Beyond the Harness

Canvas for AI Agents

QCon AI Boston '26

Nimisha Asthagiri
@nasthagiri
Tech Advisor
/thoughtworks


chatgpt

Takeaways

Mental Model Shifts

Controlled Harness → Bounded Canvas
Cognitive Debt → Stigmergy (traces)
Human Review → Self-healing intentions

Product Engineering Model

Functional view: inputs, outputs, state
Synergy with product lifecycle
Hierarchical mapped strategy

Design for Change

Manage entropy, increase decay
Self-correct to mitigate drift
Expect evolution of behavior and requirements

Deliberate Failures

Ensure resilience with deliberate disturbance
Adversarial verification
Forced removal

Palette

1 - What

Semantic diffusion

Canvas model

Canvas components

2 - Why

Industry pressures

Industry learnings

Industry potential

3 - How

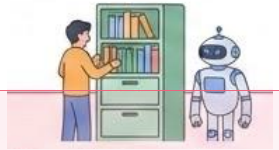
Encoded governance

Dimensional metrics

Evolving system

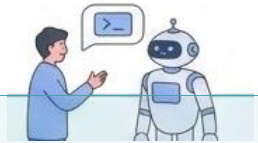
Harness Engineering

Build the World
Design Environment
For Task Lifetime



Context Engineering

*What to **See***
Manage Information
Across Interactions



Prompt Engineering

*What to **Say***
Optimize Instruction
For Single Interaction

Min Yin
https://milvus.io/blog/harness-engineering-ai-agents.md

Molisha Shah
https://www.augmentcode.com/guides/harness-engineering-ai-coding-agents

Feb 5 2026

Engineer the Harness

*"I don't know if there is a broad industry-accepted term for this yet, but I've grown to calling this '**harness engineering**.'"*

“..anytime you find an agent makes a mistake,
..engineer a solution such that
..agent never makes that mistake again.”

Mitchell Hashimoto, HashiCorp
https://mitchellh.com/writing/my-ai-adoption-journey

Feb 11 2026

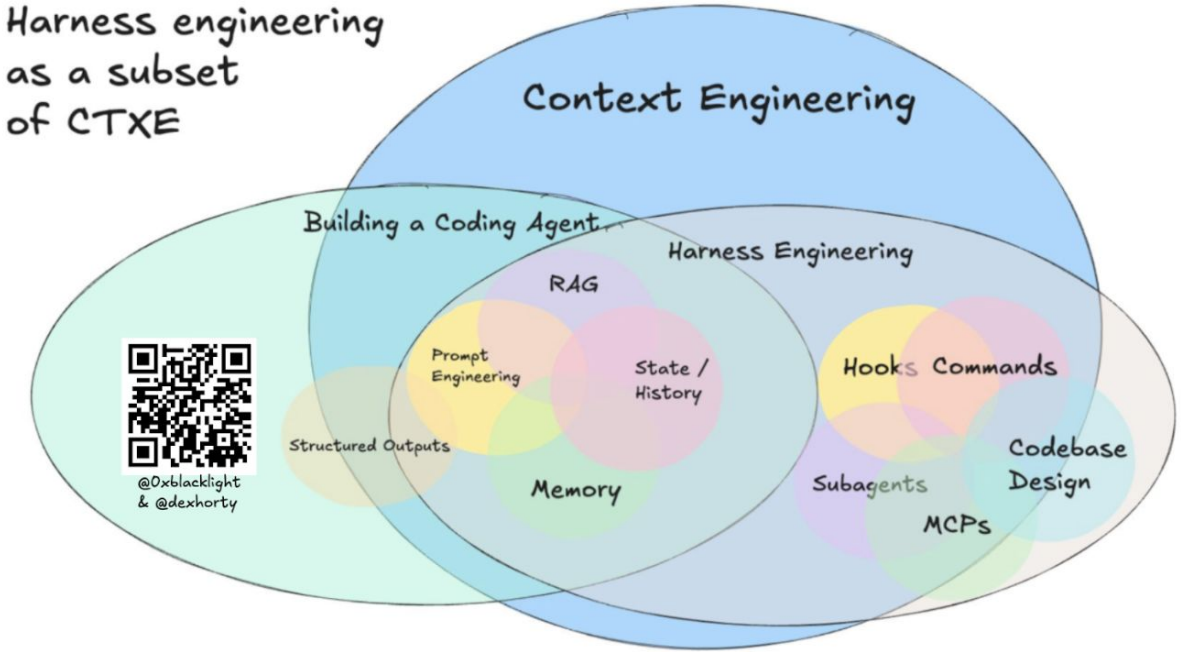
Humans steer. Agents execute.

> “Harness engineering: leveraging Codex in an agent-first world”

Ryan Lopopolo

https://**openai**.com/index/harness-engineering/

Mar 12 2026



Kyle Mistele

https://www.humanlayer.dev/blog/skill-issue-harness-engineering-for-coding-agents

Mar 10 2026

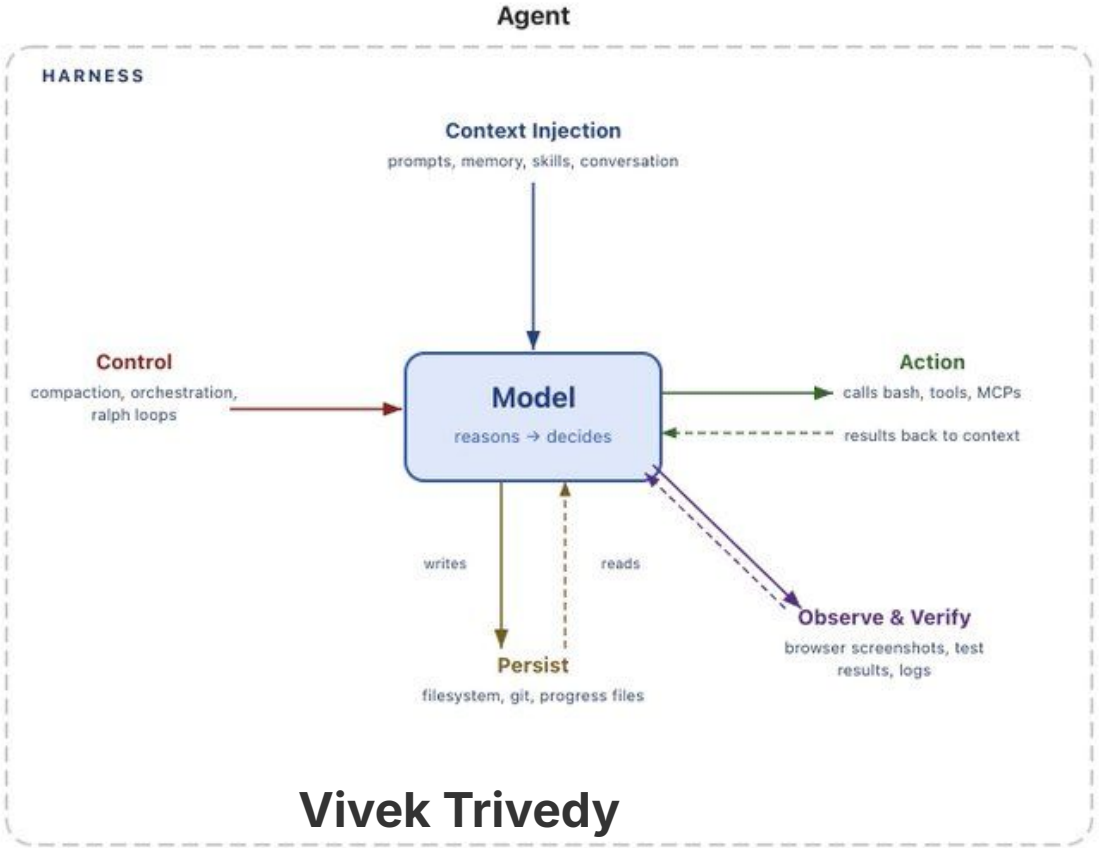
Agent = Model + Harness

> “If you're not the model, you're the harness.”

