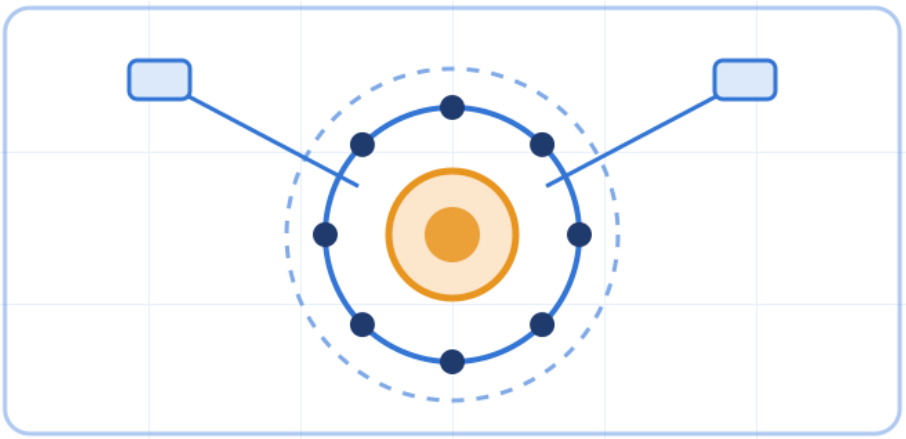
Scrum Manager
Uncovering better ways

State of the art

Harness Engineering

Reference map of current professional knowledge



Updated June 2026

Scrum Manager® · Skill Arena

About this document

This document is a curated map of the knowledge, practices and models in professional use as of its publication date for adopting AI agents with professional rigour: wrapping a model in guides and sensors that turn its output into reliable, verifiable, useful work, whatever the discipline. For each entry, the map places it in context, indicates its degree of adoption in current professional practice and points to where to find reference information.

Use it as an observatory and reference point to check whether the professional knowledge you rely on is aligned with current practice and at the leading edge.

It is complemented by an in-depth companion guide that develops each concept and by the training and assessment platform in Skill Arena. In the Harness Engineering area you can test your level of knowledge and, if you pass, obtain a diploma that provides curricular accreditation of professional competence and currency in this area.

State of knowledge

Knowledge about Harness Engineering evolves extremely fast: the discipline barely existed a year and a half ago, and its vocabulary has stabilised over the course of 2026 around a handful of reference publications. Its conceptual core (the framework of guides and sensors, the principles of context and verification) is already settled, but the frontier patterns—substantive verification, multi-agent architectures, reusable templates—are redefined almost every month. Throughout the document, the labels (ESTABLISHED, CONSOLIDATING, EMERGING) help identify the maturity of each concept:

ESTABLISHED settled consensus; knowledge taken as required.

CONSOLIDATING gaining adoption fast; not yet universal but already relevant.

EMERGING recent frontier; high relevance and high volatility.

How to read this map

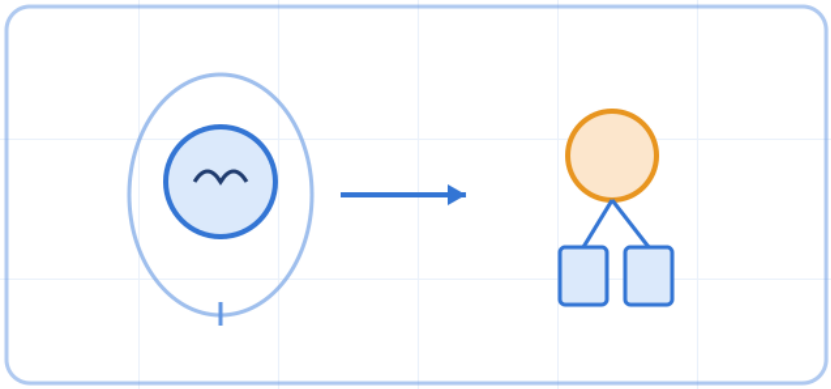
The map groups the knowledge into six blocks, running from the paradigm shift to the practice of assembly, the frontier patterns and the professional judgement that is never delegated. Each pointer states what it is, why it matters now and where to go to dig deeper. The labels keep their standard

meaning: here, an already settled conceptual core coexists with patterns still being defined, so you will see all three states represented.

