



FRONTIER AI

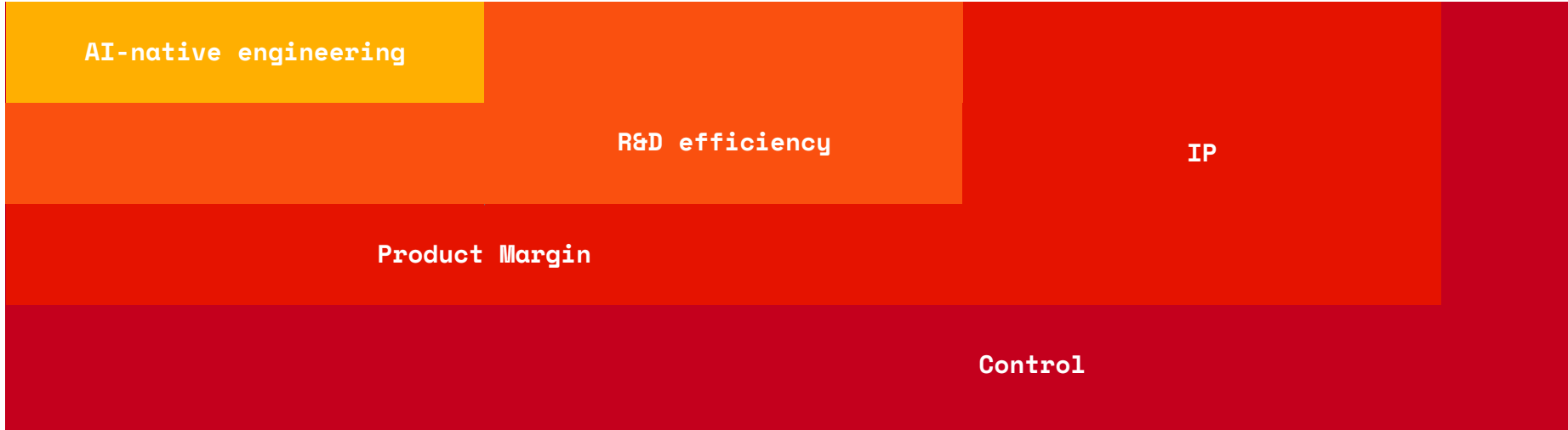
MISTRAL

Causal Continuity

The Role of Domain-Shifted Models in the
Automotive Industry

MISTRAL

If AI defines R&D, AI inherits **margin, IP and control**



Cycle time is the denominator of product margin



Three goals, one metric

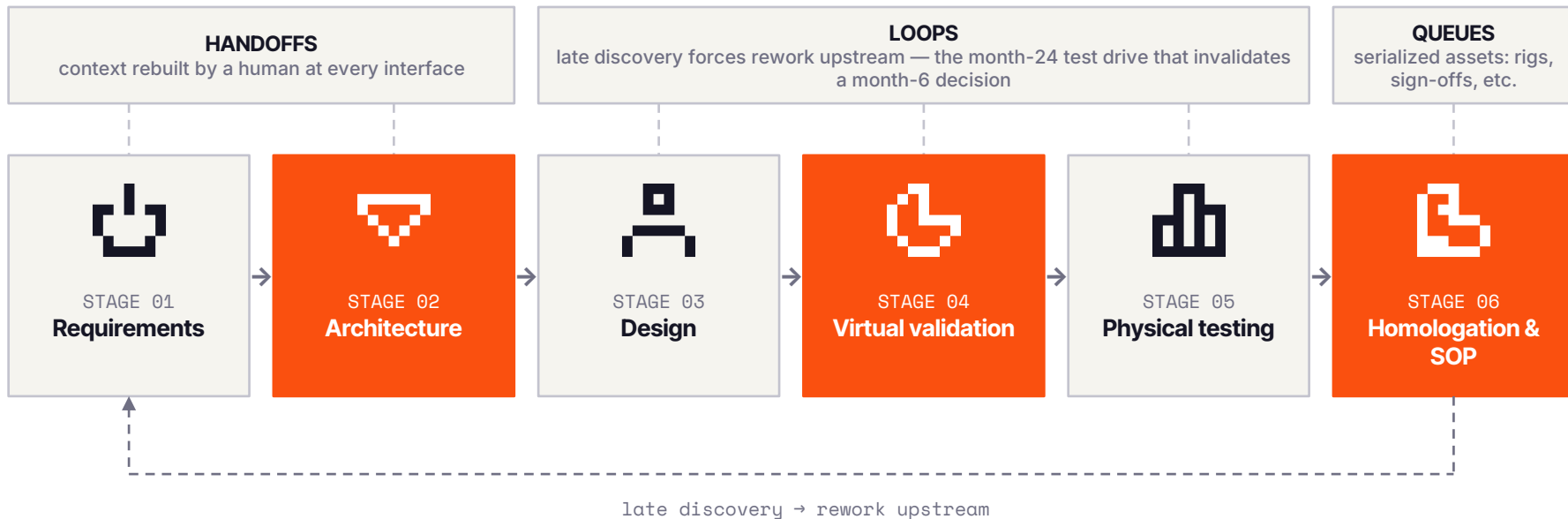
- Compress requirement → SOP
- Frontload insight — know failures before they are physical
- Absorb complexity — more variants and regulation without linear time growth



