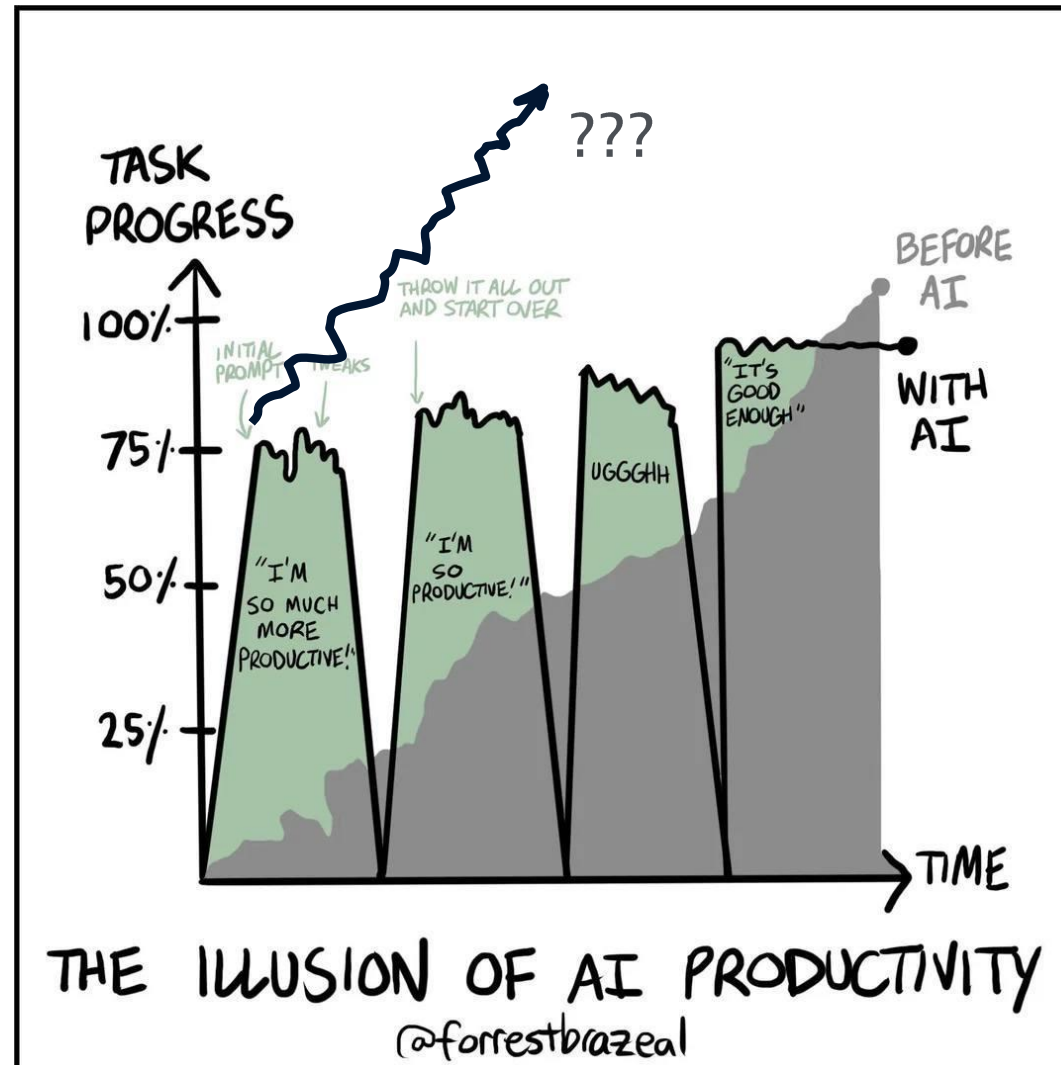




Specification-driven agentic SW development tool

2026-10-6, Mika Torhola | Atostek Oy

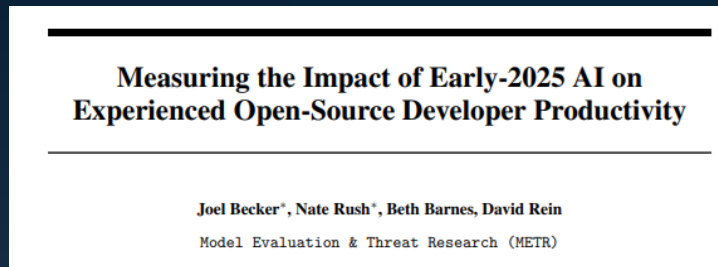
First some background context



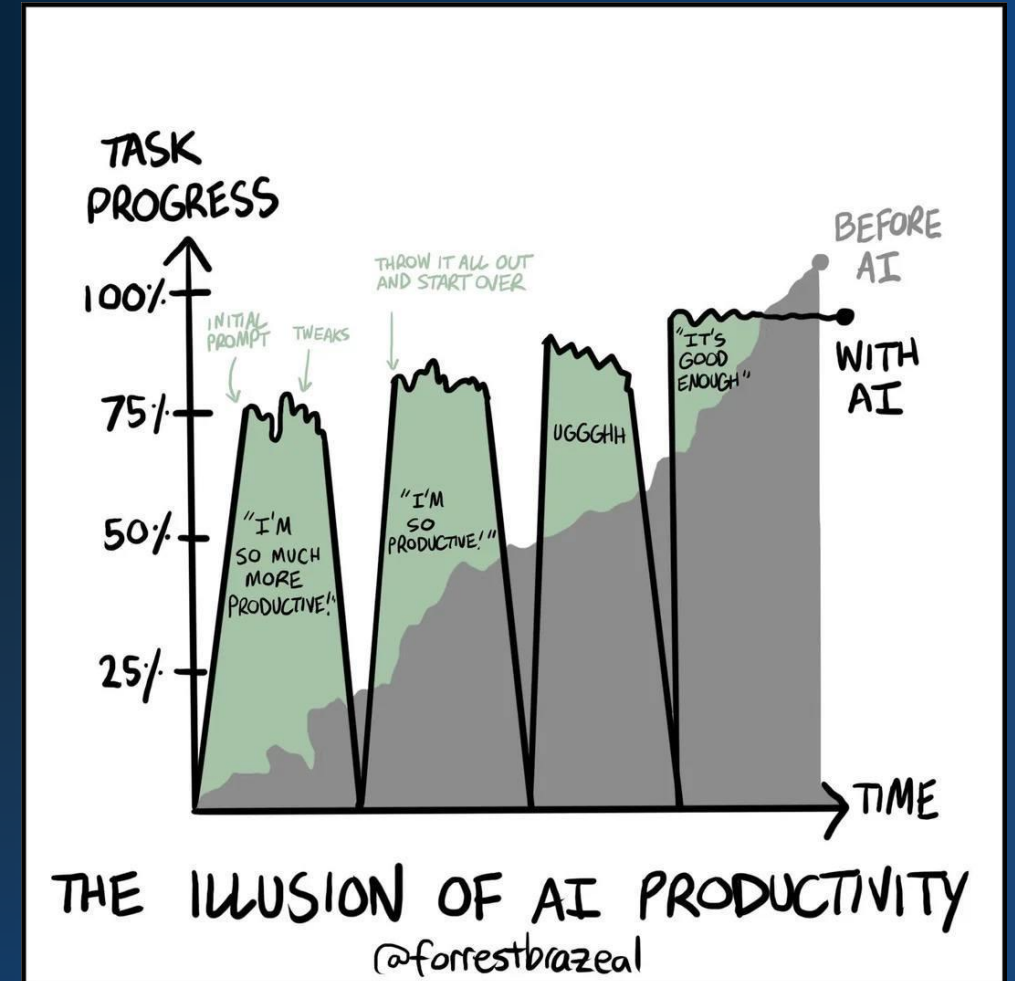
What's happening here?

**"The AI Coding Productivity Illusion:
Why Developers Feel Faster But Deliver
Slower" (7/2025)**

METR: <https://arxiv.org/pdf/2507.09089>



- Hidden quality costs that accumulate
- Hallucination problem compounds maintenance burden
- Speed without stability creates accelerated chaos



Other "measurements" – 2025

Anthropic's internal study (12/2025)

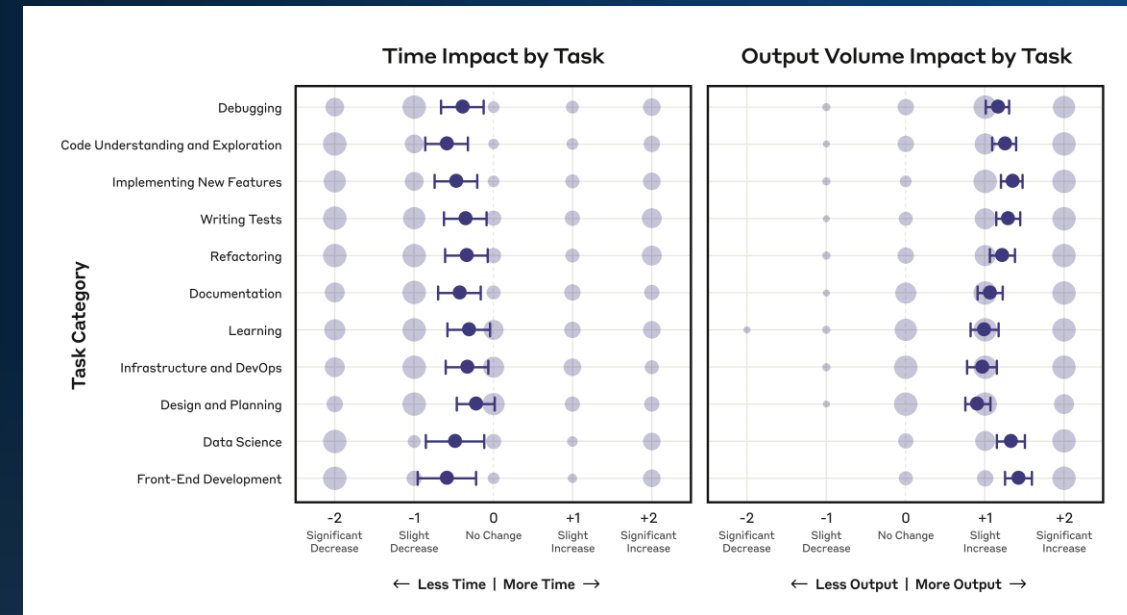
- 67% more pull requests per day
- Self-measured productivity increase 20-50%
- But, "organizational numbers had not improved. Devs felt faster, customer deliveries stayed the same."
- <https://www.anthropic.com/research/how-ai-is-transforming-work-at-anthropic>
- <https://collinwilkins.com/articles/claude-code-productivity-paradox>

DORA study of 5000 devs (9/2025)

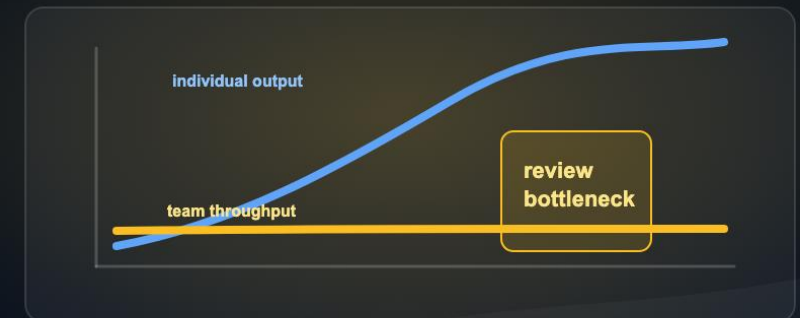
- "AI boosts individual-level output metrics."
- <https://cloud.google.com/resources/content/2025-dora-ai-assisted-software-development-report>

Faros AI study of 10 000 devs (7/2025)

- "AI coding assistants increase developer output, but not company productivity."
- <https://www.faros.ai/blog/ai-software-engineering>



Engineering Leadership



Productivity Paradox

Faros' studies key takeaway:

"We call this pattern the Acceleration Whiplash: real throughput gains at the top, compounding quality costs at every stage below."

