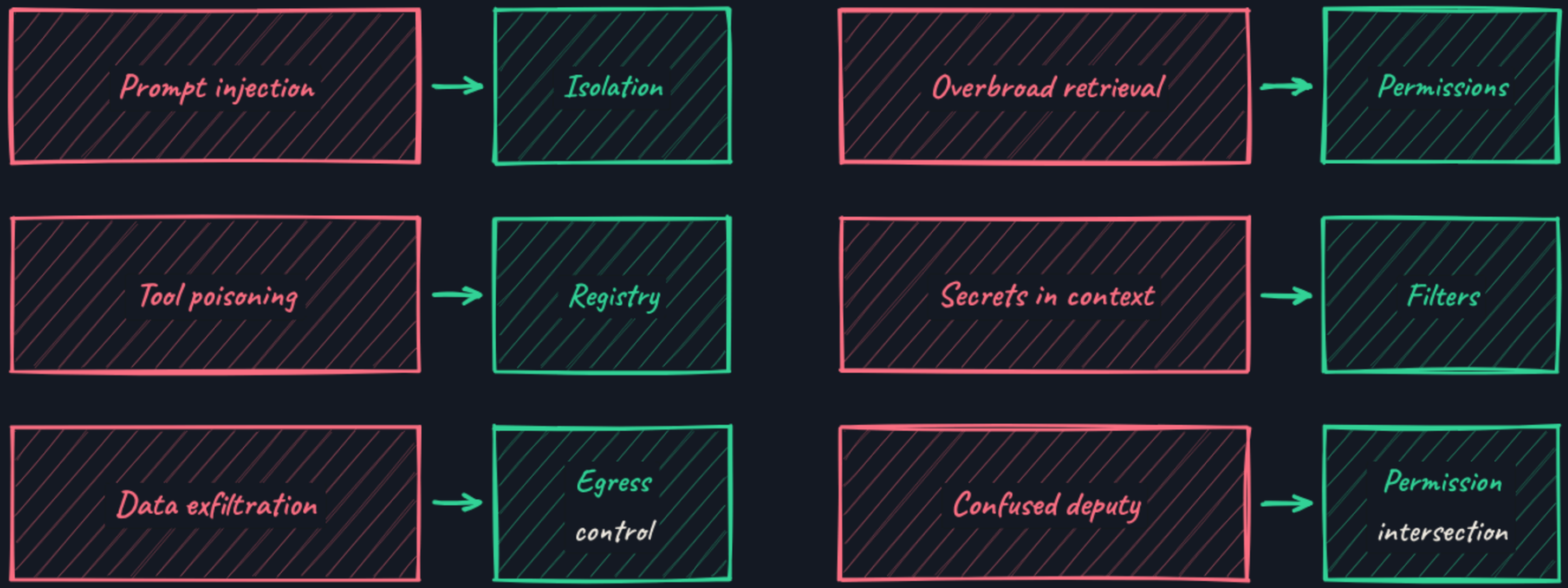
Where MCP is not enough

- State changes, cost, or risk.
- Hidden domain semantics: SLO, IAM, data.

Autonomy grows with control



Every threat needs protective measures





Part 3

03

Agents absorb the queues

Domain agents remove waiting across the SDLC

Model gateway: LLM access point

Platform Model gateway

Security and economics

- One gateway, no provider sprawl.
- Security: secrets, PII, data policy, audit.
- Quotas/costs by team/scenario.
- Models onboard centrally; clients unchanged.

Tool gateway: MCP Hub/CLI

Platform MCP

Controlled internal tools

- CLI, APIs, runbooks, service ops.
- MCP Hub: owner, scopes, schema.
- Controls: allowlist, on-behalf-of, audit, gates.
- One interface for IDE/agents.

Platform Nessy

Nessy: company-context copilot

Internal assistant

- RAG over codebase, docs, libraries.
- Knows conventions, ADRs, golden paths.
- Integrated into IDE and internal tools.
- Answers use current systems.

