14 — Convergences & Trends

🕒 state of the art 2026-07 · revised 2026-07-28 · 📖 ~7 min read

Learning objectives

By the end of this chapter, you should be able to:

1. **Enumerate** the eight architectural convergences of the first round and **explain** why independent convergence signals a consolidated discipline;
2. **Distinguish** the consolidated dimensions from the dimensions in genuine divergence, and **justify** why containment is the most consequential divergence;
3. **Apply** the expiration clause to any harness component — identifying why it exists and under what condition it expires;
4. **Evaluate** a new harness against the convergence checklist, demanding justification for each absence;
5. **Anticipate** the trends to watch in the coming rounds and what each would imply for harness design.

The problem

The previous chapters analyzed the harness dimension by dimension — context, compaction, tools, permissions, loop. What is missing is the question that gives the whole its meaning: **what is an accident of implementation and what is the anatomy of the discipline?** Without this synthesis, each chapter is a catalog of choices; with it, the reader gains a design criterion — knowing what to copy without hesitation, where a different bet is still viable, and what will disappear as models improve.

The measuring instrument is independent convergence. When teams that do not coordinate, on different stacks and from different cultures, arrive at the same architecture, that is strong evidence that the problem — not fashion — determined the solution. And the projection instrument is chapter 01's expiration clause: every harness component is a prosthesis for a current model limitation, and therefore every component should declare when it expects to become unnecessary.

The state of the art

The central finding of the first round: eight convergences

Three harnesses, three stacks (Effect-TS, TypeScript, Python), three origins (independent startup, big tech, academia/teaching gateway) — and a remarkable architectural convergence. Without coordination, all three arrived at:

1. **Hierarchical context file at the project root** — `AGENTS.md` / `GEMINI.md` / `CLAUDE.md`: the same artifact under three names (ch. 03).

2. **Staircase compaction** — truncate tools → prune → summarize via LLM, with automatic threshold-based triggering (ch. 04).
3. **Tool schemas derived from types** — Effect Schema, declarative classes, Pydantic: nobody writes JSON Schema by hand (ch. 05).
4. **MCP as the standard integration** — three full clients on the official SDKs (ch. 06).
5. **Plan mode as a permission mode** — read-only enforced by the permission system, not requested from the model (ch. 09).
6. **Lifecycle hooks** — before/after tool, compaction, session (ch. 12).
7. **Headless with structured output** — `-p` + JSON/NDJSON for scripting and CI (ch. 13).
8. **Stopping on absence of tool-call + turn limit** — the universal mechanics of the loop (ch. 02).

When independent implementations converge like this, the anatomy is consolidated: **this is the discipline**, no longer a set of idiosyncratic choices. A new harness that does not implement the eight items above must justify each absence.

Where genuine divergence remains

The dimensions without consensus are the map of the open bets:

- **Containment** (ch. 07): policy + mandatory OS sandbox (gemini-cli), policy + fixed sensitive paths (OpenHarness), or policy only (opencode)? The most consequential divergence — it is the one that defines operational risk.
- **Multi-agent** (ch. 10): a one-off tool, a service with a registry, or a persistent team with a mailbox? Three incompatible philosophies; the winner depends on how good models get at coordination.
- **Who decides to continue** (ch. 02): a structural heuristic or one extra inference per turn (next-speaker check)?
- **Model neutrality** (ch. 12): ~26 providers (opencode) versus the showcase of one ecosystem (gemini-cli). A commercial bet, not a technical one — but it defines who survives the commoditization of models.
- **Behavioral evals** (ch. 11): in round 1, only one of the three treated agent behavior as a regression surface — round 2 confirmed the prediction and the gap closed (see ch. 11). Easy prediction: in two years, this will be as mandatory as CI.

The expiration clause, applied

Returning to chapter 01's thesis — every harness component is a prosthesis for a current model limitation. The exercise every harness should do, applied to what we studied:

Component	Exists because...	Expires when...
Compaction	windows are finite and expensive	long context becomes cheap and reliable
Plan mode	models act rashly	models plan spontaneously under risk
Next-speaker check	the model does not signal end-of-turn well	model-native turn protocols
Policy engine / approvals	models are not trustworthy with destructive actions	calibrated, verifiable reliability
Prompt per model family	models respond differently to instructions	instruction-following convergence
Subagent for exploration	file dumps pollute the context	abundant context + robust attention
Repo-map / code indexes	the model does not "carry" the whole repo	usable multi-million-token context

What does **not** expire: sandbox (containment is about the world, not about model capability), interfaces, verification of the work (tests/LSP — truth external to the model), and the interoperability protocols (MCP, A2A, skill formats). Long-term harness engineering lives there: **at the boundary between the agent and the world, not in the crutch for the model's limitation**.

Trends to watch in the coming rounds

1. **Standardization of the context file** — the pressure for a vendor-neutral `AGENTS.md`.
2. **Portable skills/plugins** — OpenHarness already loads skills in the Claude Code format; an "MCP of extensibility" is taking shape.
3. **Agent-as-a-service** — A2A server, agent cards, SDKs: harnesses exposing themselves to one another.
4. **Security as a first-class dimension** — shell parsing, trusted folders, injection evals: today the exception, tomorrow the baseline (hypothesis confirmed in round 2 with Codex CLI).
5. **Reversibility** — git checkpoints with `/rewind`: when undo is cheap, the policy can be looser; expect more harnesses to copy it.
6. **The minimal harness** — against the grain of sophistication, projects like mini-swe-agent (~100 lines) test how much of the *scaffolding* the modern model can already do without. It is the expiration clause turned into an experiment.

EXECUTIVE SUMMARY

- Eight dimensions have already converged across independent implementations — they are the minimum checklist of a serious harness; absences demand justification.
- The genuine divergences (containment, multi-agent, next-speaker, model neutrality, behavioral evals) are the map of the open bets — containment is the one with the greatest operational consequence.
- The expiration clause separates temporary prostheses (compaction, plan mode, repo-map...) from what is permanent: sandbox, interfaces, external verification and interoperability protocols.
- The long-term value of harness engineering lives at the agent–world boundary; the rest changes hands or disappears as models improve.

- This chapter is the book's living scoreboard: each benchmark round confirms convergences, resolves divergences, or retires expired components.

> **See also**: the living collection Awesome Harness Engineering — Foundations gathers more consultable resources for this dimension, curated by problem.

Check your understanding

1. Why is **independent** convergence (three stacks, three origins) stronger evidence of consolidation than the adoption of a pattern by several projects that copy each other? (Re-read "The problem" and the central finding.)
2. A new harness implements neither plan mode nor a context file at the root. According to this chapter, what is the correct posture when evaluating it — and what would you demand from its author?
3. Apply the expiration clause to a component that is **not** in the table (for example, the next-speaker check is already there; pick lifecycle hooks or headless): does it exist because of a model limitation or because of a need at the agent–world boundary? Does it expire?
4. Among the five divergences listed, which one defines operational risk and which one is a commercial rather than technical bet? Justify with the text.