AI ENGINEERING REPORT 2026 | KEY FINDINGS

The Acceleration Whiplash

Telemetry data from 22,000 developers on the real-world impact of AI adoption.

OUTPUT IS UP...

Impacts of high AI adoption on engineering throughput



... WHILE PRODUCTION QUALITY IS DECLINING

Impacts of high AI adoption on production stability and quality



faros.ai/acceleration-whiplash



"Software delivery performance metrics, like the DORA metrics of lead time, deployment frequency, change failure rate, and MTTR, remain under significant pressure."

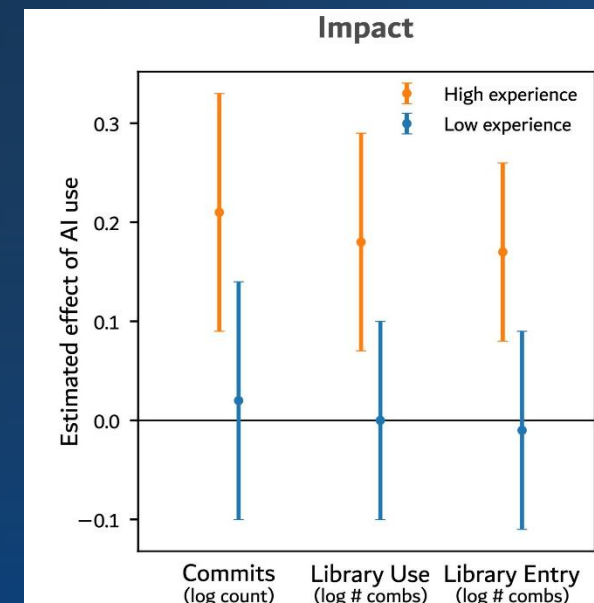
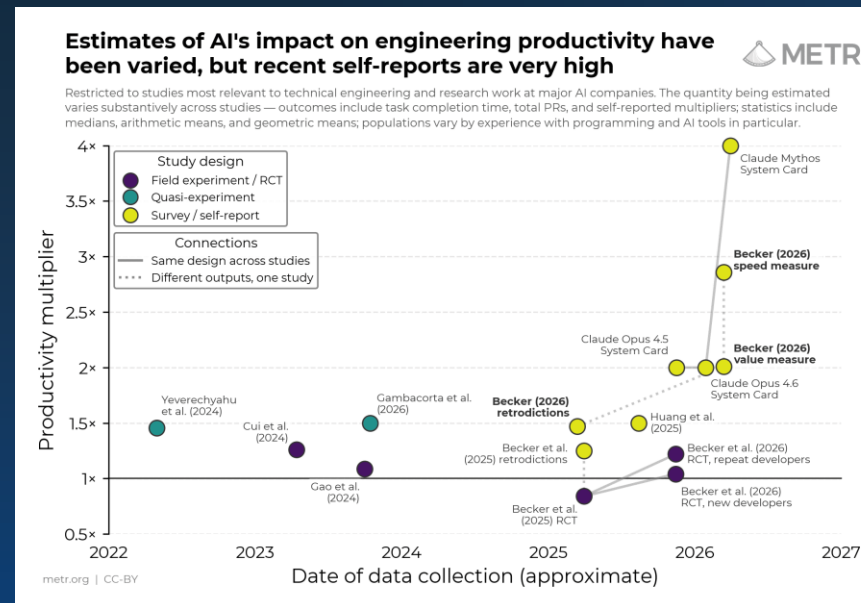
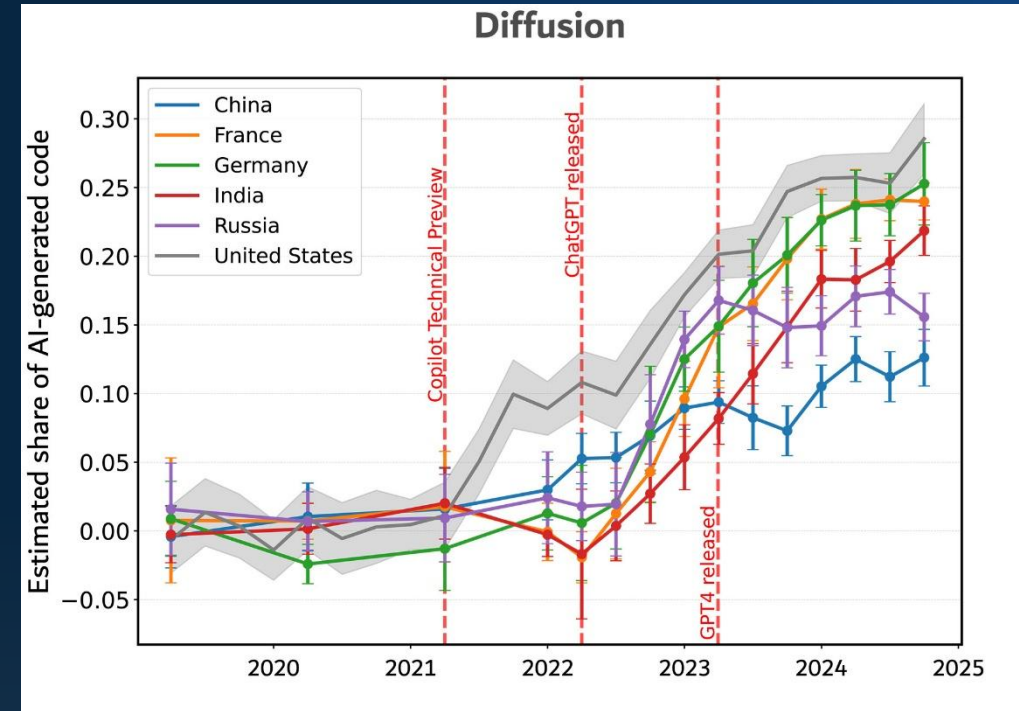
How about 2026?

Science / AAAS study / 6 countries (1/2026)

- Experienced, senior-level programmers (≥ 6 years on GitHub) capture nearly all productivity and exploration gains...
- ...while no benefits were observed for early-career programmers.
- <https://www.science.org/doi/10.1126/science.adz9311>

METR update (2-5/2026)

- "Recent self-reports are very high"
- <https://metr.org/blog/2026-05-11-ai-usage-survey/>



Amdahl's law

(Gene Amdahl, 1967)

Originally a rule for computer performance improvement

Original formulation

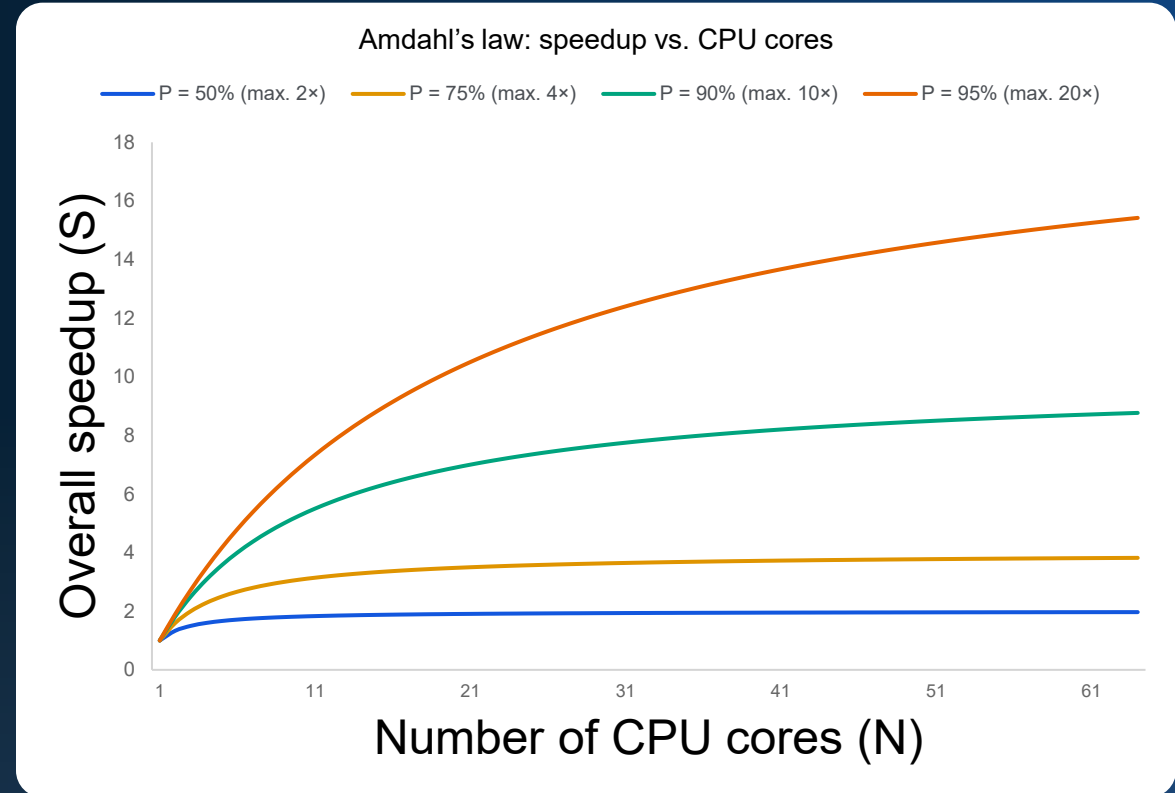
$$S = 1 / [(1 - P) + P/N]$$

S = Achievable speedup of the entire program

P = Fraction that can run in parallel (e.g. 0.80 = 80%)

1 - P = Sequential fraction that cannot run in parallel

N = Number of parallel processors (CPU cores)



AI development: what follows from faster “coding”?

