MCP Dev Summit Bengaluru

Why Agents Make Different Decisions with the Same Tools

The hidden non-determinism in agent tool selection and how we fix it before MCP scales to production.

Hi, we are 👋

Aditya Oberai

DevRel Lead

@Appwrite

Jyoti Bisht

Senior DevRel

@Harness

Animesh Pathak

DevRel

@Harness

Same agent. Same tools. Different decisions.

AGENT A

```
salesforce_query
email_send
slack_message
jira_create
```

→ Completes task in 10 seconds

AGENT B

```
salesforce_query
email_send
slack_message
jira_create
```

→ Hangs. Calls wrong tool. Fails.

Same prompt. Same MCP servers. Same context. **Different outcomes.**

A STORY YOU'VE LIVED

The silent degradation pattern

DAY 1	DAY 2-7	DAY 8	DAY 9	DAY 10+
✓	✓	↻	↓	?
Deploy	**Stable**	**Model Update**	**Drops**	**Why?**
Works locally	90% success rate	Routine upgrade	70% success rate	No clear cause

You didn't change anything. The tools didn't change. The prompts didn't change.

So what changed?

STAKES

Why this matters

Production Scale

You can't deploy what you can't predict. No SLAs. No reliability guarantees.

Compliance

"Why did your agent make this decision?" In regulated industries, you must be able to answer.

Enterprise Adoption

Enterprises don't adopt "works most of the time." They need predictability.

> This is the problem standing between MCP and **production-grade infrastructure.**

ROOT CAUSES

Five sources of divergence

None of them are tools. All of them shape tool selection.

01 **Temperature & Sampling**
Randomness in token prediction

02 **Model Version Changes**
Different training, different preferences

03 **Context Window Position**
Primacy bias favors what's listed first

04 **Tool Schema Ordering**
Sequential structure affects selection

05 **Schema Verbosity**
Description length influences selection

SOURCE #01

Temperature & Sampling

✗ temperature = 0

Greedy. Always picks highest probability token.

```
# Deterministic
same_input → same_output
same_input → same_output
same_input → same_output
```

✓ temperature = 0.7 (default)

Samples from distribution. Built-in randomness.

```
# Non-deterministic
same_input → output_A
same_input → output_B
same_input → output_C
```

Tool selection is just token prediction. Temperature affects **which tools your agent chooses.**

SOURCE #02

Model Version Changes

CLAUDE 3.5

Trained on data ending 2024-04.

```
input: "find customer info"
→ prefers salesforce_query
```

CLAUDE OPUS 4.5

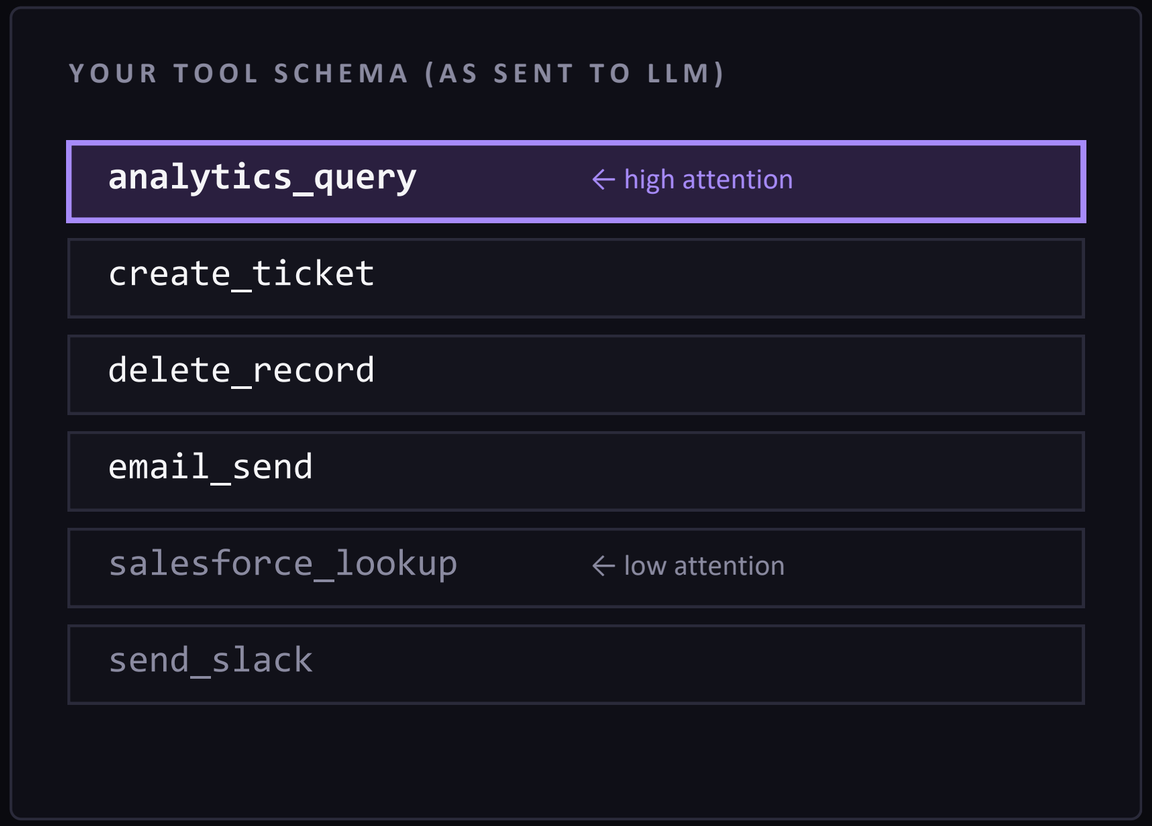
Trained on data ending 2026-01.

```
input: "find customer info"
→ prefers crm_lookup
```

