Lessons learned building a harness to rediscover and replace a legacy system

Q1-Q2 pre opus 4.6

Payfit

Founded in 2015

Payroll & HR system for Small & Medium Businesses

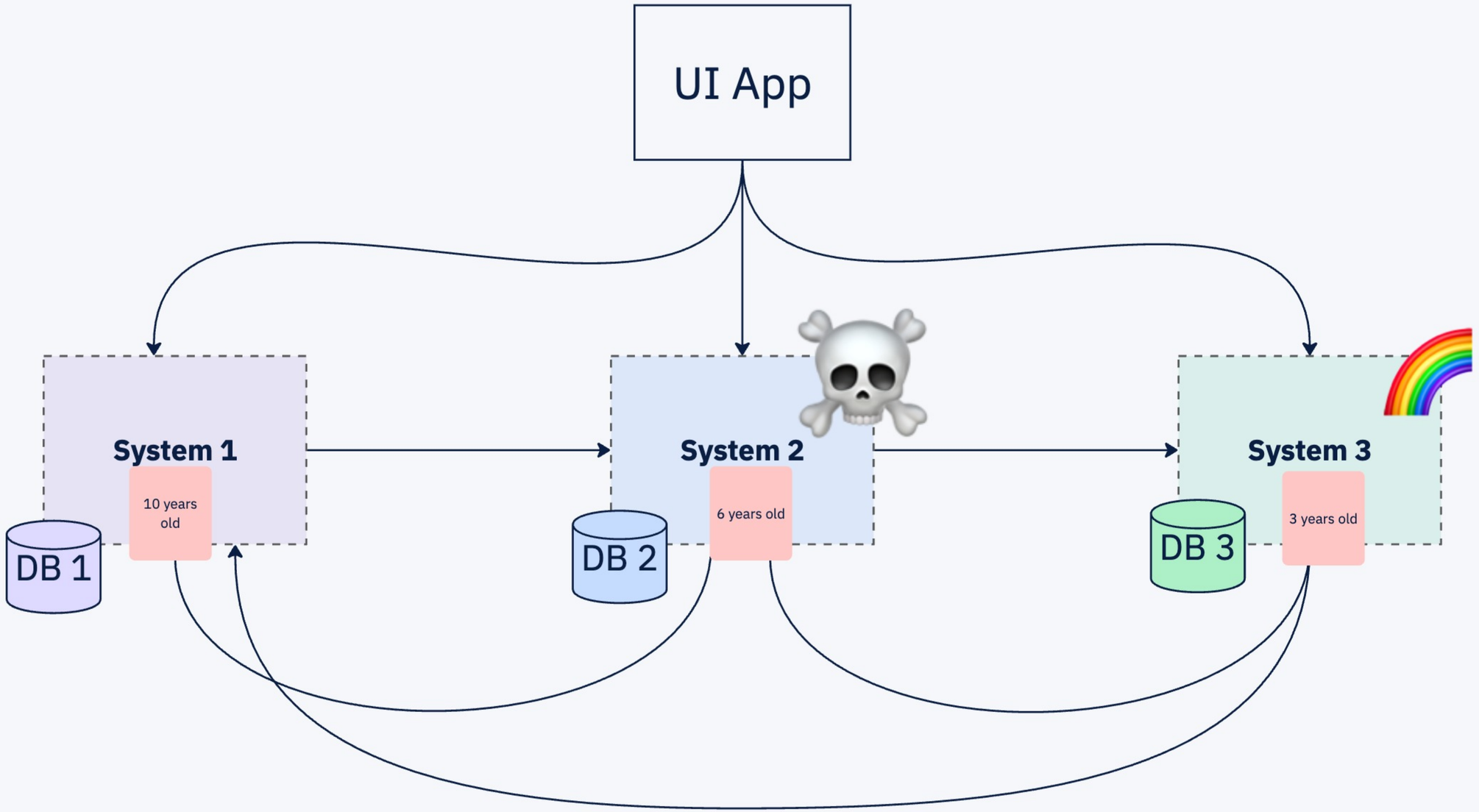


Nick Tune



Charles Desneuf

One domain/context spread across 3 systems, 3 dbs, 3 domain models.
Goal: consolidation



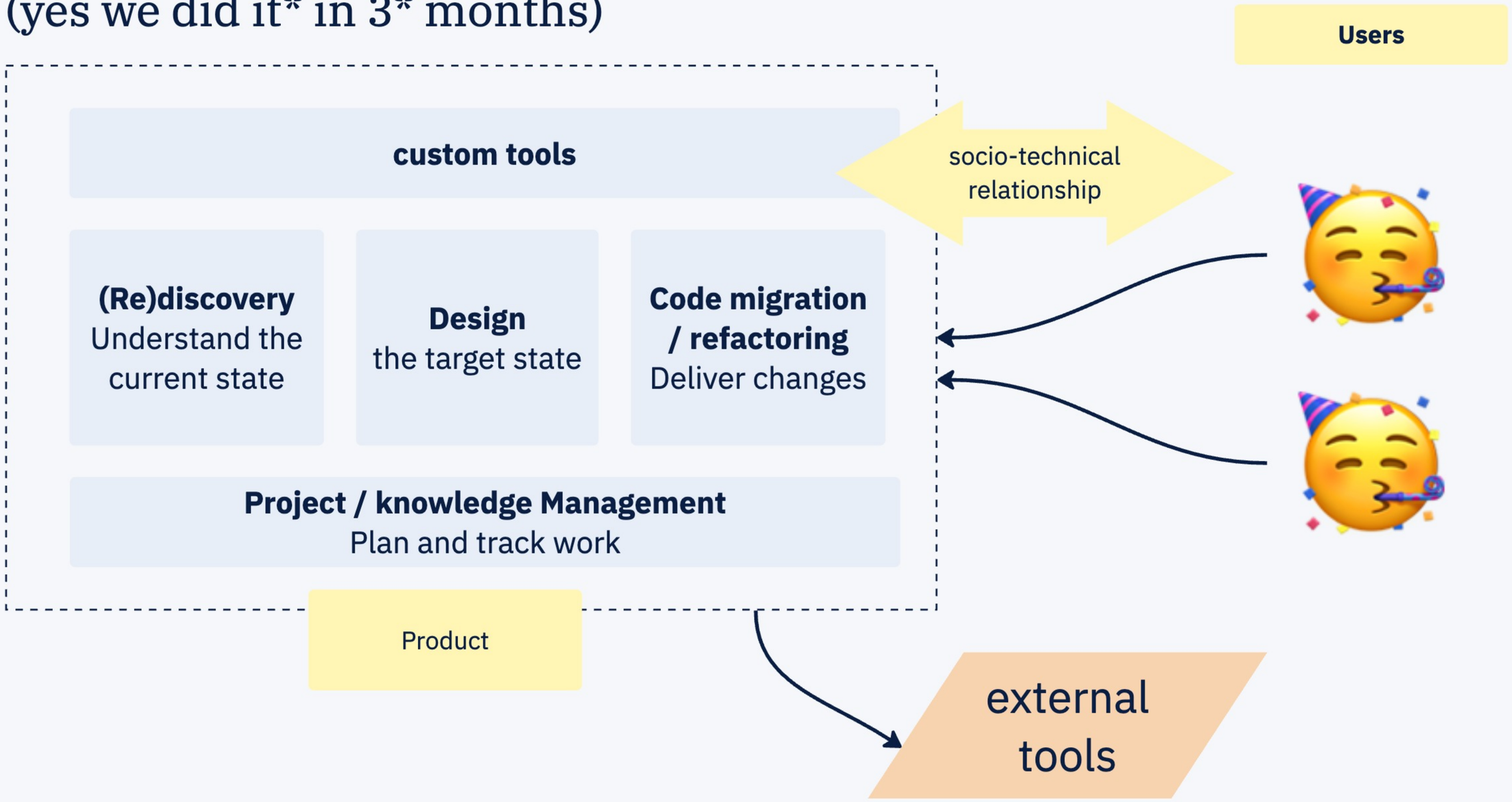
Nick: With AI, me and Charles could kill system 2 in 3 months

Big boss: That would be amazing. Is that really possible. Approved, see you in the 3 months for the system retirement party.