Spirit:
agent mode on platform

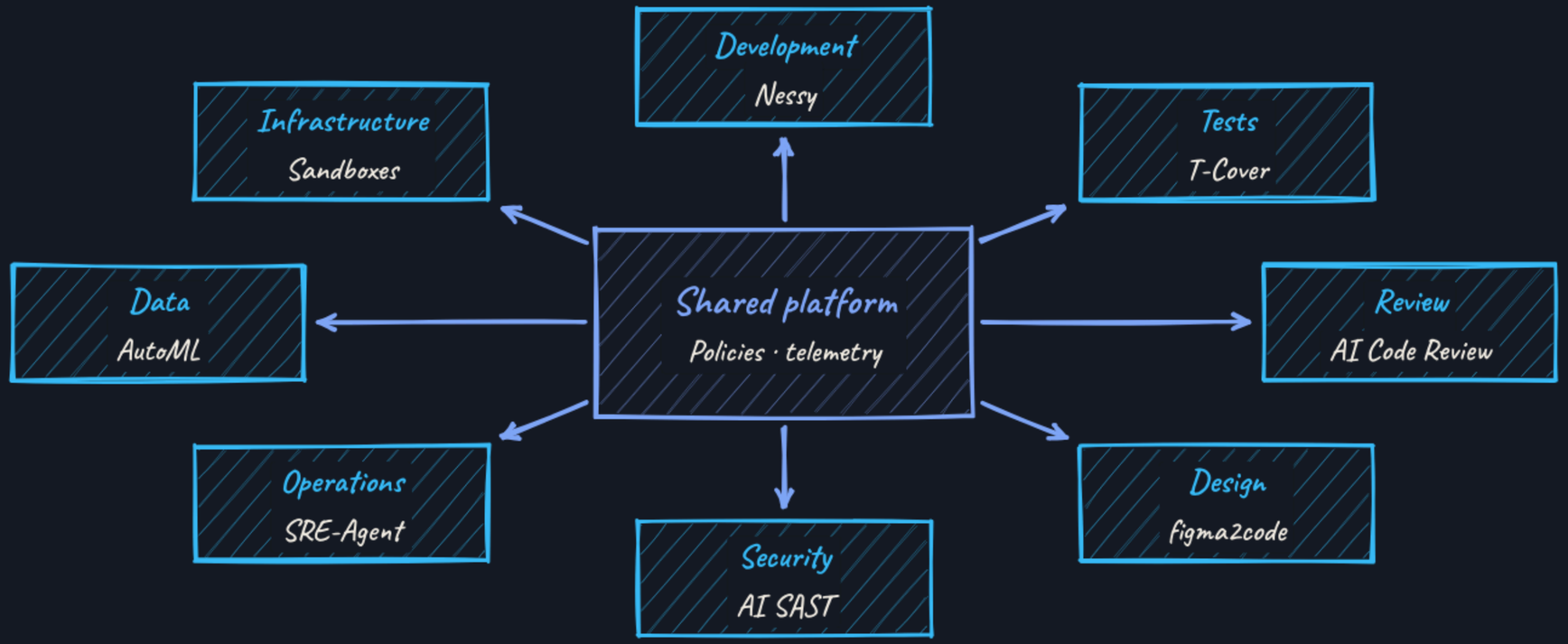
Platform Spirit

PaaS: VCS, CI/CD, agents

- VCS/CI/CD: shared policies, paths.
- Agent mode: ticket -> PR.
- Sage: AI-assisted troubleshooting.
- Agents share policy perimeter.

Domain agents share a platform

Our contour





Part 4

04

Measure the whole flow

Wait time and rework matter more than AI-code share

AI-code share is a trap

Anti-metric AI-LOC

AI-LOC cannot reveal the next queue

Tempting

- Easy to collect and present
- Grows with adoption

Harmful

- Rewards generated lines
- Hides quality, risk, and cost

Flow metrics find the queue

Wait time, rework, blocked time, safe-change time

Metrics DORA × SPACE × DevEx

Process map

- DORA: delivery and recovery
- SPACE/DevEx: flow and feedback
- Adoption is input, not proof

Signals

- review and pipeline wait
- rework and blocked time
- safe-change time and defects

Evals/telemetry instead of MAU

Metrics Telemetry

Find blocked work, not clicks

- Trace: intent -> tools -> approvals.
- Span: user, agent, tool, model, cost.
- KPIs: intent success, safe-change time.
- Evals regress agent-facing interfaces.

Reality check Time split

Developer time actually goes here

Code is only one part of the flow

- Coding: 30–35%.
- Meetings and syncs: 25–30%.
- Wiki, docs, context search: 12%.
- Code review and discussions: 10–15%.



Part 5

05

Engineers hold the control loop

Less typing, more intent, orchestration, and validation

Sociotechnics Key observation

Same platform, different effect

AI adoption is sociotechnical

- Same tools, different team results.
- The effect appears where teams change work.
- Requirements and cross-team work changed.
- Platform is necessary, not sufficient.

What made teams succeed

Cases Comparison

They rebuilt the work

No effect

- Enabled assistant, waited.
- Tasks still arrive via chat.
- Review/testing unchanged.

With effect

- Requirements became agent contracts.
- Review adapted to AI changes.
- Context/checks live in pipeline.

People Resistance

Resistance is part of adoption

Automation touches real workflows

- Fear of replacement is real.
- Education, transparent goals, quick wins.
- Team champions beat top-down orders.
- Pilot -> feedback -> scaling.

Engineer day: before / after

Practice Before / after

Less typing, more intent/context/checks

Before

- Writes code by hand.
- Context from chats/wiki/calls.
- Review reads diffs.

After

- Formulates intent/context/criteria.
- Orchestrates agents and parallel tasks.
- Validates results, more design.

Agent mode depends on the task



Engineer’s role gets wider

Role Leverage

Value shifts to steering change systems

Intent

Goal, context, constraints, DoD.

Orchestration

Split, delegate, assemble result.

Validation

Verify behavior, risks and intent.

Skills become the leverage

What helps agents amplify you

- Context engineering: manage agent context.
- Spec-driven development: verifiable contracts.
- Evals: make agent quality measurable.
- Platform literacy: know agent boundaries.

What is not happening

Myths | Reality check

Fears vs practice

- Engineers stay; judgment matters more.
- Juniors still matter: the learning path changes.
- Code review checks contracts and proof.
- Accountability stays with the human.

Five AI4SDLC shifts

Accelerate the flow, not one stage

- Coding is no longer the main constraint
- Queues move to intent and verification
- Platforms expand safe autonomy
- Evals and flow metrics prove impact
- Engineers orchestrate and validate change

- Accelerate the flow

AI4SDLC

Research and materials

Research, talk materials, and the ongoing discussion are in the AI4SDLC channel.

Alexander Polomodov

Technical Director & Fellow,
T-Technologies



@AI4SDLC