The scope is deliberately cross-disciplinary. Harness Engineering was born in software development, but its framework applies to any professional work assisted by agents: analysis, consulting, research, legal advice, journalism, writing, management or teaching. The examples alternate between code and knowledge work so as not to tie the concept to a single discipline.

Block A — The paradigm shift: Agent = Model + Harness

Before building anything, it is worth understanding why a frontier model, however brilliant, is not enough to deliver reliable professional work—and what exactly the thing that wraps it is.



Without an operating environment, the model's intelligence cannot act; the harness is the body that turns reasoning into verifiable work.

Intelligence versus agency

ESTABLISHED

Intelligence is a trained property of the model: reasoning, connecting concepts, synthesising, planning. Agency is the capacity to act on the world: reading documents, executing actions, modifying artefacts, staying coherent across sessions and producing verifiable deliverables. A model without an operating environment reasons and opines, but cannot check anything.

Why it is here now. It is the founding distinction of the discipline: it separates what the vendor trains from what you build around it, and explains why two professionals with the same model deliver results of very different quality.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Mollick, One Useful Thing

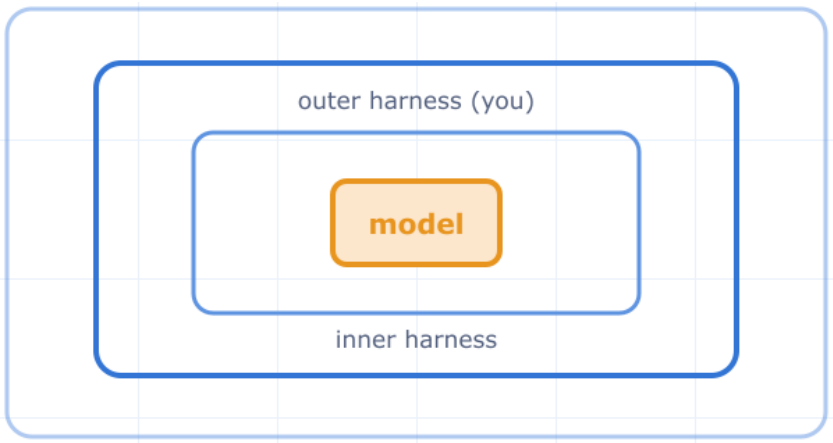
The Agent = Model + Harness equation

ESTABLISHED

The canonical formula, popularised by Birgitta Böckeler (Thoughtworks) and adopted across the industry in 2026, defines the harness as everything that is not the model: the instructions it reads on startup, the documents within its reach, the tools it can invoke, the permission policies, the orchestration loops, the persistent memory and the verification sensors. The model is a commodity; the harness is the competitive advantage.

Why it is here now. It provides the field's shared vocabulary. Without a shared mental model of what the harness is and is not, the practices that come afterwards have nothing to anchor to.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Thoughtworks · What is harness engineering



Three concentric circles: the model trained by the lab, the vendor's inner harness, and the outer harness you build yourself.