- Faster code generation simply shifts the bottleneck elsewhere
- For example, to requirements, verification or deployment
- These stages need corresponding efficiency improvements

Little's law

John D. C. Little, 1961

Origins: operations research and queueing theory

Original formulation

$$WIP = TH \times CT$$

WIP (Work in Process): work currently in the system

Example: 30 customers

TH (Throughput): units completed per unit of time

Example: 2 customers / min

CT (Cycle Time): time a unit spends in the system

Example: 15 min

The AI development paradox

Faster coding does not help customers if the bottleneck remains.

Before	AI without more verification capacity	AI + automated evidence + WIP limit
Idea → Implementation	Idea → Implementation	Idea → Implementation
Small queue → Review / V&V	Faster implementation	Faster implementation
Bottleneck → Release	Large queue → Review / V&V	Small queue → Review / V&V
	Bottleneck → Release	Automation / evidence → Release
WIP: 6	WIP: 12	WIP: 6
Throughput: 2 features / week	Throughput: 2 features / week	Throughput: 4 features / week
Cycle time: 3 weeks	Cycle time: 6 weeks	Cycle time: 1.5 weeks
Customer gets a feature in about 3 weeks.	Faster code, longer customer waits.	Now the customer benefits.

Little's law: $WIP = Throughput \times Cycle\ time$

Shorter cycle times require more bottleneck capacity or less WIP.

AI speeds up implementation. Without faster review, WIP and cycle time grow.

Benefits require more effective verification or a limit on WIP.

Three steady-state averages, not the transient period during rollout.

AI speeds up implementation, but not automatically overall throughput.

- Starting more work with the extra capacity increases WIP.
- Unless the bottleneck (e.g. review, testing, V&V) also speeds up, queues and cycle time grow.
- AI benefits require a WIP limit or more capacity at the bottleneck.

The problem we are solving

- AI makes writing code several times faster
- Requirements, design, testing and review do not scale at the same pace
- **Code piles up that nobody can account for: what it should do, how it works, or whether it works correctly**
- *"Which requirement justifies the code that AI wrote for you last week?"*

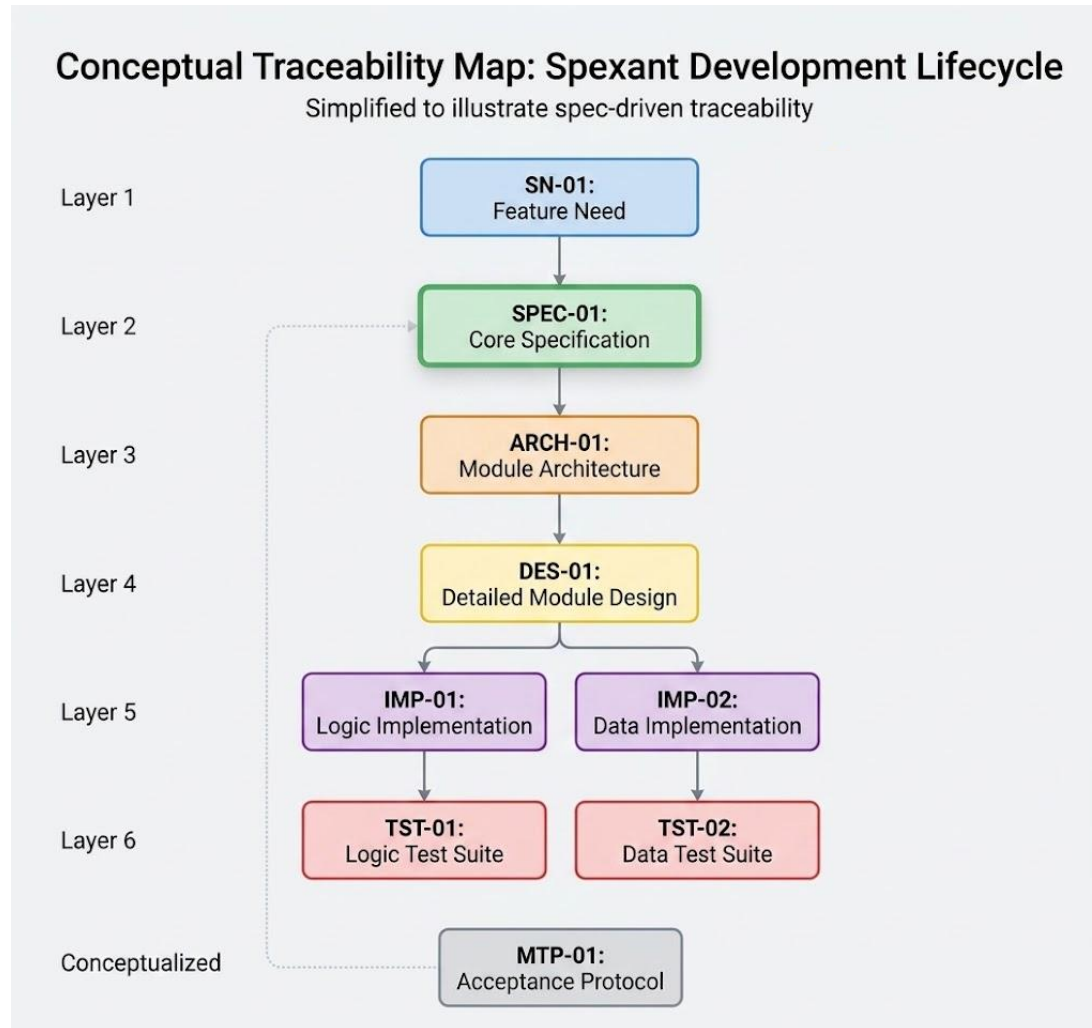


Spexant in brief



Spexant is an agent-based software development system in which AI produces the same artifacts as a human engineer **in a professional development process.** All of it under human supervision.

Which artifacts?



From Assembly to AI-Native SDD

The evolution of software development

CLOSE TO HARDWARE

RISING ABSTRACTION & PRODUCTIVITY

SPECIFICATION-FIRST, AI-ENABLED

1

1950s –
Assembly Era



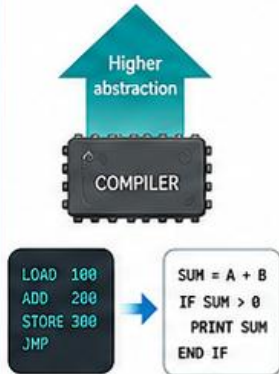
```
LOAD 100
ADD 200
STORE 300
JMP 400
```

REGISTERS	
AC	0001
PC	0400
IR	ADD
MAR	2000

Software is coded close to the hardware, in assembly and machine-oriented instructions.

2

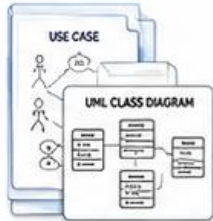
1957–1960 –
Higher-Level
Languages Raise
the Abstraction Level



FORTRAN (1957), LISP (1958), and COBOL (1959/1960) help developers move above hardware details. Compilers unlock a major abstraction jump.

3

1990s –
Careful Software
Design Is Taught



- ✓ Requirements specification
- ✓ Use cases
- ✓ Architecture plan
- ✓ UML design
- ✓ Unit tests
- ✓ Sequence, ER, and UML diagrams
- ✓ Test plan

Universities emphasize deliberate software design and rich engineering artifacts.

4

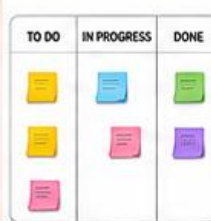
1999 –
XP (Extreme
Programming)



XP promotes iterative delivery, refactoring, tests, and close feedback.

5

2001 –
Agile Goes
Mainstream



The Agile Manifesto shifts attention toward working software, collaboration, and responsiveness to change.

6

2005 –
Martin Fowler
asks: “Is Design
Dead?”



The debate highlights design's evolving role in agile development.

7

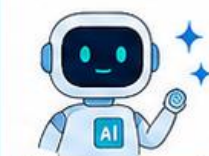
2001–2020 –
Two Decades of
Agile / Scrum



Agile and Scrum dominate practice. Many teams reduce documentation and rely heavily on backlogs, ceremonies, and continuous iteration.

8

2022–Present –
LLMs Transform
Coding

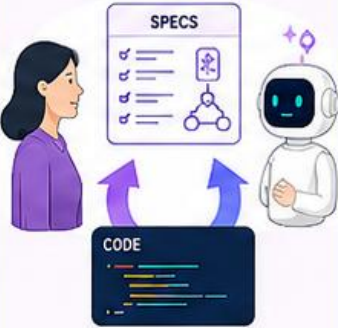


- Generate Code
- Write Tests
- Refactor

LLMs automate large parts of coding, testing, and refactoring. Manual coding becomes far less central than before.

9

Now / Next –
AI-Native SDD
(Specification-Driven Development)



Software development stays agile – but in a new way. Humans work with AI, and the most important human skill becomes specification and design ability.

- Rich artifacts return
- Requirements, use cases, architecture, UI flows, unit specs, diagrams, test plans
- Humans + AI can update any artifact
- Bidirectional traceability between specs and code is essential



KEY TAKEAWAY

AI-native SDD revives the strengths of 1990s software design – but now in collaboration with AI, with continuous agility and strict traceability between specs and code.



Design matters



Agile evolves



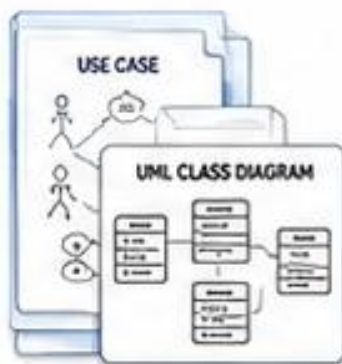
Humans + AI collaborate



Traceability ensures trust

3

1990s – Careful Software Design Is Taught



- ☒ Requirements specification
- ☒ Use cases
- ☒ Architecture plan
- ☒ UML design
- ☒ Unit tests
- ☒ Sequence, ER, and UML diagrams
- ☒ Test plan

Universities emphasize
deliberate software design and
rich engineering artifacts.