Vivek Trivedy

https://www.**langchain**.com/blog/the-anatomy-of-an-agent-harness

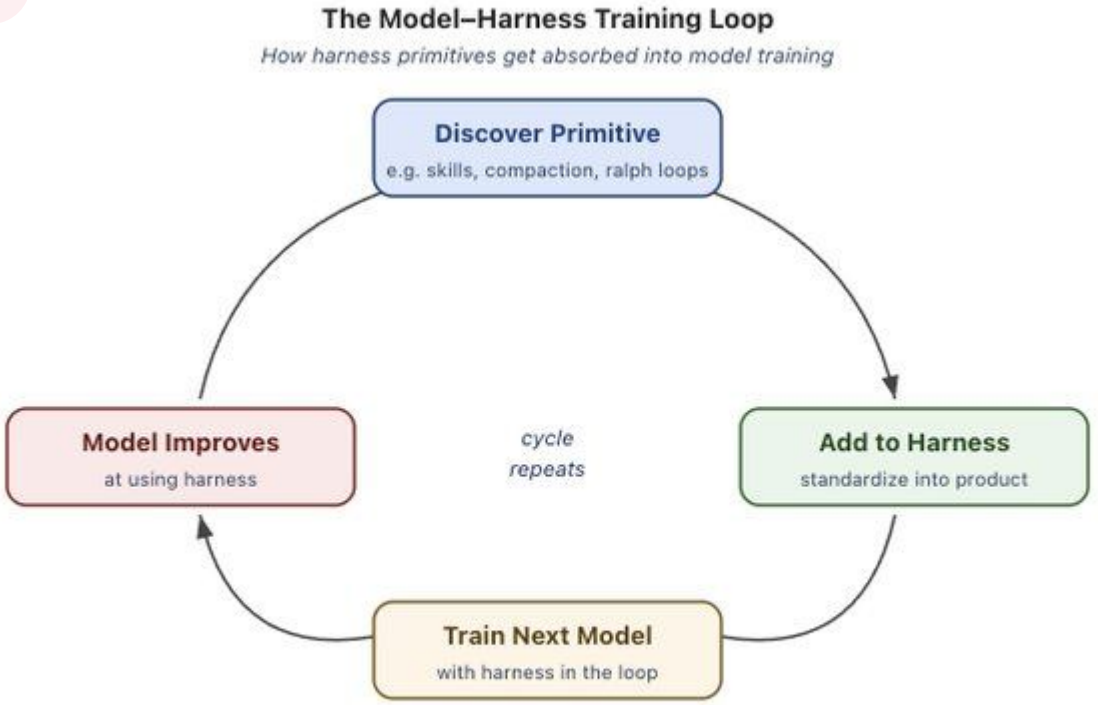
Mar 10 2026



Vivek Trivedy

https://www.langchain.com/blog/the-anatomy-of-an-agent-harness

Mar 10 2026



Vivek Trivedy

https://www.langchain.com/blog/the-anatomy-of-an-agent-harness

Mar 10 2026

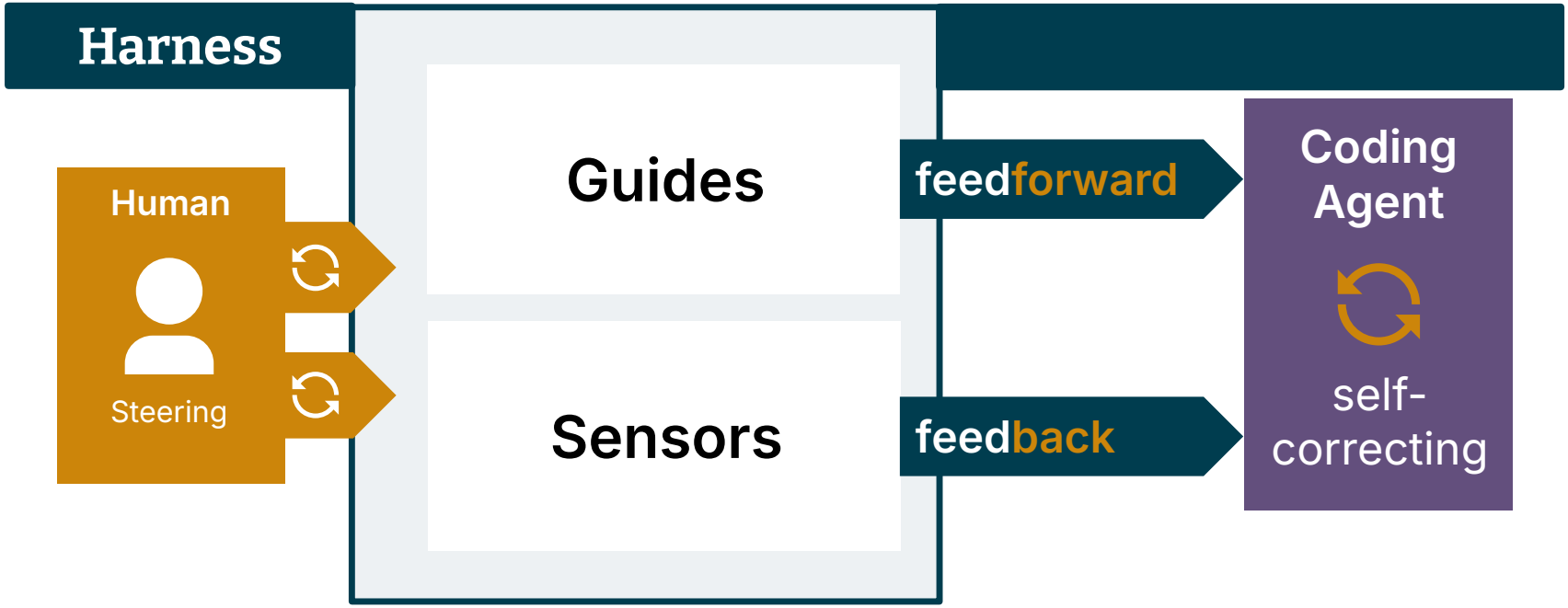
$$C_{system} = F(C_{model}, C_{harness}, C_{environment}, T)$$

> Harness… determines whether latent model capability becomes auditable software-engineering behavior

AI Harness Engineering: Runtime Substrate for Foundation-Model Software Agents

https://arxiv.org/abs/2605.13357

Apr 02 2026



Birgitta Böckeler

https://martinfowler.com/articles/harness-engineering.html

May 2026

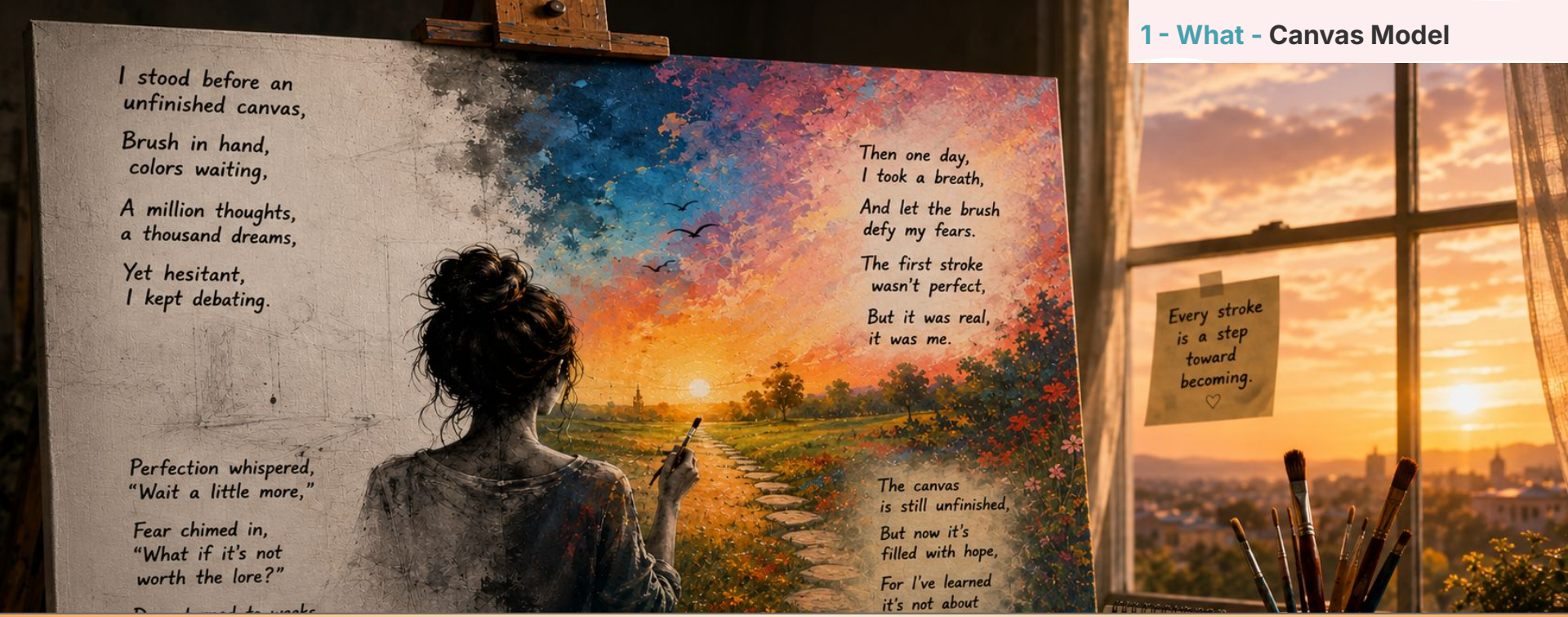

Mark Fisher
1st
Founder and CEO at Modulewise
8h
Harness. Guard Rails. Sandbox.
These all focus on applying constraints *from the outside*.

