

Digital transformation simplified Via AI for a mission critical system

A practitioner's view from payments — moving fast while staying inside the guardrails

Ajay Upadhyay

Director of Software Engineering, Mastercard

Modernization is not new — it is constant

1999

A student, freelancing

Converting a FoxPro app into a client-server application. A few hundred megabytes of plain text — and the data migration was already hard.

2026

An enterprise, in the AI era

New tools, the same friction. Systems that run perfectly and silently — while no one fully understands how.

The technology changed. The hard part didn't.

Why the old system is so hard to replace



Workarounds posing as requirements

Years of quick fixes, until the workaround became the spec no one ever wrote down.



Documentation that no longer matches

What's written down drifted away from what the system actually does.



Thin test data in lower environments

Complex business logic never gets exercised — so it stays hidden until production.



Reliable technology, scarce skills

The platform still works — but the people who understand it are leaving the market.

THE ECONOMICS

So why go through all of this?

Most of the IT budget just keeps the lights on

[__%]

Run & maintain

[__%]

Change & innovate

Budgets are consumed running and patching what already exists.

The threat landscape moves faster than ever — new software and hardware vulnerabilities surface constantly.

Standing still is a competitive risk and a security risk at the same time.

Modernization isn't optional — doing it *safely* is the real challenge.

MODERNIZATION IN FINTECH

In fintech, the guardrails are higher

Every industry feels modernization pressure. In payments it's sharper — each change is squeezed between speed and trust, inside a regulated environment.



Speed to market

Customers and the business expect new capability quickly.



Security & reliability

The same customers expect their money and data to be safe, every single time.



Regulatory review

New technology must pass review — inside the organization and outside it.

TWO PROJECTS I HAVE LIVED

Same discipline, very different stakes

Supply chain & logistics

A TYPICAL INCIDENT

A wrong invoice, order, or shipping label.

IMPACT

Hits reputation — but there is time to respond and correct.

CAN YOU UNDO IT?

Recoverable, with time to react.

Fintech & payments

A TYPICAL INCIDENT

A wrong settlement or an incorrect rewards allocation.

IMPACT

Once the money has moved, funds and trust are both on the line.

CAN YOU UNDO IT?

No — it cannot be rolled back.

Two poles of force: move fast enough — **without** leaving the guardrails.

THE GUARDRAIL

Build a shadow production environment

A shadow environment runs in parallel to production on the same real inputs — but its output has no business impact. It's where you prove the modern system against the legacy one, live and safely.

Identical real production inputs flow to both systems



Production — the legacy system

Processes live traffic. Output is applied for real: money moves, points post.

Output is real



Shadow — the modern system

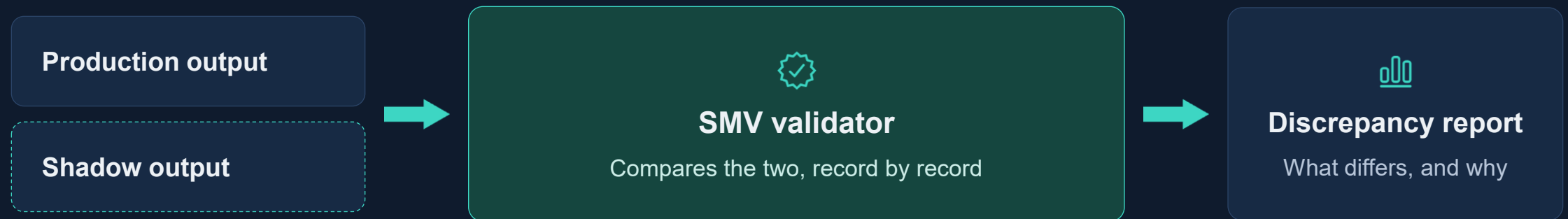
Processes the same traffic in parallel. Output is captured and compared — never applied.

Zero business impact

THE AGENTIC SAFETY NET

Silent Mode Validation (SMV)

A Gen AI validator that continuously compares shadow output against production and produces a report — so discrepancies surface before a cutover, not after.



Built from reusable Gen AI skills — the validator itself stays small and never becomes its own megaproject.

THE VALIDATOR, IN DEPTH

What the validator can actually catch — and how precisely

WHAT IT CATCHES

Value mismatches

Field by field — not just record-level pass/fail.

Missing & duplicate records

Gaps and double-counts in either stream.