Inner harness versus outer harness

ESTABLISHED

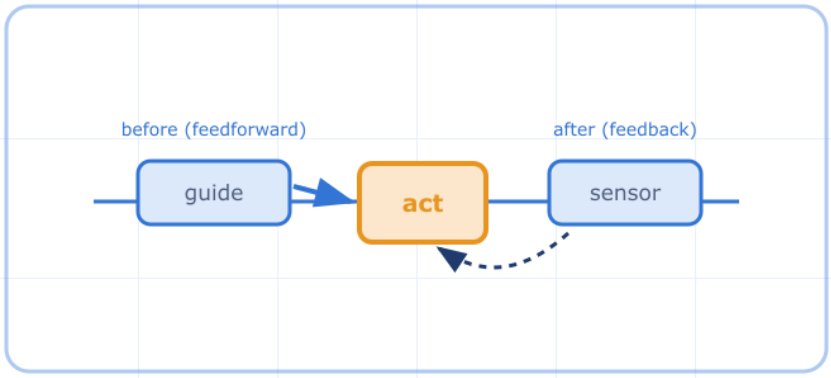
Böckeler draws the harness as concentric circles. The inner harness is built by the vendor around the model (interface, memory, search, orchestration loop, tool executors, permission system): you use it, you do not build it. The outer harness is what you build on top of it: instructions, documents, skills, connectors, verifications and templates.

Why it is here now. It draws the boundary of practical responsibility. The bulk of the value a professional extracts from agents comes from doing the outer harness well, not from understanding the inner one; knowing where one ends and the other begins prevents misdirected effort.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Anthropic · Effective harnesses for long-running agents

Block B — The two pillars: guides and sensors

Böckeler's framework breaks the outer harness down into two types of control that act at different moments of the agent's cycle. Understanding both, and why both are needed, is the operational heart of the discipline.



Guides steer before the act (feedforward); sensors observe afterwards and enable self-correction (feedback).

Guides: controls before the act

ESTABLISHED

Guides (feedforward controls) anticipate the agent's behaviour and try to steer it before it acts, increasing the probability of getting it right the first time. They are what the agent reads on startup and carries with it while it operates: instruction files, method or architecture documentation, glossaries, style guides, structural templates, reference corpora and reusable skills.

Why it is here now. It is one half of the canonical framework. An agent without guides improvises conventions in every session; with good guides, it starts from a known state and works aligned with your rules.

Where to look. Böckeler, Harness engineering (martinfowler.com) · AGENTS.md · open standard

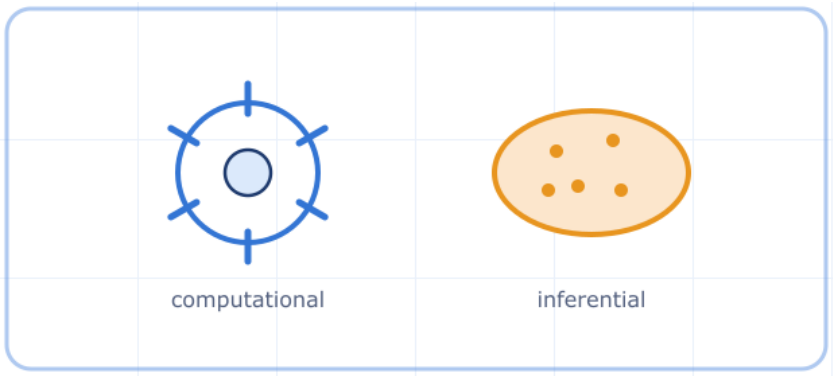
Sensors: controls after the act

ESTABLISHED

Sensors (feedback controls) observe the agent after it has produced something and allow it to self-correct. Good sensors return actionable signals, optimised for consumption by another AI: instead of a cryptic failure, they explain which claim fails, what was expected, what was observed and how to fix it. They include verification suites, checklists, fact-checking, citation verification, structural analysis and adversarial reviewers.

Why it is here now. It is the other half of the framework. Without sensors, the agent encodes rules but never knows whether its output complies with them, and repeats mistakes that are structurally valid but functionally broken.

Where to look. Böckeler, Maintainability sensors for coding agents · Böckeler, Harness engineering (martinfowler.com)



A computational control is deterministic, fast and does not lie; an inferential one is powerful for semantics but expensive and probabilistic.