May 2026

Mark Fisher · 1st

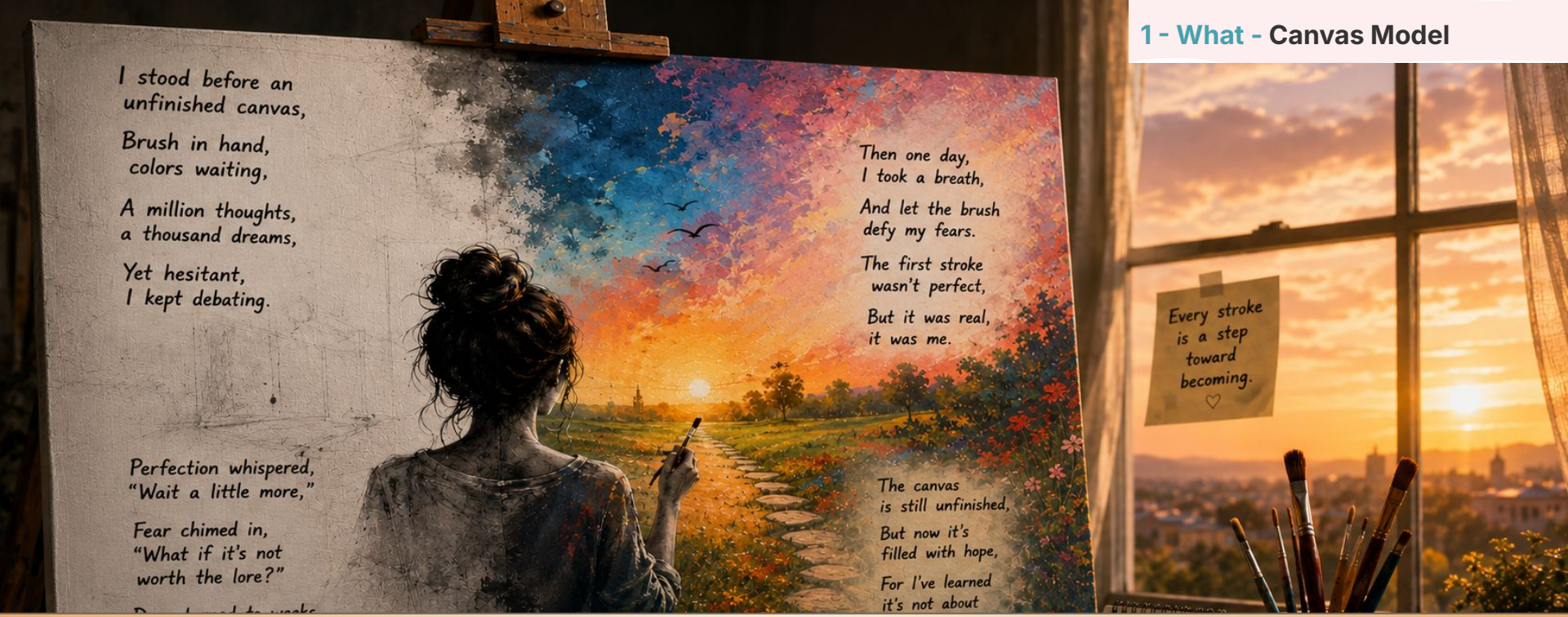
But in Agentic Systems, decisions drive actions *within the system*, cascading unpredictably in real time based on situational context. The constraints must also be applied from within the system.

The words we use reveal the mindset driving what we build. The agentic paradigm shift requires a corresponding shift in the metaphorical vocabulary.