Agility stays
In the picture:

With help of
62304-level
traceability (or
similar)

...but transforms

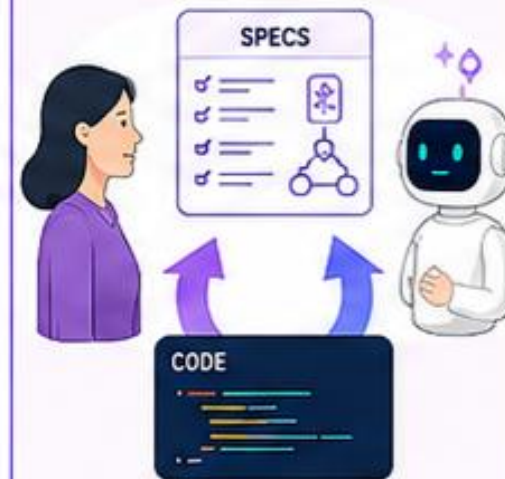


IEC
62304:2006
Medical Device Software

9

Now / Next – AI-Native SDD

(Specification-Driven Development)



Software development stays agile —
but in a new way. Humans work with AI,
and the most important human skill
becomes specification and design ability.



Rich artifacts return



Requirements, use cases, architecture,
UI flows, unit specs, diagrams, test plans



Humans + AI can update any artifact



Bidirectional traceability between
specs and code is essential

Artifact examples: stakeholder needs

Stakeholder Needs — Spexant

SN-001: AI as Productive Participant

Stakeholders need an AI system that participates as a productive member of the software development team — capable of producing the same types of artifacts that human engineers produce (requirements, designs, code, tests, documentation) — so that development speed and capacity are increased.

Source: reqspec.md §1.1, §2.3 (AM-002)

SN-002: Human Oversight and Control

Stakeholders need humans to remain in control of all decisions that affect the delivered software. No AI-generated artifact may become part of the delivered system without explicit human review and approval.

Source: reqspec.md §2.2 (HI-006), §2.7 (AI-005); usecases.md UC-PH-003

Artifact examples: requirements

▼ AM — Artifact Management Requirements (9 reqs)

100%

ID	Title	SN	Arch	Impl	Test
AM-001	Comprehensive Software Definition	✓	✓	✓	✓
AM-002	AI as Artifact Author	✓	✓	✓	✓
AM-003	Unified Artifact Repository	✓	✓	✓	✓
AM-004	Artifact Identification	✓	✓	✓	✓
AM-005	Version Control	✓	✓	✓	✓
AM-006	Baseline Management	✓	✓	✓	✓
AM-007	Document AI-Editable Flag	✓	✓	✓	✓
AM-008	Artifact Linking and Cross-Referencing	✓	✓	✓	✓
AM-009	Schema-Driven Artifact Format Validation	✓	✓	✓	✓

Artifact examples: architecture

[Dashboard](#) > [Artifact](#) > 03-architecture/architecture.md

5.1 Layer 1: Artifact Storage & Management

ID: ARCH.5.1 Type: **architecture-component** Status: **active** Path: 03-architecture/architecture.md

5.1 Layer 1: Artifact Storage & Management

Purpose: Provide a unified, versioned, permission-controlled repository for every artifact produced during the software lifecycle.

Responsibilities:

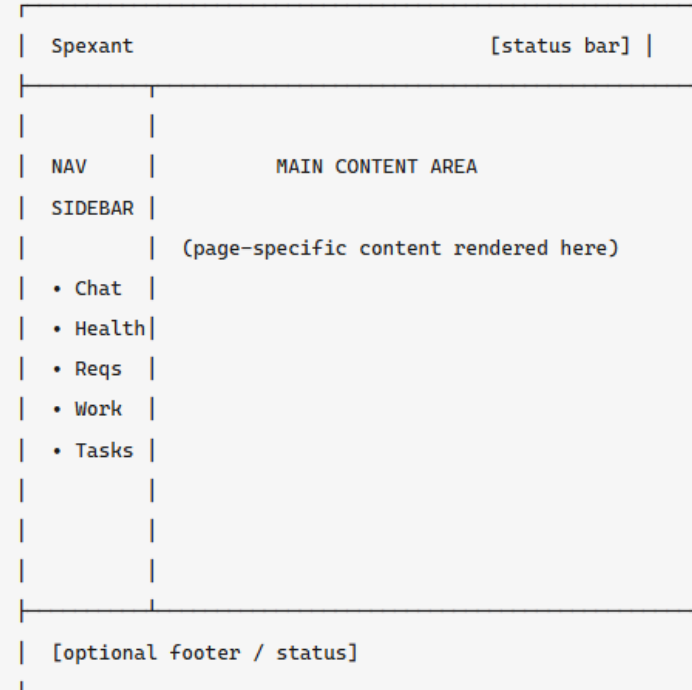
- Assign and maintain a unique persistent identifier for every artifact (AM-004)
- Store all artifact versions with timestamp, author identity, and change rationale (AM-005)
- Enforce the `ai_editable` boolean flag per artifact: `ai_editable: false` prevents any AI write; `ai_editable: true` allows AI-proposed changes through the MR workflow for human approval (AM-007)
- Support heterogeneous artifact types: requirements, designs, source code, tests, decision logs, chat transcripts, work items, regulatory records, and **binary artifacts** (images, PDFs, design files, sketches, mockups, wireframes, and any other non-text file types) as first-class managed artifacts subject to the same lifecycle controls (AM-001, AM-002, AM-003, IN-002)
- Store binary artifacts with full version history maintained via Git commit messages (commit message serves as the change rationale record for files that cannot carry inline front-matter); route files that exceed the project binary size cap through Git LFS per the binary storage policy declared in

Artifact examples: design documents

2. Application Shell and Navigation

2.1 Layout Structure

The application uses a **sidebar + main content** layout. The sidebar is persistent across all pages and provides primary navigation. The main content area fills the remaining viewport width.



Sidebar width: Fixed at 220px. Collapsible to icon-only (48px) via a toggle button, with the collapsed state persisted in `localStorage`.

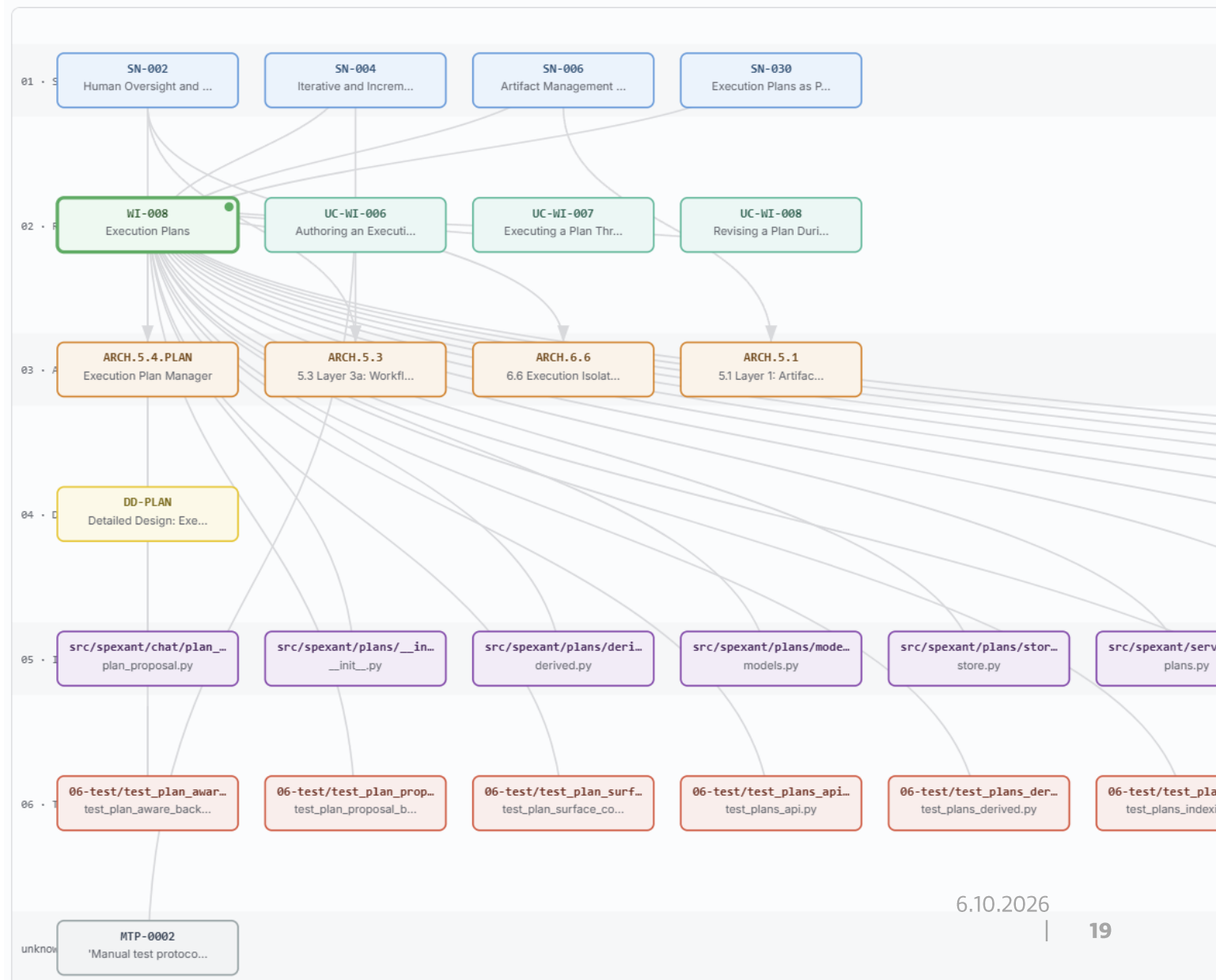
Status bar (top right): Shows project name, repository path, a cache freshness indicator (time since last traceability graph rebuild) with a manual refresh button, and a **repository sync indicator** showing whether the local default branch is up to date with the remote (synced / behind N commits / diverged). When the indicator shows "behind", a **Sync Now** button triggers a fast-forward pull (`git pull --ff-only`). The button is disabled with a tooltip while an orchestrator task is active (HI-011). After a successful sync, the traceability cache is automatically rebuilt.