Nick and Charles: Oh f**k. They agreed. What the hell are we supposed to do now? Is it really possible to kill all of that in just 3 months with AI?

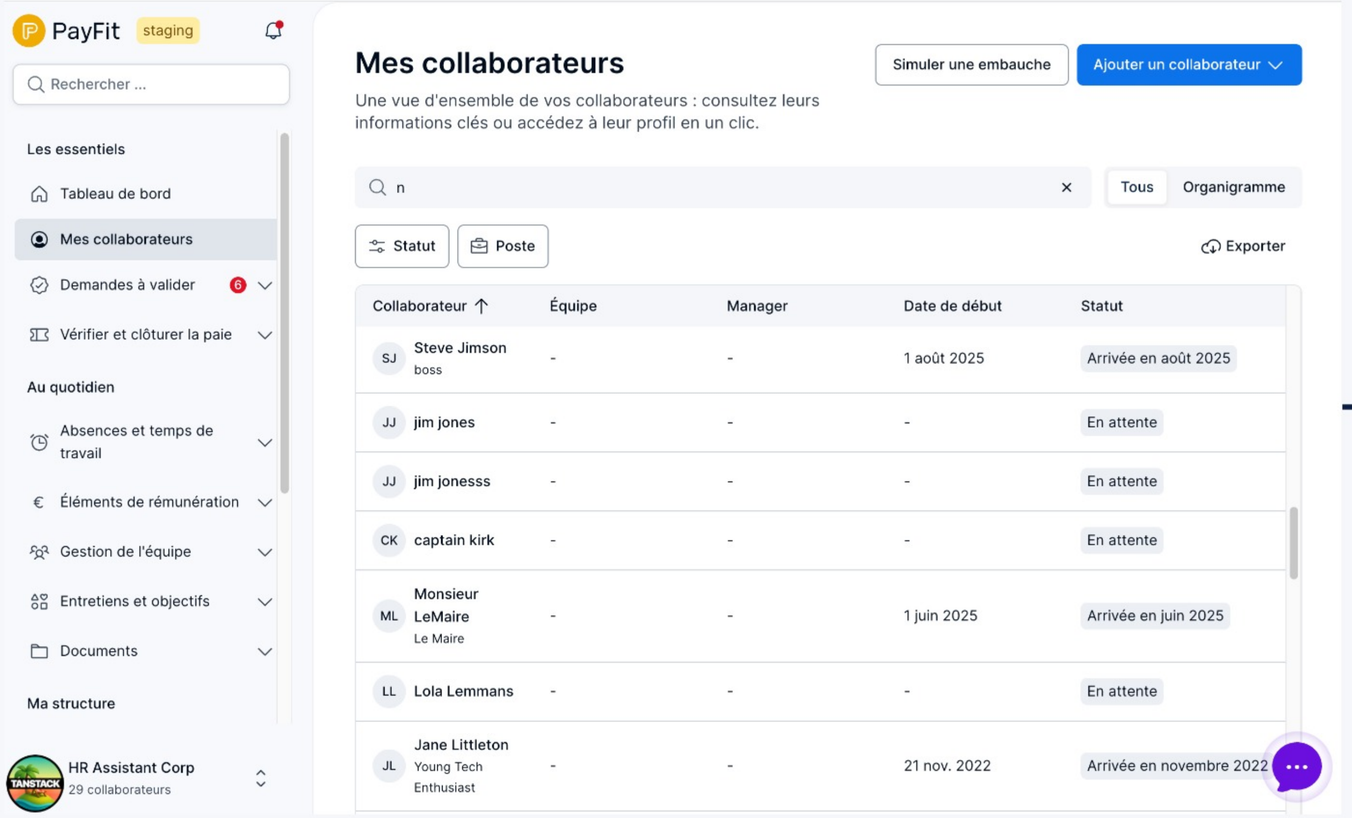
The harness we ended up with...

(yes we did it* in 3* months)



Part 1:

Is there a business case for modernizing or is it just tech reasons?



get active collaborators

System 2

Data synchronisation error = UI inconsistency / bugs



Data synchronisation error = cannot run payroll

[redacted]

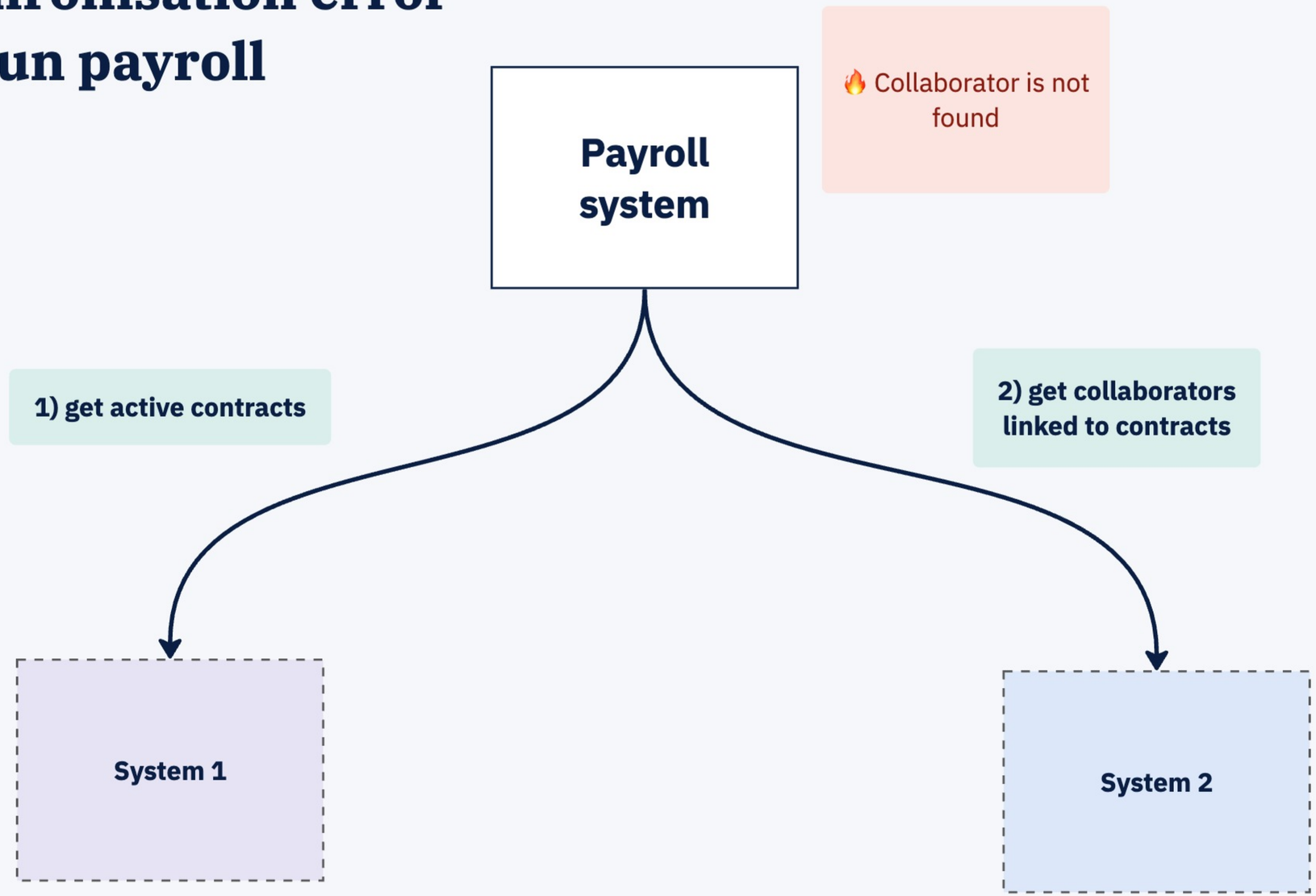
Hello there, we have currently a company which is not able to close their pay period: a contract is not linked to a collaborator

Here the companyId: [redacted]
and the contract without collaborator: [redacted]