Computational versus inferential

ESTABLISHED

Any control—guide or sensor—can be executed in two ways. A computational control is run by a deterministic tool (a linter, a test, a script, a search): fast, cheap, reliable within its scope. An inferential control is run by another model (LLM as judge): powerful for semantic judgements, but slow, expensive and probabilistic. The pragmatic rule: whenever something can be resolved with a computational control, do not use an inferential one.

Why it is here now. It separates the mature harness from the immature one. The immature harness rests everything on inferential controls because they look sophisticated, and ends up costing a fortune without delivering reliability.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Böckeler, Maintainability sensors for coding agents

The three dimensions of regulation

CONSOLIDATING

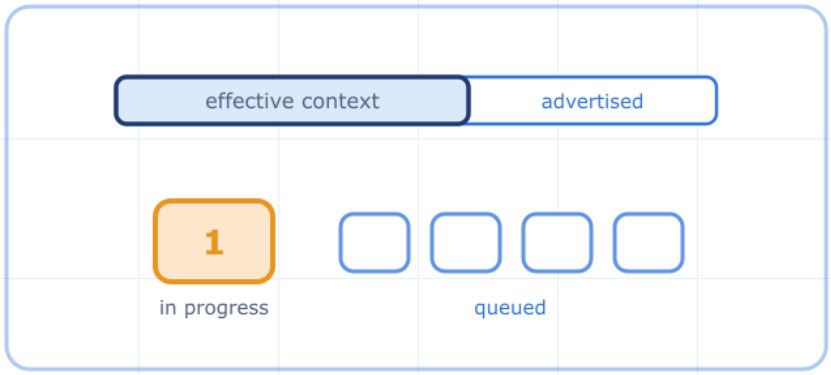
The harness acts as a governor regulating the output towards a desired state, along three dimensions: structural quality (readability, coherence, maintainability), professional form of the craft (methodological rigour, transparency about sources, handling of uncertainty, compliance) and functional behaviour (does it actually solve the problem?). The first is the most mature; the third is the open frontier. Böckeler has begun to operationalise the maintainability dimension with concrete catalogues of sensors.

Why it is here now. It organises where each verification belongs and exposes the typical imbalance: almost all available tooling covers structural quality, while substantive correctness remains underserved.

Where to look. Böckeler, Maintainability sensors for coding agents · Böckeler, Harness engineering (martinfowler.com)

Block C — Principles that govern the harness

Beneath the concrete practices lies a handful of principles that come up again and again. They are rooted in the real limits of the models and in cybernetics, and they explain why the harness is designed the way it is.



The effective context is considerably smaller than the advertised one; working on a single task at a time protects that budget.

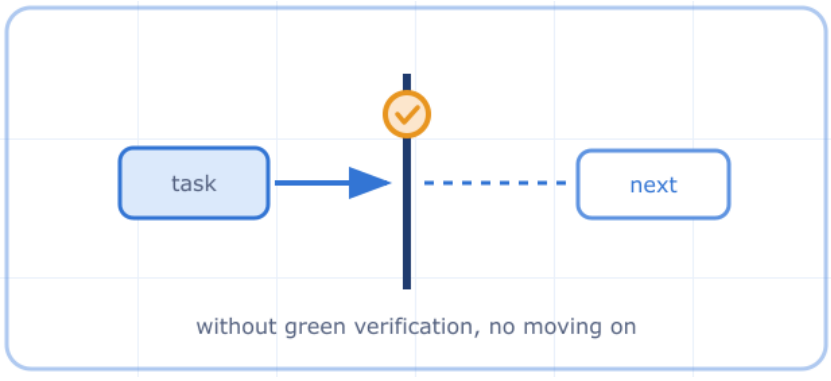
WIP = 1 and the context budget

ESTABLISHED

Work-in-progress equal to one: each session works on a single task or deliverable and does not move on until the current one is verified. The reason is physical: a model's usable context is much smaller than advertised—NVIDIA's RULER benchmark places the effective context typically at around half to two thirds of the window—and the Lost in the Middle effect degrades the processing of information placed in the middle of long contexts.