Ordering & timing drift

Right answer, wrong sequence or settlement window.

Schema & contract drift

Shape changes that silently break downstream consumers.

HOW IT STAYS PRECISE

Field-level tolerances

Allowed variance for rounding, timestamps, surrogate IDs.

Configurable match keys

Aligns records that don't share a natural key.

Severity scoring

A wrong settlement amount ranks above a cosmetic diff.

Deterministic & explainable

Every diff is reproducible and traceable to its source.



Always on: it runs continuously against live traffic — silent, with zero business impact — so coverage deepens every day the shadow runs.

WHAT MAKES IT AGENTIC

SMV triages itself, so engineers see signal



Splunk MCP · root cause

Pulls the relevant logs on demand to explain *why* each discrepancy happened.



Self-correcting ignore rules

Suppresses known noise, so the report stays signal — not spam.



Error classification

Buckets discrepancies by type instead of drowning teams in one-offs.

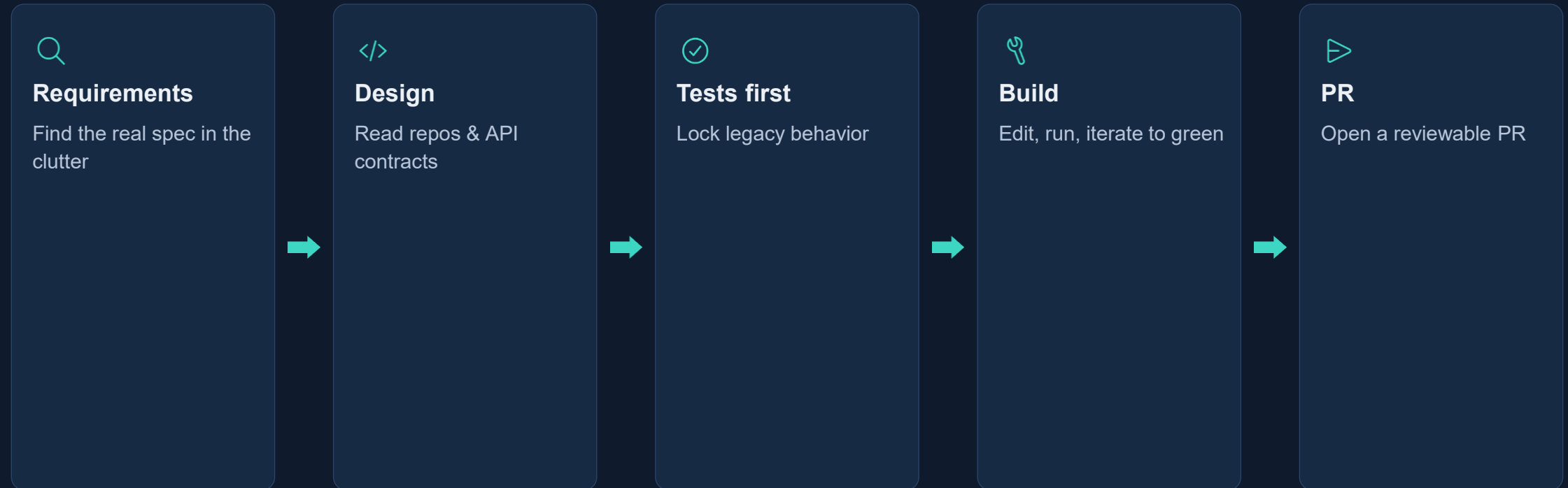


Jira MCP · one ticket per class

Opens a single ticket for each class of error, routed to the right team.

The result: engineers work on **classes of problems**, not an inbox of noise.

One agent across the lifecycle — humans at every gate



 **Human gate at every handoff** — the agent proposes and does the work; nothing advances to the next phase without sign-off.

PHASE 1 · REQUIREMENTS

Signal from the clutter

Reads through

- Obsolete docs and stale comments
- Dead code paths and old workarounds
- Scattered tickets and commit history

Produces

- Live behavior split from dead code
- Workarounds surfaced as real requirements
- Every requirement traced to its code

HOW THE AGENT DOES IT

Reachability analysis

Static call-graphs plus production traces separate live paths from dead code.

Corpus cross-reference

Semantic search over repo, commits, tickets and docs flags where they disagree.

Traceability matrix

Each candidate requirement is linked to the code and data that prove it.