- → Different training data shifts which tools "feel right" to the model.
- → Different RLHF examples reinforce different tool-calling patterns.
- → No changelog tells you this. You discover it when error rates rise.

SOURCE #03

Context Window Position



Primacy bias

LLMs remember the first thing they see best. Attention weights skew toward early tokens.

Tools listed first have a **visibility advantage** in tool selection.

Was your alphabetical ordering intentional? **Probably not.**

SOURCE #04

Tool Schema Ordering

✗ Alphabetical (Default)

- analytics_query
- create_ticket
- delete_record
- email_send
- list_files
- salesforce_lookup

No semantic structure. Random adjacency.

✓ Semantic Groups

READ

- analytics_query
- salesforce_lookup
- list_files

WRITE

- create_ticket
- delete_record

NOTIFY

- email_send

SOURCE #05

Schema Verbosity

✗ Sparse

```
name: "create_record"
description: "creates a record"
```

Low semantic surface area. Easily ignored.

✓ Verbose

```
name: "create_record"
description: "Creates customer record with audit
trail, validates email, sends notification..."
```

More keywords. More semantic matches. Wins attention.

Your efficient, simple tool loses to verbose, bloated tools — **even when it's the better choice.**

You don't get one bias. You get all five.

temperature + model + position + order + verbosity = **chaos**

~10%	50%+	∞
EACH SOURCE ALONE	ALL FIVE COMBINED	VARIANCE ACROSS PROMPTS

This is why your agent works 90% today and **70% next month.**

Until we solve this, MCP doesn't scale

Production Failures

Works in dev. Fails in prod. No way to reproduce. Debug cycle takes days.

Compliance Failures

Auditor asks "why did your agent choose this tool?" — you can't answer.

Enterprise Blockade

Need 99.9% reliability. Need SLAs. Need explainability. None of which we offer.

Without solving this, MCP remains a **hobby tool**. With it, MCP becomes **production infrastructure.**

AND, HUGE costs

Every wrong tool call costs money.

Retry tax

Wrong-tool calls trigger retries. Multiplies your token spend.

Schema bloat

Verbose tool descriptions inflate input tokens on every request.

Model-update tax

Each upgrade burns engineering hours on prompt rewrites and re-evals.

Deployment freeze

Can't trust agents at scale → underdeploy → value left on the table.

PROPOSAL · PART 1

Agent Fingerprinting

The QA layer we're missing for agents.

> Treat your agent like any other system you'd ship.
>
> Define a baseline. Run regression tests. Block bad deploys.

> We don't ship code without tests.
>
> **Why do we ship agents without behavioral guarantees?**

WHAT IT LOOKS LIKE

A fingerprint scenario

```
# A test case for your agent's behavior
scenario_id: "user_query_revenue"
input: "What's last quarter revenue?"
expected_tools: ["salesforce_query"]
expected_order: ["salesforce_query"]
expected_confidence: 0.85+
tolerance: ±5%
```

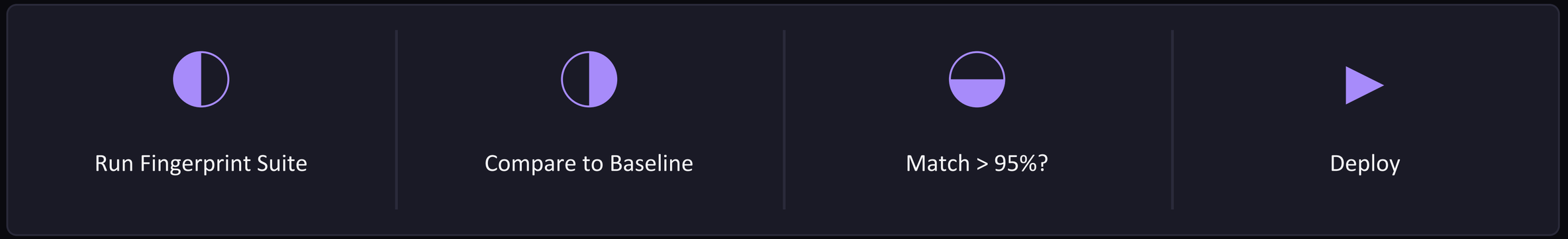
01 Collect 50-500 scenarios that cover your agent's key behaviors.