Traceability

Trace Graph WI-008

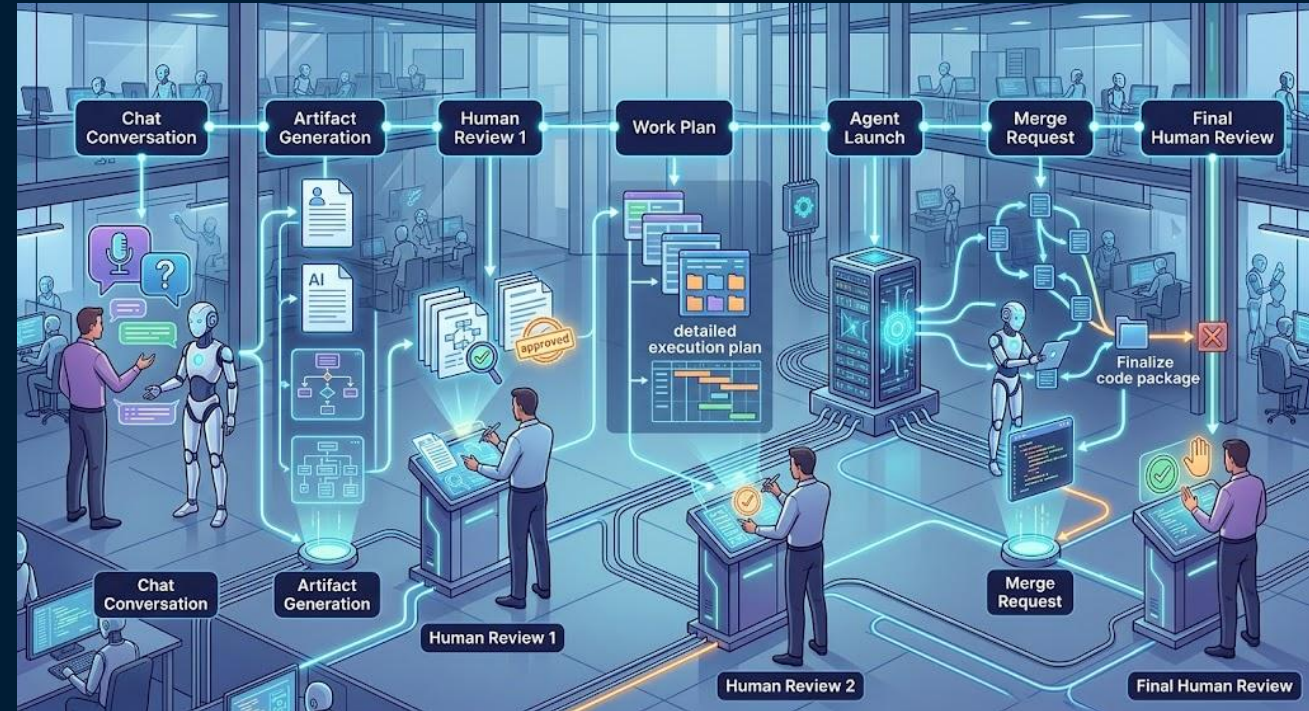
Depth: 3 35 nodes

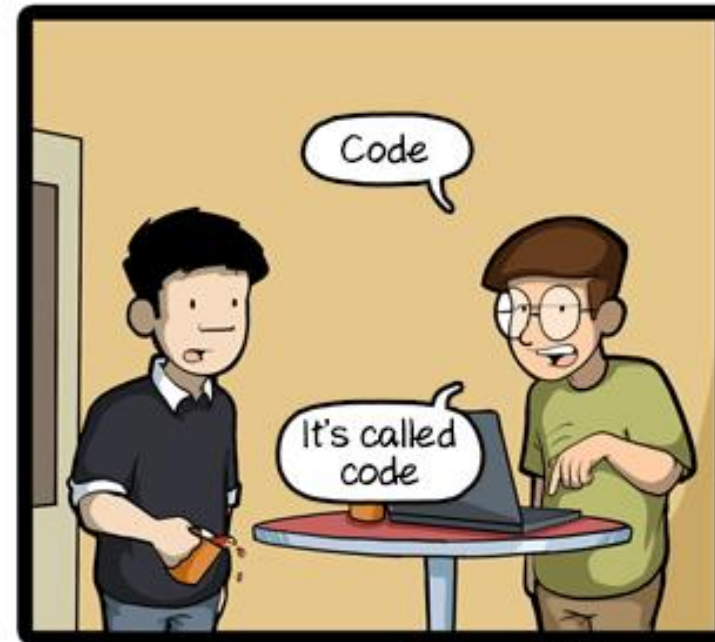
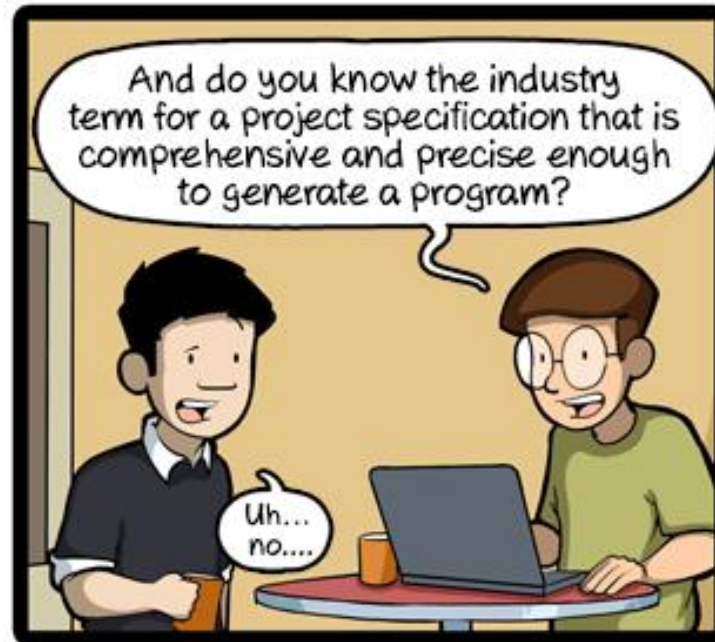
stakeholder need requirement architecture component detailed design implementation test use case gap

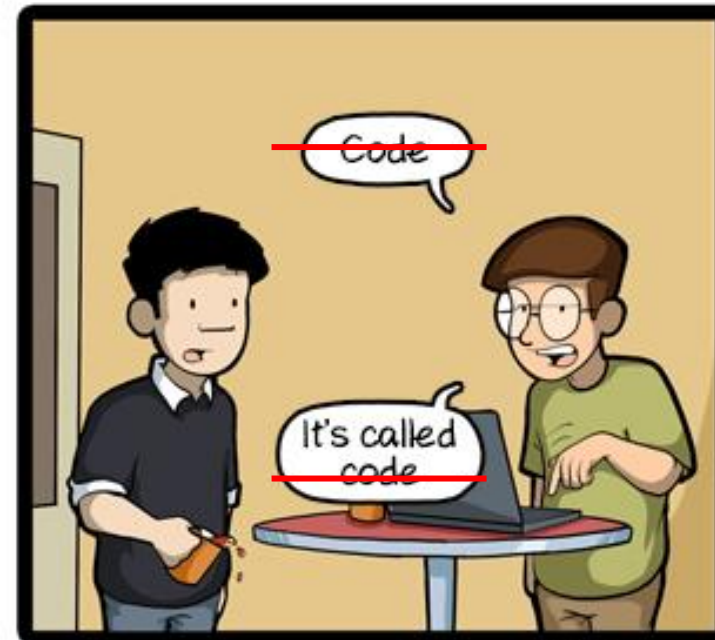
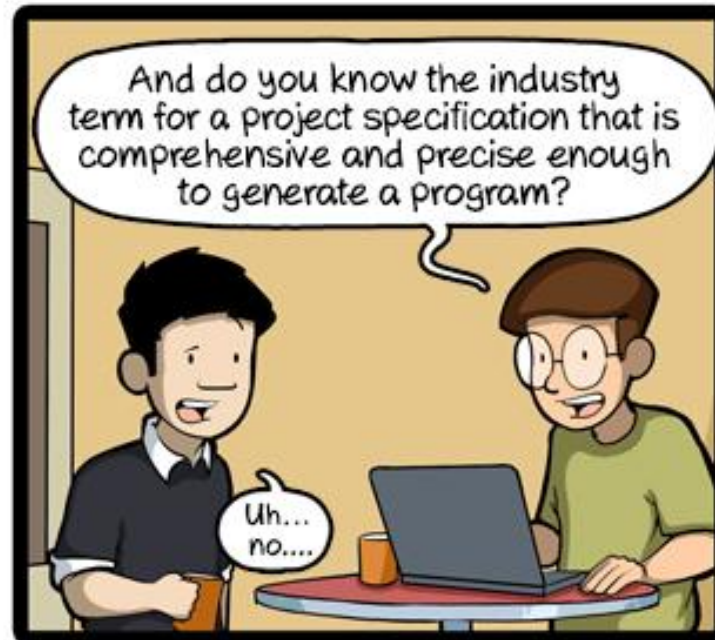


A typical Spexant workflow

- Chat conversation
 - Stakeholder needs, requirements, use cases, architecture, detailed design
 - **a human reviews and approves each one separately**
 - work items, execution plan
 - **a human reviews and approves**
 - the orchestrator launches an agent in a worktree
 - merge request
 - **a human reviews and approves**