Why it is here now. It is the operating rule with the greatest empirical impact on session success rates. Saturating the context with several tasks at once is the most common cause of agents that lurch back and forth.

Where to look. Liu et al., Lost in the Middle (arXiv) · NVIDIA RULER (arXiv)



Pass-state gating forbids moving on to the next deliverable until the current one has passed an explicit set of verifications.

Pass-state gating and the verification gap

ESTABLISHED

There is a dangerous distance between the feeling of confidence a well-presented result produces and its actual correctness. Agents are structural optimists: they declare complete what is merely

plausible. The golden rule is that “done” is not an adjective but an outcome of verifications. Pass-state gating embodies that rule: the harness forbids moving forward until the current deliverable passes an explicit set of checks.

Why it is here now. It is the direct defence against the most expensive failure of agentic work: accepting as finished something that is not, and discovering it in review or, worse, in production.

Where to look. Anthropic · Effective harnesses for long-running agents · Böckeler, Harness engineering (martinfowler.com)

The cold-start test and environment legibility

CONSOLIDATING

The most revealing test of a harness's quality: open a fresh session, with no history, and check whether the agent, reading only the project files, can identify what the system is, what the progress is and what remains to be done. If it manages in a few minutes, the harness is solid; if not, the knowledge lives in a human's head and not in the project. Its corollary: what is not in the project does not exist for the agent. Tacit knowledge must be encoded in explicit artefacts.

Why it is here now. It turns a fuzzy quality (“a good harness”) into an actionable, repeatable test, and guides what to document: not more, but what a newcomer would need.

Where to look. Lopopolo (OpenAI) · Harness engineering · Anthropic · Effective harnesses for long-running agents

Ashby's Law and variety reduction

CONSOLIDATING

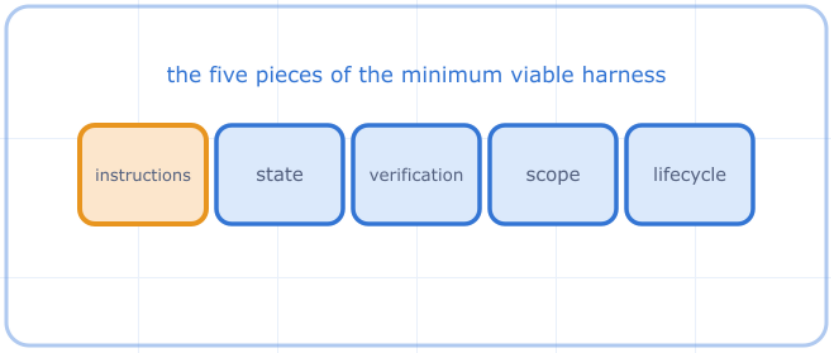
The Law of Requisite Variety (Ashby, 1956) states that a regulator needs at least as much variety as the system it governs. Since a model can produce almost anything, a harness that matched it would be impossible. The practical way out is to invert the principle: reduce the system's variety. By tying the agent to a concrete topology—a known architecture, a fixed stack, a closed template, a defined process, a bounded corpus—the space of what it can produce shrinks, and a comprehensive harness becomes feasible.

Why it is here now. It is the theoretical justification for harness templates and for the whole discipline of scoping. It explains why “restricting” does not impoverish the result but makes it governable.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Law of Requisite Variety (Ashby)

Block D — Preparing the environment

Here the discipline becomes assembly practice. Before touching any specific tool there is a foundation to prepare: the workspace, the materials, the minimum pieces of the harness and the pattern of memory across sessions.



With five pieces—instructions, state, verification, scope and lifecycle—you are already doing Harness Engineering.