Show more

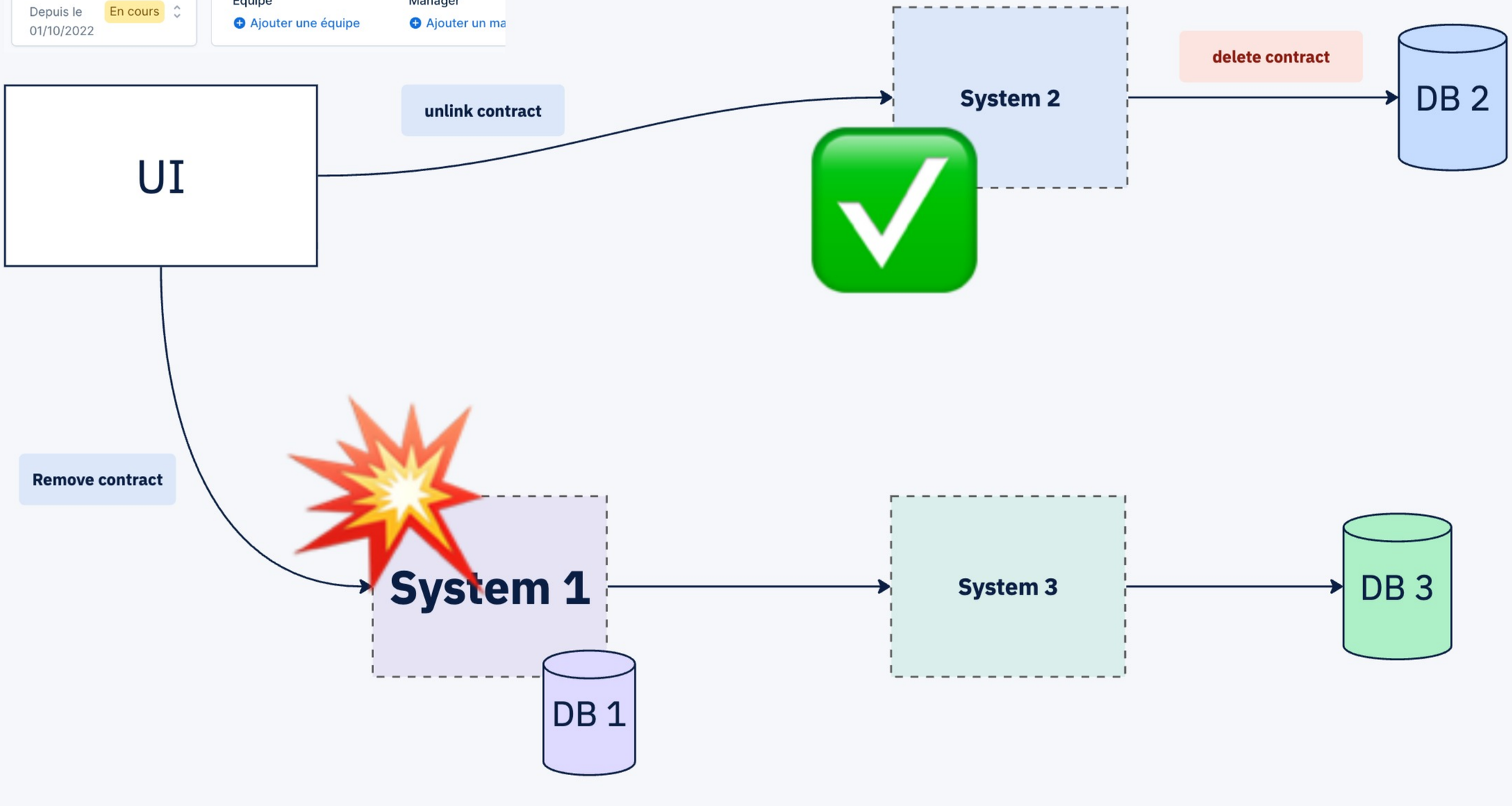
Thread in # ask-employee-management | Oct 6th, 2025 | View message

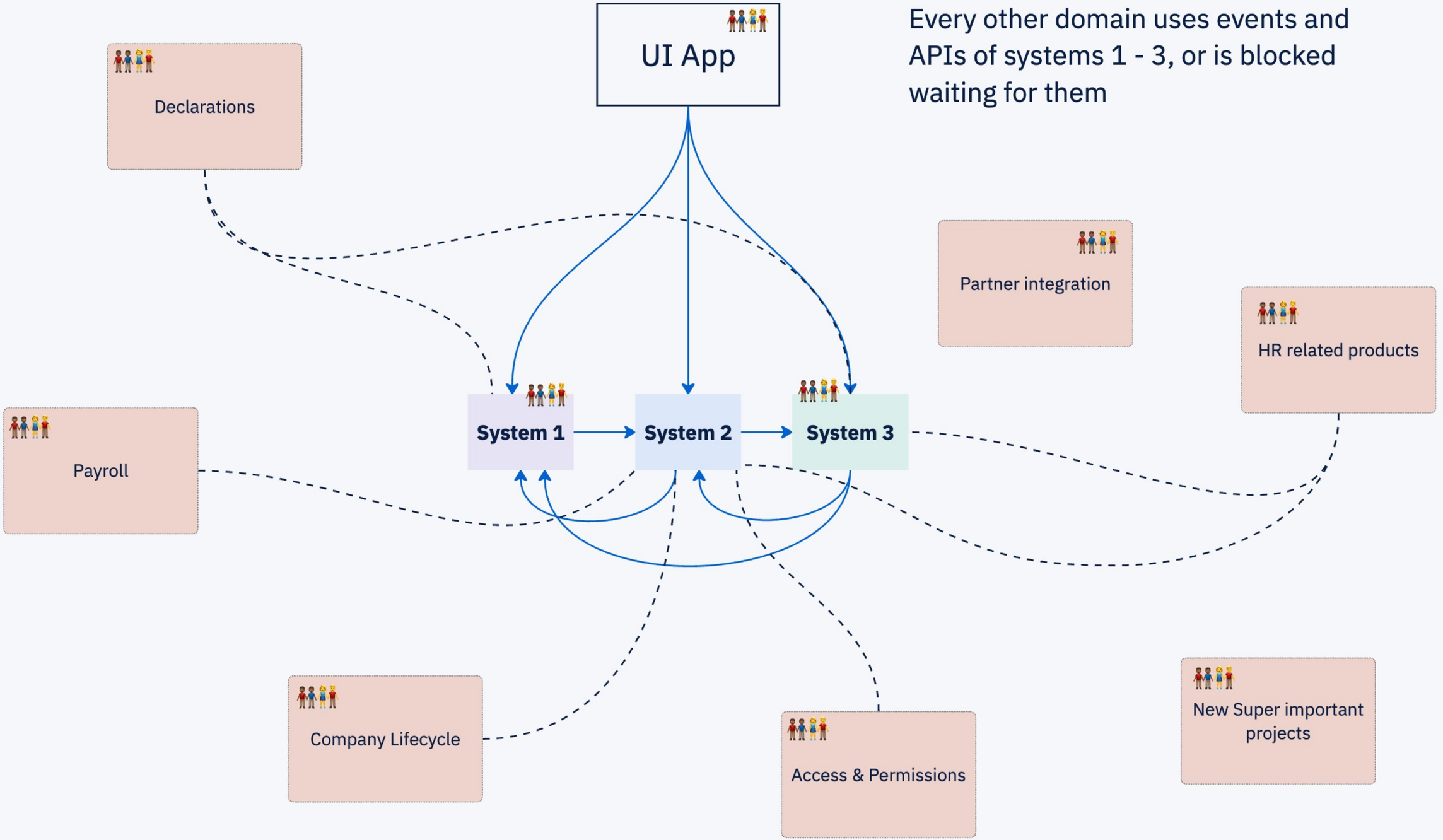
Data synchronisation error = cannot run payroll



How orphans are created






UI App
Every other domain uses events and APIs of systems 1 - 3, or is blocked waiting for them
Declarations
Partner integration
HR related products
Payroll
System 1
System 2
System 3
Company Lifecycle
Access & Permissions
New Super important projects

Part 2:

How do the users of the harness influence the design of the harness?