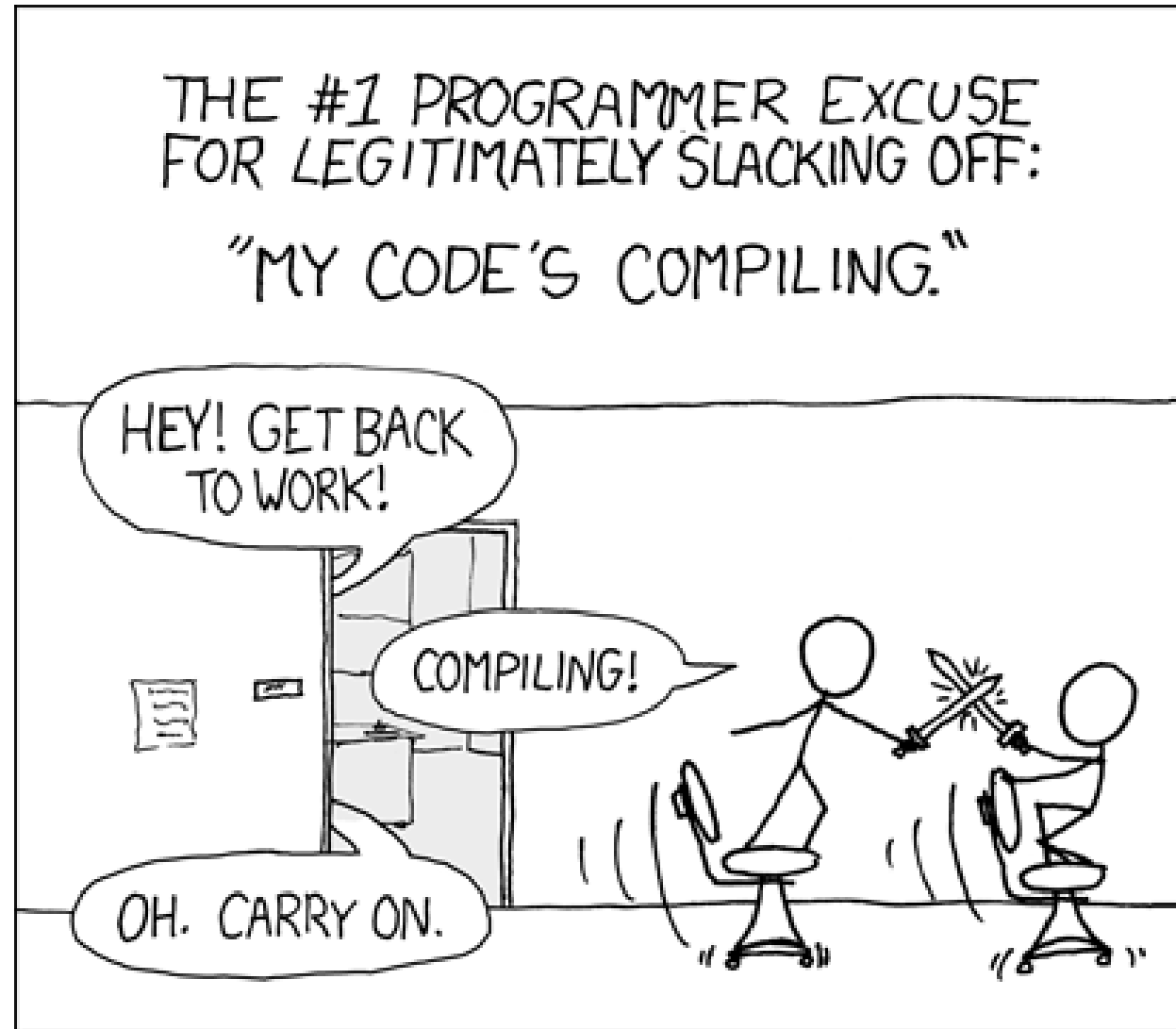




Specification Artifact
Collection (SAC)

Or Specification as
Code (SaC)

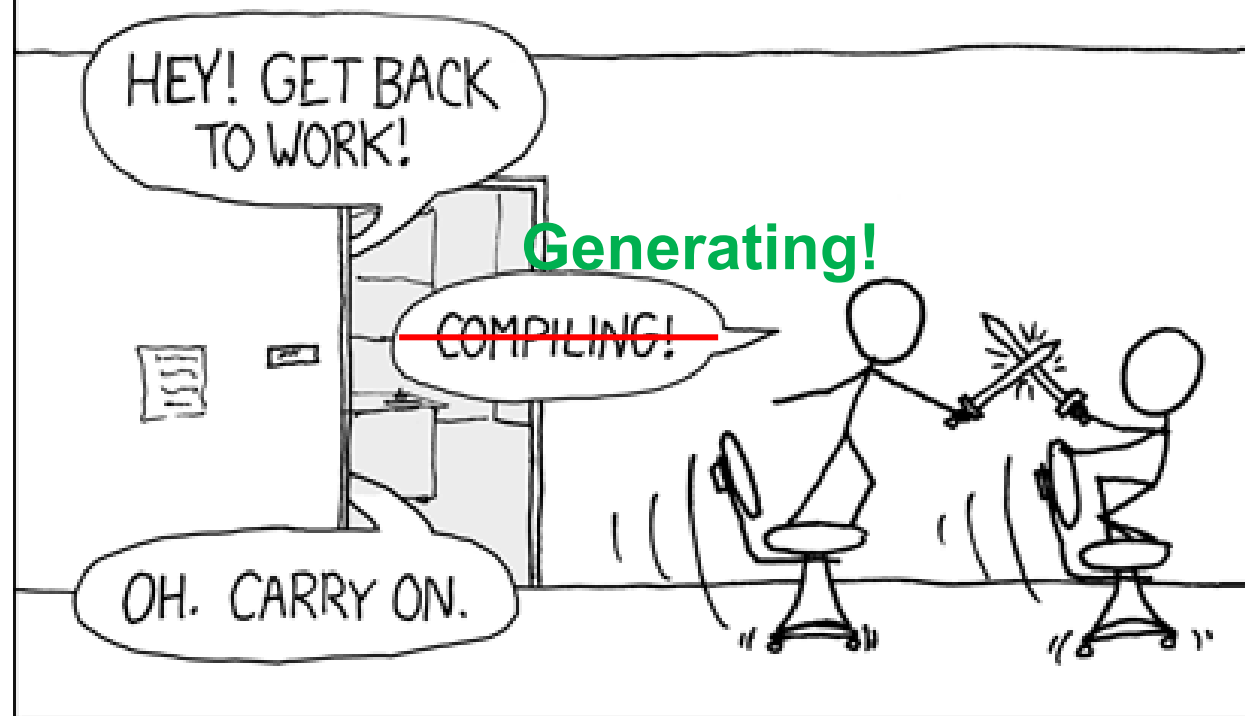
BEFORE



NOW

THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:

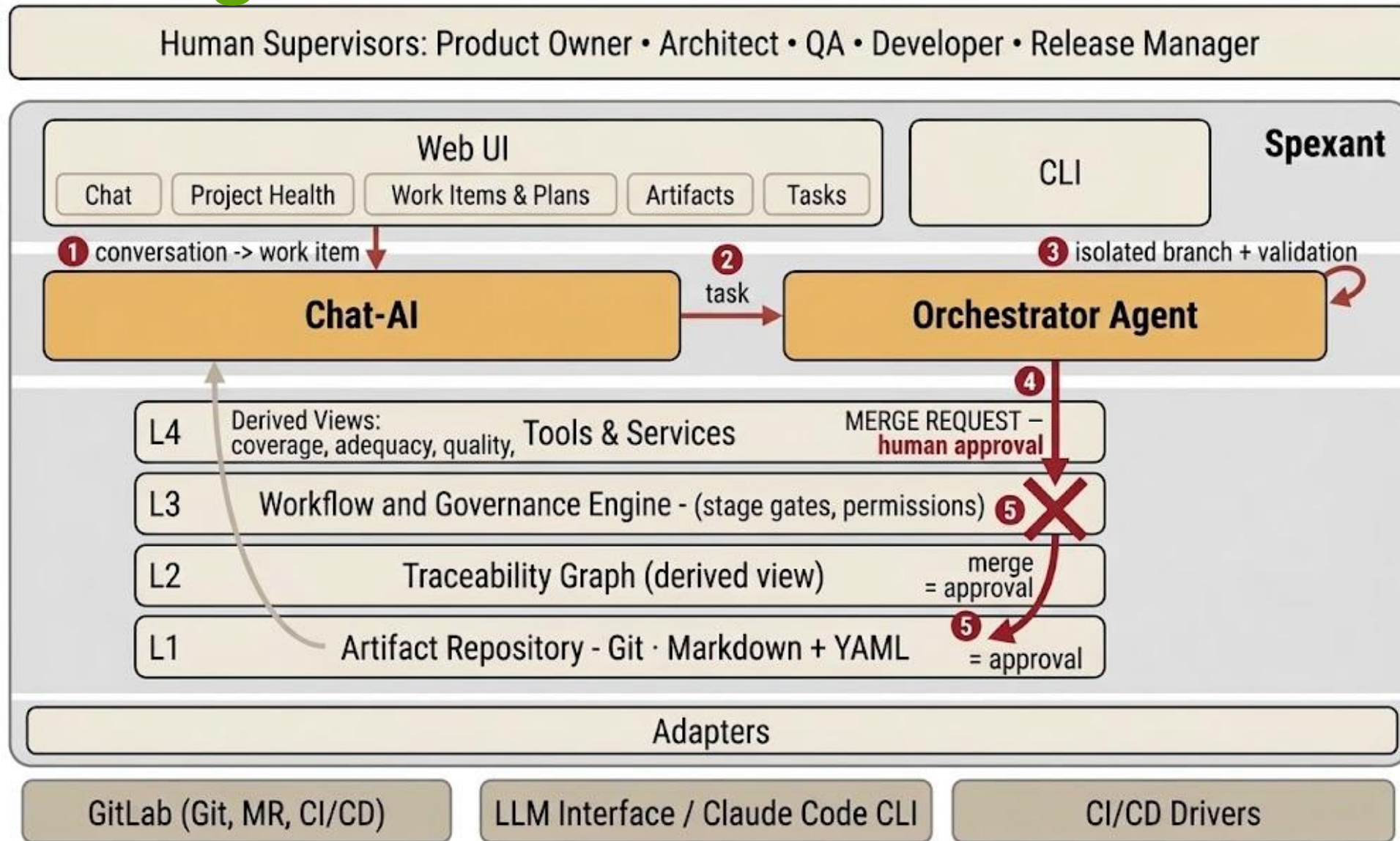
"MY CODE'S ~~COMPILING.~~ ...Being generated.



xkcd

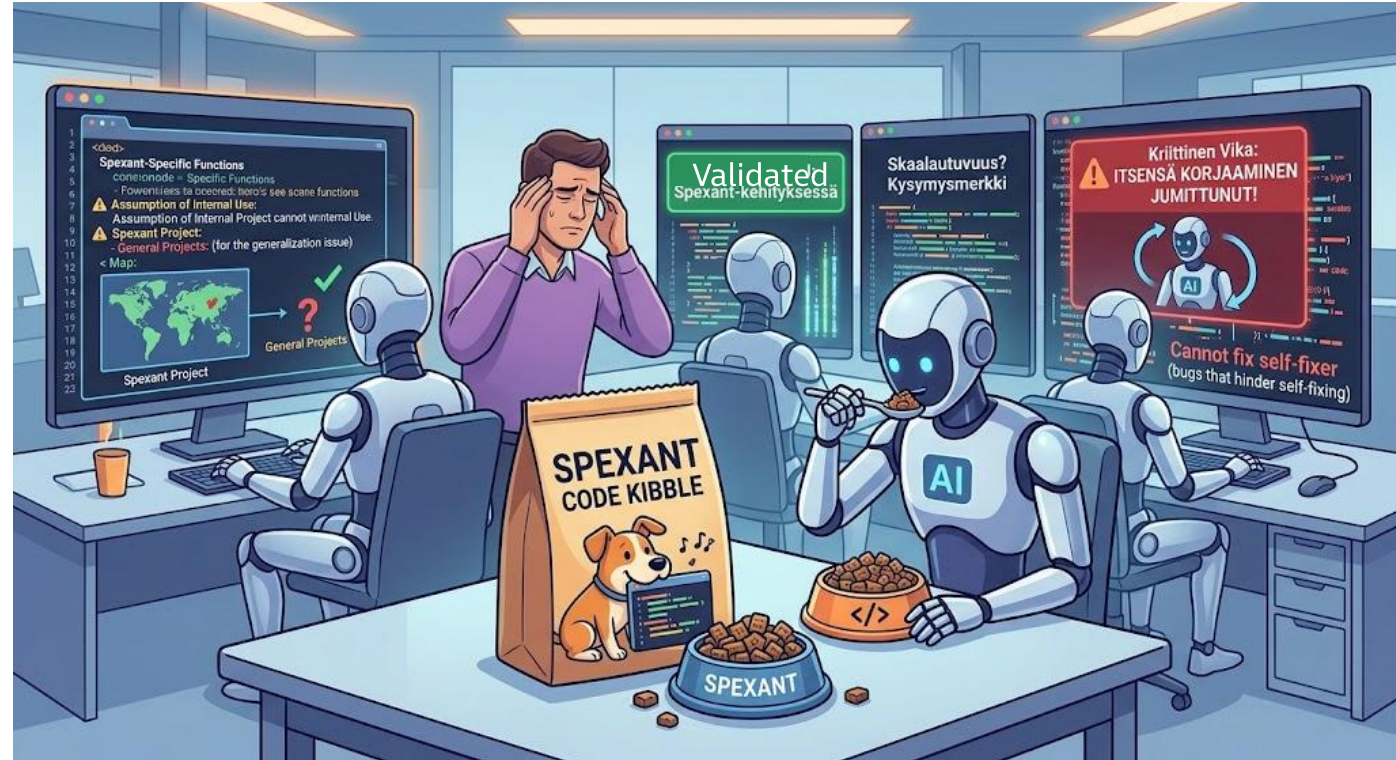
<https://xkcd.com/303/>

High-level architecture



Eat your own dog food

- **Spexant is built with Spexant**
- This constantly validates and proves that the tool works



✓

Foo2

[Edit].spe...

