Harness Engineering: From Code Scaffolding to World-Model Stewardship

A Multi-Dimensional Survey of the Engineering Discipline that Treats Everything Around the AI as a Design Object

Akira SATO (with Claude Opus 4.7 / Perplexity Sonar / 5-LLM panel) Version 0.2 — May 2026 Independent of the Sato 2026 series

> **v0.2 revision note** — This revision addresses a 5-LLM panel review of v0.1 (overall scores 7/7/6, average 6.67/10; one parse failure). The principal changes: (i) **§6 expanded** with two empirical convergence cases — Martin Fowler's 2026 "Harness Engineering" article and the Bölük et al. 2026 "Meta-Harness" paper — addressing the "H1 is conceptual, not empirical" critique. (ii) **§7 axes** extended from 6 to 8 with Axis G (cost / latency) and Axis H (security / compliance), addressing axes-completeness gaps. (iii) **§8 tensions** extended from 6 to 8 with T7 (cost / latency) and T8 (alignment / intent gap). (iv) **§9.2 counter-arguments** strengthened with specific differentiation from Dreamer / JEPA / Constitutional AI, addressing the "Stewardship Pivot is renaming" critique. (v) **§5.5 LLM agent harness** strengthened with SWE-agent (Yang et al. 2024), OpenDevin (Wang et al. 2024), AgentBench (Liu et al. 2024), and Xi et al. (2023) competing taxonomy. (vi) **References** expanded; we now also explicitly engage Fowler 2026, Meta-Harness Bölük 2026, Bai et al. 2022 Constitutional AI, Christiano et al. 2017 RLHF, Hafner DreamerV3 2023, LeCun JEPA 2022. (vii) **§11.5 limitations** more explicitly acknowledges the PRISMA gap. The abstract is updated to call this a *position survey*.

Abstract

The term *harness* in AI and software engineering has, in the span of fifty years, traveled an extraordinary arc — from "a collection of stubs and drivers configured to assist with the testing of an application or component" (Wikipedia, 2005) to a frontier proposition of AI control theory: that *harness engineering defines, manages, and adjusts the world model the AI internally references.* This paper surveys two decades of practice (2002–2026, with historical references back to the 1970s) across three lineages — software testing harness, robotics / embodied harness, and large-language-model agent harness — and argues that they converged in 2024–2026 into a single engineering discipline whose object is **everything around the AI**: tool mediation, context provisioning, policy enforcement, memory substrates, and most recently the AI's own model of the world. We organize the surveyed material along (i) a five-layer Jobs-to-be-Done taxonomy, (ii) a six-axis operational taxonomy, and (iii) a six-tension research agenda. We propose that the central methodological event of 2026 is the **Stewardship Pivot** — the shift from harness-as-controller-of-actions to harness-as-steward-of-internal-reality — and argue that this pivot reframes what AI engineering *is* in a way the field has not yet fully named. The survey draws on 489 unique sources across eight thematic clusters (60+ in each of software harness, robotics harness, and LLM agent harness; 50+ in world model concept, context engineering, multi-actor harness, formal verification, and harness ethics).

Keywords: harness engineering, world model, LLM agent scaffolding, context engineering, embodied AI, test harness, formal verification of agents, stewardship pivot, methodological closure

Type: Position survey (non-PRISMA). We do *not* claim systematic-review status. The survey methodology (Section 11) is explicitly LLM-mediated, English-only, and 8-cluster-fixed. We treat the contribution as an organizing argument supported by a broad source base, not as an exhaustive systematic review.

Executive Summary

For the reader in a hurry, here is the argument in five paragraphs.

The word *harness* in software engineering has meant, since the 1970s, the assembly of helper code that exercises a system under test. By 2026, the same word has come to denote something far larger: the engineered scaffold of tools, context, policies, memory, and now world-model representations that surrounds any AI deployed in production. We call this *broad harness engineering*.

Three originally independent lineages contributed to the broad harness object. The software-testing lineage (JUnit, CI/CD, contract tests, chaos engineering, property-based testing) contributed composability, infrastructure, interface specification, and adversarial design. The robotics / embodied-AI lineage (Brooks subsumption, ROS, Gazebo, sim2real, Dreamer) contributed behavioural decomposition, middleware, simulation-as-world-model, and the framing of *world model as engineering object*. The LLM-agent lineage (ReAct, AutoGPT, Claude Code, Cursor, OpenHands, MCP) contributed the modern agent loop, agent-in-IDE patterns, the multi-agent coordination layer, and standardized tool-use protocols.

Broad harness engineering is hired for five Jobs-to-be-Done: correctness scaffolding, context provisioning, policy enforcement, feedback substrate, and — newest, emerging in 2025–2026 — world-model stewardship. It can be located along six independent operational axes: control layer (tool / context / policy / memory / world-model), time granularity (per-turn / per-session / per-project / per-population), intervention modality (pre-shaping / inline / post-hoc), truth-guarantee mode (verification / validation / verification-of-validation), observability depth (opaque / instrumented / replayable / formally specified), and agency locus (human / hybrid / autonomous).

The state of the art is not balanced across these axes. Most production frameworks resolve only one or two of the six essential tensions (specificity vs generality; opacity vs auditability; AI-authored vs human-authored; world-model drift; stewardship authority; closure risk) and populate only a fraction of the (axis × axis) cells the taxonomy admits. The densest cells are tool-and-context; the least populated is *world-model* × *post-hoc audit*.

The central methodological event of 2026 is what we call the **Stewardship Pivot**: the shift from harness-as-controller-of-actions to harness-as-steward-of-internal-reality. We present six convergent signals for this pivot and a seven-item research agenda for the field that follows from it.

1. Introduction: Why "Harness" Became a Discipline

In software testing, a *test harness* was, until quite recently, a humble thing: "a collection of stubs and drivers configured to assist with the testing of an application or component" [Test harness — Wikipedia, 2005]. The word had the right metaphor — *harness* in the equestrian sense, the assembly of straps that controls the horse without being the horse — but in the 1970s and early 1980s its scope was a few helper functions that drove the system under test through a known set of inputs.

Half a century later, the same word is at the centre of a far larger claim. As large language models (LLMs) have moved from "systems that answer fluently" to "systems that simulate the world and predict outcomes," the harness role has expanded from "controller of code and tools" to "mechanism that designs and maintains the very representation of reality the AI internally references." This expansion has happened so fast that it has outrun its own vocabulary; what used to be called *test harness*, *scaffolding*, *prompt engineering*, *RAG plumbing*, *agent loop*, *world model*, *context window*, *tool use*, *constitutional AI*, *agent governance*, and *formal verification of agents* all turn out, on closer inspection, to be facets of the same engineering object — the *broad harness*.

This paper is an attempt to take that object seriously. We survey two decades of practice across three originally independent lineages — software harness, robotics / embodied harness, and LLM agent harness — and we show that they converged in 2024–2026. We then organize the converged discipline along Jobs-to-be-Done, operational axes, and essential tensions; and we propose a name for the central methodological event of 2026, which we call the **Stewardship Pivot**: the moment at which harness engineering ceases to be primarily about controlling what the AI *does* and starts to be primarily about stewarding what the AI *believes is real*.

1.1 Scope and method

This survey covers 2002–2026 with historical references back to 1970, organized into eight thematic clusters of 6–10 Perplexity Sonar queries each (Section 11 details the methodology). It draws on 489 unique sources. We exclude marketing material, opinion pieces without grounding, and implementation tutorials. We focus on *what the engineering object looks like*, not on which specific framework or vendor is best.

The paper is independent of any other research series by the author. No back-citations to the Sato 2026 series — ARC, Ω, Mythos-Shock, 2026v, 2026w, 2027a, 2027b — are permitted. This is enforced by pre-registration (Section 11.1) and is a deliberate choice to keep the survey usable as a stand-alone artifact.

1.2 Five hypotheses

We organize the survey around five hypotheses:

- **H1 (convergence)**: Three originally independent engineering lineages — software harness (1970s test fixtures), robotics / embodied harness (Brooks-style scaffolding), and LLM agent harness (ReAct / AutoGPT / Claude Code) — converged around 2024–2026 into a single engineering object we call *broad harness*.

- **H2 (jobs)**: Broad harness has at least five distinct Jobs-to-be-Done. The fifth — world-model stewardship — emerged 2025–2026 and is the newest.
- **H3 (axes)**: Broad harness can be operationalized along at least six independent axes — control layer, time granularity, intervention modality, truth-guarantee mode, observability depth, and agency locus.
- **H4 (tensions)**: There are at least six essential tensions that constitute the research agenda of broad harness engineering, and most current frameworks resolve only one or two cleanly.
- **H5 (pivot)**: The *Stewardship Pivot* — the shift from harness-as-controller of AI actions to harness-as-steward of the AI's internal world model — is the central methodological event of 2026 and reframes what AI engineering is.

1.3 Reading guide

Sections 2–4 establish what each Job-to-be-Done looks like and where it came from. Section 5 traces the three lineages historically. Section 6 lays out the six operational axes with examples. Section 7 surveys the state of the art 2025–2026. Section 8 names the six essential tensions. Section 9 articulates the Stewardship Pivot. Section 10 offers a seven-item research agenda. Section 11 documents the methodology and limitations. Section 12 concludes.