Harness design is socio-technical design

Before we talk about all of the technical, we think it's important to describe how the choice of people on a project made a big difference to how we were able to benefit from AI and how it shaped the harness.

Two profiles, two perspectives

Nick



2 years working in EM domain

Already have a good understanding of the system

Already used AI to analyze, automate bug investigations etc

Charles



6 months in the company, but not EM domain

Needs to learn EM domain

just moved from another team - 1st project in EM

Nick: "AI was an extension of my brain!"



- I felt a lot more productive with AI
- All the artefacts it produced fitted into my existing mental model and was like an extension of my memory
- I was able to use the tooling to work on more things in parallel because I understood relationships and dependencies
- I knew all the places to look to harvest existing sources of truth
- I was able to calibrate output with my own knowledge and know when there were blind spots and teach AI how to address them
- I knew our system conventions and built AI tooling around them

Charles: "AI overloaded my brain!"

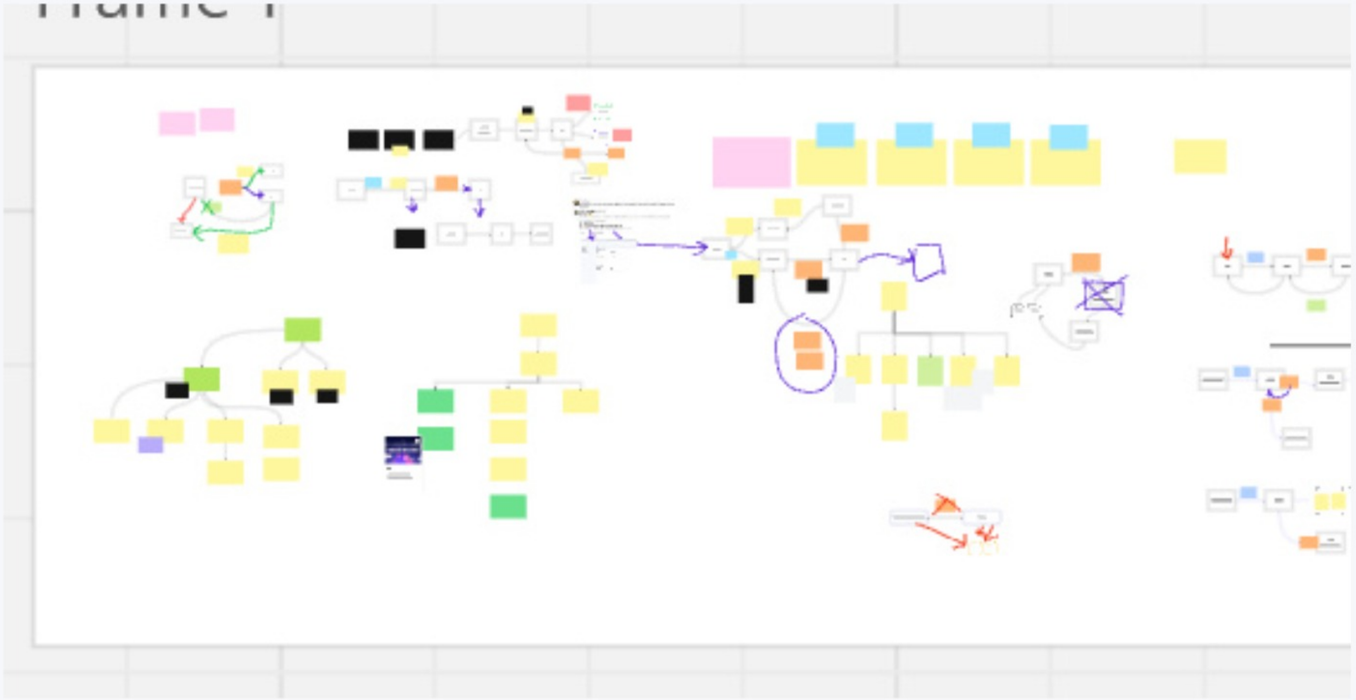


- Everything was new to me: I didn't have an existing mental model
- When AI proposed solutions, I was less able to understand misassumptions, blind spots
- I had less confidence to trust AI and verify it's output

BUT

- It did help me to learn faster
- It did make me aware of questions I needed to ask (and that's why having someone experience on the project was also crucial...)

We still used Miro a lot



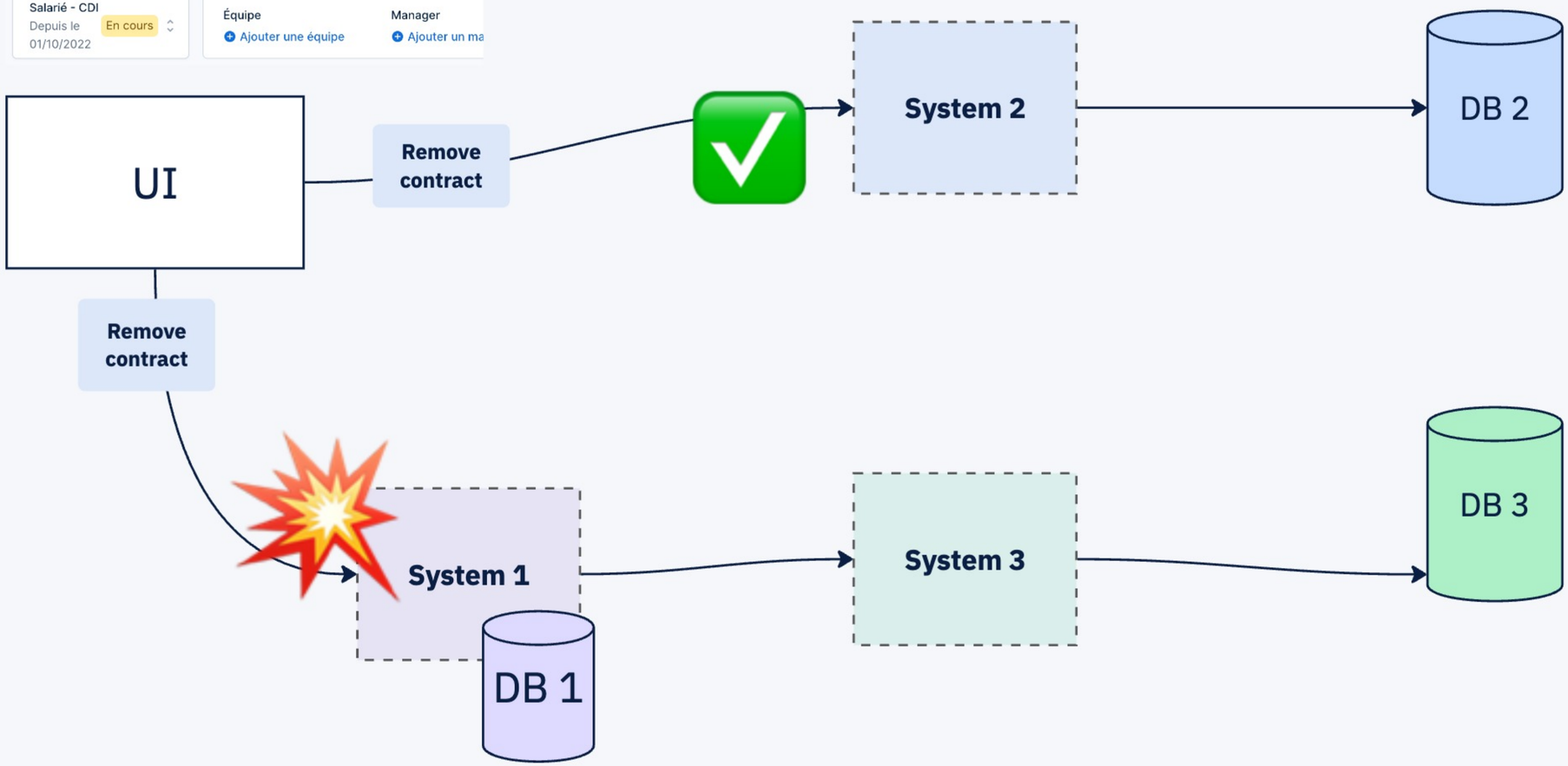
- Sketching out complex dependency flows across many services
- exploring what happens if we migrate a single dependency

Major Breakthrough: the red trash button

Remember this?

