The future of AI isn't just intelligent. It's accountable.

Alex Porcelli

Mark Cuban
@mcuban

𝕏.com

I'm coming to the conclusion that the biggest challenge for Enterprise AI, and AI in general , as of now, is that it's still impossible to make sure that everyone gets the same answer to the same question, every time.

Which is a great response to the doomers. AI doesn't know the consequences of its output.

Judgement and the ability to challenge AI output is becoming increasingly necessary, and valuable.

Which makes domain knowledge more valuable by the second.

Am I wrong ?

9:24 AM · 5/4/26 · **1.3M** Views

 1.8K  675  6K  1.5K 

Accountability, not just consistency.

Mark isn't wrong. **Consistency matters.**

But in enterprise AI, consistency is only the symptom. The deeper requirement is accountability.

When AI participates in approving a loan, classifying risk, paying a claim, or applying a treatment protocol — a "good answer" isn't enough.

AN ACCOUNTABLE DECISION IS

01 **Reproducible**

02 **Explainable**

03 **Aligned with policy**

04 **Governed**

versioned, reviewed, and approved before it ever goes live.

Drools

The most adopted open-source rule and decision engine in the world.

20 years spent making policy executable.

I've worked on this layer since the early **Drools** days.

I help lead the open source as an **Apache KIE PPMC** member, and led the commercial products built on it at **IBM** and **Red Hat**.