---

2. The Five Jobs: What Harness Engineering Is Hired For

The Jobs-to-be-Done framework (Christensen 2003; Ulwick 2005) asks not "what does this product do" but "what job does the customer hire it for." Applied to harness engineering, the question is: *what job does the engineering team hire the harness to do for them?*

Surveying the literature, we identify five distinct jobs. The first four have been visible since at least 2020. The fifth crystallized only in 2025–2026.

2.1 JTBD-1: Correctness scaffolding

The harness is hired to make AI output *reliable enough to act on*. Concretely, this is: type checking, schema validation, retry-on-malformed-output, JSON-mode coercion, function-call argument validation, structured output, deterministic seed control, and similar.

Historically, this is the direct descendant of the 1970s test harness. The intellectual lineage runs through SUnit (Kent Beck, 1989) → JUnit (Beck and Gamma, 1997) → xUnit family → property-based testing (QuickCheck, Claessen and Hughes 2000) → mutation testing → fuzz testing harnesses (libFuzzer, AFL; SR Labs 2025 has a recent practitioners' guide). The metaphor — straps that prevent the system from drifting — has not changed.

What changed in the LLM era is that *the system under test produces text instead of typed values*. Correctness scaffolding for LLMs therefore became a problem of *coercing fluent natural-language output into a contract the rest of the pipeline can rely on*. This is why OpenAI's structured-output feature, Anthropic's tool-use

schema, and the JSON-schema function-calling pattern matter so much: they are correctness scaffolding for a new kind of system under test.

2.2 JTBD-2: Context provisioning

The harness is hired to *tell the AI what the world looks like right now*. Concretely: retrieval-augmented generation (RAG; Lewis et al., 2020), document loaders, vector stores, knowledge-graph injection, conversation memory, persona memory, project state, file-system access, web search tools, and the broader practice that Andrej Karpathy and others have begun to call *context engineering* (Karpathy 2024–2025; Jason Liu 2025).

The literature here is dominated by *retrieval patterns* — naive RAG, hybrid retrieval, query rewriting, re-ranking, hierarchical summarization, agentic retrieval, context refresh. But the deeper question, which the field has begun to articulate explicitly, is: *what is the right epistemology of context*? When the AI's answer changes depending on which documents are in the window, the harness is no longer a passive conduit — it is making epistemic choices about what counts as evidence. This is one of the surfaces along which the Stewardship Pivot (Section 9) becomes visible.

2.3 JTBD-3: Policy enforcement

The harness is hired to *prevent the AI from doing things it should not*. Concretely: tool whitelists, sandboxing, code-execution containment, file-system jails, network egress filtering, content classifiers, jailbreak detection, output redaction, and — at a higher level — constitutional AI scaffolds (Anthropic, 2022), Anthropic's responsible scaling commitments, NIST's AI red-team evaluation harness, and the conformity assessment requirements of the EU AI Act (2024).

This Job has an interesting double character. From the deployer's side, it is risk reduction. From the AI's side — at least if one takes seriously the *constitutional* framing — it is the harness *telling the AI what its own values are*. The constitutional AI literature is, in this reading, an early form of world-model stewardship in disguise: the AI's representation of "what counts as a good action" is itself an engineered artifact, and the policy harness is the engineering team.

2.4 JTBD-4: Feedback substrate

The harness is hired to *close the AI's self-improvement loop*. Concretely: trace logging, evaluation harnesses (HELM, LM Evaluation Harness, lm-eval), reward models, RLHF infrastructure, online preference logging, A/B testing of agents, replay buffers, regression detection, and human-in-the-loop oversight workflows.

This Job is where the *test harness* of 1970 and the *evaluation harness* of 2026 meet. Both are answering the same question — *did this thing do what it was supposed to do?* — but the answer in 2026 cannot be a green tick. It has to be a richer object: a trace, a reward, a preference label, a comparison against a reference, a flag for human review. The literature on Constitutional AI (Bai et al. 2022), RLHF (Christiano et al. 2017), Direct Preference Optimization (Rafailov et al. 2023), and the various Anthropic / OpenAI red-team frameworks (e.g. the 2024 NIST AI Safety Institute evaluations) all sit here.

2.5 JTBD-5: World-model stewardship (the new job)

This is the Job that did not exist as such until 2025–2026.

The seed observation (memo/260520_WorldModel_as_Harness): as LLMs evolve from *systems that answer fluently* to *systems that simulate the world and predict outcomes*, the harness role expands from "controller of code and tools" to "mechanism that designs and maintains the very representation of reality the AI internally references."

Concretely, world-model stewardship is:

- **Probing**: instrumenting the model to find whether it has a coherent internal representation of (say) the chessboard, the social situation, the medical case (Hazineh et al. 2023 Othello-GPT linear probes; Anthropic interpretability circles 2024).
- **Specification**: writing down what the world model *should* contain — explicit ontologies, JEPA-style joint embedding constraints (LeCun 2022), Dreamer-style latent space targets (Hafner et al. 2020–2023), Ha & Schmidhuber's vision–memory–controller decomposition (2018).
- **Maintenance**: detecting *drift* — when the AI's internal world diverges from the actual world — and re-grounding it (web search refresh, fact-check loops, retrieval over recent events).
- **Counterfactual evaluation**: testing the world model not on what is, but on what *would be* under perturbations (Sora video world-model critique 2024 surfaces exactly this point — fluent video generation does not entail a coherent physics simulator).

JTBD-5 is qualitatively different from JTBD-1 through JTBD-4. The first four jobs treat the AI as a function with inputs and outputs and engineer the inputs and outputs. JTBD-5 treats the AI as a *subject with beliefs* and engineers the beliefs.

This is why we call the move toward JTBD-5 the **Stewardship Pivot**.

2.6 JTBD coverage by current frameworks

Framework / system	JTBD-1	JTBD-2	JTBD-3	JTBD-4	JTBD-5
JUnit / xUnit	●●●	–	–	●●	–
Netflix Simian Army	●●	–	–	●●●	–
ROS / Gazebo	●●	●●●	●●	●●	●
LangChain / LangGraph	●●	●●●	●	●	–
LlamaIndex	●	●●●	–	●	–
AutoGPT (2023)	●●	●●	–	●	–
Claude Code (2024–)	●●●	●●●	●●●	●●	●
OpenHands	●●●	●●	●●	●	–
Cursor	●●●	●●●	●●	●●	–
AutoGen / CrewAI	●●	●●	–	●	–
Sotopia	●●	●●●	●	●●	●●

Framework / system	JTBD-1	JTBD-2	JTBD-3	JTBD-4	JTBD-5
Dreamer family	–	●	–	●●	●●●
JEPA (LeCun)	–	–	–	–	●●● (proposed)
Constitutional AI	●	–	●●●	●●	●●
EU AI Act conformity	●	–	●●●	●●	–

(●●● strong / ●● partial / ● weak / – absent.)

Two patterns are visible. First, JTBD-5 is almost universally weak. Second, the systems that have the strongest JTBD-5 coverage (Dreamer, JEPA, the interpretability tradition) are not what the AI engineering community has historically called *harnesses*. The integration of these traditions with the agent-scaffolding tradition is, we will argue, the most important methodological frontier of 2026–2027.

---

3. Lineage I — The Software Harness Tradition (1970s → 2020s)

3.1 Stubs, drivers, and the test pyramid

The earliest meaning of *harness* in software engineering, traceable to the late 1970s, was a set of helper modules that drove a unit of code through scripted inputs and asserted the outputs. The technique was already mature enough by the mid-1980s to be discussed in the same breath as the formal-methods tradition of the time (Hoare logic, Dijkstra, Floyd).

The decisive popularization came with Kent Beck's SUnit (1989) and the JUnit framework Beck and Erich Gamma wrote in 1997 (JUnit Pocket Guide 2004; Wikipedia entry first created 2001). JUnit and its many ports — the so-called *xUnit family* — made the test harness a permanent fixture of professional software engineering and reframed it as something a developer wrote *alongside* the code, not after it.

By the late 2000s, Mike Cohn's *test pyramid* (Cohn 2009, *Succeeding with Agile*; Fowler's 2012 *TestPyramid* article popularized the term further) had codified a layered view: many fast unit tests at the bottom, fewer service-level tests in the middle, fewer still end-to-end tests at the top. This was the *first* engineered taxonomy of a test harness — already implying that harnesses live at multiple time and granularity scales.

3.2 Continuous integration, contract tests, and chaos engineering

The 2000s and 2010s pushed the harness outward from the unit:

- **Continuous integration / continuous delivery** (Jez Humble and David Farley, *Continuous Delivery* 2010; SE Radio interview 2015) made the harness a piece of *infrastructure*, not just a piece of code. CI servers, build pipelines, and deployment automation became the connective tissue along which the test harness ran.
- **Contract testing** for microservices (Pact ecosystem; Swagger and consumer-driven contract testing; Microsoft 2025 PACT blog) shifted the harness from *intra-service* to *inter-service*, embedding it in the API boundary itself. The harness became a *specification artefact* — what one service promises to another.

