AI Dev Conf · May 21, 2026

State of AI4SDLC: AI assistants to agents



Why the development bottleneck moves from coding to specs, evals and the engineering operating model

Talk plan

Navigation Roadmap

- Research: adoption high, trust lower.
- SE 2.0 shifts bottlenecks to delivery.
- Task management becomes a work graph.
- SDD and evals make agents governable.

Part 1

01

What AI4SDLC Research really shows

Adoption has already happened; sustained impact comes through verification, process and platform

AI is no longer an IDE experiment

Context Shift

Manage the new development physics

Mass usage

default tool

AI writes code, tests, docs, explanations.

Uneven impact

task-dependent

Coding speeds up before delivery.

Management question

verify-first

Manage adoption, throughput, quality/risk, economics.

Why one productivity metric is not enough

Research Method

Read AI4SDLC as signals, not one magic speedup

What we measured

- Public research from 2023-2025.
- Survey of engineers and leads.

What to remember

- Self-report is not causality.
- Impact depends on maturity.

Adoption is high, trust is lower

Engineering systems must close this gap

Everyday tooling

Part of the workday.

Quality needs proof

Review, tests and evals.

ROI through process

IDE pilots need an operating model.

Coding faster, delivery not

Main takeaway Bottleneck

Existing bottlenecks grow

- Review/tests/releases get more changes.
- Weak framing generates wrong results faster.
- Without metrics, teams debate impressions.
- Next level: managed work loop.

Part 2

02

SE 1.0 -> SE 2.0

From role-based SDLC to agent-based SDLC: not a tool change, but a delivery loop change

Engineers steer the whole process

Two modes

Loading diagram...

The engineer’s role gets wider

Shift Leverage

Manage the system that produces changes

Framing

Define goal, context, constraints and DoD.

Orchestration

Split work, delegate, integrate results.

Validation

Check behavior, regressions, security and intent.

Team works by new rules

Practice Transition

Not just tools

- Tasks target humans and agents.
- Context/checks become dev interface.
- CI, tests, logs become evidence.
- Parallel agents need rules.

Part 3

03

AI-native operating model

When artifacts are created faster, manage the work graph, platform and metrics

Ticket becomes work signal

Work graph knows context earlier

Task management Work graph

Classic tracker

- Task and status written manually.
- Context spread across chats, PRs, docs.

AI-native work graph

- Intake and status partly computed.
- Intent, spec, PRs and tests connected.

AI-native rollout is platform work

Scale requires shared capabilities

Operating model Platform

Instructions and context

Repo rules, ADRs, conventions, safe commands.

Verification and observability

Evals, CI, telemetry, cost, latency.

Guardrails and economics

Permissions, secrets, limits, approvals.

Manage chain, not LOC

Metrics Chain

LOC is weak signal

- Adoption: who uses AI, where.
- Throughput: lead time, review, deploys.
- Quality/risk: defects, incidents, eval score.
- Economics: tokens, infra, waiting, rework.