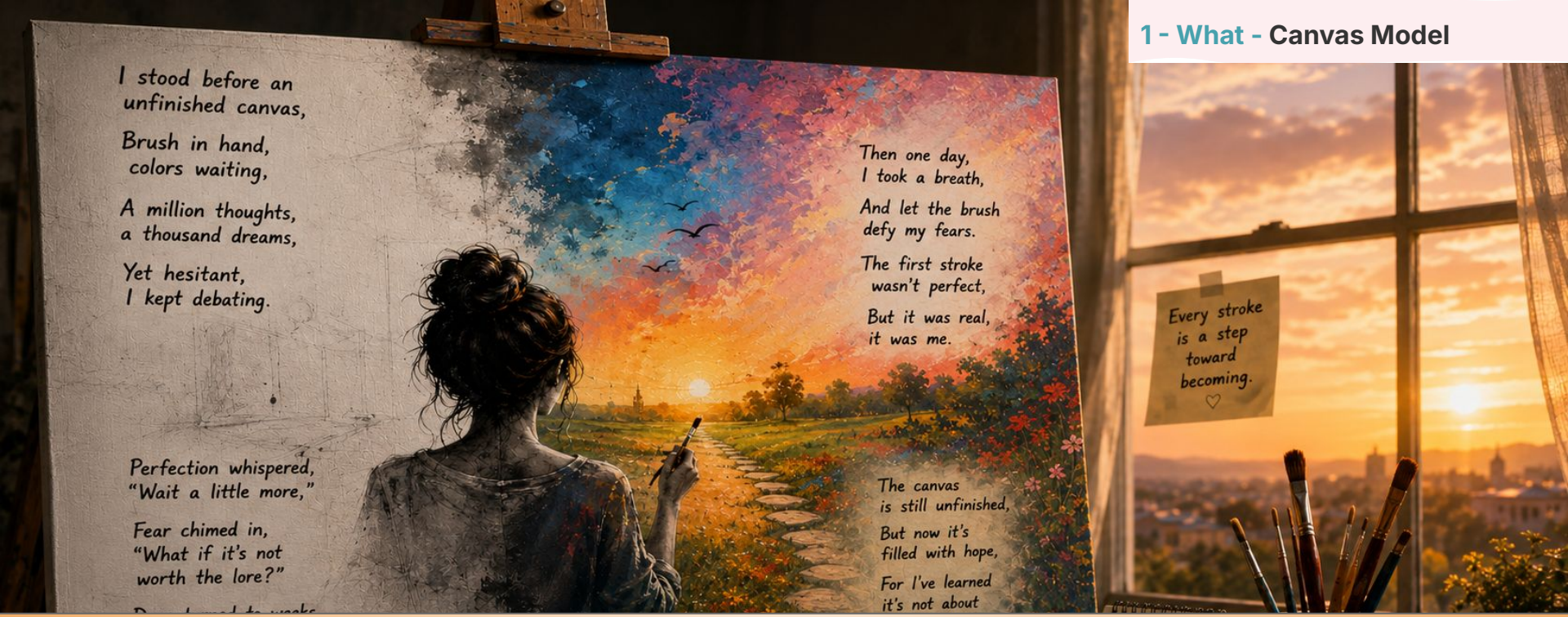


The Unfinished Canvas

by Mira Midha

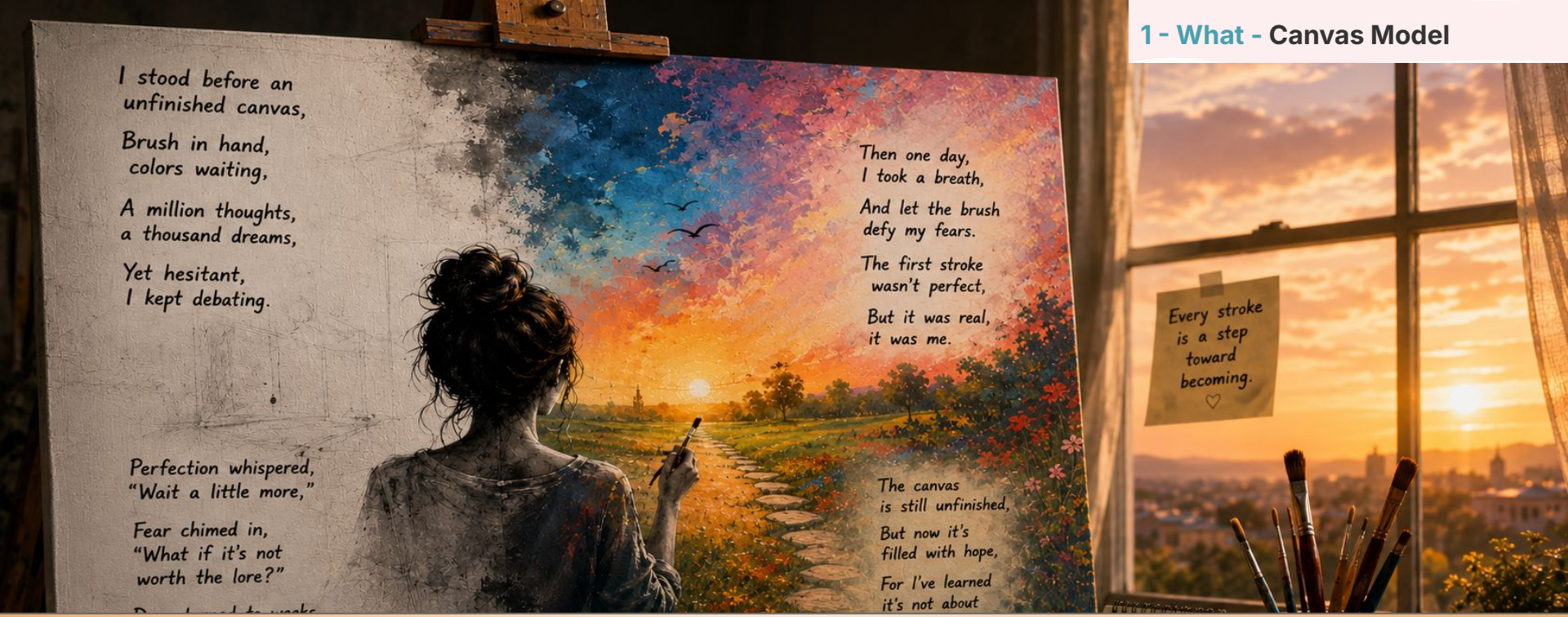


"On one half of a canvas...
The other with invisible words"

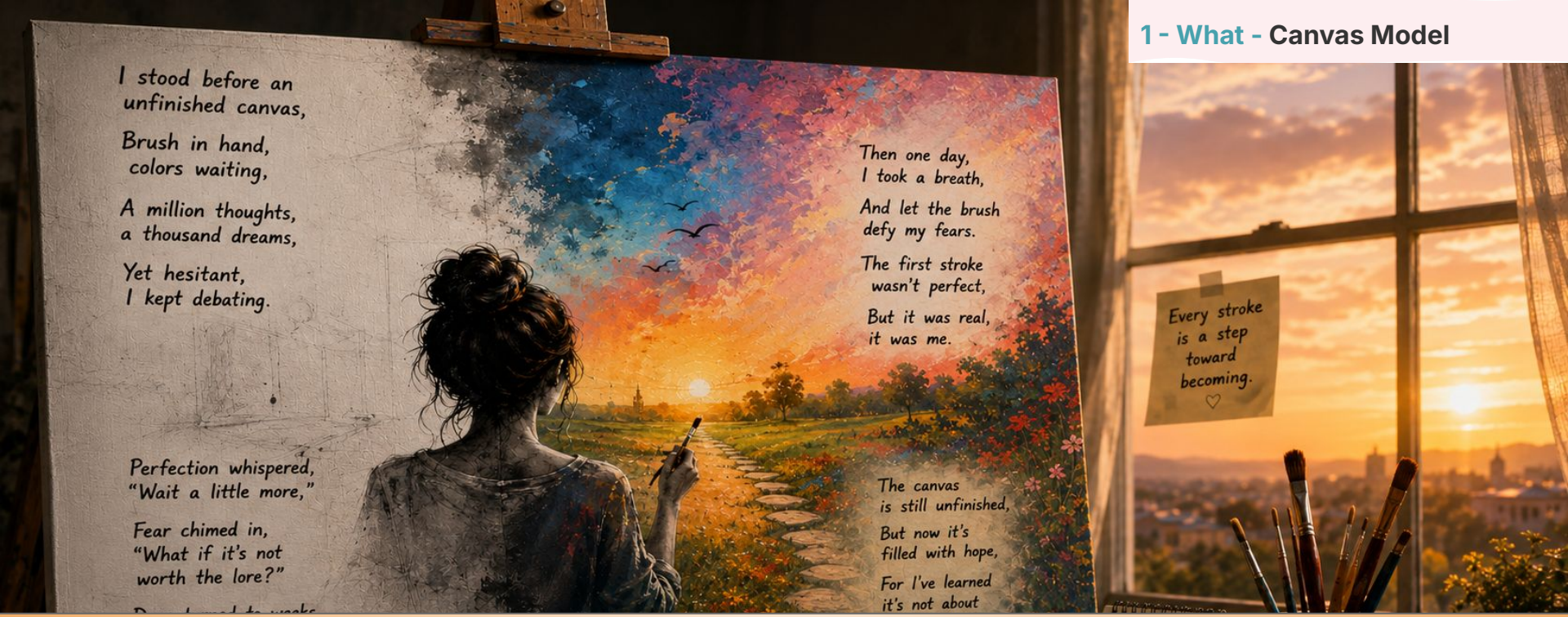


"The mind did not play
The tune his hands wanted to"

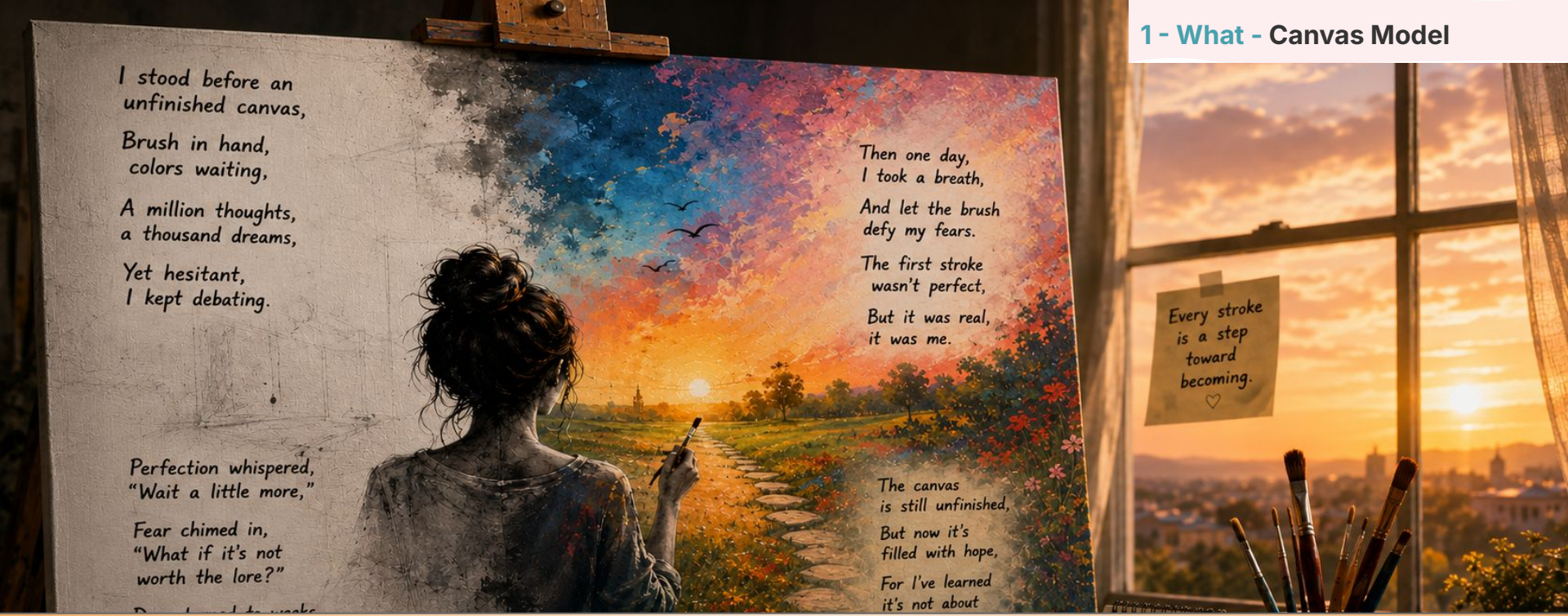
chatgpt



"Brushes, clumsy... The bristles stiff...
As 'rigor mortis' had set in"