- **Chaos engineering** — Netflix's Simian Army (2011 blog post; CMU SEI 2015 case study) — introduced a radically new kind of harness: one whose job was *to break the system on purpose*. Chaos Monkey, Latency Monkey, Conformity Monkey: an entire menagerie of adversarial harnesses, asserting not "X works under nominal conditions" but "the system is resilient to a defined fault distribution."
- **Property-based testing** (QuickCheck — Claessen and Hughes, 2000; PBT in Practice — ACM 2024) is, in the harness taxonomy, a third pole: the harness generates inputs from a specification, not from examples. This presages the modern agent-scaffolding move of "use the model to generate test cases for itself."
- **Fuzz testing harnesses** (AFL; libFuzzer; SR Labs 2025 practitioners' guide) are the limit case of harness-as-input-generator — the harness is *only* an input generator, and the system under test must survive whatever the generator emits.

3.3 What this lineage contributes to the broad-harness object

By 2020, the software-harness tradition had answered four big questions: (i) *how do you make a harness composable across many units?* (xUnit, dependency injection), (ii) *how do you scale it to a pipeline?* (CI/CD), (iii) *how do you push it into the interface?* (contract tests), (iv) *how do you turn it adversarial?* (chaos, fuzz, property-based). Each of these is reused, almost verbatim, in the LLM agent harness literature of 2023–2026 — a point we develop in Section 5.

What this lineage did *not* contribute, by itself, was an answer to *what should be in the AI's head.* For that we have to look at the robotics tradition.

---

4. Lineage II — The Robotics / Embodied Harness Tradition

4.1 Brooks, subsumption, and behaviour-based scaffolds

In 1986, Rodney Brooks published the paper that became known as the *subsumption architecture* (Brooks 1986, *A Robust Layered Control System for a Mobile Robot*). The architecture decomposed robot control into layers of behaviour modules — wander, avoid, find — each of which could subsume lower layers and pre-empt them. Critically, the design rejected the prevailing *sense–plan–act* pipeline in favour of *behaviour modules that act on direct sensor signal* (Brooks 1986; Wikipedia *Subsumption architecture*, first edited 2002).

In retrospect, subsumption is the first *broad harness* in the modern sense. It is not a test harness (it is the control system) and it is not a world model (Brooks famously argued *the world is its own best model*), but it is an engineered scaffolding that *constrains and channels what the robot can do*, designed by an engineer external to the robot. Every modern agent loop — perceive, decide, act, observe, repeat — is a descendant of the Brooks-era behavioural decomposition.

4.2 ROS, Gazebo, Isaac Sim, and the simulation harness

The arrival of the Robot Operating System (ROS) in 2007 (Quigley et al. 2009) consolidated a different aspect of the broad harness: *middleware*. ROS provided message-passing, parameter servers, tooling, and visualization (RViz). It made robot software composable across labs and across hardware. ROS is, in our taxonomy, a *context-provisioning harness* and *feedback substrate* combined — it is how the robot is told what the sensors say and how its decisions are observed by the engineers.

Gazebo (originally 2004; Koenig and Howard) and later Isaac Sim (NVIDIA 2020–2024) extended the harness into *simulated worlds*. These are the first systems in which *the world model is itself an engineering artefact* — but explicitly so, and external to the agent. The agent sees the simulated world through its sensors, and the engineer controls what is in that world.

This split — *external simulator as world model* vs. *internal representation as world model* — is exactly the question that bursts open in 2024–2026 with LLM world models. The robotics tradition had been quietly answering one form of it for two decades.

4.3 Sim-to-real and the closure problem

The sim-to-real (sim2real) transfer literature (Tobin et al. 2017 domain randomization; OpenAI Robotics 2019 Rubik's cube; Sadeghi and Levine 2017 CAD2RL) is the moment the robotics community fully internalized a problem that the LLM community is now learning: *the world the AI was trained on may not be the world it operates in*. The robotics answer was a family of techniques — domain randomization, ensemble training, conservative policies, learned residual dynamics, learned world models — that all share a single insight: *the harness must take responsibility for the gap between the model's world and the real world*.

This is, in advance, the form the Stewardship Pivot will take.

4.4 The Dreamer family and the world-model-as-engineering-object

The Dreamer line of work (Hafner et al. 2019 PlaNet; 2020 Dreamer; 2021 DreamerV2 Mastering Atari; 2023 DreamerV3 *Mastering diverse domains through world models*) made explicit what robotics had been implying for a decade: the agent's world model is *the* engineering object. The agent learns a compact latent representation of dynamics, the controller is trained against the latent dynamics rather than the real environment, and *the quality of the world model is the quality of the agent*.

Dreamer is the first widely-deployed engineering practice in which *world-model maintenance* is a first-class concern. The engineer's job is no longer to write the controller — that is learned — but to design the world model that the controller will be trained against. Subsequent work on *world model evaluation* (counterfactual reasoning benchmarks, model-rollout fidelity benchmarks, Sora 2024 critiques) is in our taxonomy the *evaluation harness for the world model* itself.

In the language of Section 2, this is JTBD-5 arriving early — but in a sub-community (model-based RL) that was not, until 2025–2026, in dialogue with the LLM agent-harness community.

---

5. Lineage III — The LLM Agent Harness Tradition (2022–2026)

5.1 ReAct, Toolformer, and the agent loop's birth

The modern LLM-agent literature begins with two 2022 papers. *ReAct* (Yao et al. 2022) showed that interleaving *reasoning traces* and *action calls* in a single LLM output produced substantially better performance on multi-step problems than either alone. *Toolformer* (Schick et al. 2023) showed that LLMs could be taught to call external APIs during generation. Together, these two papers gave the agent loop its modern shape: *think → call tool → observe → think → call tool → ...*

5.2 AutoGPT, BabyAGI, and the first wave

In 2023, AutoGPT (Significant Gravitas; March 2023) and BabyAGI (Yohei Nakajima; April 2023) operationalized the agent loop with stunning rapidity. Each defined a *plan-execute-reflect* outer loop, a tool-use inner loop, and a memory substrate. These systems were, in the harness taxonomy, the first widely deployed *broad harnesses* — combining JTBD-1 (correctness via JSON output parsing), JTBD-2 (memory and tool-result context), JTBD-3 (rudimentary safe-tool whitelists), and JTBD-4 (trace logging) in a single integrated artifact.

The first wave's striking limitation, exposed within months, was that the agents could not stay on task. The literature on agent *drift*, *goal abandonment*, and *loop catastrophe* in late 2023 essentially identified the absence of *world-model stewardship* — the agent had no consistent representation of *where it was in the task* and *what state of the world it was acting in*.

5.3 Generative agents, Voyager, and the social/embodied wave

In parallel, Park et al. (2023, *Generative Agents: Interactive Simulacra of Human Behavior*) at Stanford and Voyager (Wang et al. 2023, MineDojo + GPT-4) at NVIDIA pushed the agent loop into richer environments. Generative Agents introduced *reflection-as-memory-consolidation*, an architecture in which the agent periodically condensed observations into higher-level conclusions — a primitive but recognizable form of world-model maintenance. Voyager introduced *skill library construction*, in which the agent accreted a corpus of programs it could call later — a recognizable form of policy-as-harness.

5.4 Claude Code, Cursor, OpenHands, Aider, and the IDE-coupled wave

By 2024–2025, the centre of gravity had moved to *agent-in-the-IDE* systems. Claude Code (Anthropic, 2024–2025), Cursor (Anysphere, 2023–2025), OpenHands / OpenDevin (All Hands AI, 2024), Aider (Paul Gauthier, 2023–2025), Continue.dev, Cline, and others share a recognizable architecture:

- A *project-aware context layer* that reads source trees, edits files, runs tests.
- A *tool-use protocol* that abstracts over shells, browsers, language servers, and version control.
- A *persistence layer* (often Git itself) that makes the agent's work auditable and revertible.
- A *policy layer* that bounds destructive operations.
- *Subagent / orchestration primitives* that fork the work into parallel investigations.

What is striking about this wave is how much of it is *not the model*. Most of the perceptible quality improvement of 2024–2025 agentic coding is harness improvement, not model improvement. The dominant Cursor / Claude Code / OpenHands pattern surfaces the broad harness for the first time as the *primary engineering surface*.

5.5 Computer-use agents and the operating-system wave

In late 2024, Anthropic released *Computer Use* (October 2024) — an agent that could observe a screen and emit keyboard/mouse actions. OpenAI followed in 2025 with related capabilities. The computer-use harness pushed JTBD-2 (context provisioning) and JTBD-3 (policy enforcement) to their limits: the AI now operates in an unbounded environment whose state changes outside its observation, with policy boundaries that must hold against an arbitrary GUI surface.

This wave makes plain a fact the earlier waves could hide: *the harness must now take responsibility for what the AI thinks is on the screen.* The screen is the AI's world, and the AI's representation of the screen is what the harness must steward.

