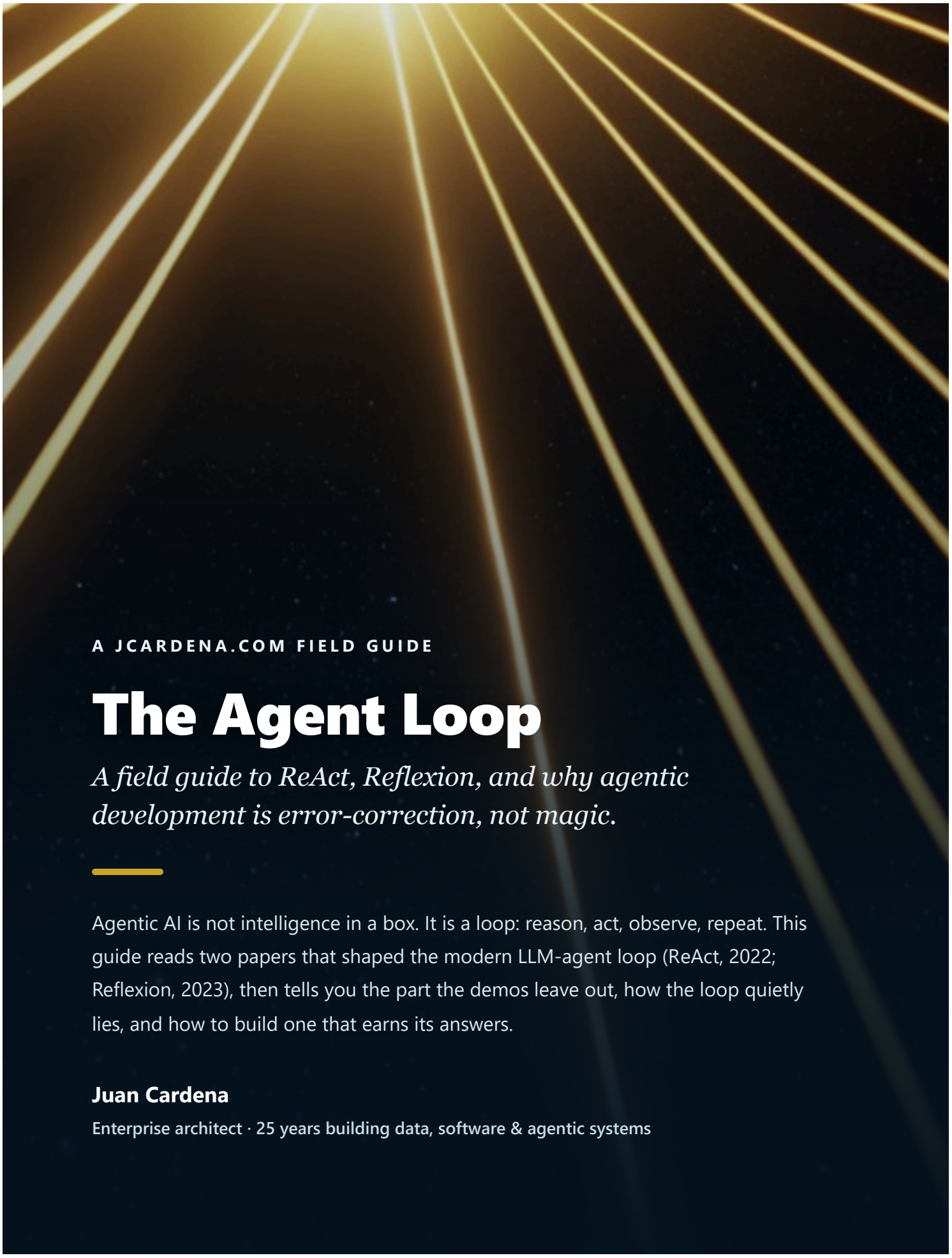

A JCARDENA.COM FIELD GUIDE

The Agent Loop

A field guide to ReAct, Reflexion, and why agentic development is error-correction, not magic.

Agentic AI is not intelligence in a box. It is a loop: reason, act, observe, repeat. This guide reads two papers that shaped the modern LLM-agent loop (ReAct, 2022; Reflexion, 2023), then tells you the part the demos leave out, how the loop quietly lies, and how to build one that earns its answers.

