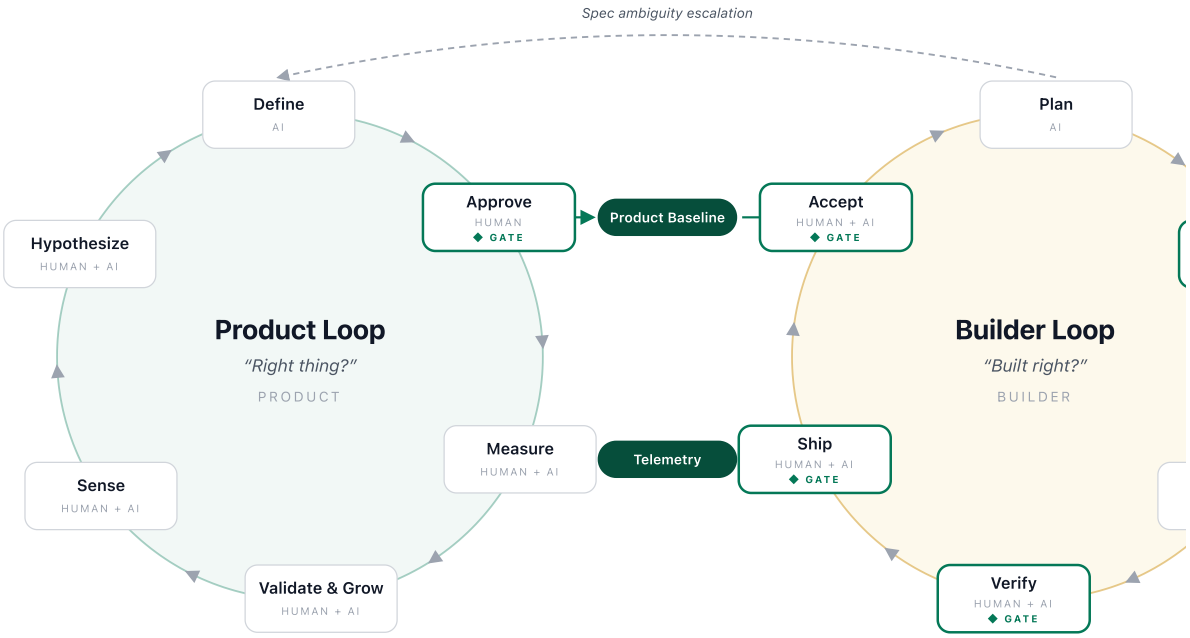
The Product Loop & Builder Loop

The Two-Loop Operating Model for AI-Native Product Development

AI-DLC, Spec-Kit, and OpenSpec run a single engineering loop: the task arrives from outside, and verification means the code matches the task. The two-loop model adds the missing Product Loop — originating what to build and verifying it worked — the same move Scrum@Scale made for Scrum.

PBHQ Original v1.0 Last updated August 2026

two-loop product-loop builder-loop operating-model ai-dlc working-backwards



The execution side of AI-native development has been solved with remarkable speed. AWS AI-DLC steers an agent through requirements, design, construction, and testing. GitHub Spec-Kit drives any coding agent from a feature description through specification, planning, and implementation. OpenSpec turns change proposals into delta specs, tasks, and merged code. Each installs workflow

instructions into a coding agent — Claude Code, Codex, Cursor, Copilot — and the agent does the work under human gates.

Look closely at what these systems share. AWS publishes AI-DLC's development cycle explicitly:

> Humans provide task
> AI creates plan, seeks clarification
> Humans provide clarification
> AI refines plan
> Humans approve plan
> AI executes plan
> Humans verify outcome — and the cycle repeats

This is a genuinely good loop. It is also, structurally, half a system. Two questions have no accountable owner anywhere inside it:

Where did the task come from? Step 1 assumes someone already knows what to build. The loop begins *after* the hardest product judgment has been made — and offers no machinery for making it.

Did the work achieve its goal? Step 7 verifies the code matches the task. It does not ask whether the task was worth doing — whether anyone adopted the feature, whether revenue moved, whether the hypothesis behind the work survived contact with users.

A loop that starts with a task from nowhere and ends with "matches the task" is an engineering loop. It optimizes *built right* and is silent on *right thing*. That is not a criticism of AI-DLC, Spec-Kit, or OpenSpec — it is a description of their scope. The gap is above them.

We Have Seen This Before

Scrum had the same structure and the same gap. The standard Scrum cycle — backlog, sprint planning, sprint, review, retrospective — is a delivery engine that treats the backlog as given. It optimizes the team's loop and leaves product origination and outcome accountability outside the system.