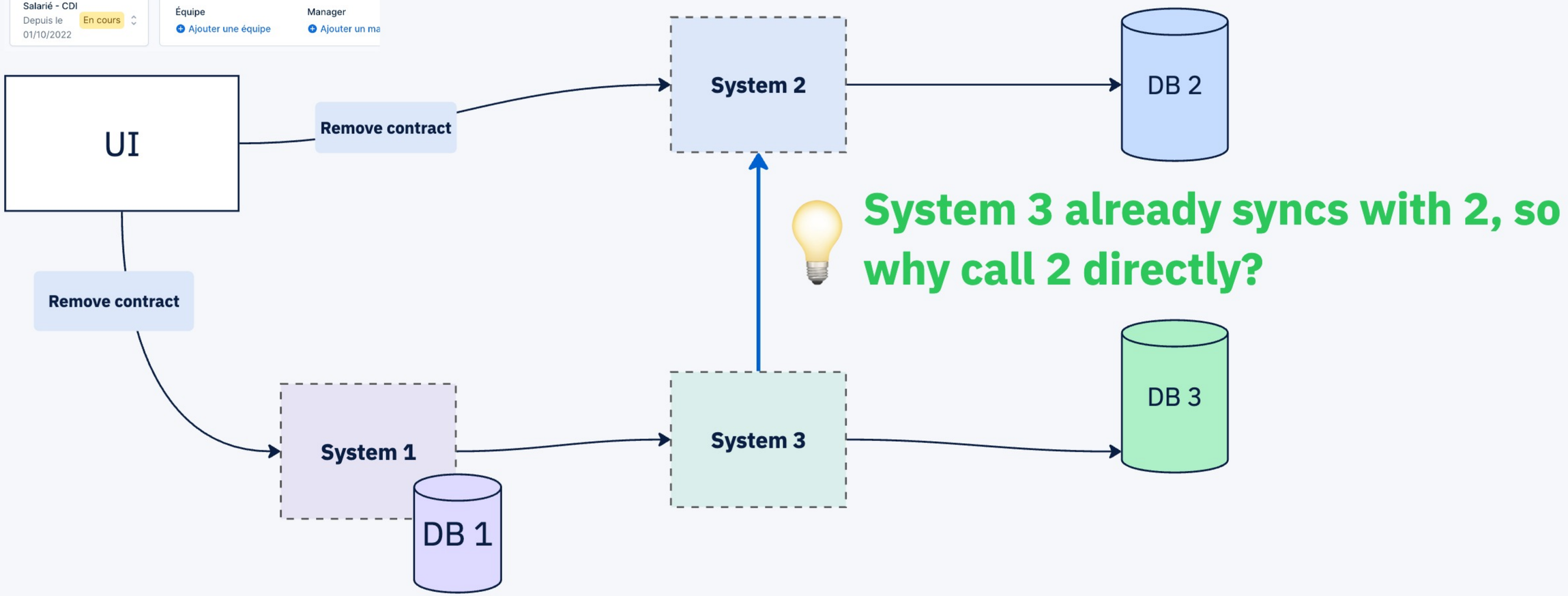
We were going to spend weeks redesigning UX and rewiring dependencies



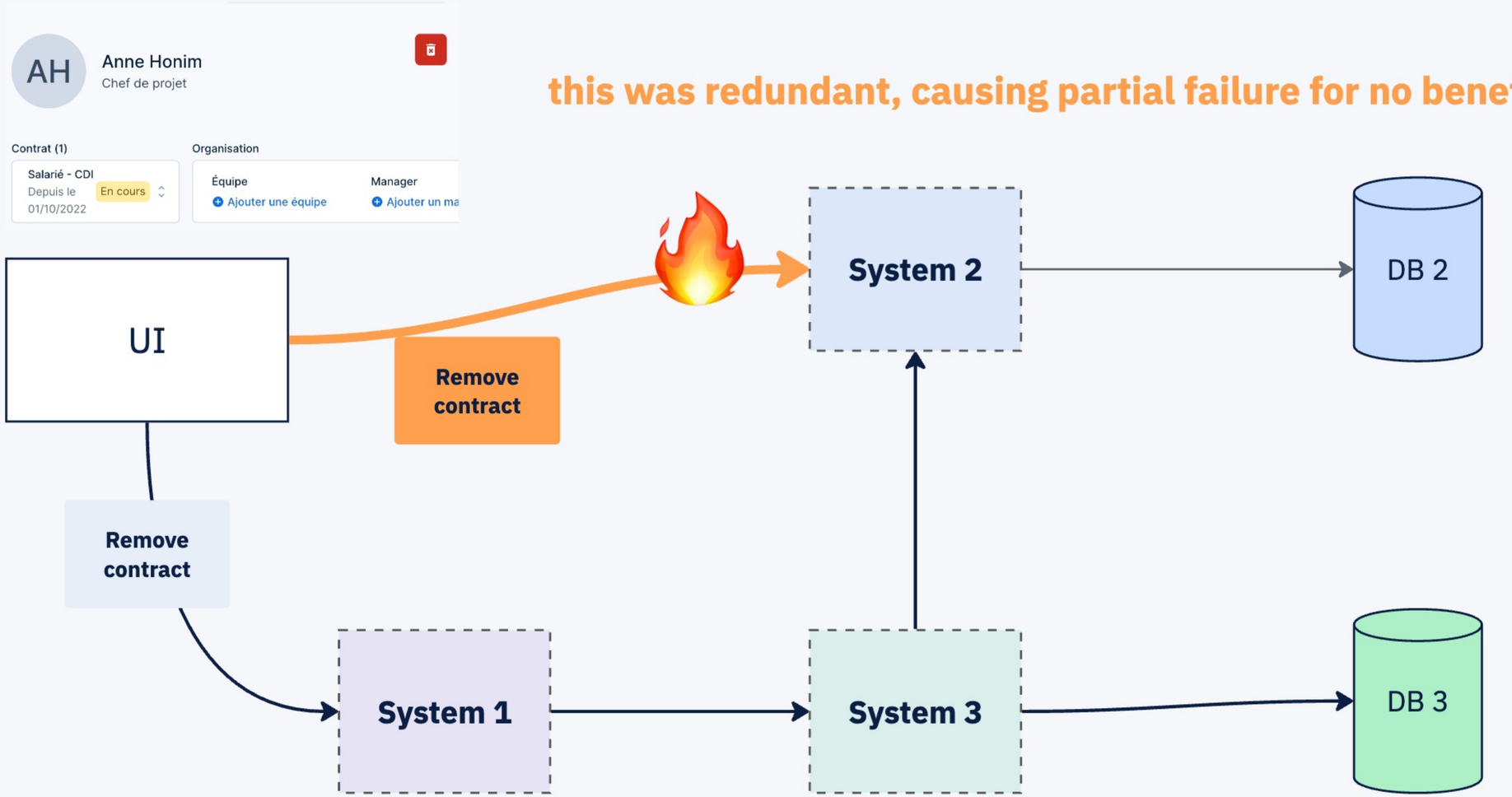


During a miro session we realised a simpler solution => AI didn't catch this for us





that means we could fix the whole problem by just deleting this API call



Explore graph-building capabilities

AI* can find better solutions and understand the impact of choices with a modelled graph.

It misses things that are scattered across multiple diagrams / documents.