Juan Cardena

Enterprise architect · 25 years building data, software & agentic systems

CONTENTS

CHAPTER ONE

The Shape Under the Hype

People keep telling me agentic AI started in 2024, the year "agent" was on every slide. It didn't. The idea that matters was written down quietly in October 2022, and its lasting contribution is not a model or a benchmark. It is a *shape*: a loop.

Once you see the loop clearly, most of what gets sold as "agency" turns out to be marketing wrapped around one small, durable structural truth. This guide is for the builder who wants that truth, not the slide. I read these systems the way I read a system in production: not "is it clever," but "what does it buy you, and where does it quietly fail."

And I want to be honest about history, because the brand of everything I write is trust. The loop was not invented from nothing. Robotics had sense-reason-act loops for decades (the classical BDI and SOAR architectures). In 2021, WebGPT had a model navigate a browser across multiple steps. In mid-2022, Inner Monologue was already feeding environment feedback into an LLM's reasoning, and SayCan was grounding an LLM's action selection in robotic affordances. So when I say ReAct "started it," I mean something narrower and defensible: **ReAct gave the LLM era the cleanest, promptable form of the loop**, the one a builder could pick up and use in an afternoon. That is a real contribution. It just isn't a virgin birth.

> *Agency is not intelligence. It is a loop, and the loop's real job is not motion. It is to make error* **visible**.

CHAPTER TWO

Anatomy of the Loop

ReAct's move is almost embarrassingly simple. Make the model produce, in one interleaved stream, both a **Thought** ("I should look up X") and an **Action** (actually look up X), then feed back an **Observation** (what came back), and let it think again. In the authors' words: it generates "both reasoning traces and task-specific actions in an interleaved manner."

Here is the subtlety almost every explainer gets wrong, including my own first draft. The loop is not all "inside the model." The model writes the Thought and the Action. The **Observation comes from outside**: a search engine, an API, a file system, a test runner, a compiler. That asymmetry is the entire point.

Read it as feedback control

We call this "agentic," but a loop is not agency; a thermostat has a loop. Structurally, ReAct is **error-correction**. The Thought proposes, the Action intervenes, and the Observation returns the gap between what the model expected and what the environment reports. The next step is constrained by what *happened*, not by what merely *sounded plausible*.

That reframing changes the hero of the story. Most writing centers the model and its cleverness. The real protagonist is the **Observation**, and behind it the environment. An ungrounded model is a closed system: left to talk to itself, it drifts toward fluent, confident nonsense. Every genuine Observation is a small injection of outside order, the compiler that refuses to lie, the test that goes red.

> *The environment is the only thing in the room that can push back, but only as well as the instrument exposing it.*

CHAPTER THREE

The Loop Learns (and What That Doesn't Mean)

In 2023, Reflexion (lead author Noah Shinn, with ReAct's Shunyu Yao among the coauthors) added a second turn to the screw. After a failure, the agent writes a short verbal reflection, "I assumed the file was sorted; it wasn't", and keeps it in an episodic memory, so the next attempt is better. This is the trial-to-trial improvement effect.

Be precise, because the hype is not: Reflexion does **not** train the model. No weights change. It learns *in context*, by carrying a written lesson forward. That is powerful and cheap. It is also fragile, because a memory can preserve a *wrong* lesson just as faithfully as a right one, and the model must diagnose its own failure using the same reasoning that caused it.

THE HONEST ONE-LINER

Reflexion is in-context error memory, not weight learning. It makes the loop better at remembering what broke last time, not wiser about the world.

CHAPTER FOUR

When the Loop Lies Quietly

The seductive version of this story ends at the previous chapter: the agent reasons, acts, observes, reflects, and converges on truth. The honest version does not, and anyone who has shipped these systems knows exactly where the floor gives way.

FAILURE MODE 1 — THE OBSERVATION IS NOT TRUTH

Tools return mediated signals: stale search, a poisoned web page, a flaky test, a misleading benchmark. "Grounding" is only as honest as the instrument doing the grounding.

FAILURE MODE 2 — THE LOOP LAUNDERS ERROR

A wrong first premise leads to a wrong tool call, which retrieves confirming evidence, which makes the model *more* confident. Iteration does not guarantee correction; sometimes it is recursive rationalization with better footnotes.

FAILURE MODE 3 — REFLECTION RATIONALIZES INSTEAD OF REPAIRS

Asked to explain its own failure, the model often writes a tidy, wrong lesson and marches on, now wrong *and* documented.

FAILURE MODE 4 — LOOPS DRIFT, AND THEY COST

Tokens, latency, quota, money. In practice quality often degrades after a handful of turns. "Every answer is earned" needs a companion: not every answer is worth earning this way.