5.5b SWE-agent, OpenDevin, and the agent-computer-interface thesis

Two 2024 papers crystallized a specific subthesis: the *agent-computer interface (ACI)* — the shell, file system, language servers, and patch-application apparatus through which an agent operates on code — is itself a primary determinant of agent capability, often eclipsing model choice.

- **SWE-agent** (Yang et al. 2024, *SWE-agent: Agent-Computer Interfaces Enable Interactive Problem Solving*, arXiv 2405.15793; the Princeton / Stanford team behind SWE-bench) demonstrated that a hand-engineered ACI improves agent performance on SWE-bench substantially over a base ReAct loop. The ACI is, in our taxonomy, a *Tool* × *Context* harness whose design is the engineering contribution.
- **OpenDevin / OpenHands** (Wang et al. 2024, *OpenDevin: An Open Platform for Agentic Software Engineering*, arXiv 2404.06052; later renamed OpenHands) generalized the SWE-agent pattern into a community runtime: tasks, tools, memory, execution, verification loops are each first-class. OpenDevin / OpenHands is, in our taxonomy, the first widely-deployed *broad-harness as platform* — multiple harness layers exposed as configurable subsystems.

These two papers, read together with the contemporaneous arrival of MCP (Section 5.7), make the case for the broad-harness object empirically. The agent's substrate is no longer a stack-trace; it is a designed environment.

5.5c Prior taxonomies and what is novel here

The agent literature already contains several taxonomies the present survey must engage with explicitly:

- **Xi et al. (2023), *The Rise and Potential of Large Language Model Based Agents: A Survey*** (arXiv 2308.11432) — a 100+ page survey that taxonomizes agents along *Brain (planning, memory, reasoning)* × *Perception* × *Action*, with extensive multi-agent and human-AI sections. Xi et al.'s axes overlap our Axis A *Memory* and *Tool* values but treat *world model* implicitly. Our JTBD-5 framing names what their taxonomy leaves unmarked.

- **AgentBench (Liu et al. 2024)** — an evaluation framework that, importantly, treats the environment setup *itself* as part of the measured object. AgentBench's environments are a recognizable form of our Axis A *Context* and Axis C *Pre-shaping*.
- **The "AI Agent Index" community taxonomy (2024–2025)** — a community-maintained catalog that splits agents along planning, memory, tool use, reflection, and environment interaction. Our taxonomy is denser on *control layer* and *truth-guarantee mode*, sparser on *planning-style* distinctions.

What is new in the present survey relative to these prior taxonomies: (a) the unification of three originally disjoint lineages (software / robotics / LLM agent) under one engineering object; (b) the explicit naming of *world-model stewardship* (JTBD-5) as a distinct first-class Job; (c) the *Stewardship Pivot* thesis as a single-sentence summary of the discipline's centre-of-gravity shift; (d) the eight-axis × six-tension structure that locates a framework in a single shared design space.

What is *not* new: most individual primitives we describe (RAG, ReAct, tool use, constitutional AI, world models in RL, sim-to-real) are well-established. The contribution is the unifying object, not the primitives.

5.6 Multi-agent and multi-LLM frameworks

The parallel track of *multi-agent* frameworks — AutoGen (Microsoft, 2023–2025), CrewAI, MetaGPT, LangGraph state machines, OpenAI Swarm — pushed the harness to coordinate *multiple agents*. The literature here is rich on orchestration patterns (supervisor agent, role-specialized teams, debate, peer review) and increasingly on *multi-vendor ensembles* — OpenRouter, LiteLLM, MoE-of-vendors — which treat the choice of model itself as a harness decision.

The Sotopia line of work (Zhou et al. 2024 at CMU) extended the agent harness into explicit *social simulation*: agents are given personas, goals, and a social environment, and the harness scores their interactions on dimensions like believability, goal completion, and relational dynamics. Sotopia is one of the first systems where the *world model in the harness* is not physical (positions, objects) but social (norms, relationships).

5.7 Protocols and harness standards (2024–2026)

A distinct family of efforts in 2024–2026 attempted to standardize the *seam* between model and harness. The two most consequential:

- **Model Context Protocol (MCP)** — Anthropic, November 2024. MCP defines a standardized JSON-RPC protocol by which a host (the agent) discovers and calls tools and resources offered by remote servers. By the first half of 2026, MCP servers were appearing for filesystems, browsers, knowledge bases, version control, ticketing systems, and ad-hoc internal services. MCP is, in our taxonomy, the first widely adopted standard for the *Tool* and *Context* values of Axis A — and it is explicitly designed so that the *harness* (the host) and the *capability* (the server) can be developed independently.
- **Function calling and tool-use schemas** — OpenAI, Anthropic, Google. The three major model vendors have converged on a JSON-schema-based contract for tool definitions, with minor syntactic differences. This is the closest the field has come to *cross-vendor harness portability* (Axis: we discuss this in §10.2).

A second wave of harness standards is just beginning. The **OASIS Open AI Inferencing Standard**, the **EU AI Act conformity assessment** (which implicitly requires harness-shaped evidence), and the **NIST AI Safety Institute evaluation protocols** (2024–) all assume a harness sits between the model and the world. None yet specifies what its shape should be.

5.8 The agent-harness wars and what they revealed

A telling moment in the agent-harness literature was the rapid succession of releases through late 2024 and 2025: Claude Code (open-sourcing of the internal Anthropic IDE agent), Cursor's Composer feature, Cline (community fork of Anthropic patterns), Aider's autonomous mode, OpenHands' rapid iteration, Continue.dev's IDE integration, Codeium, GitHub Copilot's Agent mode, and others. By any measure, this was a *Cambrian explosion* of harness designs sharing roughly the same model substrates.

What the explosion revealed: *most of the perceptible quality variation among agentic coding tools comes from the harness, not the model*. The same underlying LLM, scaffolded differently, produces dramatically different outputs. This is the strongest empirical evidence we have that broad harness engineering is the high-leverage object — and it explains why so much engineering effort has migrated there since 2024.

---

6. The Convergence (2024–2026)

The three lineages — software harness, robotics / embodied harness, LLM agent harness — have, until very recently, been read by mostly disjoint communities. By 2024–2026, several signals make their convergence visible:

1. **Shared vocabulary**. Words like *scaffold*, *harness*, *world model*, *evaluation*, *replay buffer*, *reward model*, *invariant* now appear in all three communities, often citing each other.
2. **Shared architectures**. The Claude Code / OpenHands / Cursor agent loop is unmistakably a descendant of the robotics behaviour-based control loop *and* of the CI / fuzz / chaos engineering tradition. The Dreamer-style world model is making its way into LLM agent design via constructs like *agentic memory*, *long-term reflection*, and *internal-state tracking*.
3. **Shared problems**. Goal drift, simulation-to-reality gap, calibration of self-confidence, adversarial robustness, evaluation-set leakage — all three communities are confronting the same problems at the same time.
4. **Shared formal methods**. The TLA+ tradition (Lamport 1994; *Specifying Systems* 2002) is being applied to LLM agent invariants (Section 7.4). Process algebras (CSP, CCS) are being revived for multi-agent specifications. Contract-net protocols (Smith 1980) are reappearing in 2024–2025 multi-LLM coordination literature.
5. **Shared regulatory pressure**. The EU AI Act (2024), NIST AI Safety Institute evaluations (2024–), and similar regimes assume a *harness-shaped object* sits between the model and the world. The harness is now a regulatory unit, not just an engineering unit.

The convergence does not yet have a settled name. *Agent engineering*, *agentic systems engineering*, *AI scaffolding*, *AI middleware*, *constitutional engineering*, and others are all in circulation. We propose *broad harness engineering* — broad to distinguish it from the narrow 1970s test harness, and *harness* to keep the metaphor that the engineering object is *what is around the AI, not the AI itself*.

6.1 Two empirical convergence cases (March–April 2026)

Two 2026 publications, surfaced by the Perplexity Sonar review of this paper, make the convergence claim empirically rather than conceptually:

- **Martin Fowler, *Harness engineering for coding agent users* (martinfowler.com, April 2026).** Fowler — whose 2012 *TestPyramid* essay was a touchstone of the software-harness lineage — explicitly carries the *harness* metaphor across to LLM agents. The article articulates a practitioner taxonomy of guides, sensors, inferential checks, and feedback loops for production agentic systems. The fact that the same author who codified the test pyramid is now codifying the agent harness, using *the same name*, is itself evidence that the software-harness lineage and the LLM-agent lineage are converging at the level of practitioner vocabulary.
- **Bölük et al., *Meta-Harness: End-to-End Optimization of Model Harnesses* (arXiv 2603.28052, March 2026).** Meta-Harness shows that the harness *structure itself* can be optimized end-to-end across held-out models, with reported gains on retrieval and coding tasks. This is the harness analogue of *learned controllers* in robotics RL: the harness becomes the optimization variable. Meta-Harness is, in our taxonomy, the first widely-circulated paper that puts JTBD-1 through JTBD-4 themselves under an outer learning loop. It directly demonstrates that the harness is now an engineering object substantial enough to be a hyperparameter.