*or even just normal code

Part 3:

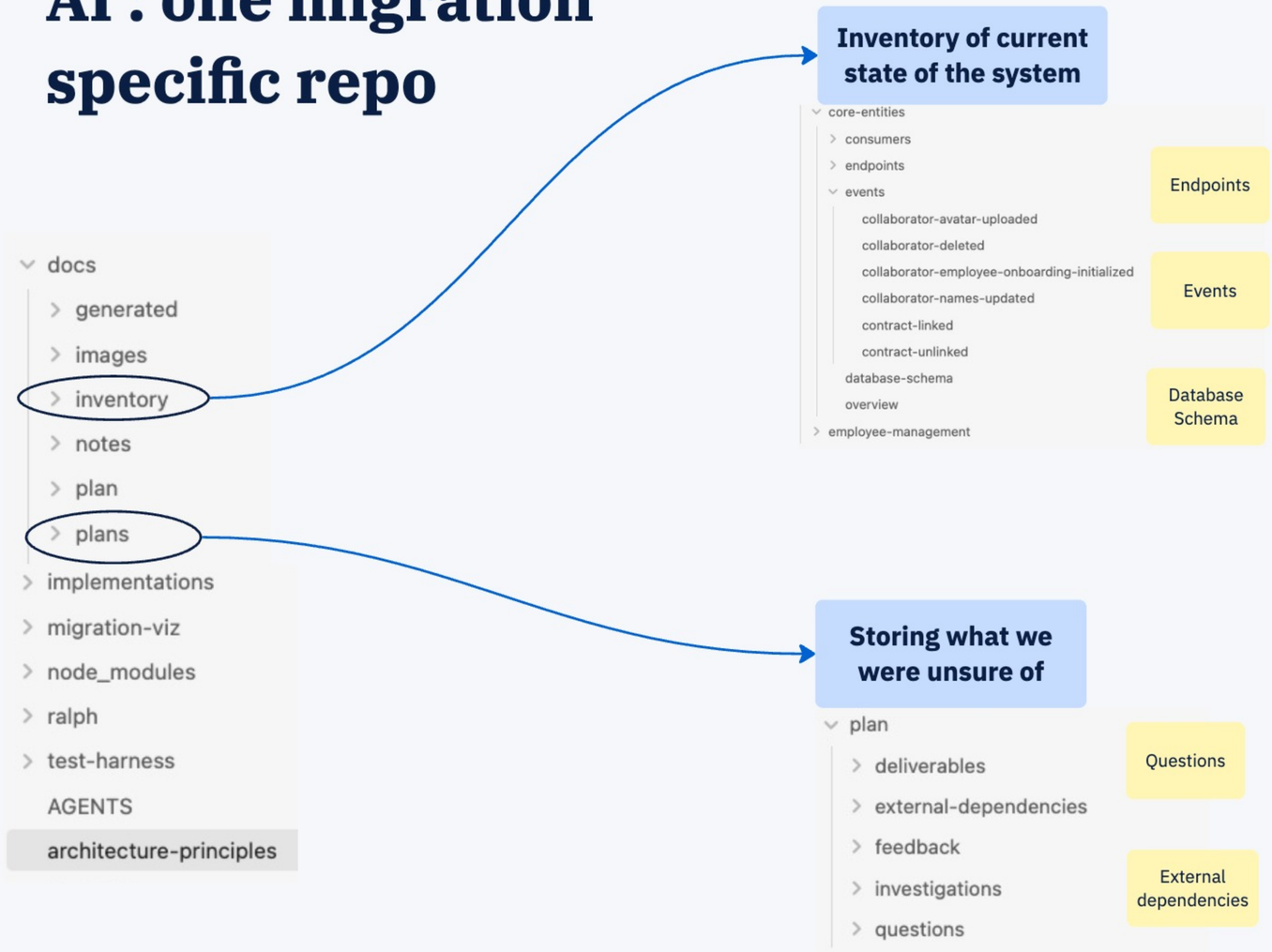
How do we collect and store the information we need (and ensure it's accurate)?

Create a shared brain between humans and AI

If AI knows everything you know*, it can do more for you.

* where to find knowledge, how to interpret it, how to navigate, when it's out of date

Storing all we know for AI : one migration specific repo



Mainly markdown files

why not jira, confluence, notion etc.?

Everything is a markdown file in git

Make the User Experience you want

Build the harness to get the most of the information you have.
No need to be constrained by staying just in a conversation with agent.
Use visualizations if that works better for you.
Today agents are offering more and more visual capabilities
Build the tools you need, the cost of building your own tooling is almost 0.

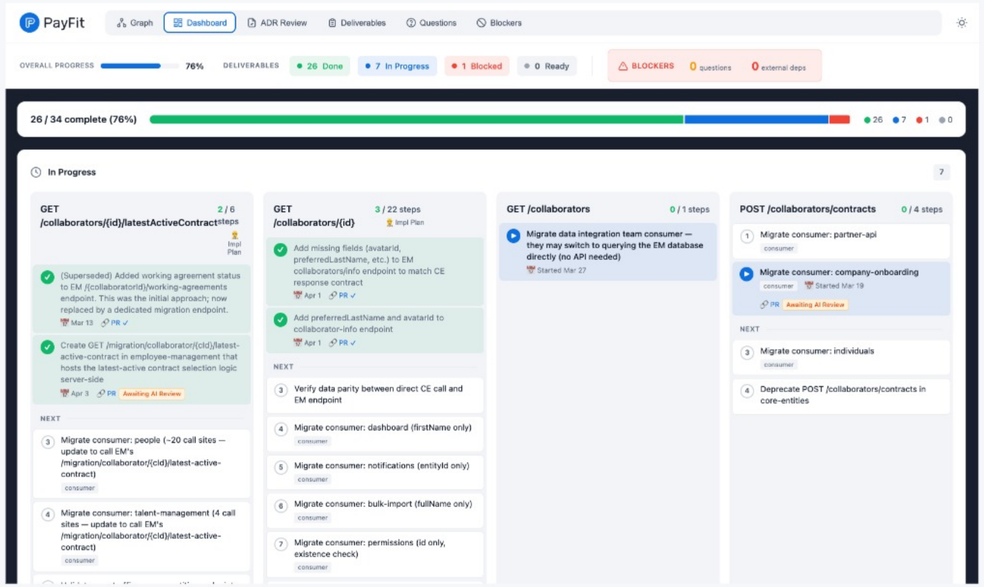
Mikado dependency viewer and dashboard



Blocked

In progress

Dashboard



How to generate UIs from markdown files ?

mikadomethod.info

Generating UIs from AI generated markdown files

```yaml
---
id: kill-collaborator-events-jetlang-consumer-remove-employee-and-contract
    -registered-handlers
name: "Remove employee and contract registered handlers from collaborator-events
    -jetlang-consumer"
title: "Remove the employee registered and contract registered handlers from
    collaborator-events-jetlang-consumer"
type: endpoint
started_date: "2026-03-13"
completed_date: "2026-03-18"
dependencies: []
prs:
  - url: https://github.com/PayFit/core-entities/pull/2121
    status: merged
category: D
risk: medium
effort: M
target_service: employee-management
internal_review_status: draft
adr_accepted_date: null
migration_steps:
  - id: step-1
    description: Remove employee registered handlers from core-entities consumer
```

deterministic, fast, reliable, no AI

AI hallucinates yaml fields that don't exist and doesn't follow rules

Frontmatter schemas

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Migration Deliverable",
  "type": "object",
  "additionalProperties": false,
  "required": ["id", "name", "title", "type", "target_service",
      "category", "effort"],
  "properties": {
    "id": {
      "type": "string",
      "description": "Unique identifier for this deliverable"
    },
    "name": {
      "type": "string",
      "description": "Short display name (e.g., 'GET /employees')"
    },
    "title": {
      "type": "string",
      "description": "Full descriptive title"
    },
    "type": {
      "type": "string",
      "enum": ["endpoint", "event", "prerequisite", "infrastructure"],
      "description": "Type of deliverable"
    },
```

Enforcement hooks