FINANCIAL SERVICES · INSURANCE · HEALTHCARE · GOVERNMENT

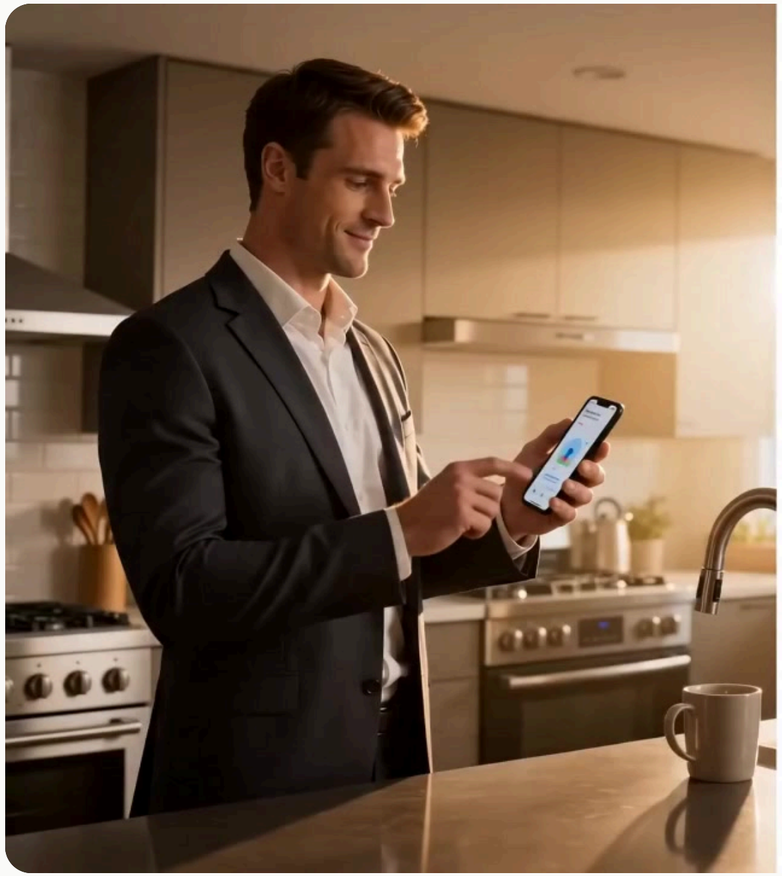
KIE
Knowledge platform

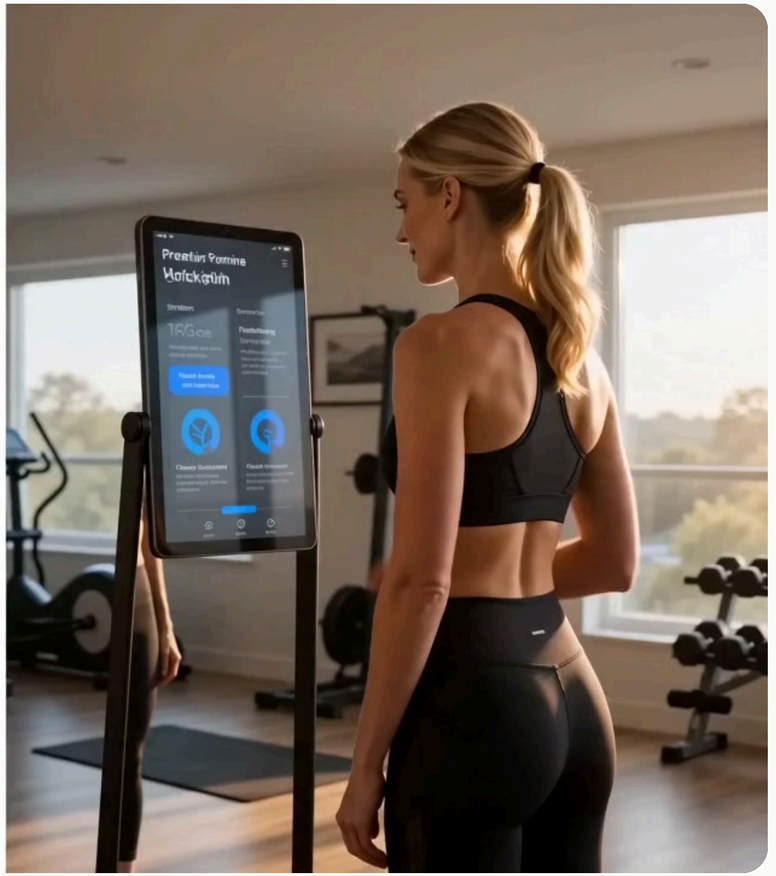
jBPM
Process orchestration

Kogito
Cloud-native automation

Same AI.
One gets legacy. One gets leverage.

Both start the morning with an agent. One walks into work and is handed a 30-year-old form — the other walks in and keeps it.





We hired a human.
We gave them data entry.

A trained professional spends the day **collecting context, clicking through screens, and re-typing it into a 100-field form** — so the system can make the call.

It's the one job they shouldn't be doing — **and the exact work AI should absorb.**

FREE THE HUMAN FOR WHAT MATTERS

- Prevent fraud and coercion
- Read local market signals
- Grow trusted relationships
- Own high-value exceptions
- Recover trust when it breaks

Automating the typing is ~~RPA~~.
AI should make the call.

YESTERDAY — RPA

A bot fills the form faster.

Same workflow, same screens, same 100 fields. The decision still waits for a human to assemble it.

The work is unchanged.

THE SHIFT — AI MAKES THE CALL

AI captures context naturally, talks to the customer, and runs the decision itself.

No form. No adapter. The experience changes — the bank shows up **where the customer already is.**

→ A different experience, not a faster one.

LIVE DEMO

Let's give the LLM one customer to classify.

No rules. No policy. Just the model, a profile, and a confident answer.

DEMO RECAP

Confident. And wrong.

openclaw · classify-customer

```
> classification: AAA

rationale:
customer self-described as "responsible," maintains a
consistent gym routine, and follows a disciplined nutrition
plan.
```

It’s funny because it failed loudly.

The dangerous version fails **politely** — plausible, polished, and unauditable.

Same architecture. Same model. The reasoning was just vibes.

I know what you're thinking...

REFLEX 01

"Let AI generate the deterministic code."

THE INSTINCT

"If the LLM is unreliable, ask it to write code that isn't."

Generate Python, SQL, a workflow — and just run it.

Deterministic ≠ governable.

Generated code runs. But the policy itself disappears into control flow that's **hard to review, version, audit, or explain.**

You traded one black box for another.

REFLEX 02

"Add RAG, MCP, logs. Done."

THE INSTINCT

"Plug in the right context, give it the right tools, capture the logs."

Better retrieval. Better access. Better observability.

All useful. None of it is the decision.

The decision logic still lives scattered across **prompts, retrieved docs, tool calls, and model behavior** — never named, never versioned, never **governed**.

You logged the outcome. You can't reproduce the reasoning — and there's nothing for governance, risk, or audit to actually review.