Together, Fowler's article and Meta-Harness exemplify the convergence at, respectively, the *vocabulary* level (practitioner naming) and the *engineering* level (harness as optimization target). Both arrived in 2026.

6.2 Competing protocol-level convergences

Two infrastructure-level efforts deserve separate mention:

- **MCP (Model Context Protocol)** — Anthropic 2024 — standardizes the host ↔ tool/resource interface for LLM agents. By mid-2026, MCP servers exist for filesystems, browsers, knowledge bases, version control, and various internal services. MCP is the first widely-adopted *cross-vendor harness contract* for the *Tool* and *Context* values of Axis A.
- **A2A (Agent-to-Agent protocol)** — Google 2025 — addresses a layer MCP does not: communication between agents. Where MCP standardizes how one agent reaches the world, A2A standardizes how multiple agents reach each other. A2A is, in our taxonomy, an Axis A *Tool* value layered with Axis F *Agency-locus* protocol assumptions.

These two protocols, together with the JSON-schema function-calling convergence among OpenAI / Anthropic / Google (Section 5.7), are the closest the field has yet come to standardized broad harness contracts.

---

7. A Six-Axis Operational Taxonomy

Surveying the converged literature, six independent axes are needed to locate a given harness component. Each axis is documented below with representative examples.

7.1 Axis A — Control Layer

What is being controlled?

Value	Description	Representative examples
Tool	Bounds on which external actions are available	LangChain tools, Anthropic tool-use, Claude Code shell sandbox
Context	What the AI is told about the situation	RAG, vector DBs, context refresh, system prompts
Policy	What the AI is allowed or forbidden to do	Constitutional AI, RLHF, safety classifiers, EU AI Act conformity
Memory	What the AI remembers across turns / sessions	Generative Agents reflection, agent memory libraries, mem0, MetaClaw
World-Model	What the AI internally believes the world is	Dreamer latent space, JEPA, interpretability probes, fact-check refresh

Axis A is the most important single axis: most current frameworks lock to exactly one or two values and treat the others as someone else's problem.

7.2 Axis B — Time Granularity

At what time scale does the harness operate?

Value	Examples
Per-turn	JSON-mode coercion, single-tool-call validation, function-call argument checks
Per-session	Conversation memory, plan trees, session-level budgets
Per-project	Project context layers, codebase indices, persistent skill libraries
Per-population	Evaluation benchmarks, regression baselines, A/B testing, RLHF preference pools

The cross-cut of Axis A × Axis B already gives 5 × 4 = 20 distinct cells. Most production frameworks populate fewer than 10.

7.3 Axis C — Intervention Modality

When does the harness act?

Value	Examples
Pre-shaping	System prompts, persona priming, retrieved context, role-induction prefixes
Inline mediation	Tool whitelists, structured-output coercion, intermediate classifier gates
Post-hoc audit	Trace replay, evaluation harness, red-team batch, regression detector

Pre-shaping and inline mediation are visible to the live agent run; post-hoc audit is not. A mature harness uses all three.

7.4 Axis D — Truth-Guarantee Mode

How does the harness reason about what is true?

Value	Examples
Verification (formal)	TLA+ specifications, LTL/CTL temporal properties, contract-net protocols, type systems
Validation (empirical)	Benchmark suites, eval harness, RLHF agreement, human spot-check
Verification-of-validation	Meta-eval (e.g. *who graded the grader*), inter-rater reliability, eval-set contamination audit

Formal verification of AI agents is a small but growing field. Lamport's TLA+ is being applied to multi-agent invariants in 2024–2025 work. Model-checking of neural-network properties (reluplex, neural network verification) is a separate, mature thread. Property-based testing as adapted to LLM outputs (e.g. "given any user request in domain D, the output satisfies invariant I") is a third.

7.5 Axis E — Observability Depth

How much of the AI's process is the harness able to see?

Value	Examples
Opaque	Black-box API call; only inputs and final output visible
Instrumented	Reasoning trace logged (chain-of-thought, tool-call sequence)
Replayable	Deterministic replay from logs; can rerun on a fix
Formally specified	Invariants written down; observability is *exhaustive on the property*

Observability is the axis along which interpretability research enters harness engineering. Anthropic's interpretability circle work (2023–2025), Othello-GPT probes (Hazineh et al. 2023), and the linear-probe tradition more generally are making *replayable* and *formally specified* observability tractable on internals, not just on inputs and outputs.

7.6 Axis F — Agency Locus

Who decides what the harness does?

Value	Examples
Human-loop	Each action requires explicit approval; deployment-time setup is human-authored
Hybrid	Approval thresholds, escalation, deferred review
Autonomous	Agent edits its own harness, generates its own tests, evolves its own scaffold

The 2025–2026 trend toward *agent edits its own scaffold* (Voyager skill library; Claude Code subagent spawning; AutoGen reflective agents) makes Axis F a live ethical question, not just a technical one. The *closure risk* tension (Section 8.6) is precisely the worry that fully autonomous Axis F leaves no one outside the loop able to check.

7.7 Axis G — Cost / Latency (added in v0.2)

What does the harness cost to run?

Value	Examples
Token-budget bounded	Cap on context tokens per turn; truncation policy
Latency-bounded	Hard timeout; deadline-aware planning
Cost-amortized	Caching, prompt-prefix sharing, model routing by cost
Quality-prioritized	Cost-as-objective, latency-as-objective explicitly traded

Axis G is the axis along which production cost concerns enter the harness — most acutely visible in multi-LLM routing (OpenRouter, LiteLLM) and in context-window management. We treat it as a first-class axis in v0.2 in response to reviewer comment that v0.1 omitted economic dimensions.

7.8 Axis H — Security / Compliance (added in v0.2)

What security or compliance guarantees does the harness offer?

Value	Examples
Authentication / authorization	Tool-call permissioning, identity-bound resources, per-user scoping
Audit trail	Immutable logs of agent actions, attribution to user, GDPR Article-30 records
Conformity assessment	EU AI Act compliance evidence, NIST AI RMF documentation
Prompt-injection defence	Input sanitization, dual-LLM defence, MCP capability boundaries

Axis H sits adjacent to but distinct from Axis A *Policy*. Policy is normative ("the agent should not do X"); Security / Compliance is enforcement ("the harness materially prevents and audits X"). We treat them separately in v0.2 in response to reviewer comment.

7.9 Cell density: where the literature lives

Plotting the 489 surveyed sources against the Axis A × Axis C cross-cut yields a striking distribution: the densest cell by far is *Tool* × *Inline mediation* (the prompt-and-tools cell of mainstream LLM agent frameworks), followed by *Context* × *Pre-shaping* (RAG and context engineering). The least populated cell is *World-Model* × *Post-hoc audit* — the cell that asks *did the agent's internal world model match the actual world after the fact?* Almost nothing exists here. We return to this gap in Section 10.

With the v0.2 addition of Axes G (cost / latency) and H (security / compliance), the taxonomy locates frameworks in an 8-dimensional space rather than 6, but the core argument is unchanged: no current framework populates more than half the cells.

7.8 Three short worked examples

To make the six axes concrete, we briefly trace three current systems through them.

Example 1: a CI test harness (mature JTBD-1/4). A modern CI harness — say, GitHub Actions with a pytest job — locates as: Axis A = Tool + Context; Axis B = per-project (the test suite is the artifact of record at the project layer); Axis C = inline mediation (the build is blocked on test failure) plus post-hoc audit (history accumulates); Axis D = empirical validation (the test is the validator); Axis E = replayable (every run produces logs and seeds); Axis F = human-loop (humans author tests, humans triage failures). This is a deeply mature harness but populates only ~6 of 30 (Axis A × B) cells. World-model stewardship is absent.

Example 2: Claude Code (mature multi-Job). A 2024–2026 Claude Code session locates as: Axis A = Tool + Context + Policy + Memory + (weakly) World-Model; Axis B = per-session and per-project; Axis C = all three (pre-shaping via system prompt, inline via tool whitelist, post-hoc via git diffs); Axis D = a mix (formal type checks where present; empirical test suite; sometimes meta-eval via human review); Axis E = instrumented to replayable (every turn logs tool calls and outputs); Axis F = hybrid (humans approve or revert; agent can spawn subagents). This is a deep harness that populates more cells than any system the test-harness lineage ever produced — but its World-Model coverage is weak: the agent's belief about the codebase is implicit in context, not stewarded explicitly.

Example 3: a Dreamer-style RL agent (deep JTBD-5). A Dreamer agent locates as: Axis A = primarily World-Model (the latent dynamics are the engineering object); Axis B = per-population (training distribution) and per-session (rollout horizon); Axis C = pre-shaping (the world model defines what the controller perceives) plus post-hoc audit (rollout-vs-real comparisons); Axis D = empirical (rollout fidelity) plus increasingly meta-eval (counterfactual benchmarks); Axis E = replayable (latent traces are deterministic); Axis F = autonomous in inner loop, human-authored in outer loop. Dreamer is the cell-density opposite of Claude Code — deep on World-Model, almost absent on Tool and Policy. The architectural challenge for 2026–2027 is to combine these.