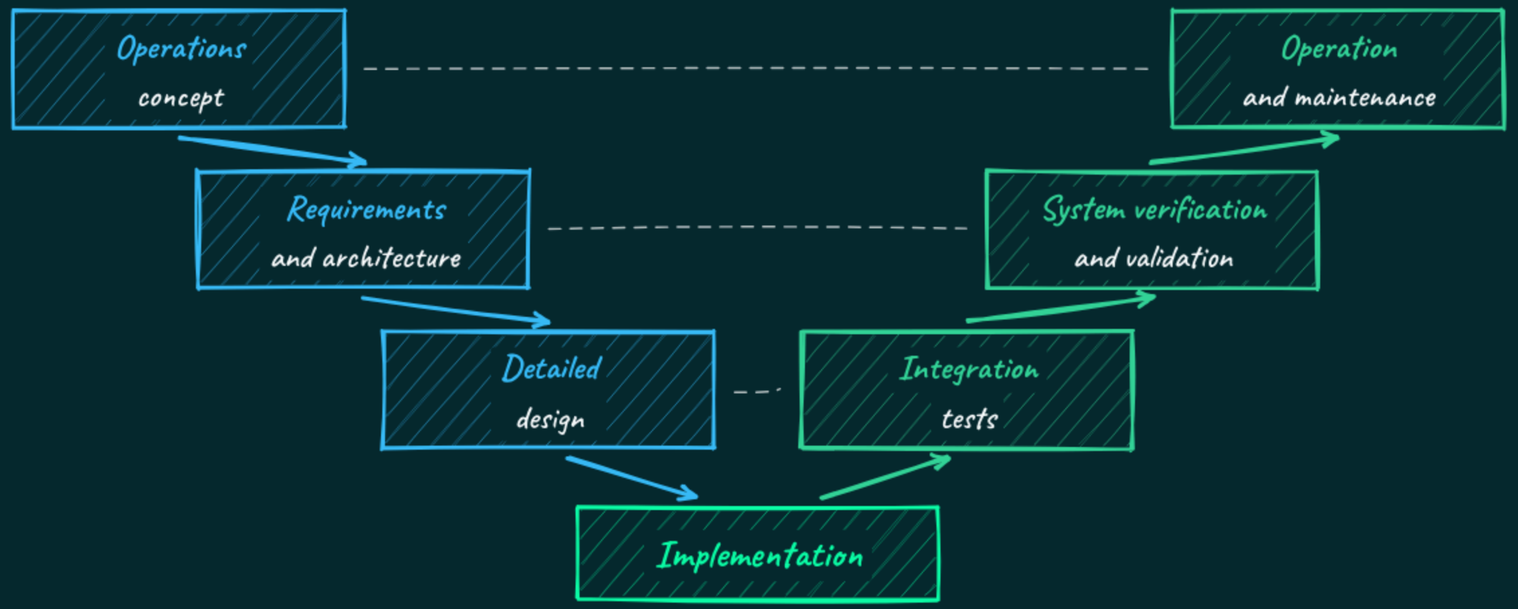
Part 4

04

Spec-driven development

AI brought specs back, but now they are a working interface for agents, not bureaucracy

The V-model links requirements and verification



SDD resembles V-Model, for a new reason

Why it returned SDD

Old specs traced work; modern SDD delegates

Old spec-driven

- Requirements -> design -> code -> verification.
- Strength: expectation traceability.

Modern SDD

- Spec becomes executable context.
- Agents need intent, boundaries, checks.

Interface Human → Agent

Specification: contract with the agent

Agents need managed context

Change

Goal, impact, flows, expected behavior.

Do not break

Compatibility, data, security, APIs.

Proof of done

Criteria, checks, tests, logs, rollback.

SDD: intent → proof

Chain Flow

Docs matter through checks

- Intent: why and what outcome.
- Criteria: success and regression boundaries.
- Plan: verifiable work steps.
- Verification: tests and review confirm intent.

Four practical SDD variants

Make the spec a portable contract

Tools | Patterns

GitHub Spec Kit

Spec -> plan -> tasks -> implement.

Kiro + OpenSpec

Structured and open spec formats.

Lightweight SDD

AGENTS.md, tests, PRD/RFC, checks.

Bad specs speed wrong code

Risk Reality check

SDD makes weak framing visible

- Unclear intent amplifies ambiguity.
- Empty criteria make review taste-based.
- No tests/evals means no proof.
- Large tasks need verifiable chunks.

Part 5

05

Evals for SDLC agents

Evaluate not response elegance, but whether an agent reaches a goal in a replayable environment

Basic template for one eval case

A small replayable development episode

Eval case Template

Frozen state

Fixed repo, data, issue, environment.

Agent contract

Allowed changes, commands, boundaries.

Hidden executable judge

Judge runs checks; cases stay fresh.

Evals must cover real work types

Scenarios SDLC

One coding benchmark is not enough

Product / Engineering

- Requirements: brief -> stories.
- Feature/bug: spec + regression oracle.

Quality / Operations

- Code review: merge-blocking issues.
- Tests/incidents: regression, units, incident bundle.

Scorecard Autonomy

Agent autonomy cannot be one score

Score result, trajectory and cost

Outcome

Task solved; criteria pass.

Trajectory

Plan, changes, rollbacks, tool use.

Cost & safety

Repeatability, latency, constraints.

Standard: spec, evals, CI, review

Standard Spec + evals

Minimum agentic loop

- Artifact: tasks, PRs, bugs, incidents.
- Frozen state: start and conditions.
- Hidden judge: executable checks.
- Scorecard: outcome, safety, spec fit.

AI-native development is not managed by prompts

The practical takeaway

- Bottleneck moves to intent and verification.
- Specs become human-agent contracts.
- Evals make agent quality measurable.
- Manage work graph, quality/risk, economics together.

- ↗ aidevconf.org · ↗ ai4sdlc-research.space · ↗ tellmeabout.tech

Links and materials

Sources References

Sources and practices

- AI Dev Conf and AI4SDLC Research.
- Spec-driven development: Telegram post.
- GitHub Spec Kit and AWS Kiro specs.
- OpenAI Codex, harness engineering, evals.