Dr. Jeff Sutherland's answer in Scrum@Scale was not a better sprint. It was a **second cycle**: the framework splits into a **Product Owner Cycle** (strategic vision, backlog prioritization, release planning, feedback) and a **Scrum Master Cycle** (team process, delivery, impediment removal), meeting at defined touchpoints.

Today's AI execution workflows are the Scrum Master Cycle of the AI era — dramatically accelerated, increasingly autonomous, and still waiting for their other half.

The two-loop model is that other half.

The Two-Loop Model

The model is two cycles with the same anatomy and different payloads:

> The **Product Loop** originates what to build and verifies it worked. Its verification question is **"Right thing?"** — measured in adoption, engagement, and revenue.
> The **Builder Loop** implements what was handed off and verifies it matches. Its verification question is **"Built right?"** — measured against specs and scenarios.

Both loops run the same sequence — intent, AI drafts, human clarification, an approval gate, AI executes, verification — because that gate anatomy is what human-gated AI work *is*. What changes between the loops is the artifact (specs versus code) and the verification question. What changes over time, as a team climbs the Software Delivery Autonomy Levels, is who holds each gate: human authority progressively becomes policy authority, from the inside of the system outward.

The Product Loop

Station	Actor	What Happens
Sense	Human + AI	Market signals, user research, and production telemetry arriving from the Builder Loop. Continuous discovery lives here.
Hypothesize	Human + AI	Frame the opportunities worth testing: hypothesis documents, Opportunity Specs, opportunity solution trees.
Define	AI (human-clarified)	AI drafts the definition specs — MRD, Press Release, FAQ, 6-Pager, PRD, UXD — through definition workflows such as **AWS Working Backwards**, evaluated by **LLM-as-a-Judge**.
Approve ◆	Human	The Product Baseline is approved — the versioned handoff contract to the Builder Loop. This gate stays human at every autonomy level: it is where the human's market knowledge advantage is largest.
Measure	Human + AI	Instrument adoption, engagement, and revenue against the hypothesis, using **AI-native product metrics**.
Validate & Grow	Human + AI	Did the hypothesis survive contact with users? Double down or kill. Learnings feed Sense, and the loop turns.

The Builder Loop

Station	Actor	What Happens
Accept ◆	Human + AI	Receive the Product Baseline; the builder accepts the spec as buildable.
Plan	AI (human-clarified)	AI drafts the implementation plan. Spec ambiguity **escalates to the Product Loop** rather than being resolved by builder guesswork — the third seam between the loops.
Approve ◆	Human → policy	The implementation plan is approved. Human at Level 5 and below; policy-gated at Level 6+.
Build	AI	Code, tests, iterate. The agentic coding loop runs unattended inside this station — this is where execution workflows and **Loop Engineering** operate.
Verify ◆	Human → policy	Built-right verification against spec and scenarios. Human review through Level 5; LLM-as-a-Judge and scenario validation at Level 6.
Ship ◆	Human → policy	Release per operational policy. Human gates through Level 6; autonomous operations at Level 7. Ship emits telemetry back to the Product Loop.

The Three Seams

The loops connect at exactly three points, and each is a contract:

The Product Baseline (Product Approve → Builder Accept). The versioned handoff between product and builder — what the **PDLC specification** formalizes: complete definition, canonical layout, quality gates.

Telemetry (Builder Ship → Product Measure). Usage, adoption, and operational data flowing back. Without this seam the Product Loop cannot verify anything, which is precisely the state most AI-accelerated teams are in today: shipping faster into a void.

Escalation (Builder Plan → Product Define). When the spec is ambiguous, the question goes back to the loop that owns the answer. This seam is what makes “the spec is the contract” enforceable rather than aspirational.

The Builder Loop Generalizes, It Does Not Compete

ProductBuildersHQ deliberately does not build execution workflows. The third-party systems are excellent, open, and improving fast — and they all instantiate the Builder Loop:

Builder Loop Station	AWS AI-DLC	GitHub Spec-Kit	OpenSpec
Accept	Humans provide task	`/speckit.specify` input	`/opsx:propose` intent
Plan	AI creates / refines plan	specify → clarify → plan → tasks	proposal, delta specs, design, tasks
Approve ◆	Humans approve plan	plan review	proposal review
Build	AI executes plan	`/speckit.implement`	`/opsx:apply`
Verify ◆	Humans verify outcome	`/speckit.analyze`, `/speckit.converge`	`/opsx:verify`
Ship ◆	— (Operations phase reserved)	—	`/opsx:archive` merges specs

None of these systems contains its own coding model. Each installs workflow instructions into the coding agent you already run, and the agent reads the repository, edits code, and runs the tests. They are engineering-native by design — which is exactly why the **definition workflows** exist: to supply the product half these systems do not attempt, and feed it into them through **defined pipelines**.