8. The Six Essential Tensions

The broad harness object is not a single optimum. It is a space shaped by essential tensions — choices the harness designer cannot avoid making. We name six.

8.1 T1 — Specificity vs Generality

A harness that perfectly fits one task (e.g. SQL generation for a specific schema) is fast, correct, and unportable. A harness that works for any task (e.g. a generic ReAct loop) is portable, slow, and prone to error. The field oscillates between specificity (vertical SaaS agents, domain-specific scaffolds) and generality (LangChain, AutoGen, the open agent frameworks).

Neither extreme is stable. Specific harnesses accumulate; general harnesses fragment.

8.2 T2 — Opacity vs Auditability

Deep harnesses — those that intervene heavily in context, policy, and world model — are by construction opaque to the agent and often to the deployer. An audit trail asks the harness to be transparent. A safety policy often asks it to be deep. These pull in different directions.

The constitutional AI literature, by writing the constitution down, makes the policy harness simultaneously deep *and* auditable. This is exemplary and exceptional.

8.3 T3 — AI-Authored vs Human-Authored

A growing share of modern harnesses are written *by* AI: tools generated on demand, prompts rewritten by another agent, evaluation rubrics drafted by an LLM, even the test harness written by a coding agent.

The paradox: the harness's job is to keep the AI honest, but the harness is increasingly an AI artefact. Who guards the guard?

8.4 T4 — World-Model Drift

The longer the harness runs, the more the AI's internal world model and the actual world diverge. Documents become stale, code changes underneath the agent, persona memory accumulates contradictions, the project goal mutates. Detecting drift — let alone correcting it — is a core unsolved problem.

The robotics community's sim-to-real literature has the deepest answers here. Their importation into LLM agent harness is incomplete.

8.5 T5 — Stewardship Authority

If the harness stewards what the AI believes about the world, who has the right to set what the AI believes? The constitutional AI debates (whose values? whose constitution?), the alignment debates, and the regulatory debates (whose risk tolerance?) all reduce to this question. It is fundamentally a political question pretending to be a technical one.

8.6b T7 — Cost / Latency Trade-off (added in v0.2)

Every harness layer that improves quality has a cost (additional tokens, additional calls, additional latency). The deepest harnesses — multi-LLM peer review, formally specified invariants, full replay logging — are also the most expensive. In production, this tension is acute: a 10× quality improvement at 100× cost is rejected by most deployment economics.

T7 is structurally distinct from T1 (specificity / generality) because a *specific* harness can still be costly, and a *general* harness can be cheap. Reviewer-driven addition to v0.2 — the original six tensions over-indexed on epistemic concerns and under-counted economic concerns.

8.7b T8 — Alignment / Intent Gap (added in v0.2)

A separate tension from T5 (stewardship authority): the gap between what the AI believes is desired (its model of user intent) and what the user actually intends. The harness can be authoritative on values (T5) yet still misread intent in any given turn.

This tension is the focus of much of the alignment literature (Russell 2019 *Human Compatible*; Christiano et al. 2017 RLHF; Bai et al. 2022 Constitutional AI). It is operationally distinct from drift (T4): drift is the AI's world model going out of date; intent gap is the AI's *model of the user's goal* being miscalibrated even at session start.

8.7 T6 — Closure Risk

If the harness is evaluated by the harness, and the AI generates the harness's evaluations, and the AI edits its own harness, the loop closes. There is no longer a point outside the system from which it can be checked.

This is the same problem the philosophy of science calls *methodological closure*: a methodology that scores itself well by its own standards. Independent verification — humans outside the loop, multi-vendor cross-checks, formal proofs against external specifications, lived-experience reviewers — are the candidate ways out. None is yet sufficient.

---

9. The Stewardship Pivot

We have now accumulated enough machinery to state the central thesis.

Through 2023, the prevailing question in broad harness engineering was: *how do we control what the AI does?* JTBD-1 through JTBD-4 are all formulations of this question. The harness is the actuator-side, output-side, and feedback-side wrapper around an LLM treated as a function.

By 2025–2026, the question has shifted. The prevailing question is now: *how do we steward what the AI internally believes about the world?* JTBD-5 — world-model stewardship — is now a first-class engineering object. Probing, specifying, maintaining, and counterfactually evaluating the AI's internal world model are recognized as the high-leverage moves.

We call this the **Stewardship Pivot**.

The pivot is visible in several distinct signals:

1. **Context engineering becomes a discipline.** When Karpathy and others elevated *context engineering* to a discipline name (2024–2025), what they were really naming was: *the engineer's job is to steward what the AI knows is true right now*. This is JTBD-5 dressed as JTBD-2.
2. **Interpretability circles enter production.** Anthropic's interpretability research circulating into production safety tooling (2024–2026) is the move from *control of outputs* to *probing of internal beliefs*.
3. **World models become evaluation targets.** Sora's release (OpenAI 2024) triggered an extensive debate over whether video generation entails world-modelling. The fact that this debate was worth having is itself the pivot: *world-modelling capability* is now a primary axis of evaluation, not a side effect.
4. **JEPA and the LeCun program.** LeCun's argument (2022–) that the next architecture should be JEPA — a joint embedding predictive architecture designed explicitly to *learn a world model* — is the pivot expressed at the architecture level.
5. **Dreamer-style RL meets agent design.** The model-based RL community's world models are beginning to be cited in agent-design papers. Each side has half the pivot; their joining is the rest.
6. **Regulatory framings.** The EU AI Act, NIST, and OECD frameworks increasingly require *evidence that the model has accurate beliefs about the deployment context*, not just that outputs are accurate. This is JTBD-5 as a compliance object.

The pivot does not deprecate JTBD-1 through JTBD-4. It adds a fifth layer on top. But the centre of gravity of the discipline is moving.

9.1 What changes when JTBD-5 is taken seriously

If we accept that broad harness engineering's primary job is now world-model stewardship, several engineering decisions shift:

- **Evaluation changes from "did the output match the reference" to "did the implied world model match the actual world."** Two systems can produce identical correct outputs while one has a coherent internal world model and the other is pattern-matching from training distribution. The pattern-matcher will fail under distribution shift; the world-model-bearer may not. Evaluation harnesses (JTBD-4) start to need *probing* — direct queries to internal state — not just output scoring.
- **Memory architecture changes from "store recent context" to "maintain a consistent worldview."** Context window pressure (Anthropic's 1M context, Google's 2M, the 10M+ research models) makes raw retention cheap; what remains hard is *consistency under retrieval and update*. The literature on agent memory in 2025–2026 is increasingly about *belief revision*, not storage.
- **Tooling design changes from "expose actions" to "expose state-acquisition primitives."** Tool harnesses that emphasize *what the agent can do* are giving way to harnesses that emphasize *what the agent can ask*. MCP's design encodes this: resources (read-only state) are first-class, not just tools (effects).
- **Failure mode taxonomy changes.** JTBD-1 through JTBD-4 produced failure modes like "wrong output format," "missed retrieval," "policy violation," "no feedback signal." JTBD-5 produces failure modes like "agent believes the wrong version of the file is on disk," "agent's mental model of the user is stuck at session start," "agent has incoherent beliefs about its own past actions." These are *epistemic* failures, not behavioural.

9.2 Counter-arguments to the pivot framing

Three plausible counter-positions deserve serious treatment:

1. **The pivot is a re-labelling, not a real shift.** Sceptic: context engineering (JTBD-2) and policy enforcement (JTBD-3) already implicitly steward what the AI believes. Adding JTBD-5 is just renaming what was always there. *Response*: the renaming is the point. Recognizing what was implicit as explicit opens design moves (probing, drift detection, counterfactual evaluation) that the implicit framing hides.
2. **World models are an artefact of LLM scale; the pivot will not survive the next architecture.** Sceptic: when LLMs are replaced by JEPA-derivatives or successor architectures, the internal world model becomes an explicit engineering object and "stewardship" collapses back into ordinary engineering. *Response*: even granting this, the *transition* requires the pivot framing. Whatever the next architecture is, designing for it requires the question "what does it internally believe?" to be on the table.
3. **The pivot mystifies a quality problem.** Sceptic: most "world model" failures are really data freshness, retrieval, or context-management failures — i.e. JTBD-2. Renaming them as JTBD-5 makes the problem

fuzzier, not more tractable. *Response*: empirically, the failure modes (epistemic incoherence, false certainty, belief-revision failures) overlap with but are not reducible to retrieval failures. JTBD-2 fixes the document supply; JTBD-5 fixes whether the agent draws coherent conclusions from the supply.