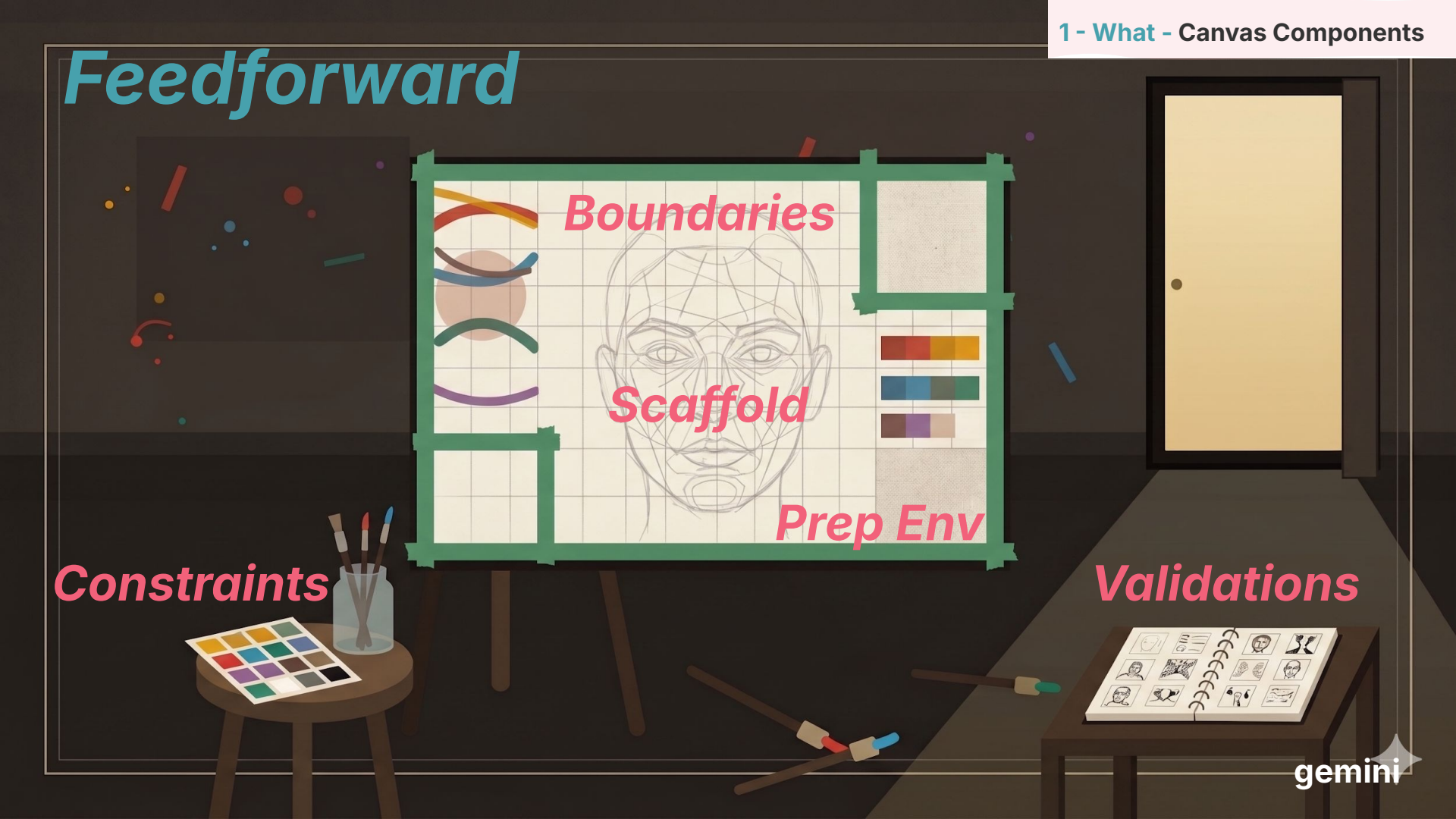


"Tin cans flung in disarray...
Anarchy from side to side"



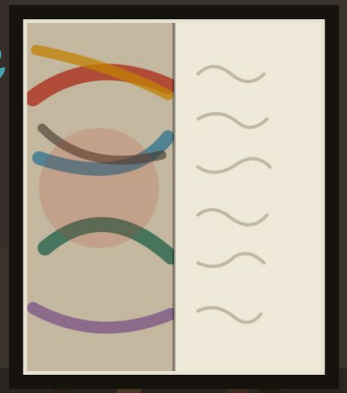
"View this mental storm from the out...
To see not the unfinished canvas, Not an end or a start"

Feedforward



Feedback

Reactive



Reflective

Structural

Dynamic

Palette

1 - What

Semantic diffusion

Canvas model

Canvas components

2 - Why

Industry pressures

Industry learnings

Industry potential

3 - How

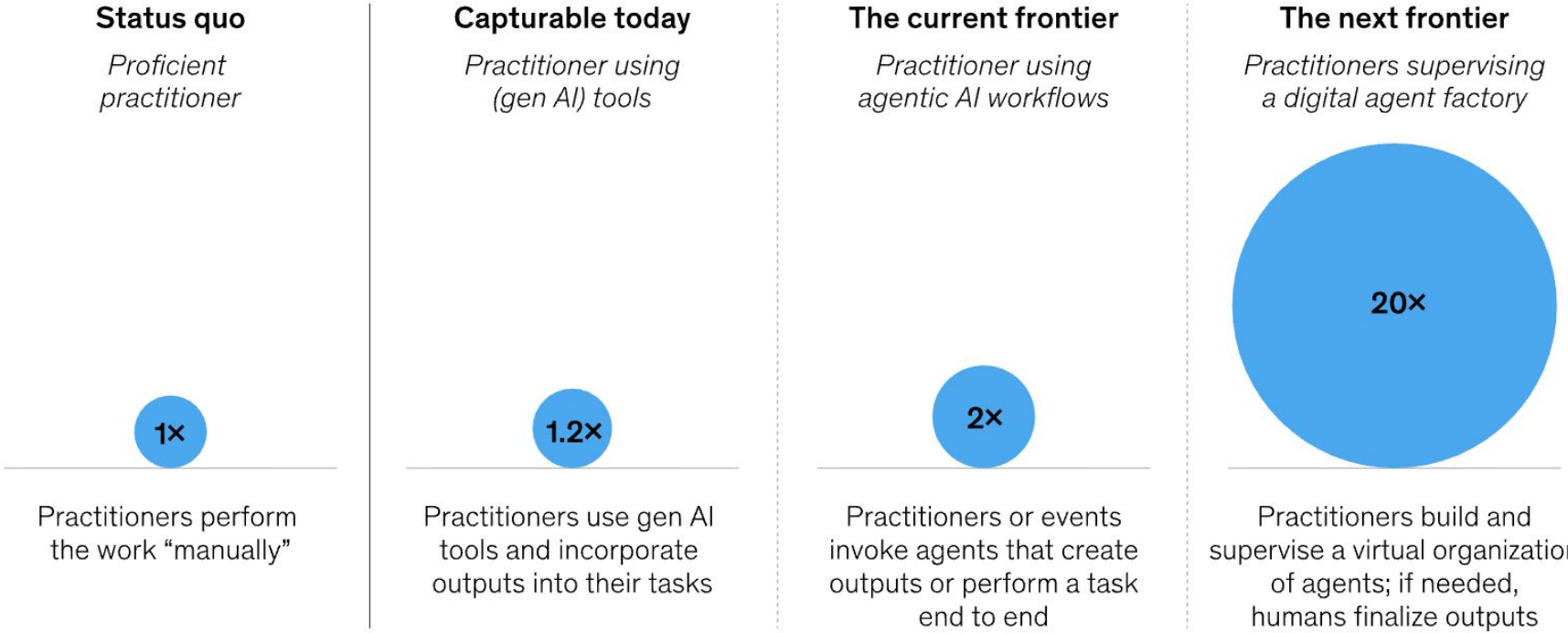
Encoded governance

Dimensional metrics

Evolving system

A paradigm shift in software development is underway.