Pre-flight: workspace, confidentiality and materials

ESTABLISHED

Before starting, three things. A dedicated, version-controlled workspace, not a loose conversation. A minimum set of confidentiality rules: never paste credentials or upload sensitive data to consumer versions, anonymise whenever possible, document each project's confidentiality level. And bounded, curated materials: a trustworthy reference corpus produces verifiable work, whereas an agent that “searches for whatever it can find” produces plausible but unverifiable work.

Why it is here now. These are the cheapest mistakes to prevent and the most expensive to drag along. Most substantive failures of agentic work are incubated in a poorly prepared environment, not in the model.

Where to look. Lopopolo (OpenAI) · Harness engineering · Anthropic · Effective context engineering

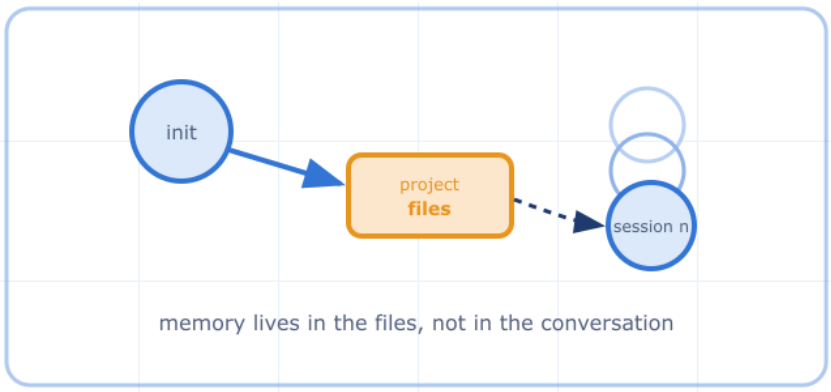
The minimum viable harness and AGENTS.md

ESTABLISHED

A minimum viable outer harness has five pieces: instructions (the master guide the agent reads on startup), state (persistent memory across sessions: deliverables list, decision and progress log, version control), verification (the sensors), scope (rules restricting what it does in each session) and lifecycle (bootstrap, persistence, cleanup). Across much of the ecosystem, the instruction file has standardised as AGENTS.md, an open format maintained under the Linux Foundation.

Why it is here now. It provides a concrete, complete starting point. If the five pieces are in place, you are already doing Harness Engineering; everything else—hooks, skills, subagents—amplifies this foundation but does not replace it.

Where to look. AGENTS.md · open standard · Lopopolo (OpenAI) · Harness engineering



One agent initialises the environment once; another advances session after session, reading and updating the project files.

Initializer + working agent: memory in files

CONSOLIDATING

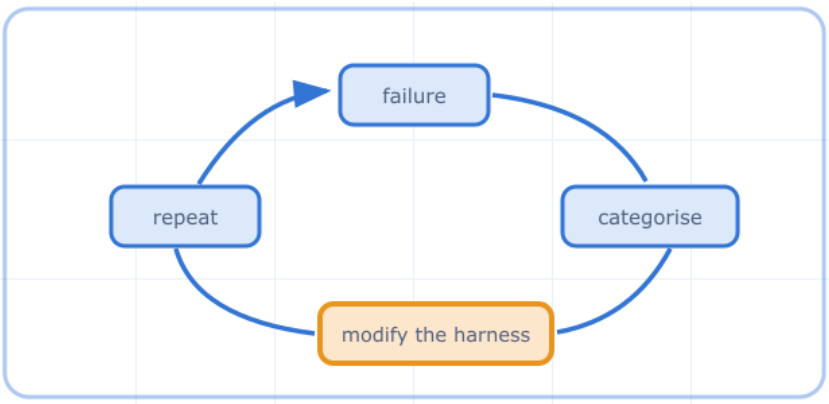
Anthropic's canonical pattern for long tasks separates two roles. An initializer runs only once: it expands the brief into a structured list of deliverables with acceptance criteria, sets up the workspace and prepares the verifications. A working agent wakes up session after session: it re-grounds itself by reading the files, reads the progress log, picks a pending entry, produces and verifies it, updates the progress and closes. The key: persistent memory lives in the project files, not in the conversation.