None of this means the loop is a lie. It means the loop is a tool, with a blade and a handle, and most demos only ever show you the handle.

CHAPTER FIVE

Building a Loop That Earns

Point the loop at a codebase and it becomes a coding agent: read the file, run the test, read the failure, fix, run again. This is why ReAct's little loop is one ancestor of many agentic dev tools you use. But "the agent earns its answer" cannot be a vibe. It has to be a contract, with every link present.

```
claim → action / evidence → observation → revision → stop-condition
                          → accountable answer
```

Miss any link and the answer is not earned, only fluent. In practice that means five disciplines:

THE FIVE DISCIPLINES OF A LOOP THAT EARNS

1. **Trustworthy observations.** Invest in the instrument: real tests, typed APIs, deterministic tools. A loop is only as good as what it is allowed to see.
2. **A real verification step.** Not "ask the model if it's done." A separate check the model cannot talk its way past, a test, a schema, a second tool.
3. **A human at the stop.** Specific decision rights: approve irreversible actions, inspect evidence, set risk tolerance, own the final judgment, and pull the cord on a loop spending money to convince itself.
4. **A budget.** Max turns, max spend, a circuit breaker. Loops that cannot stop do not converge; they bleed.
5. **Knowing when not to loop.** A single confident answer is sometimes correct and far cheaper. Reserve the loop for work where the environment can actually tell you that you were wrong.

The uncomfortable truth in the human gate: the person is not decoration. The person is the part that isn't finished. We are still there because the loop is not closed; we have not solved autonomous grounding.

CHAPTER SIX

The Two Chapters Demos Skip: Security & Evaluation

Security: a tool-using agent is a new attack surface

The moment your loop can act, it can be made to act for someone else. Treat these as first-class, not afterthoughts:

- **Prompt injection through observations.** A retrieved web page or a file can contain instructions. If the loop reads it as data *and* as commands, you have handed the wheel to whoever wrote the page.
- **Credential and data leakage.** An agent with broad tool access will, eventually, send something somewhere it shouldn't. Scope every tool; least privilege is not optional.
- **Unsafe execution.** "Run this code" and "run this command" are loaded guns. Sandbox, allowlist, and never let the loop hold a key it doesn't strictly need.

Evaluation: how you know the loop worked

If you cannot measure it, you are not running an agent; you are running a slot machine with good manners. The minimum kit:

- Pass/fail **task tests** with held-out cases, not vibes.
- **Source and tool-call logs**, so you can see what it actually observed.
- **Calibration**: does its confidence track its correctness?
- A **regression suite**, because a prompt change that fixes one task quietly breaks three.
- **Escalation thresholds**: when uncertainty is high, the loop should hand up to a human, not loop harder.

> *An agent you cannot evaluate is not autonomous. It is merely*

> *uninspected.*

CHAPTER SEVEN

What an Architect Keeps

Strip away the format and the hype, and ReAct leaves one durable idea: a model becomes useful the moment it is forced to leave language, touch the world, receive a correction, and carry that correction forward. Much of the recent LLM-agent stack, planning, memory, multi-agent teams, can be read as scaffolding around that primitive.

The loop is leverage on a capable model, plus a trustworthy observation, plus a human who owns the stop. Take any of the three away and iteration becomes only a more elaborate way to be wrong. That is why the tool was always the fashion and the principle is the work. The ReAct prompt format will date; some of us already write agents that look nothing like it. But the shape underneath, propose, intervene, get corrected, decide whether you are done, is what makes an answer something other than a confident guess.

> **Tools are fashion. Trust is the work.** *On the AI layer, that has one slogan, and now you can see the machinery behind it:* **every answer is earned.**

APPENDIX

Glossary

ReAct (2022)

Yao et al., arXiv:2210.03629. The promptable reason-act-observe loop for LLMs.

Reflexion (2023)

Shinn et al., arXiv:2303.11366. Verbal self-reflection stored in episodic memory; in-context improvement, not weight updates.

Thought / Action / Observation

The three moves of the loop. The first two are the model's; the third comes from the environment.

Grounding

Constraining the model's next step with a real signal from the world. Only as trustworthy as the instrument.

Feedback control

The control-theory lens: the Observation is an error signal that corrects the next action.

The human gate

The explicit decision rights a person keeps over an autonomous loop, especially the stop.

Agent slop

Fluent output from a loop that never actually got corrected, an answer that sounds earned but isn't.