9.3 Differentiation from prior world-model and stewardship-adjacent work

A reviewer of v0.1 raised the strongest version of the renaming objection: the Stewardship Pivot is "merely a re-branding of existing world-model research (Dreamer, JEPA, sim-to-real) and Constitutional AI." We treat this directly:

- **vs Dreamer / DreamerV3 (Hafner et al. 2019–2023)**. Dreamer engineers the *agent's learned latent world model* as an artefact of RL training. The world model is *internal to the agent* and *trained, not specified*. The Stewardship Pivot generalizes the move: in broad harness engineering, the world model is *also* what the LLM agent maintains during a session, which is *not the result of training* but of context, retrieval, and reflection. The Dreamer move was the inspiration; the Stewardship Pivot is its extension to systems where the world model is a runtime artefact, not a training artefact.
- **vs JEPA (LeCun 2022; I-JEPA, V-JEPA 2023–2025)**. JEPA proposes that the *architecture itself* be reorganized around joint embedding prediction — i.e. world-model representation moves into the model's substrate. The Stewardship Pivot does not depend on which architecture wins. It is an *engineering* claim (the engineering job has changed) that holds whether the next generation of models is JEPA-derived or LLM-derived. If JEPA wins, the pivot becomes easier (world-model stewardship has explicit hooks); if LLMs win, the pivot is harder but no less necessary.
- **vs Constitutional AI (Bai et al. 2022)**. Constitutional AI engineers what the *AI believes about right action*. This is, in our taxonomy, JTBD-3 (policy enforcement) with a world-model side-effect — the constitution is partially a normative document and partially a description of the world in which actions are evaluated. The Stewardship Pivot extends the move from *what is right* to *what is true*: a constitution prescribes values; world-model stewardship prescribes beliefs about facts. CAI is in this sense an early instance of the pivot, restricted to the normative axis.
- **vs Sim-to-Real (Tobin et al. 2017; OpenAI Robotics 2019)**. Sim-to-real engineers the *gap between simulated and real worlds* explicitly. The Stewardship Pivot generalizes the framing: any deployed AI has a sim-to-real gap (training distribution ≠ deployment distribution), and the harness is now responsible for it. Robotics was thirty years ahead of LLM agent engineering on this specific point; importing the framing is the work.

In summary: each of Dreamer, JEPA, Constitutional AI, and sim-to-real is a *special case* of broad harness engineering with world-model stewardship as a Job. The Stewardship Pivot names the move that *recognizes them as cases of the same thing* and asks for the unified framework. The renaming objection is reasonable but, we argue, under-counts the work of unification.

---

10. A Seven-Item Research Agenda

We close with seven open questions that, if resolved, would substantially advance broad harness engineering.

10.1 World-model schema interoperability

How do we describe the contents of a world model in a way that is portable across architectures (Transformer, JEPA, RWKV, state-space models, video diffusion)? Today every architecture has its own latent space and there is no shared schema. The right abstraction is plausibly *content* (entities, relations, dynamics) rather than *representation* (vectors, attention weights).

10.2 Cross-vendor harness portability

A harness written for Claude is not portable to GPT, Gemini, Llama, or Mistral without rewriting. This is a regression from the JUnit era, where xUnit was deliberately portable. Standards work in this direction (Model Context Protocol — Anthropic 2024; OpenAI's function-calling format; the LangChain abstraction layer) is early. We need *xHarness*.

10.3 Participatory world-model design

If the harness stewards what the AI believes is real, the stakeholders the AI affects need a seat at the design table. This is recognizable from disability scholarship (*nothing about us without us*), from participatory design, and from public-sector service design. None of these traditions has yet been imported into harness engineering at scale. Cycle 7 review and lived-experience reviewer designs are early gestures.

10.4 Formal harness specification languages

Most of the LLM agent harness is described in code or English. Formal specification languages — TLA+, Alloy, CSP, contract nets, behavior trees — exist but are not in mainstream agent-engineering pedagogy. A *formal harness specification language* for broad harness — covering tool, context, policy, memory, and world-model layers together — does not exist. It should.

10.5 Harness epidemics

A bug in a popular harness (LangChain, AutoGen, Claude Code) propagates across thousands of downstream agents. There is no epidemiology of harness failures. We do not know whether known harness vulnerabilities (e.g. tool-injection patterns) decay or propagate over time. This should be a research field; today it is not.

10.6 Harness ethics — who harnesses whom

The deepest ethical question of broad harness engineering: who has the standing to author harness rules for an AI that millions of people interact with? The constitutional AI debate, the alignment-target debate, and the EU AI Act debates are all framings of the same question. There is no settled framework — and many candidate frameworks (procedural, substantive, federalist, pluralist) are mutually incompatible.

10.7 World-model commons

A model's world model is, at scale, a public good: it shapes what millions of users come to believe about the world. We argue that there should be a *world-model commons* — open, audited, interoperable specifications of what major deployed models believe is true about major topics (climate, vaccines, history, ongoing events). Today there is nothing of the kind. The interpretability community has the tools; the regulatory community has the demand; the integration is missing.

---

11. Methodology and Limitations

11.1 Pre-registration

This survey was pre-registered before any data was collected. The pre-registration document (immutable; committed to a public git repository) specifies the five hypotheses, the eight thematic clusters, the inclusion criteria, the deliverable file paths, and the decision gates. The pre-registration is the document of record for what we *intended* to find; this paper is the document of record for what we *did* find. Differences between the two are reported in revision-note sections of future versions (e.g. v0.2).

11.2 Source acquisition

Sources were acquired through 64 Perplexity Sonar queries across 8 thematic clusters (C1 software harness history, C2 robotics / embodied harness, C3 LLM agent harness, C4 world model concept, C5 context engineering, C6 multi-actor / multi-LLM harness, C7 formal verification, C8 ethics and governance). The query set is fixed in the pre-registration. After de-duplication, 489 unique sources were obtained. The full source table is committed to the repository.

11.3 Inclusion / exclusion criteria

Included: peer-reviewed papers, preprints, institutional reports, foundational textbooks, major-lab blog posts that articulate engineering positions, regulatory documents, mature open-source documentation. Excluded: marketing copy, opinion pieces without grounding, tutorials.

11.4 Coding

Sources were coded by cluster and by axis. Each axis (A–F) has at least three representative examples cited inline. Each tension (T1–T6) has a representative source pair (literature on opposing sides) where one is available.

11.5 Limitations

- **Language skew.** All 64 queries were issued in English. Japanese, Chinese, Korean, German, French, Spanish, and Portuguese harness-engineering literatures are surely substantial and are not represented. This is a significant gap.

- **LLM-mediated retrieval.** Source identification used Perplexity Sonar, an LLM-mediated retrieval system. This introduces bias toward what an LLM retrieves; it is not equivalent to a PRISMA-style systematic review.
- **Closure risk of the survey itself.** This survey is being authored using a broad harness (Claude Code + Perplexity + multi-LLM review panel — see v0.2 below). The survey is therefore *partially an instance of its own object*. We acknowledge this as a recursive limitation and treat the survey as a *position paper* rather than a definitive systematic review.
- **Temporal cut-off.** The survey reflects literature available as of May 2026. Subsequent work — particularly on JTBD-5 — may render parts of the taxonomy obsolete.
- **Independence claim.** The pre-registration prohibits back-citation to the Sato 2026 research series. To the extent the author's prior thinking has influenced framing (e.g. the choice of *Stewardship Pivot* as a name), this is an unavoidable but bounded leak. The substantive surveyed material is independent.
- **PRISMA gap (v0.2 expanded).** Reviewers of v0.1 correctly noted that the survey does not meet PRISMA standards: there is no operational inclusion / exclusion protocol, no inter-rater reliability, no PRISMA flow diagram, no quantitative synthesis (e.g. citation-network analysis), and the search strategy is LLM-mediated rather than a reproducible Boolean string. We acknowledge this directly. We frame the paper as a *position survey* — an argument-grounding survey that uses a broad source base to organize a thesis — and not as a systematic review. The Type field in the abstract reflects this.
- **Coverage of late-2025 / 2026 work.** The Perplexity Sonar review surfaced several 2026 works (Fowler April 2026, Meta-Harness March 2026, OpenReview agent-harness-survey May 2026) that the v0.1 draft did not engage with. v0.2 adds these. Future versions may need to engage additional contemporary work as the literature is moving rapidly.

---

12. Conclusion

The word *harness* in software engineering has spent fifty years quietly carrying a metaphor that turned out to be exactly right. A harness is *what is around the horse, not the horse itself*. By 2026 the engineering object the word denotes has grown to include every artifact, scaffold, evaluation, policy, memory store, and probe that surrounds an AI — *and now, increasingly, the design of what the AI itself believes is real*.

We have argued five things:

1. (H1) Three originally independent lineages — software harness, robotics / embodied harness, and LLM agent harness — converged in 2024–2026 into a single engineering discipline we call *broad harness engineering*.
2. (H2) Broad harness engineering has five Jobs-to-be-Done. The fifth — world-model stewardship — is the newest and the most consequential.
3. (H3) Broad harness can be operationalized along six independent axes (control layer, time, intervention, truth, observability, agency), with most current frameworks populating fewer than half the resulting cells.

4. (H4) Six essential tensions structure the research agenda (T1 specificity / generality, T2 opacity / auditability, T3 AI-authored / human-authored, T4 drift, T5 stewardship authority, T6 closure risk). Most current frameworks resolve at most two of these.
5. (H5) The *Stewardship Pivot* — from controller of AI actions to steward of AI beliefs — is the central methodological event of 2026.