Mika Torh...

[File] 05-in...

Varastokirj...

127.0.0.1:8...

Tavarat - L...

05-implem...

Human im...

Human im...

Foo2

+

−

□

×

←→↺127.0.0.1:8000/needs☆MNew Chrome available⋮

SPEXANT

<

Project Health

Chat

Artifacts

Stakeholder Needs

Requirements

Use Cases

Architecture

Detailed Design

Implementation

Tests

Test Files

Release

Source Materials

Work Items

Tasks

Settings

v0.5.8

Foo2SyncedSync Now✓ Dev-ready⚙ Light

Stakeholder Needs

Upload Image

SN-001: Selainpohjainen 2D-avaruustaistelupeli

Stakeholder(s): Pelaaja, Kehittäjä

Need statement:

Tarvitaan selaimessa toimiva 2D-avaruuspeli, jossa kaksi alusta taistelevat toisiaan vastaan maapallon ympäristössä.

Acceptance criteria:

• Peli käynnistyy ja on pelattavissa modernissa selaimessa ilman asennuksia.

• Peli renderöi 2D-pelikentän reaaliaikaisesti.

Constraints / context:

Käytettävän teknologian (pelikirjasto ym.) tulee olla kustannusvapaata.

id: SN-002 title: "Kahden aluksen taistelupelimekaniikka" type: stakeholder-need status: draft priority: high source: "Elicitation-keskustelu 2025-04-21" created: 2025-04-21T00:00:00

SN-002: Kahden aluksen taistelupelimekaniikka

Stakeholder(s): Pelaaja

Need statement:

Pelissä on kaksi avaruusalusta — ihmisen ohjaama alus ja alien-alus — jotka taistelevat toisiaan vastaan. Alien-alusta ohjaa joko tekoäly tai toinen pelaaja.

Acceptance criteria:

• Molemmat alukset voivat liikkua vapaasti ja ampua ammuksia.

• Peli päättyy kun toinen aluksista tuhoutuu.

• Alien-alus voidaan vaihtaa AI-ohjauksesta toisen pelaajan ohjaukseen.

Constraints / context:

POC-versiossa alien-alusta ohjaa AI; kaksinpeli lisätään myöhemmin.

id: SN-003 title: "Avaruusympäristö maapallon ympärillä" type: stakeholder-need status: draft priority: high source: "Elicitation-keskustelu 2025-04-21" created: 2025-04-21T00:00:00

SN-003: Avaruusympäristö maapallon ympärillä

Stakeholder(s): Pelaaja

Need statement:

Pelikenttä esittää avaruutta, jossa maapallo on keskellä esteenä. Pelikentän reunat ovat wraparound-tyyppiset: reunan yli menevä objekti ilmestyy vastakkaiselta puolelta.

Acceptance criteria:

• Maapallo renderöidään keskelle pelikenttää ja se toimii fyysisenä esteenä.

• Maapallo on tuhoutumaton.

INDEX

SN-001

SN-002

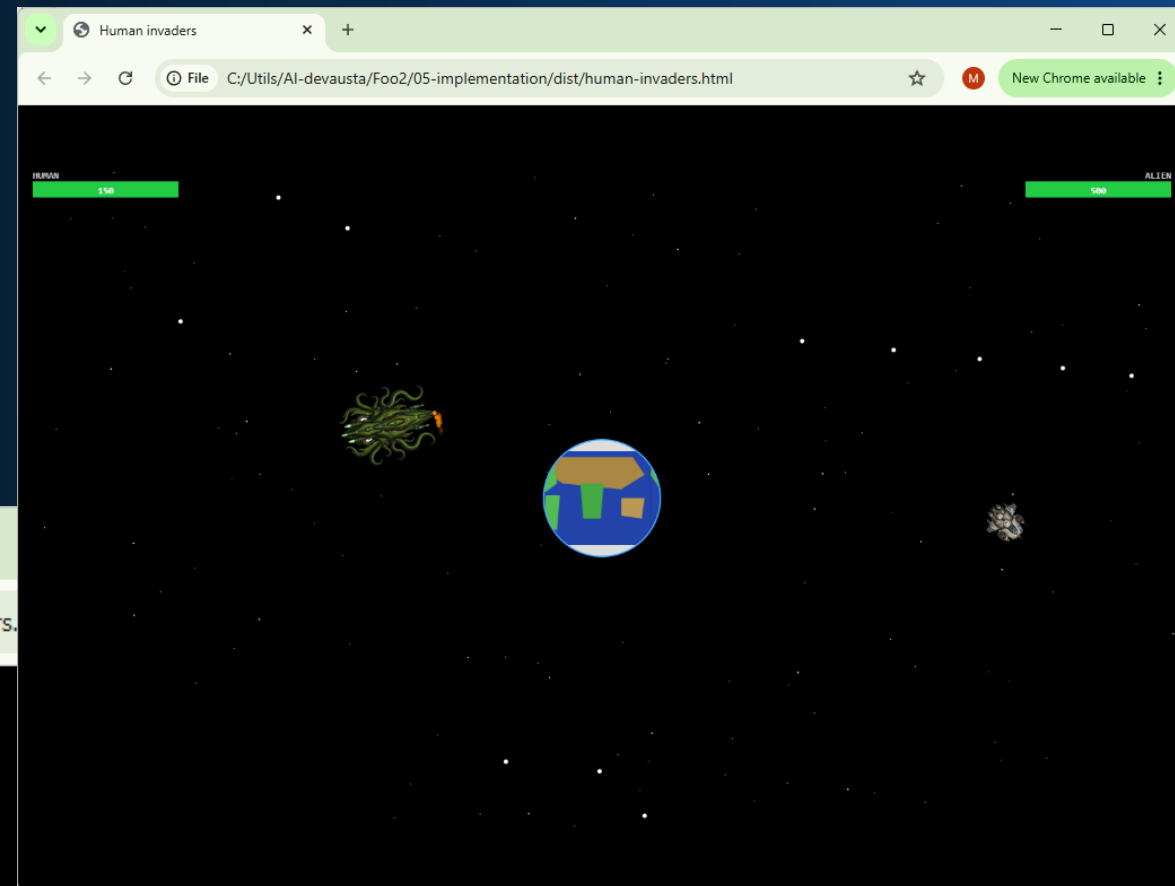
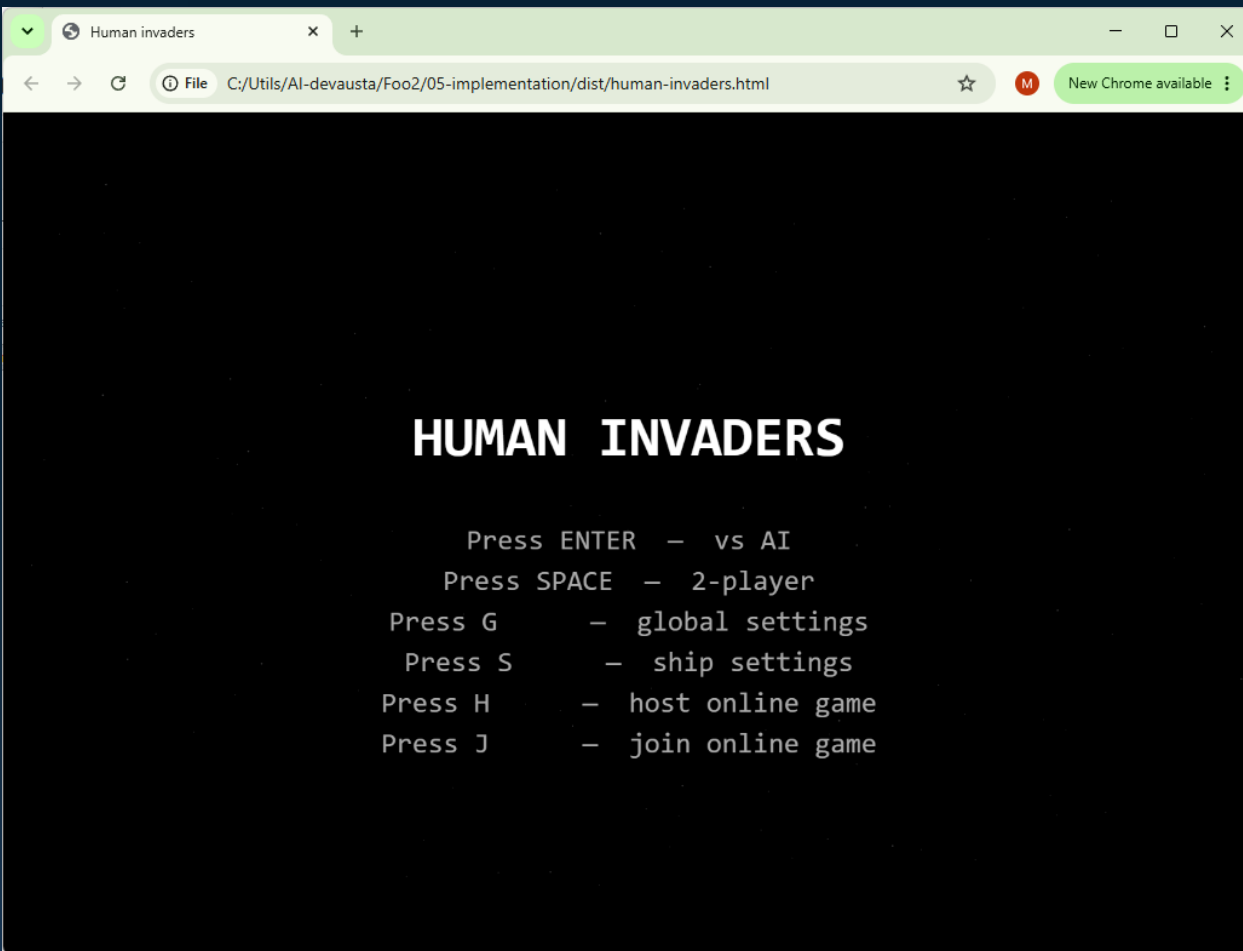
SN-003