Why it is here now. It solves the central problem of tasks that do not fit in a single context window: how not to lose what was decided between sessions. Anthropic has also published patterns and a reference repository for implementing it.

Where to look. Anthropic · Effective harnesses for long-running agents · Anthropic · cwc-long-running-agents (GitHub)

Block E — Frontier patterns

Where the discipline is still being written. The harness improvement loop is already common practice; substantive verification and multi-agent architectures are ground in motion, with high relevance and high volatility.



When something fails, the error is categorised and the harness is modified; scolding the model does not change its behaviour.

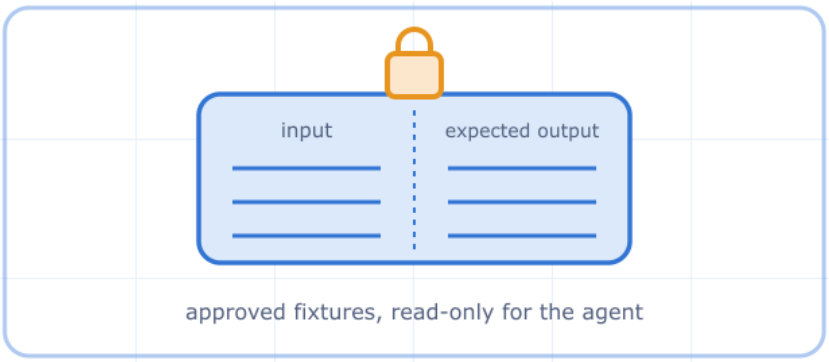
The steering loop and the harness journal

CONSOLIDATING

The steering loop is the human practice of iterating on the harness when something fails, instead of iterating on the prompt or scolding the model. The procedure: do not scold; categorise the error (is a guide missing? a sensor? should the scope be narrowed? the legibility improved?), modify the harness accordingly, repeat the task and check. It works because the model does not learn from scolding—it does not update its weights—but it does change behaviour when its instructions, its error signals and its materials change. A harness journal records each change and its outcome.

Why it is here now. It is the mechanism that turns working with agents into a cumulative discipline rather than a succession of frustrations. Every well-harvested failure improves the system permanently.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Anthropic · Scaling Managed Agents



(Input, expected output) pairs marked read-only: the agent must satisfy them; it cannot rewrite them in order to "pass".

Behaviour harness, approved fixtures and adversarial reviewers

EMERGING

Verifying that a deliverable actually does what it was supposed to do—not just that it is well formed—is the field's open front. Verifications generated by the agent itself tend to be circular: they validate the misunderstanding if there was one. Promising patterns include approved fixtures (input/output or minimal question/answer pairs that the human fixes as read-only and the agent cannot modify), property-based testing, adversarial reviewers (an agent whose only job is to object) and LLM-as-judge in a fresh context, always as a second line behind computational controls.

Why it is here now. It is the discipline's unsolved problem and, at the same time, the one with the greatest consequences: the more critical substantive correctness is, the less autonomy the agent should be given and the greater the human role must be. The patterns are redefined almost every month.