Corroboration: The Three Loops

In The Batch #359, Andrew Ng describes AI-assisted development as three nested loops: an **agentic coding loop** (the AI iterating against specs and evals, cycling in minutes), a **developer feedback loop** (a human steering the agent and making product decisions, in tens of minutes to hours), and an **external feedback loop** (real users and the market, in hours to weeks — the slowest).

Ng's loops are timescale rings; the two-loop model is an accountability structure. They reconcile cleanly: his agentic coding loop and developer feedback loop are both inside the Builder Loop — the same accountable human spans them, and the boundary between them *is* the team's autonomy level, dissolving at Level 6 when human code review ends. His external feedback loop is the Product Loop's verification half.

The instructive difference is that Ng's outer loop is passive — feedback *arrives*, slowly. The Product Loop makes it operated: definition workflows originate deliberately, metrics and monitoring verify deliberately, and the loop closes by design rather than by osmosis.

Ng also supplies the criterion for where human gates belong: human-in-the-loop is needed *so long as the human knows something the AI does not*. The human's information advantage is smallest inside Build — the spec says everything — and largest at Sense, Approve, and Validate & Grow, where market and customer knowledge live. That is why, as autonomy rises, gates automate from the inside of the system outward, and why the Product Approve gate is the last one a human should ever surrender.

Gates and Autonomy

The two-loop model composes with the autonomy levels rather than duplicating them. Each gate's authority is a function of level:

Gate	L4–L5	L6	L7
Product: Approve Baseline	Human	Human	Human
Builder: Accept	Human	Human	Policy
Builder: Approve Plan	Human	Policy	Policy
Builder: Verify	Human review	LLM-as-a-Judge + scenarios	Policy
Builder: Ship	Human	Human	Autonomous operations

Read the table bottom-up and the pattern is the point: **autonomy consumes the Builder Loop from the inside out and never reaches the Product Loop's origination gates.** A Level 7 organization still has humans deciding what is worth building and whether it worked — that is what the Governor role governs *with*.

This also yields a practical metric: the share of Builder Loop wall-clock spent in the unattended agentic loop is a clean autonomy proxy — it is the boundary Ng's loop 1 and loop 2 share, measured.

Where Each Framework Fits

The two-loop model is the organizational architecture; the rest of the ProductBuildersHQ system operates it:

Concern	Framework
The person who runs both loops	**Product Builder Maturity Model** — the converged Level 5 Product Builder
How much of the Builder Loop runs without humans	**Software Delivery Autonomy Levels**
Engineering the agentic loop inside Build	**Loop Engineering**
Automated verification at the gates	**LLM-as-a-Judge**
Keeping AI-built systems maintainable	**SCALE**
The Product Loop's definition machinery	**Definition workflows** — Working Backwards, Big Tech, PBHQ Lite
The Builder Loop's execution machinery	**Execution workflows** — AI-DLC, Spec-Kit, OpenSpec, Kiro
The Product Baseline handoff contract	**PDLC**
The Product Loop's outcome instruments	**AI-Native PM Metrics**

The model itself is machine-readable: both loops — stations, actors, gates, seams, and their mappings to workflows and integrations — are defined as data in **specification-workflow-spec** (`loop.schema.json`), alongside the AI-DLC single-loop system as a reference. The diagram above is rendered from that data, and **VisionStudio** consumes the same definitions.

Running the Loops Today

You do not adopt the two-loop model by reorganizing. You adopt it by closing the seams you are missing:

> **If tasks appear from nowhere**, install the Product Loop's front half: pick a **definition workflow** matched to your context and produce a real Product Baseline before work enters the Builder Loop.
> **If you ship into a void**, close the Telemetry seam: instrument adoption and revenue against the hypothesis that justified the work, and put Measure and Validate & Grow on the calendar with the same discipline as sprint review.
> **If builders resolve spec ambiguity by guessing**, open the Escalation seam: ambiguities route back to Define, and the spec gets better instead of the drift getting worse.
> **Then raise autonomy deliberately**: automate Builder Loop gates from the inside out as your **autonomy level** earns it, and never automate Product Approve.

The execution half of AI-native development arrived first and arrived well. The two-loop model is the claim that it was always half — and the machinery for the other half already exists.

Attribution

The Product Loop & Builder Loop model is original ProductBuildersHQ work. It builds gratefully on published third-party work referenced throughout: AWS's AI-DLC development cycle (**awslabs/aidlc-workflows**, MIT-0), GitHub Spec-Kit (**github/spec-kit**, MIT), OpenSpec (**Fission-AI/OpenSpec**, MIT), Dr. Jeff Sutherland's Scrum@Scale two-cycle architecture, Amazon's Working Backwards methodology, and Andrew Ng's analysis in The Batch (**#349**, **#359**). None of these parties is affiliated with or has endorsed this model.