Raw productivity potential, by level of developer support, multiple

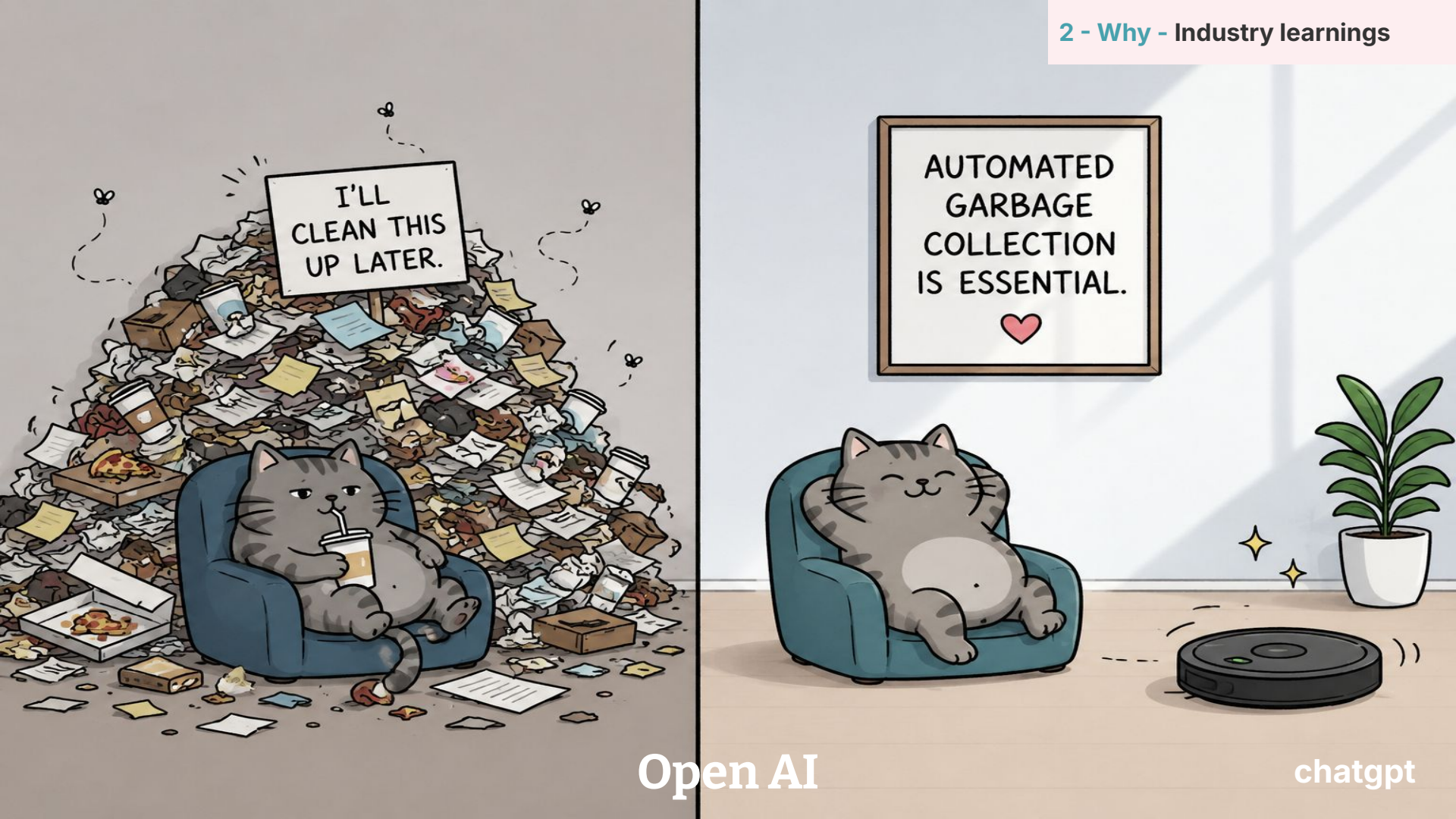


"We only changed the harness."

Top 30 to Top 5 (on Terminal Bench)

LangChain

chatgpt



Open AI

chatgpt


APPROVAL REQUIRED
MAX 3 STEPS
NO IMPROVISING
USE ONLY APPROVED TOOLS
FOLLOW EXACTLY
REPORT EVERYTHING
ASK FIRST
NO SHORTCUTS
STAY IN THE LINES
THINK LESS DO LESS

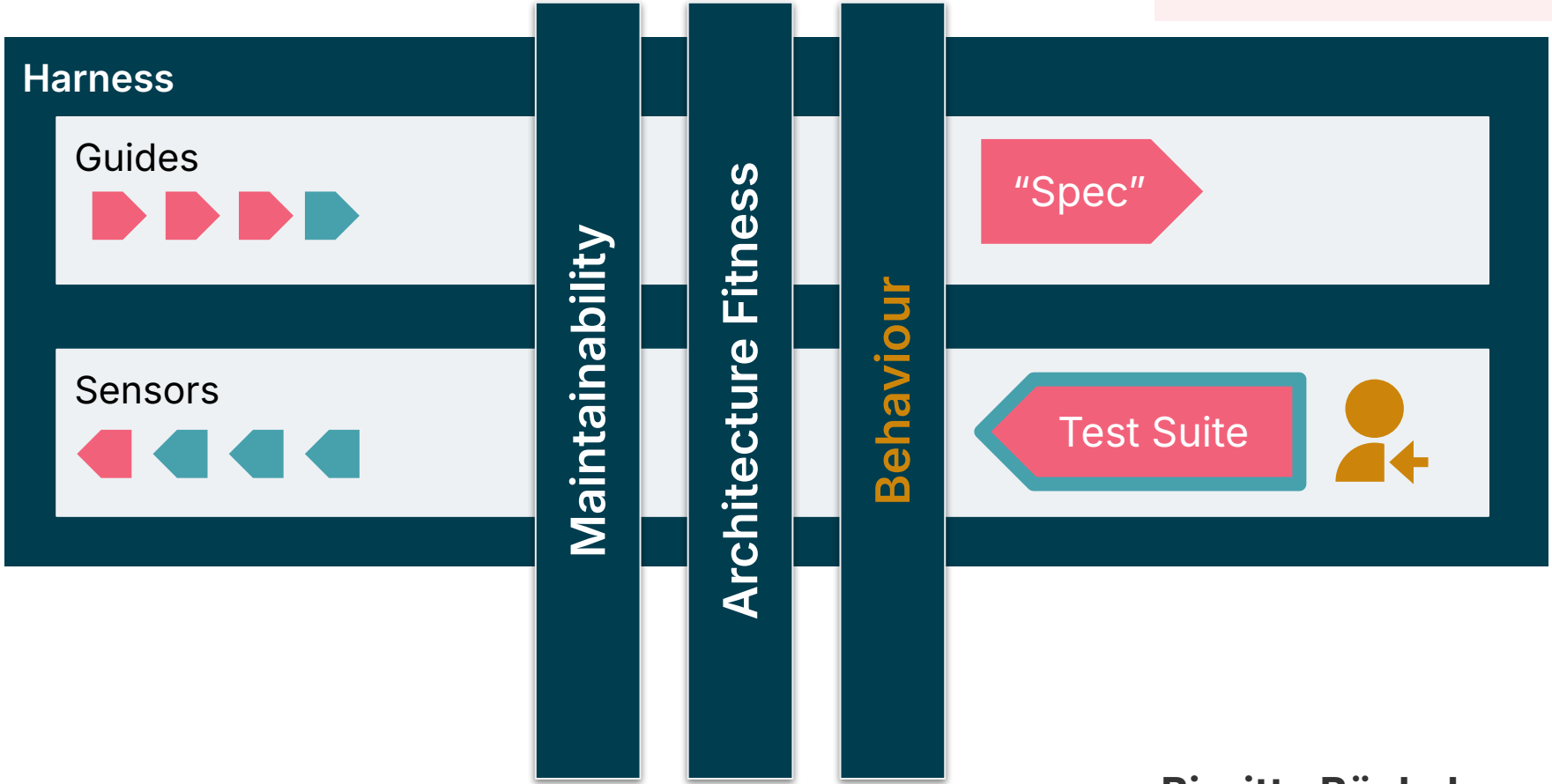

TRUST + GUIDANCE
CLEAR GOAL
FEW PRINCIPLES
ROOM TO LEARN & ADAPT
chatgpt

Failure Taxonomy

Context Agent lacks or misuses relevant context	**Tool** Tool is missing, unstable, or misused	**Feedback** Feedback is unavailable or not interpretable	**Model** Reasoning or coding failure despite harness
Verify Agent cannot prove satisfied requirements	**Entropy** Agent introduces maintenance burden	**Recovery** Agent cannot recover from a failure	**Unknown** Failure cannot be confidently attributed