Where to look. Anthropic · Harness Design for Long-Running App Development · Böckeler, Maintainability sensors for coding agents

Multi-agent architectures and agentic cleanup

EMERGING

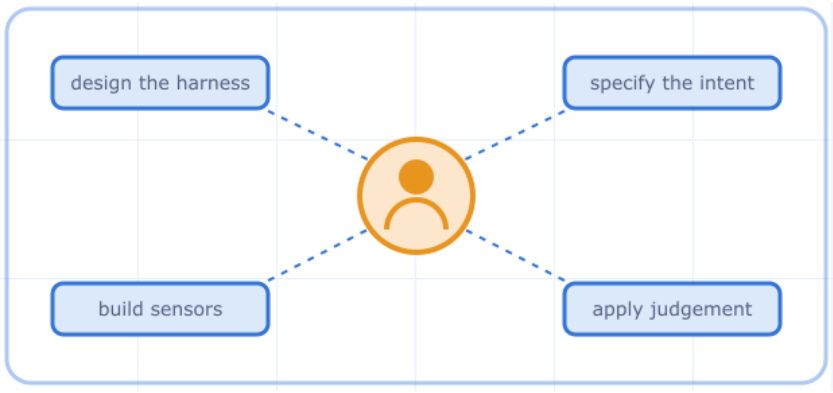
The orchestration frontier. Anthropic has documented three-role architectures (planner, generator, evaluator), inspired by generative adversarial networks, where an evaluator with fresh context and no write permissions judges the generator's work in successive cycles; and the Brain/Hands/Session separation, which decouples the model and its loop from the ephemeral sandboxes and the session log. In parallel, agentic garbage collection uses background agents to clean up project drift: dead code, stale sources, orphaned dependencies, terminological inconsistencies.

Why it is here now. It marks where the harness is heading for long, autonomous tasks. It is powerful but volatile: the specific architectures and tools change fast, and they are worth following without adopting them out of fashion.

Where to look. Anthropic · Harness Design for Long-Running App Development · Anthropic · Scaling Managed Agents

Block F — Professional judgement and the human role

A well-designed harness does not remove the human: it concentrates them where they are worth most. This block gathers what is never delegated—measuring, recognising antipatterns, deciding when not to use an agent, and sustaining professional judgement over the deliverable.



The human stops writing every line and instead designs the environment, specifies the intent, builds the feedback loops and applies judgement where it matters.

Measuring the harness, avoiding antipatterns and sustaining judgement

CONSOLIDATING

A harness is governed with indicators: rebuild cost (time from an empty session to useful work), first-pass completion rate, recipient objections, sensor coverage, harness improvement latency. Against them, the recurring antipatterns: over-specified instructions the model stops reading, endless exploration, premature confidence, bypassing out of desperation, subagent proliferation and blind faith in the inferential reviewer. And the limit of the method itself: there are tasks where the right call is not to use an agent—when the context exists only in your head, when the responsibility cannot be transferred, when you would not know how to evaluate the answer, when confidentiality is paramount. The human designs the harness, specifies the intent, builds the feedback loops and applies judgement where the machine must not decide alone.

Why it is here now. It closes the map with the layer that protects the validity of everything before it: without measurement there is no improvement, without recognising antipatterns they repeat, and without human judgement an impeccable harness can sign off work that nobody has genuinely thought through.

Where to look. Böckeler, Harness engineering (martinfowler.com) · Mollick, One Useful Thing

What to watch in the next revision

Signals of change the editorial team anticipates for the next edition of this map:

- Consolidation of substantive verification: if approved fixtures, property-based testing or adversarial reviewers mature into standard practice, the behaviour harness would move from EMERGING to CONSOLIDATING.
- Reusable harness templates: the appearance of public bundles by topology (strategy report, legal memorandum, microservice, academic paper) could justify a pointer of its own if the movement takes hold.
- Stabilisation of multi-agent architectures: planner/generator/evaluator and the Brain/Hands/Session separation are still being redefined; it is worth reviewing whether any of them settles as the reference.
- Evolution of the instruction standard: the adoption of AGENTS.md and its governance under the Linux Foundation may change the practical recommendations on instruction files.
- Obsolescence through model improvement: every harness component encodes an assumption about what the model cannot do on its own; as models improve, some pieces become dead weight and should be retired.

A note on the references

The links included were available and verified as of the update date. Given the pace of change in this area, some may change; each revision of the map also updates its references.