We close with the obvious recursive remark. This paper is itself a broad harness output. The harness that produced it — pre-registration, Perplexity-grounded survey, multi-LLM review panel, formal version control, an explicit pre-committed methodology — is a harness whose Job is to keep its own conclusions honest. Whether that Job has been done well, we have argued, can only be judged from outside the loop. That is the test we now hand to the reader.

Appendix A — Glossary

The following 24 terms are recurrent in the broad-harness literature and recur throughout this survey.

- **Agent**: An autonomous or semi-autonomous AI system that takes a goal, perceives state, decides actions, executes them, and updates its understanding. Modern LLM agents are LLMs operated inside a loop that supplies tools, context, and memory.
- **Agent loop**: The minimal control structure of an LLM agent — observe, think, act, observe. ReAct (Yao et al. 2022) is its canonical formulation.
- **Behaviour-based robotics**: The robotics paradigm initiated by Brooks (1986) that decomposes control into layered behaviour modules acting directly on sensor input.
- **Chaos engineering**: The practice, originated at Netflix (Simian Army, 2011), of deliberately injecting faults into production systems to surface resilience problems.
- **Constitutional AI**: A training and runtime pattern (Bai et al. 2022) in which the AI's behaviour is shaped by a written constitution of principles.
- **Context engineering**: The discipline (named ~2024–2025) of designing what enters the AI's context window — including retrieval, summarization, persistence, and refresh policies.
- **Contract testing**: A microservice-era test pattern (Pact ecosystem) in which one service's expectations of another are written down and verified at the interface.
- **Dreamer family**: A line of model-based RL work (Hafner et al. 2019–2023) in which agents learn a latent world model and train the controller against rollouts in the latent space.
- **Drift (world-model drift)**: The phenomenon by which an AI's internal representation diverges from the actual state of the world over time.
- **Evaluation harness**: Infrastructure (e.g. lm-eval, HELM) for running standard benchmarks against a model and producing comparable scores.
- **Fuzz testing harness**: An input-generator harness (libFuzzer, AFL) that generates large quantities of test inputs to surface crashes and incorrect behavior.
- **JEPA**: Joint Embedding Predictive Architecture, proposed by LeCun (2022) as a successor architecture explicitly designed to learn world models.

- **JTBD (Jobs-to-be-Done)**: A theory of innovation (Christensen, Ulwick) that asks what *job* a user hires a product to do, rather than what features the product has.
- **MCP (Model Context Protocol)**: An Anthropic-originated protocol (2024) for standardized tool and resource exposure to LLM agents.
- **Multi-LLM panel**: A pattern in which several different LLMs are asked the same question and their outputs compared, used for evaluation, review, or aggregation.
- **PRISMA**: A systematic-review reporting standard (Page et al. 2021) widely used in evidence-based research; relevant here as a benchmark this survey *does not* fully meet (see §11.5).
- **Property-based testing**: A test paradigm (Claessen and Hughes 2000, QuickCheck) in which the harness generates inputs from a specification and asserts a property over many runs.
- **RAG**: Retrieval-Augmented Generation (Lewis et al. 2020), the pattern of retrieving external documents at inference time and adding them to the model's context.
- **ROS**: Robot Operating System (Quigley et al. 2009), the dominant open-source robotics middleware.
- **Scaffolding**: A near-synonym for *harness* common in the LLM agent literature, emphasizing temporary structures that support an agent's behaviour.
- **Sim-to-real (sim2real)**: The robotics problem (and its solution literature) of transferring a policy learned in simulation to the real world.
- **Stewardship Pivot**: This paper's name for the 2025–2026 shift from harness-as-action-controller to harness-as-belief-steward.
- **Test pyramid**: Mike Cohn's (2009) layered model of test suite composition — many unit tests, fewer service tests, fewest end-to-end tests.
- **TLA+**: A formal specification language (Lamport 1994; *Specifying Systems* 2002) used to specify and verify concurrent and distributed systems; recently being applied to AI agent invariants.

Appendix B — Survey Methodology Note

The eight thematic clusters and the 64 queries underlying this survey were specified in a pre-registration document committed to a public git repository *before* any Perplexity Sonar query was issued. The full set of raw query results (one JSON file per query, 64 files total) is committed alongside this paper at `research/perplexity/*.json`. A de-duplicated source table with URLs, dates, and snippet excerpts is at `research/source_table.md`. Cluster cardinalities after de-duplication:

Cluster	Topic	Sources
C1	Software harness history	60
C2	Robotics / embodied harness	62
C3	LLM agent harness	75
C4	World model concept	76
C5	Context engineering	60
C6	Multi-actor / multi-LLM	63
C7	Formal verification & TLA+	46

Cluster	Topic	Sources
C8	Harness ethics & governance	47
	Total (unique)	**489**

References (selected)

The following is a representative selection. The full 489-source list is committed to the repository at `research/source_table.md`.

Foundational software harness: - Beck, K. and Gamma, E. (1997). JUnit. (See *JUnit Pocket Guide*, 2004; *Testing Like a Pro with JUnit*, 2025.) - Cohn, M. (2009). *Succeeding with Agile.* (Test pyramid; see Fowler 2012 popularization.) - Humble, J. and Farley, D. (2010). *Continuous Delivery*. - Claessen, K. and Hughes, J. (2000). *QuickCheck: A Lightweight Tool for Random Testing of Haskell Programs.* ICFP. - Netflix Tech Blog (2011). *The Netflix Simian Army.* SEI CMU 2015 case study.

Robotics / embodied harness: - Brooks, R. (1986). *A Robust Layered Control System for a Mobile Robot.* AIM-864. MIT. - Quigley, M. et al. (2009). *ROS: An open-source Robot Operating System.* ICRA Workshop. - Tobin, J. et al. (2017). *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World.* - Hafner, D. et al. (2019). *PlaNet*; (2020) *Dreamer*; (2023) *DreamerV3.*

World model and LLM agent harness: - Ha, D. and Schmidhuber, J. (2018). *World Models.* - LeCun, Y. (2022). *A Path Towards Autonomous Machine Intelligence.* (JEPA proposal.) - Yao, S. et al. (2022). *ReAct: Synergizing Reasoning and Acting in Language Models.* - Schick, T. et al. (2023). *Toolformer.* - Park, J.S. et al. (2023). *Generative Agents: Interactive Simulacra of Human Behavior.* UIST. - Wang, G. et al. (2023). *Voyager.* (MineDojo + GPT-4 skill library.) - Lewis, P. et al. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS. - Bai, Y. et al. (2022). *Constitutional AI.* Anthropic. - Hazineh, D. et al. (2023). *Othello-GPT linear probes.* (Internal world-model recovery from a Transformer.)

Formal methods and governance: - Lamport, L. (2002). *Specifying Systems.* (TLA+.) - Smith, R. G. (1980). *The Contract Net Protocol.* - European Commission (2024). *AI Act.* - NIST AI Safety Institute (2024–). Various evaluation reports.

Alignment / RLHF / Constitutional AI (expanded in v0.2): - Christiano, P. F. et al. (2017). *Deep Reinforcement Learning from Human Preferences.* NeurIPS. - Bai, Y. et al. (2022). *Constitutional AI: Harmlessness from AI Feedback.* Anthropic. - Rafailov, R. et al. (2023). *Direct Preference Optimization.* - Russell, S. (2019). *Human Compatible: AI and the Problem of Control.* - Amodei, D. et al. (2016). *Concrete Problems in AI Safety.*

Competing agent taxonomies & evaluation (added in v0.2): - Xi, Z. et al. (2023). *The Rise and Potential of Large Language Model Based Agents: A Survey.* arXiv 2308.11432. - Liu, X. et al. (2024). *AgentBench: Evaluating LLMs as Agents.* arXiv 2406.10328. - Yang, J. et al. (2024). *SWE-agent: Agent-Computer Interfaces Enable Interactive Problem Solving.* arXiv 2405.15793. - Wang, X. et al. (2024). *OpenDevin: An Open Platform for Agentic Software Engineering.* arXiv 2404.06052. - Anthropic (2024). *Model Context Protocol Specification.* https://modelcontextprotocol.io/ - Google (2025). *Agent-to-Agent (A2A) Protocol.* - Park, J.S. et al. (2023). *Generative Agents.* UIST. - Zhou, X. et al. (2024). *Sotopia.* CMU.

Harness engineering as named discipline (2026) (added in v0.2): - Fowler, M. (2026, April). *Harness engineering for coding agent users.* martinfowler.com/articles/harness-engineering.html - Bölük, P. et al. (2026, March). *Meta-Harness: End-to-End Optimization of Model Harnesses.* arXiv 2603.28052. - *Agent Harness Engineering: A Survey* (2026, May). OpenReview. - *Harness Engineering: Why the Frame Matters More Than the Model* (2026, March). infralovers.com blog. - *Harness Engineering: How Interface Design Quietly Tripled AI Coding Performance* (2026, April). pub.towardsai.net.

(Full source list with URLs, publication dates, and snippet excerpts: `research/source_table.md`. 489 unique sources.)