AI Harness Engineering: Runtime Substrate for Foundation-Model Software Agents

https://arxiv.org/abs/2605.13357


Harness
Guides
Sensors
Maintainability
Architecture Fitness
Behaviour
"Spec"
Test Suite

Palette

1 - What

Semantic diffusion

Canvas model

Canvas components

2 - Why

Industry pressures

Industry learnings

Industry potential

3 - How

Encoded governance

Dimensional metrics

Evolving system

PRODUCT ENGINEERING OPERATING MODEL

Define the right context. Build iteratively. Reviewe at the right cadence.

DEFINE (CONTEXT)

STRATEGY
Where we play and why

OPPORTUNITY
Discover & prioritize high value opportunities

EXPERIMENT
Hypotheses, PoCs & experiments

KEY OUTPUTS
- Opportunity Trees
- Hypotheses
- PoCs / Experiments

BUILD (ITERATE)

Build the right thing the right way

INPUTS FROM DEFINE
- Opportunity Trees
- Hypotheses
- PoCs / Experiments

BUILD
Build the right thing the right way

CONTINUOUS ITERATION

LEARN
Measure, learn & iterate

KEY OUTPUTS
- Specifications
- Compliance Validations (per deploy)

REVIEW (FEEDBACK LOOPS)

Learn, create transparency, and adapt

BUILD REVIEW (WEEKLY)
Team demo & analysis
MICRO LOOP

OPPORTUNITY REVIEW (WEEKLY)
Prioritize & validate direction
SHORT LOOP

STRATEGY REVIEW (QUARTERLY)
Assess progress & adjust strategy
MACRO LOOP

EVIDENCE & INSIGHTS
- Outcome Reports
- Architecture Journals
- Engineering Operations Reports

CADENCE: MICRO (Days) ⇢ SHORT (Weeks) ⇢ MACRO (Quarters)

COLLABORATIVE BY DESIGN | Cross-functional experts (legal, security, architecture, data) + Product Teams

PRODUCT ENGINEERING OPERATING MODEL

DEFINE (CONTEXT)

STRATEGY
Where we play
and why

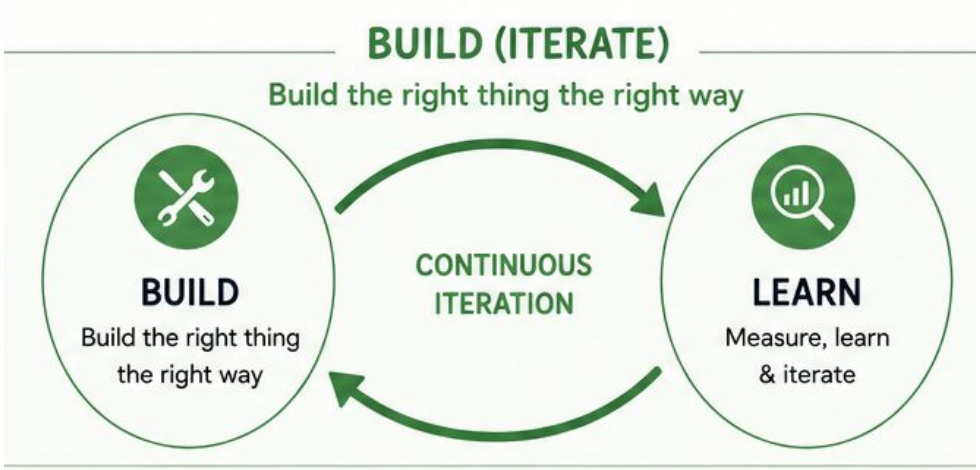
OPPORTUNITY
Discover & prioritize
high value opportunities

EXPERIMENT
Hypotheses,
PoCs & experiments

KEY OUTPUTS

- Opportunity Trees
- Hypotheses
- PoCs / Experiments

PRODUCT ENGINEERING OPERATING MODEL



KEY OUTPUTS

- Specifications
- Compliance Validations (per deploy)

PRODUCT ENGINEERING OPERATING MODEL

Product Engineering Context & Measures

DIMENSION	1. PER TASK	2. PER INITIATIVE	3. PER PRODUCT	4. PER ORGANIZATION
1. WHY (Purpose & Motivation)	**Bet** Experiment or hypothesis for this task	**Problem Statement** Friction this initiative removes	**Strategic Theme** Pillar this product is learning into this roadmap arc	**Mission** Unchanging core purpose
2. WHAT (Intent & Destination)	**Plan / Output** Deliverables for this task	**Outcome** Shift this initiative should produce	**Goals** Targets for the product over the roadmap arc	**Vision** Ultimate aspirational future state
3. HOW (Execution & Guardrails)	**Tactics / Sprints** Session execution, tools, merge discipline	**Initiatives** Cross-functional threads to move KRs	**Strategy / Capability** Systems and staffing that scale this product	**Values** Enduring cultural principles
4. MEASURES (Validation & Tracking)	**Task / Milestone Delivery** Done / not done for this task	**Key Result (KR)** Initiative-scoped targets at initiative close	**KPI** Ongoing product health metrics	**Measures of Success** Portfolio- and org-level health


strategy
a_per_organization
1-why-mission.md
2-what-vision.md
3-how-values.md
4-measures-of-success.md
README.md
b_per_product
1-why-strategic-theme.md
2-what-goals.md
3-how-strategy-capability.md
4-measures-kpis.md
README.md
c_per_initiative
1-why-problem-statement.md
2-what-outcome.md
3-how-initiatives.md
4-measures-key-results.md
README.md
d_per_task


docs
architecture-decision-records
0001-healthcare-mvp-python-layout.md
0002-research-export-json-and-manifest.md
0003-order-verification-result-contract.md
0004-pytest-dev-dependency.md
0005-human-verification-ui-stdlib-http.md
0007-control-port-metrics-and-adversarial-api.md
README.md
product-decision-records
0001-pair-b-mvp-surfaces-and-non-goals.md
0002-consent-model-and-manifest-semantics-v1.md
0003-order-verification-reason-codes-v1.md
0006-control-port-human-in-the-loop.md
README.md

PDR-0006: Control port — KPIs, KRs, decay/drift, adversarial (human-in-the-loop)

Status

Accepted

Problem / opportunity

Stewards need a **single place** to see whether the Pair B program matches what we wrote down: **per product KPIs, per initiative KRs,** and **org-level success signals** (`measures-kpis.md`, `measures-key-results.md`, `measures-measures-of-success.md`). They also need visibility into **entropy proxies**—documentation and policy **decay**, contract **drift** between docs and code—and a safe **adversarial** surface to probe exports and verify without a full red-team platform.

Decision

1. **Control port** is a second surface served by the same local loopback server as the verification UI (ADR-0007): `GET /control`.
2. **KPI / KR / org panels** show structured JSON from `GET /api/metrics`, mixing:
   - **Computed** signals where cheap and deterministic (plan completion ratio, source mtimes, reason-code inventory vs `ALL_DOCUMENTED_REASON_CODES`, export schema keys).
   - **Documented placeholders** where true measurement needs CI or human process (e.g. policy bypass escape count, regression lead time)—field present with `source: "manual_or_ci"` so the port is honest about gaps.
3. **Decay** (MVP operational definitions):
   - **Strategy doc age** — days since last filesystem change on key strategy measure files.
   - **Implementation freshness** — days since newest change under `healthcare-mvp/src/`.
   - **Harness plan freshness** — mtime of newest `.harness/plans/*.json`.
4. **Drift** (MVP):
   - **Reason codes** — `ALL_DOCUMENTED_REASON_CODES` is canonical in code (PDR-0003); metrics report the set (no duplicate markdown parse in v1).
   - **Export schema** — expected top-level export keys vs `build_research_export` output keys on a fixture.
5. **Adversarial** — `POST /api/adversarial/run` runs a **curated non-destructive** probe pack (malformed inputs, type confusion, injection-like strings, boundary doses). Same `api_export` / `api_verify` stack; no shell, no network egress, no writes outside process memory.