SN-004

SN-005

SN-006

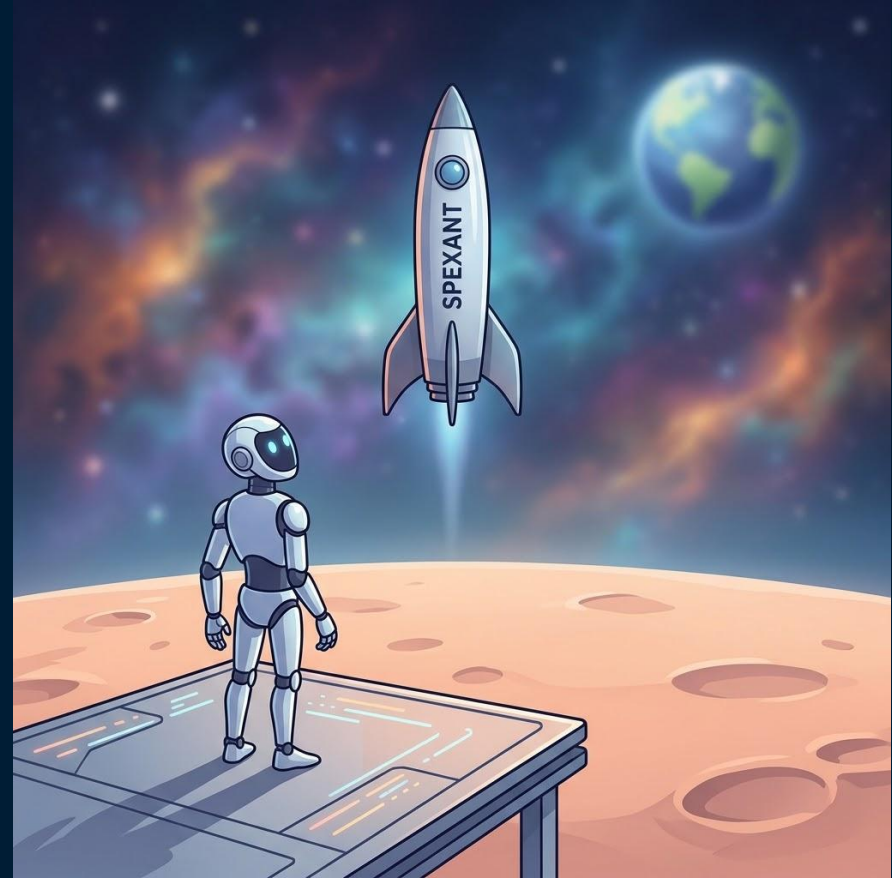
ATOSTEK



Where is Spexant going?

Directions of development:

- Team use
- Continuous verification
- Visibility and metrics
- Larger autonomous units of work



Larger autonomous units of work

- Steps forward:

Step	Human review	Status

Larger autonomous units of work

- Steps forward:

Step	Human review	Status
1. A single task	Every merge request	Works today

Larger autonomous units of work

- Steps forward:

Step	Human review	Status
1. A single task	Every merge request	Works today
2. A plan with its steps	After every step	Works today

Larger autonomous units of work

- Steps forward:

Step	Human review	Status
1. A single task	Every merge request	Works today
2. A plan with its steps	After every step	Works today
3. Automated verification	Machine-generated evidence replaces some manual checks	Being built

Larger autonomous units of work

- Steps forward:

Step	Human review	Status
1. A single task	Every merge request	Works today
2. A plan with its steps	After every step	Works today
3. Automated verification	Machine-generated evidence replaces some manual checks	Being built
4. A plan runs to completion autonomously	One review of the plan's end result	The goal we are working toward

Execution Plan

< Plans

PLAN-0008 Type-parameterised artifact link routing — eliminate prefix-collision mis-routing

active · owner Risto Pitkänen · from chat chat_session_20260907_090257_762898

2/7

Re-order steps

Insert step at end

Change status

✓ 1 WORK-ITEM WI-0959 Type-parameterised artifact link routing — resolver parameter and server-side kind-to-route endpoint COMPLETE
resolved · high · unassigned

Launch task

Open WI

Insert above

Insert below

Remove

✓ 2 WORK-ITEM WI-0961 Per-ID disambiguation for ambiguous artifact prefixes in prose linkification COMPLETE
resolved · medium · unassigned

Launch task

Open WI

Insert above

Insert below

Remove

▶ 3 GATE manual-test CURRENT ← you are here

Confirm both families sharing the colliding prefix now resolve correctly, before WI-0936 changes what the cache contains. On the Project Health requirement row whose ID uses the colliding prefix and confirm it opens the requirement detail page rather than a missing work item. In chat, ping from each side of the collision and confirm the two links land on different pages. Open a work item detail page and confirm its affected_artifact_id. Passes when all three resolve to the right destination and no link 404s.
no outcome recorded

Record outcome

Insert above

Insert below

Remove

⌚ 4 WORK-ITEM WI-0936 Make traceability exclude_patterns additive to the built-in defaults and exclude the generated record stores BLOCKED
active · high · unassigned
blocked behind step 3 — actions stay enabled; the gate is advisory.

Launch task

Open WI

Insert above

Insert below

Remove

⌚ 5 GATE manual-test BLOCKED

Re-run the previous gate's three checks after WI-0936 has removed work items and the generated record stores from the traceability cache, so that the overrides are computed from. Additionally confirm the Sync Now control and a server restart no longer block on scanning the record stores. Resolution is unchanged from the previous gate and the rebuild is visibly faster.
no outcome recorded
blocked behind step 3 — actions stay enabled; the gate is advisory.

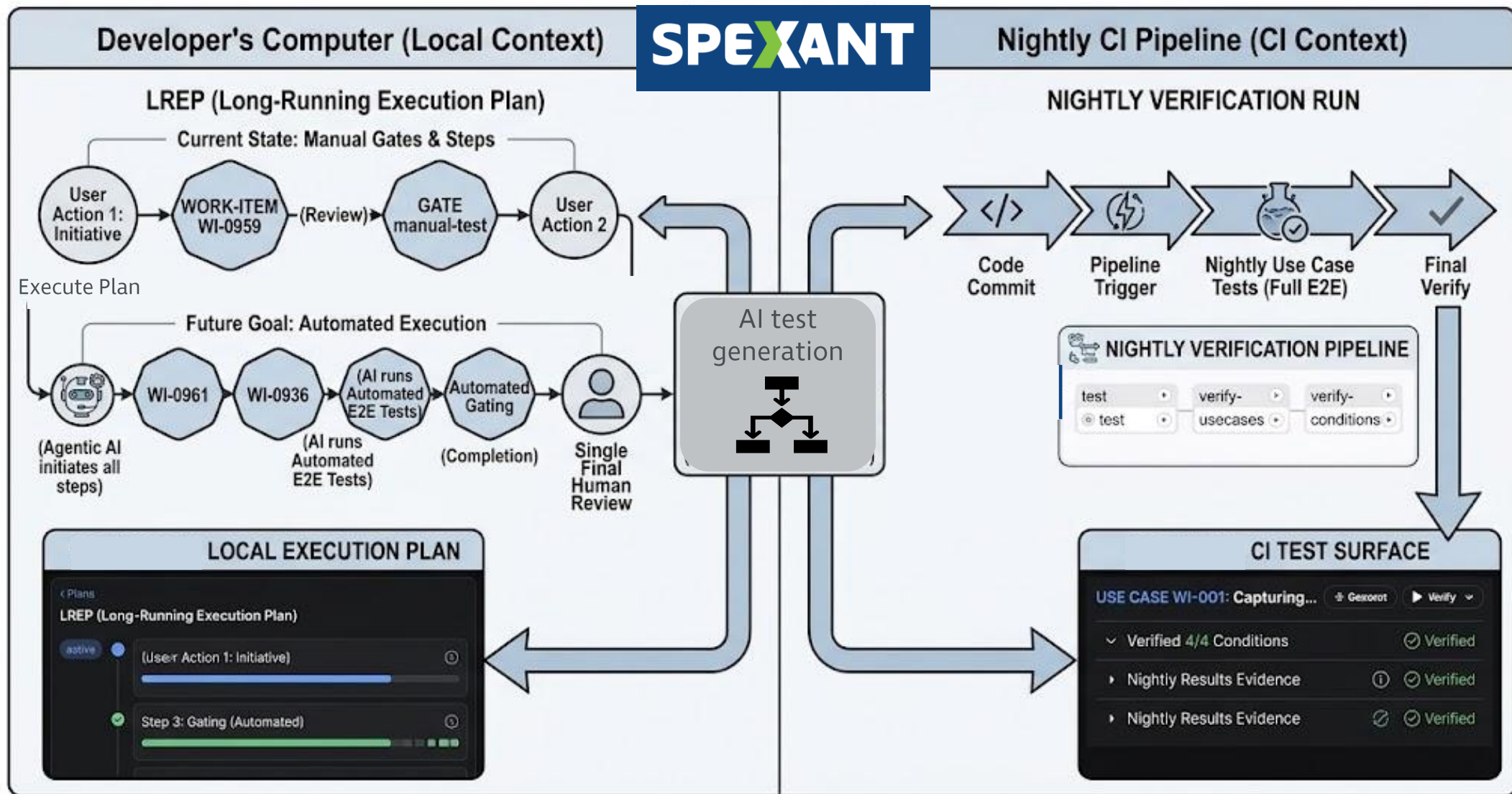
Record outcome

Insert above

Insert below

Remove

⌚ 6 WORK-ITEM WI-0960 Carry the resolved target artifact kind in trace-link, affected-artifact, and health-drilldown payloads BLOCKED
active · medium · unassigned
blocked behind step 3 — actions stay enabled; the gate is advisory.





Thank you!



Mika Torhola

Director of new Business

mika.torhola@atostek.com



Risto Pitkanen

Chief AI Officer

risto.pitkanen@atostek.com