 **Human gate:** product and engineering decide what to carry forward and what to retire.

PHASE 2 · DESIGN

Grounded in the code that exists

Reads

- Repos, modules and dependency graphs
- Live API contracts and their consumers
- Data schemas and critical flows

Proposes

- Target boundaries and data model
- A strangler-fig plan of thin slices
- Backward-compatible contracts

HOW THE AGENT DOES IT

AST & dependency mapping

Parses the codebase into module and call graphs to locate the natural seams.

Contract extraction

Reads OpenAPI/IDL or infers from handlers; checks real usage in gateway logs.

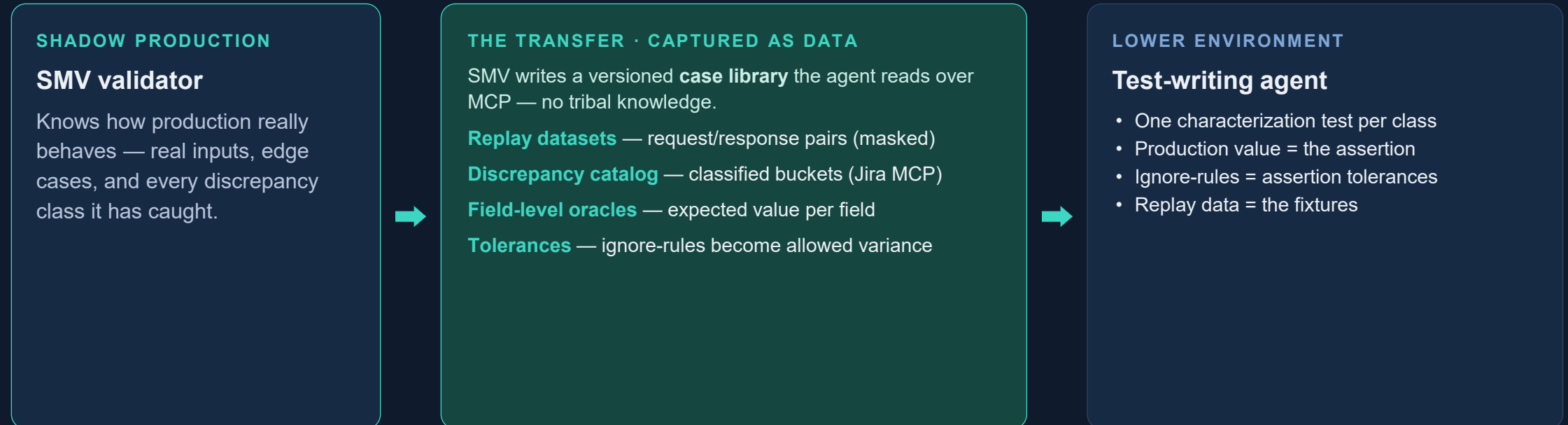
Options with trade-offs

Generates candidate designs and ADRs, each weighed on cost, risk and blast radius.

 **Human gate:** architects own the boundaries, the contracts, and the sequencing.

PHASE 3 · TESTS FIRST

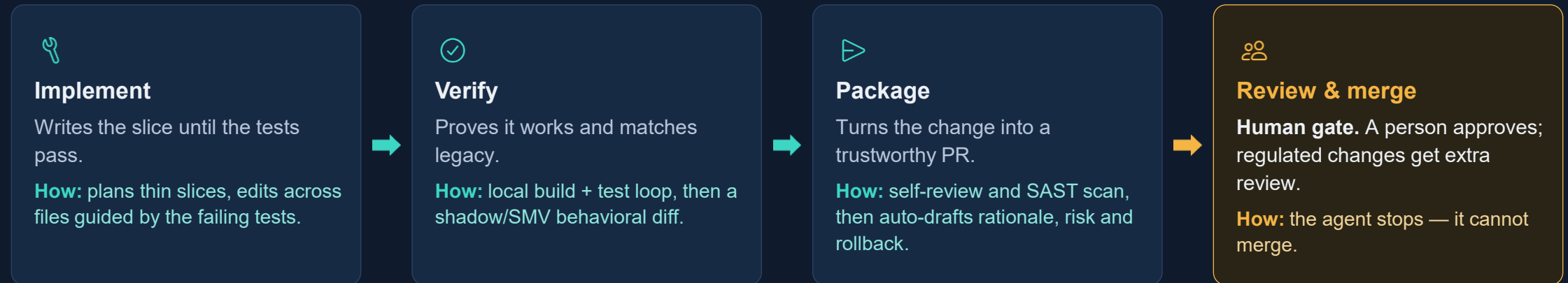
How SMV's knowledge reaches the test-writing agent



 **Masked at the boundary:** PAN, PII and account identifiers are tokenized before anything leaves production — format-preserving tokens keep the business logic testable in the lower environment.