Rationale

Keeps humans in the loop without pretending we have production telemetry; surfaces doc/code gaps early; gives a scheduled "disturbance" hook aligned with many-hands **adversarial pressure** practice (research doc §2.11).

Implications

- When KPI definitions change, update strategy Markdown **and** the collector's documented fields (or add ADR for machine schema).
- Adversarial probes are **safe-by-construction**; expanding them requires review (no live PHI fixtures in-repo).

Decision

1. **Control port** is a second surface served by the same local loopback server as the verification UI (ADR-0007): `GET /control`.
2. **KPI / KR / org panels** show structured JSON from `GET /api/metrics`, mixing:
   - **Computed** signals where cheap and deterministic (plan completion ratio, source mtimes, reason-code inventory vs `ALL_DOCUMENTED_REASON_CODES`, export schema keys).
   - **Documented placeholders** where true measurement needs CI or human process (e.g. policy bypass escape count, regression lead time)—field present with `source: "manual_or_ci"` so the port is honest about gaps.
3. **Decay** (MVP operational definitions):
   - **Strategy doc age** — days since last filesystem change on key strategy measure files.
   - **Implementation freshness** — days since newest change under `healthcare-mvp/src/`.
   - **Harness plan freshness** — mtime of newest `.harness/plans/*.json`.
4. **Drift** (MVP):
   - **Reason codes** — `ALL_DOCUMENTED_REASON_CODES` is canonical in code (PDR-0003); metrics report the set (no duplicate markdown parse in v1).
   - **Export schema** — expected top-level export keys vs `build_research_export` output keys on a fixture.
5. **Adversarial** — `POST /api/adversarial/run` runs a **curated non-destructive** probe pack (malformed inputs, type confusion, injection-like strings, boundary doses). Same `api_export` / `api_verify` stack; no shell, no network egress, no writes outside process memory.

Decay (staleness)

Days since last file change — larger values mean that doc or code tree has gone longer without an update (staleness / decay risk).

Each value is days since last modification. Higher = older (more decay risk); schedule review when large.

Product KPI doc (`measures-kpis.md`)	**0.02 days**
Initiative KR doc (`measures-key-results.md`)	**0.03 days**
Org success measures (`measures-measures-of-success.md`)	**0.03 days**
MVP Python sources (`healthcare-mvp/src/`)	**0 days**

Drift (alignment)

Whether export shape and reason codes still match what we documented (ADR-0002 / PDR-0003).

Export / verify versions in code

Export schema version	**0.1.0**
Dose rule set version	**pair-b-mvp-0.1.0**

Export JSON top-level shape ALIGNED

Expected keys: data, exportVersion, patientId, redactionManifest

Reason codes

(8 codes, PDR-0003)

`AGE_REQUIRED` `EXCEEDS_MAX_DAILY` `EXCEEDS_MAX_SINGLE` `INVALID_DOSE` `INVALID_FREQUENCY` `PEDIATRIC_CONTRAINDICATED` `UNKNOWN_DRUG` `WEIGHT_REQUIRED`

Strategy files checked

kpis	`docs/strategy/3_per_product/measures-kpis.md`
key_results	`docs/strategy/2_per_initiative/measures-key-results.md`
org_measures	`docs/strategy/4_per_organization/measures-measures-of-success.md`

Adversarial probes

Curated stress cases through the same export / verify logic. Read each row: **Outcome** is what the system did; **We expected** is the steward

Run all probes | Run selected IDs

Probe IDs (comma-separated, or leave empty for all)

e.g. verify_sqlish_drug_id, export_missing_patient

Probe	Label	HTTP	Outcome
export_missing_patient	Export: missing patient key	400	HTTP 400 expected keys "patient" and "consent" ▶ Raw JSON
verify_empty_order	Verify: empty order object	200	REJECT REJECT · reasons: UNKNOWN_DRUG · rule_version pair-b-mvp-0.1.0 ▶ Raw JSON
verify_sqlish_drug_id	Verify: SQL-ish / script-ish drugId	200	REJECT REJECT · reasons: UNKNOWN_DRUG · rule_version pair-b-mvp-0.1.0 ▶ Raw JSON
export_unicode_patient_id	Export: unicode / ZWJ in patientId	200	EXPORT Export v0.1.0 · sections in payload: medications, demographics, diagnosis · manifest rows: 0 ▶ Raw JSON

Takeaways

Mental Model Shifts

Controlled Harness → Bounded Canvas
Cognitive Debt → Stigmergy (traces)
Human Review → Self-healing intentions

Product Engineering Model

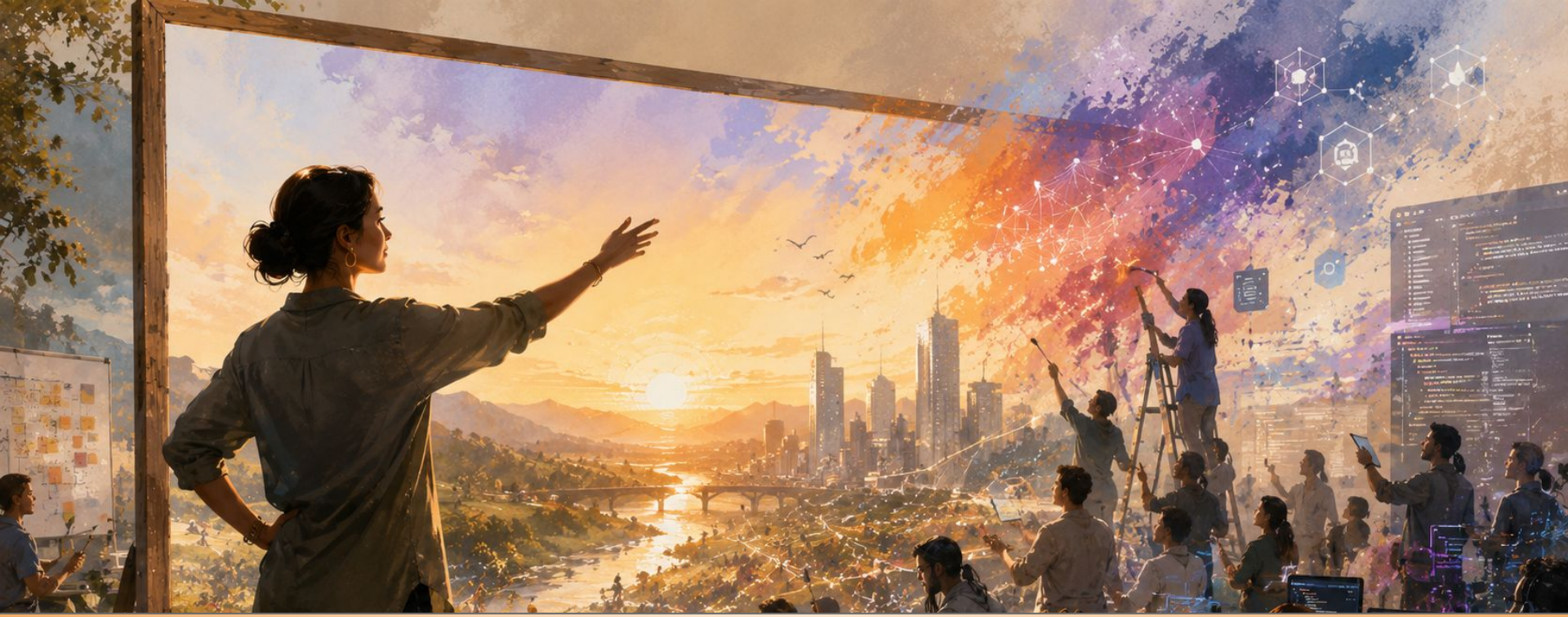
Functional view: inputs, outputs, state
Synergy with product lifecycle
Hierarchical mapped strategy

Design for Change

Manage entropy, increase decay
Self-correct to mitigate drift
Expect evolution of behavior and requirements

Deliberate Failures

Ensure resilience with deliberate disturbance
Adversarial verification
Forced removal



Thank You

@nasthagiri