ALETYX TECHNOLOGY IN PRODUCTION · ANONYMIZED FINANCIAL-SERVICES CUSTOMER

One of many use cases. Real data from production.

20M

Real data pipeline CUSTOMER-PROFILE CLASSIFICATION STEP

18k+

<5ms P99 PER DECISION · **24/7** IN PRODUCTION

Classifying customers as **AAA, AAB**, and so on — driving products, pricing, fees, and downstream decisions. Powered by **Drools**, the most adopted rule and decision engine in the world.

Executable policy isn't a theory.

Decision Models.

A way to make a policy **explicit**: named inputs, named outputs, named rules. Readable by a business analyst. Executable by a runtime.

NAMED VERSIONED EXECUTABLE TRACEABLE

Business people model the world in spreadsheets.

A **decision table** is the most common way to write down policy: rows are rules, columns are conditions, the last column is the result.

customer-profile.xlsx — DECISION TABLE

#	WHEN · CONDITIONS			THEN · RESULT
	Nationality	**Income**	**Age**	**Profile**
1	Foreign	–	–	BBC
2	US	< 40000	< 25	BBC
3	US	≥ 80000	35..59	AAA
4	–	–	–	BBB

Read row-by-row. Each row is a rule. Row 2 fires for this customer → profile **BBC**.

LIVE DEMO — MODEL THE POLICY

From spreadsheet to executable policy.

The familiar piece — a decision table — becomes a **DMN model**: testable, versioned, fast enough for production.

DEMO SEQUENCE

01

Start from the spreadsheet

02

AI assistant generates the DMN model

03

Generate tests · run · trace

04

Publish · execute · explain

SWITCHING TO TERMINAL…

Not a generic API. A business skill.

SKILL.md — generated-from: decision-model

```
---
name: classify-customer
description: Governed customer classification.
license: Proprietary
metadata:
  unit:    simple_decision_classification
  model:   classify_customer
  version: latest
  source:  decision-model
---

## Scope
Deterministic classification from
Nationality, Annual Income, Age.
Out of scope → stop. Never infer.

## Reporting
Lead with the decision. Cite the policy.
```

WHAT THE AGENT RECEIVES

Discoverable by name

The description tells the agent when to call it.

Scoped, not open-ended

In scope and out of scope are explicit. No padding.

Policy as the source of truth

Reporting cites the unit, model, and version.

The agent plans. The skill decides.

Every decision leaves a trace.

Inputs, outputs, model version, execution ID — captured for every call. Ask the system **"why?"** and it answers.

monitoring / executions — LAST 7 DAYS

TIMESTAMP	STATUS	VER.
05/28/2026, 15:05:18	SUCCESS	3
05/28/2026, 15:04:39	SUCCESS	1
05/28/2026, 15:04:08	SUCCESS	1
05/28/2026, 12:43:10	SUCCESS	1
05/28/2026, 12:42:35	SUCCESS	1