Four faces of the same time tax

- Silos — handoff time
- Serialized physical testing — waiting time
- Expertise locked in heads — search time
- Expanding validation burden — proving time

Process speed is captured in the **connections**



What crosses every interface is context — requirements, boundary conditions, assumptions, models, data, decisions — and it is re-created by hand.

Frontier intelligence is episodic. The process needs **ambient**.



Finally, LLMs can carry it

- memorize the rationale
- handle diverse modalities
- reason along the process

THE OPPORTUNITY

Connecting the dots



But, the economics break

- Frontier price/token $\times$ latency $\times$ data governance
- Thousands of always-on touchpoints
- Ambient need $\times$ episodic economics

THE EQUATION

Ambient $\neq$ scalable



Unless, domain-shifted

- do not carry the weight(s) of the whole world
- instead, purpose-build models for what is relevant to your task
- Let it continuously evolve

THE PRIMITIVE

Domain-shifts for maximum efficiency

Causal continuity — and how it's build

Causal continuity

Unbroken causal understanding along the engineering process — every why traceable, every upstream change propagating correctly downstream.

Domain understanding

Interprets the context — 'a coolant valve has entirely different safety obligations depending on whether it's in a battery thermal loop, an A/C compressor, or a powertrain cooling circuit.'

Domain reasoning

Reasons the flow — 'if acceleration curve looks like this, parameter setup of next simulation should be like that'

Domain capabilities

Knows how to handle the tools along the chain — 'for a CFD simulation you need as input params viscosity, density, turbulence model parameters'

Derivations you trust

Every model embedded in the process becomes a link in the chain itself — a model grounded in the domain's causal structure is a link you trust, and own.

Self-Distillation Finetuning

Prompt asymmetry — one question, two prompts

STUDENT no context · ships at inference

system: Answer the question.
user: Why does fast charging
 slow down above 80% SoC?

answer: "to avoid overheating?" — a guess



KL

student
pulled
towards
teacher

TEACHER with docs · training only

system: Answer using the context.
user: [same question]
context: pack & BMS docs —
 CC-CV charging curve,
 anode overpotential limits...

answer: "the CV phase — current
 tapers near full" — grounded

HOW IT WORKS — ONE LOOP

01 Generate — the student answers on-policy (vLLM); the teacher gets the docs

02 Learn — one question, two forward passes — the student is pulled toward the teacher via KL

03 Update — the teacher becomes an EMA of the student — a slightly better self, every round

SAME WEIGHTS

the teacher isn't a bigger model — it's an EMA copy of the student itself, one step ahead.

THE ESSENCE

What it does with the docs, it learns to do without them.

SFT Traces

HOW IT WORKS

- 01 Record** — run the tasks, capture every thought, action and observation
- 02 Filter** — keep only the traces with a verified outcome
- 03 Fine-tune** — train on them; loss only on the agent's own steps

WHAT THE MODEL LEARNS

PLAIN SFT

prompt → answer. The model sees the answer, but not how it was found.

SFT ON TRACES

the whole decision process — which tool to call, how to react to surprises, when to stop.

An engineering example

TASK

Diagnose the capacity drift in pack 8.

THINK

"Pull the cyclor logs before changing any config."

ACT

```
read pack8_cyclor_logs --last 30d
```

OBSERVE

DRIFTED — cells 12 & 47 down 6% vs neighbors

THINK

"Balance never engages → check the BMS config."

ACT

```
run check_balance --module 3
```

RESULT

Found — threshold 20 mV, reset to 10 mV, drift flat.

THE ESSENCE

Teach the path, not just the answer.

Harness Engineering

One task, two harnesses — same model, same task

task: Which cells drift out of balance in pack 8?