```
"Stop": [
  {
    "hooks": [
      {
        "type": "command",
        "command": "${CLAUDE_PLUGIN_ROOT}/bin/validate-schemas.sh",
        "timeout": 60
      }
    ]
  }
]
```

Don't rely on agents to follow instructions use hooks as guardrails

Agents will regularly ignore markdown files, particularly when they hit a problem and try to be clever and find workarounds.

Hooks are strict enforcement that you can rely on.

Forcing precision

Code walkthrough with files and line references

Step 2: Deep Code Walkthrough

Follow the complete code path from route to response:

1. Route definition → Controller method
2. Controller → Service layer
3. Service → Repository/Data access
4. Service → External API clients

Document each step with **exact file path and line number**.

Assert number of created files matches input

8. Count results

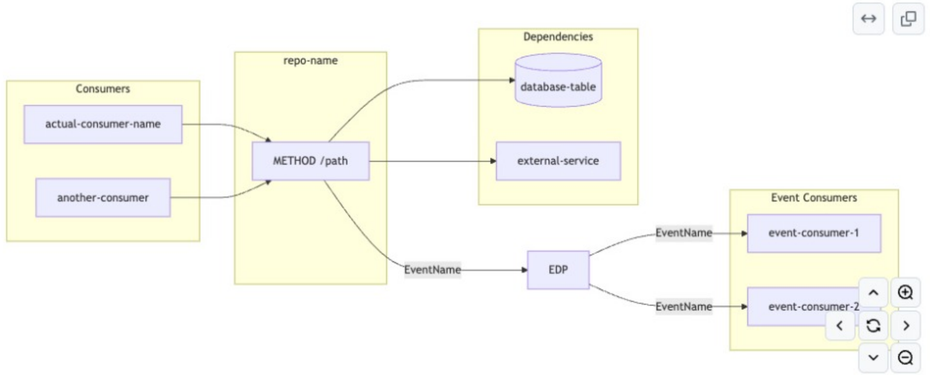
Count files created:

- `docs/inventory/*/endpoints/*.md` → endpoint count
- `docs/inventory/*/events/*.md` → event count

Extensive use of templates

Step 8: Generate Context Diagram

Create a mermaid flowchart showing the endpoint in its ecosystem, including published events and their consumers.



Rules:

- Consumers on LEFT (actual service names, not SDK wrappers)
- Endpoint in CENTER
- Dependencies on RIGHT (databases with cylinder shape, services with rectangles)
- Use actual names, not generic labels
- Solid lines for all connections
- If consumers unknown yet, use placeholder with comment

Events:

- Show EDP as a central node for event routing
- Use labeled arrows: `E -->|EventName| EDP -->|EventName| Consumer`
- Group event consumers in their own subgraph
- If endpoint publishes multiple events, show each with its own labeled arrows
- If no events published, omit the EDP and Event Consumers sections

Part 4:

How can we use AI to exploit all the information we have collected?

The harness can be your project manager & architect all-in-one

Imagine a project manager, admin, and architect all rolled into one.
That's what your harness should be.