02 Run the fingerprint suite against your agent. Record outcomes.

03 If 95%+ scenarios match → ship. If less → investigate.

DEPLOYMENT PIPELINE

Block bad deploys before they ship



Match < 95%? Stop deployment. Investigate the divergence.

Maybe retune prompts. Maybe update fingerprints. But **never blind-deploy.**

PROPOSAL · PART 2

The MCP spec has a gap

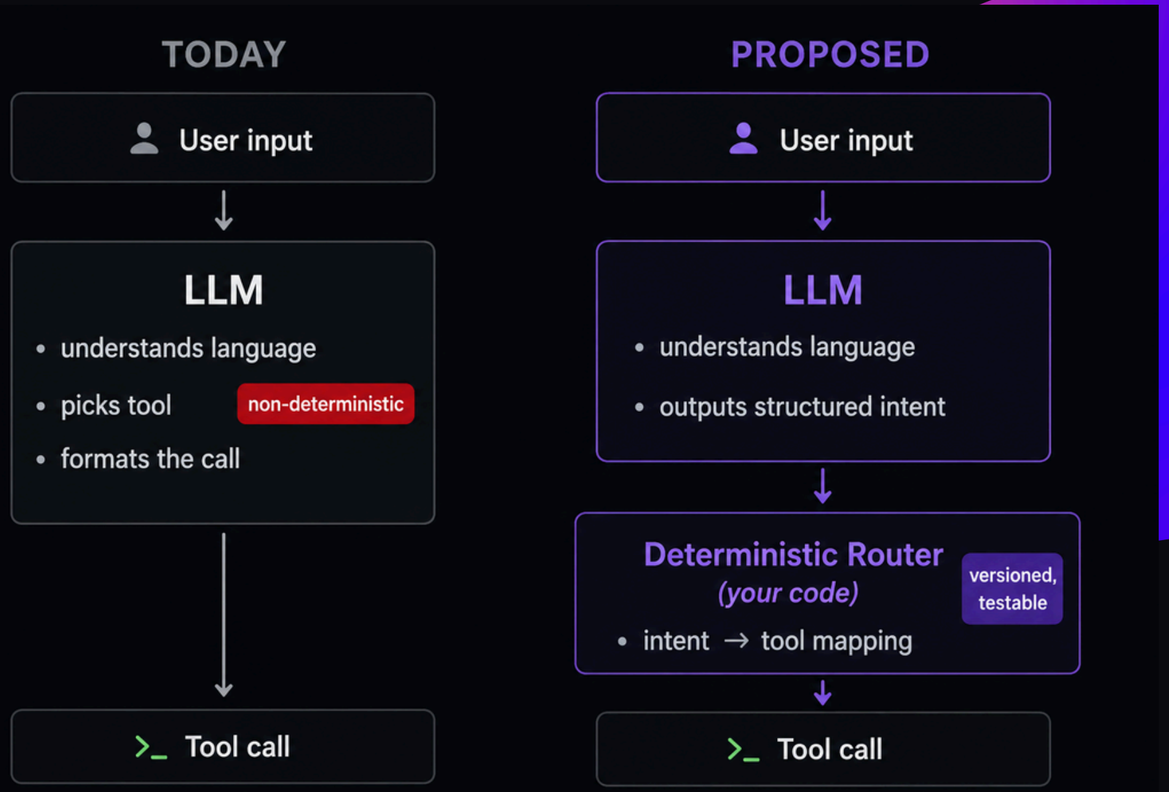
✓	Required tools format
✓	Schema structure
✓	Tool naming conventions
✓	Capability negotiation
✓	Transport protocols
✗	Tool ordering — server authors decide ad hoc

Server authors guess. Most default to alphabetical. **Order matters — and it's not specified.**

PROPOSAL · BONUS

Deterministic Routing

LLMs are great at understanding language. They're bad at making consistent choices. So why are we asking them to do both in one step?



WHAT WE'RE ASKING

Three asks

Test for divergence

Change your tool order. Bump temperature. Switch models. Measure what happens.

You'll be shocked.

Spread awareness

Write about this. Tweet about this. Most teams haven't noticed it yet.

Make it visible.

Standardize ordering

Get tool ordering into the MCP specification. Small change.

Big ecosystem impact.

> **We solve it together or it doesn't get solved.**

Thank you.

@joeyousss

@adityaoberai

@sonichigo