 **Human gate:** engineers review the SMV-sourced cases — the tests define “done.”

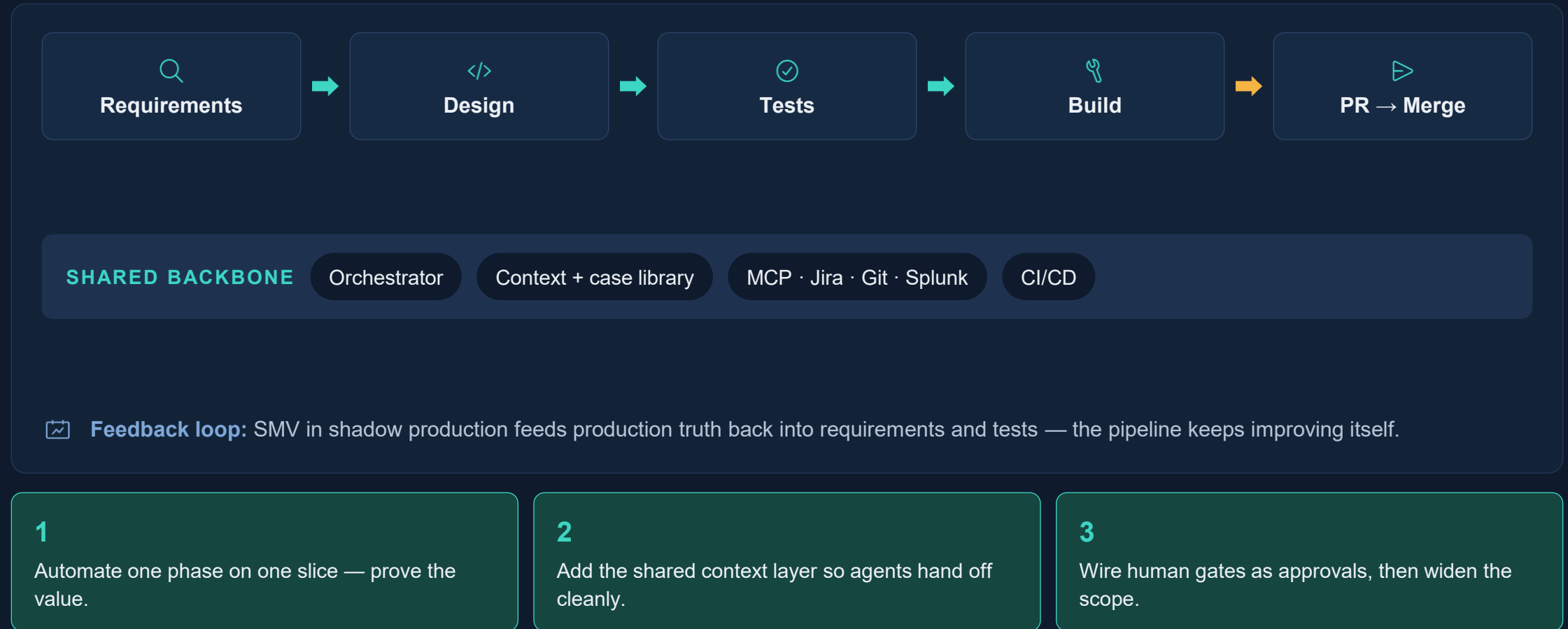
From implementation to a reviewable PR



The PR opens through Git + Jira MCP — linked to the requirement and the ticket, so the trace runs end to end. A human still says “merge.”

THE NEXT STEP

Stitch the agents into one development pipeline



What I'd like you to take away

- 1 Modernization is constant — and the hardest parts are organizational, not just technical.
- 2 In fintech, safety isn't a phase. Build the guardrail before you build the feature.
- 3 A shadow environment plus an agentic validator turns “we hope it works” into “we can prove it — silently.”
- 4 Agentic AI makes the work faster. Humans stay at the gates.

Thank you.

Ajay Upadhyay · [linkedin.com/in/upadhyay-ajay24](https://www.linkedin.com/in/upadhyay-ajay24)