HARNESS A

minimal · single-shot

prompt: task text only
sampling: greedy
tools: none
verifier: none

answer: guesses from memory —
 wrong cells

=

SAME
WEIGHTS

HARNESS B

engineered · looped

prompt: task + tool docs
loop: sample → act → observe
tools: read_log, run_check
verifier: cross-check the drift

answer: cells 12 & 47 — confirmed by
 the log

Same checkpoint, same task — the only variable is the harness: context, tools, a loop, a verifier.
That's the delta between a wrong answer and a verified one.

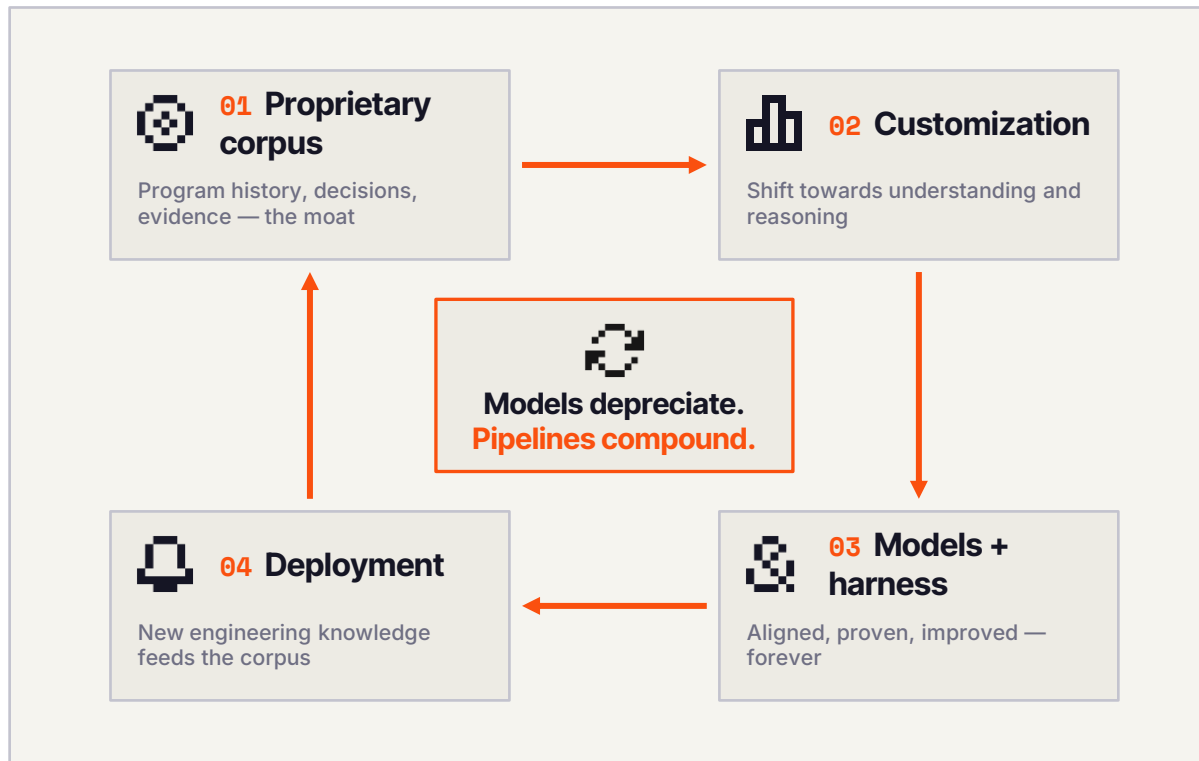
HOW IT WORKS — ANATOMY OF A HARNESS

- 01 Assemble** — prompt, few-shot examples, tool schemas, sampling params — everything the model will see
- 02 Execute** — the loop: sample → parse → call tools → observe — repeat until done
- 03 Verify** — an explicit check decides success — tests, rubrics, graders. No vibes.
- 04 Log** — capture the full trace — every run reproducible, debuggable, comparable

THE ESSENCE

A model is a component — the harness makes it a system.

The loop compounds



NO FINAL STATE

- A checkpoint is a perishable good
- Organization evolves, so does your model

THE COMPOUNDING EFFECT

- Every loop, improves the outcome
- Assets are durable and scalable

THEY CAN'T BUY YOUR LOOP

- Competitors can rent the same intelligence
- They can never buy your loop

The AI layer defines margin, IP, independence.

01 Hold the chain — causal continuity, end to end

02 Own the corpus — the moat is your program history

03 Build in systems — models depreciate, pipelines compound

Thank you.

mistral.ai