monitoring / execution-detail — EXPLAIN

EXECUTION ID aa829f4e-f718-4bcf-8d62

MODEL classify_customer

STATUS SUCCESS

VERSION 3

MODEL HASH b6f4b4a4a5dfcb1b4eb18fdc441c426d43b0c8138588034dd13817ff69e5d8d1

INPUT

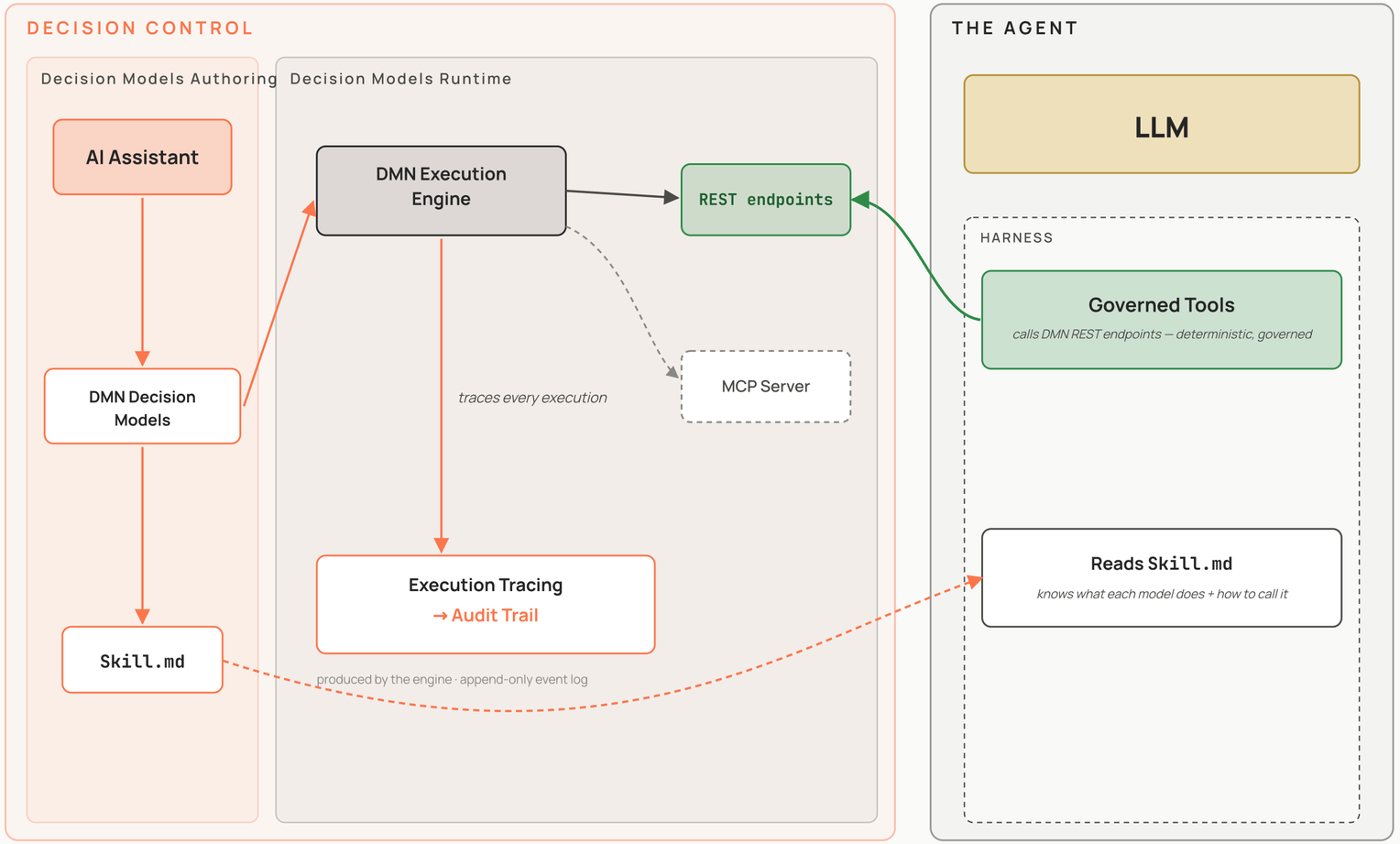
Nationality	Brazil
Annual Income	100000
Age	25

OUTPUT

classify_customer	AAA
Annual Income	100000
Nationality	Brazil
Age	25

The architecture, end-to-end.

Decision control on the left — author, run, trace. The agent on the right — LLM with governed tools.



LIVE DEMO

Now with the decision model in place.

The customer that earned a confident **AAA** on gym-routine vibes. Ask the question again — this time the agent has to defer to the decision skill, and it owes us a trace.

ONE MORE THING…

Wire a decision model into NVIDIA NeMo as a guardrail.

KEY TAKEAWAYS · WHAT TO WALK OUT WITH

Three things to take back.

01 THE PATTERN

Decision models as agent skills.

Existing enterprise decision models and rule engines become agent skills — for deterministic, explainable execution.

02 THE CHOICE

Decision model vs. regular code.

When to use a decision model instead of regular code in an agentic system — and the architectural trade-offs that drive that choice.

03 THE PAYOFF

Business logic, separated.

Isolating decisions in a dedicated model separates business logic from control flow — so teams, including non-developers, can evolve, test, and validate them with a narrowed focus.

Let AI handle ambiguity.
Give it governed decision skills.

SCAN ME

GitHub

SCAN ME

LinkedIn