ENTERPRISE UX & PRODUCT STRATEGY LEADER

Deterministic vs. Probabilistic Systems

Stochastic reasoning, recursive thinking, and what UX teams need to understand about each

Rusty Metty
rustymetty.com

Executive summary

One of the most useful distinctions I've found for explaining AI inside large organizations is the difference between deterministic systems and probabilistic ones. It's a useful distinction for engineers. It's an essential one for UX and product teams, because it describes not just how these systems behave, but how the people using them need to think differently depending on which kind of system they're standing in front of.

Two kinds of systems

Most enterprise software has traditionally been **deterministic**. A user takes an action, the system follows a defined rule, the database updates, and the result is predictable, repeatable, and auditable. These systems are essential for transactions, identity, eligibility, enrollment, compliance, payments, and permissions — anywhere consistency isn't optional.

AI systems behave differently. They infer, summarize, generate, compare, recommend, and surface patterns. Their outputs are **probabilistic** — genuinely useful, but they require context, evaluation, and human judgment in a way a deterministic rule never did.

Stochastic reasoning: how the model thinks

AI systems, particularly large language models, reason stochastically. They sample likely continuations from learned patterns rather than executing fixed logic. Every output carries fluency and confidence — but fluency is not the same thing as correctness. A stochastic system can produce a confident assumption dressed in the grammar of evidence: an answer that reads as validated because it's well-formed, not because it's true.

This is not a flaw to be engineered away. It's the nature of the tool. Treating a probabilistic system's output as if it carried deterministic certainty is where most AI-adoption failures actually originate — not in the model, but in the expectation brought to it.

Recursive thinking: what humans must do in response

If the model reasons stochastically, the human counterpart has to reason **recursively**. Recursive thinking is the metacognitive loop a person runs when evaluating stochastic output: not just reading the answer, but questioning the assumptions behind how the model likely arrived at it, then questioning the assumptions behind that judgment, before acting.

Deterministic systems reward linear trust — if a rule fired correctly once, it will fire correctly again under the same conditions. Probabilistic systems require recursive trust: each output needs to be re-evaluated against what's changed, because trust doesn't transfer automatically from one output to the next, even when the prompt looks the same.

Why this matters for UX specifically

UX teams are the ones who design how humans actually encounter probabilistic systems — how uncertainty gets communicated, how confidence is displayed without being overstated, how a correction or recovery path works, and how much recursive thinking a design asks of the user versus absorbs on the user's behalf.

This is not a new problem for me. Early in my career, designing Cora — an enterprise conversational assistant at Microsoft — meant answering the same question in a different form: how does a system stay useful and trustworthy when its responses are generated, not guaranteed? The tools have changed. The underlying design question — how much certainty can this interface honestly claim, and how does it invite the right amount of human scrutiny — has not.

Recent market evidence backs this up directly. A 2026 UCLA study of 2.26 million freelance contracts found that as AI narrows the gap between weak and strong performers' output, clients increasingly hire on price alone — human capital signals like credentials and reputation lost 7.8% of their hiring weight after ChatGPT's launch, and the discount kept deepening for three years. The researchers' conclusion mirrors this paper's: as execution becomes commoditized, the scarce, differentiating skill is judgment — deciding what to build, evaluating AI output, and knowing when to override it.

Source: Siddiq & Zhang, UCLA Anderson School of Management (2026), as analyzed by Jakob Nielsen, "AI Turns Competence into a Commodity," UX Tigers, 2026 (uxtigers.com/post/ai-skills-gap).

A practical framework for teams

Instead of asking whether AI should be used at all, teams get more value asking:

- What kind of task is this — does it need deterministic consistency, or does it tolerate probabilistic variation?
- What level of variation is acceptable in the output?
- Where does a human need to review before this moves forward?

- What evidence or control does the person on the other end of this actually require to trust it?

Those four questions do more to prevent AI-adoption failures than any policy document, because they force the deterministic-vs-probabilistic distinction into every individual decision, rather than settling it once at the top of the organization.

Conclusion

The best enterprise systems are hybrid by design: deterministic where consistency is non-negotiable, probabilistic where inference and generation add real value, and deliberately paired rather than left to blur into each other. The skill teams need to build isn't blanket AI skepticism. It's recursive judgment, applied consistently, every time a stochastic system hands back an answer that sounds sure of itself.