Agent as project manager + architect + router

Skill: /update-plan

Usage

/update-plan <natural language describing what changed>

Examples

Single update:

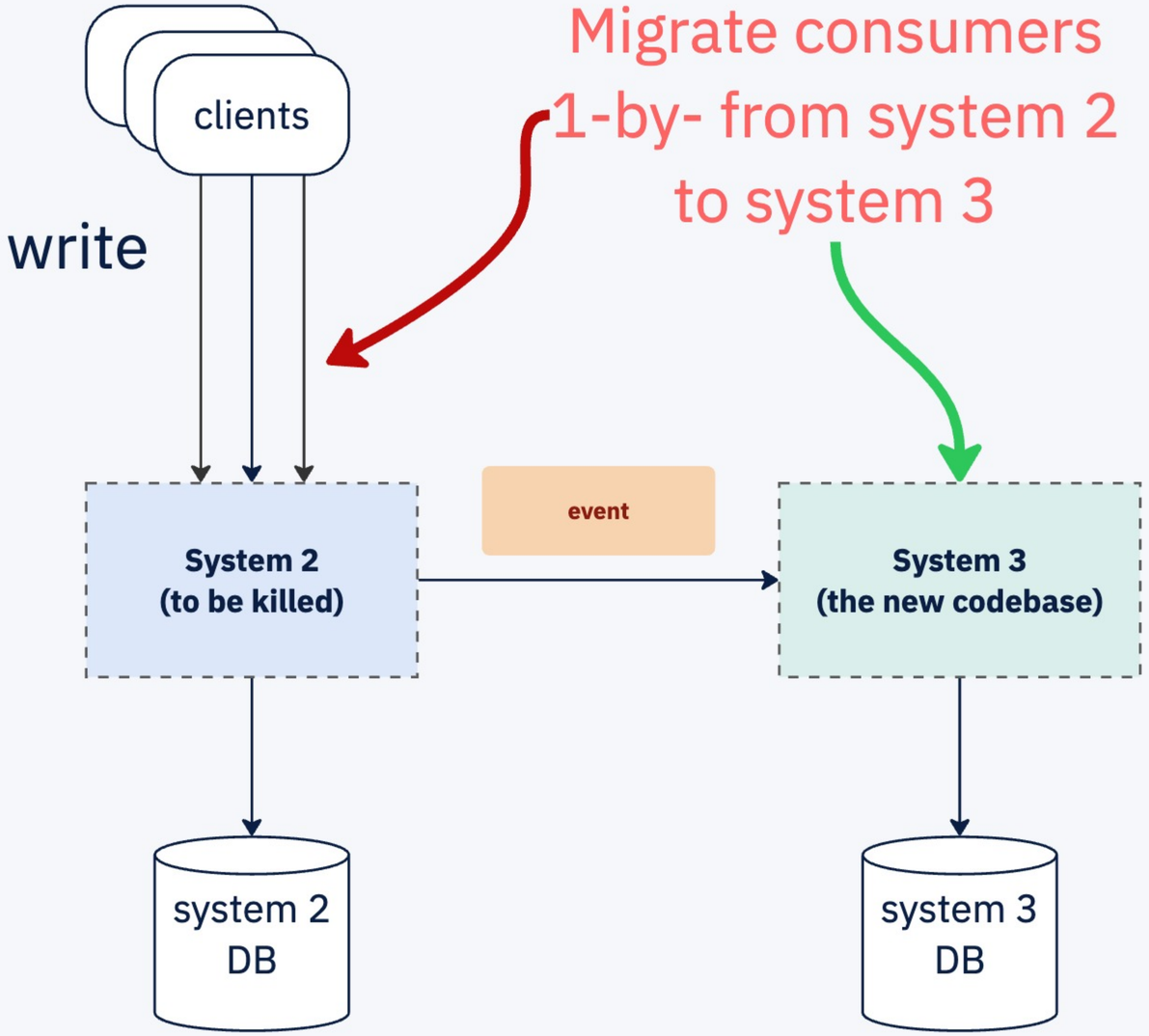
/update-plan I approved the ADR for migrate-get-employees

3. **Route to appropriate action**

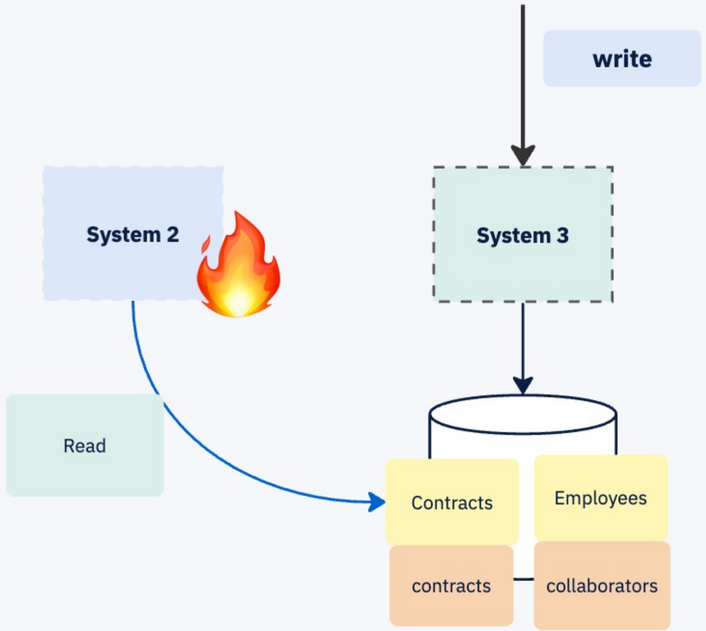
Intent	Action
ADR approved	Edit deliverable: set `adr_accepted_date`
Work started	Edit deliverable: set `started_date`
Work completed	Edit deliverable: set `completed_date`
PR (no step specified)	Edit deliverable: add to deliverable's `prs[]`
PR for step	Edit deliverable: add to step's `prs[]`, set step `status: in_progress`
Step done	Edit deliverable: set step `status: done`
Add step/subtask	Edit deliverable: append to `migration_steps[]`
Multi-phase work detected	**Offer to create subtasks** - list phases found, ask user to confirm
Time-gated work ("after X weeks/days")	**Offer subtask** with time gate in description
Multiple PRs for different phases	**Ask if these are phases** needing separate subtasks
New question	Invoke `architecture-migration-skill:add-question`
Question resolved	Invoke `architecture-migration-skill:resolve-question`
New external dependency	Invoke `architecture-migration-skill:add-external-dependency`
Dependency done	Invoke `architecture-migration-skill:resolve-external-dependency`
Feedback received	Invoke `architecture-migration-skill:add-feedback`

How to keep systems synchronised as we migrate

Which **migration pattern** to use to ensure all consumers see the exact same data as they do now?



Jointly decided (humans and AI): merge DBs first

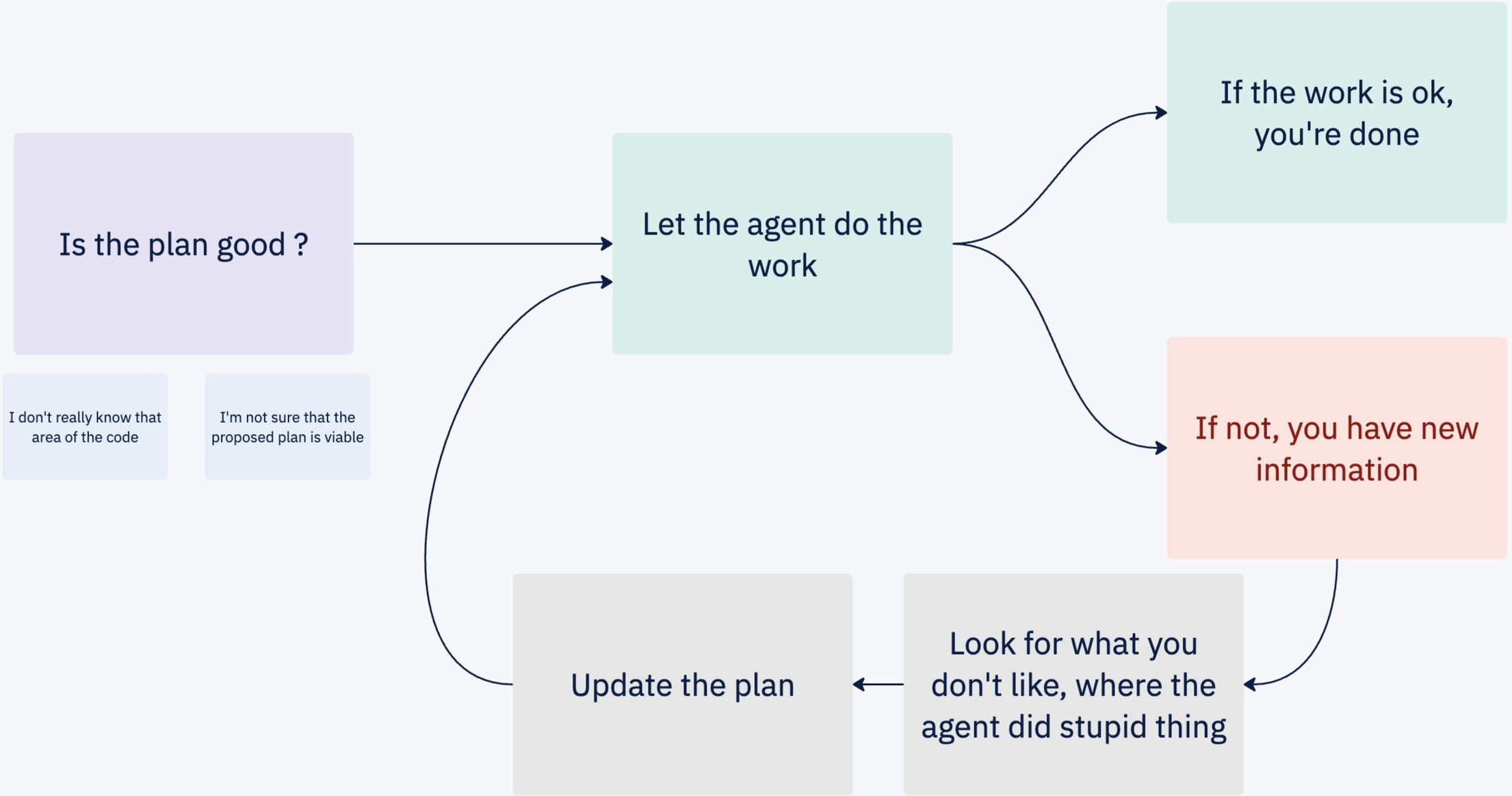


Getting this decision right was an absolute game-changer

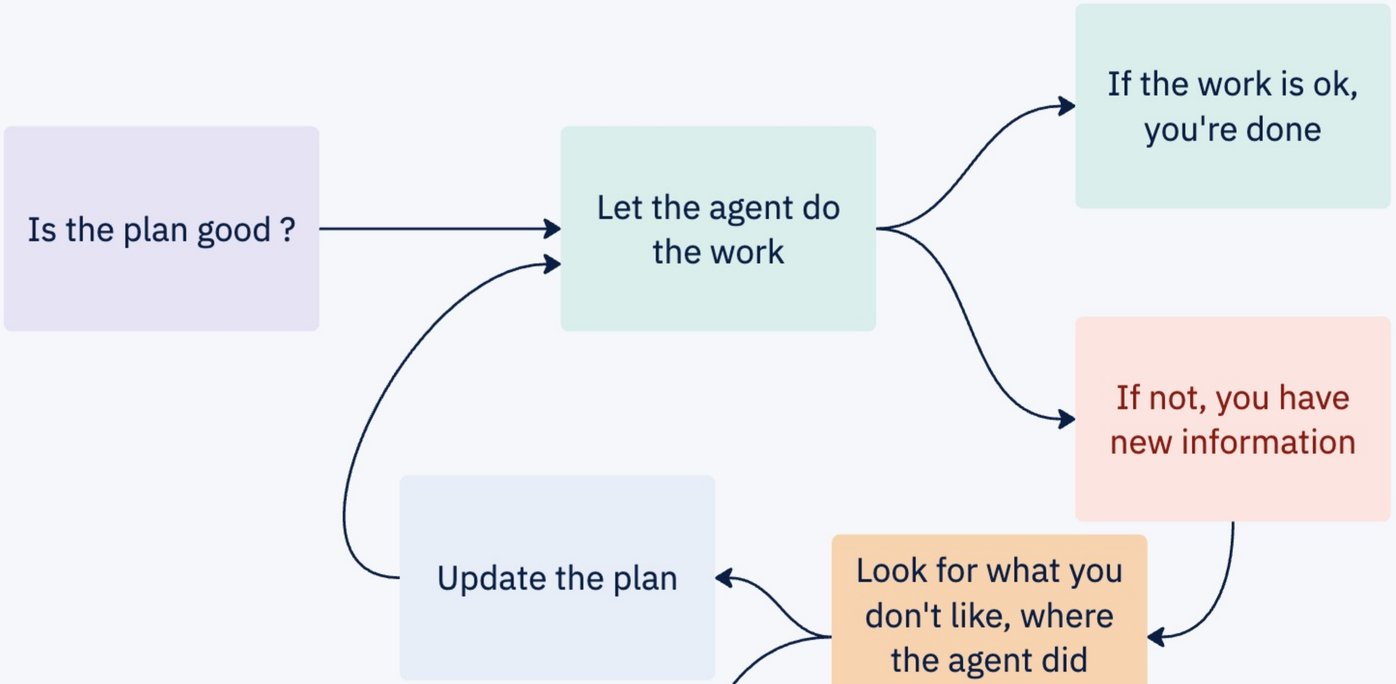
1. Transactional dual write removed exacerbating data sync issues
2. Much simpler migration (no need to create newer temporary events)
3. Refactor and simplify domain model safely

Part 5:
When is the harness finished?

Let the agent fail



Continuous Improvement



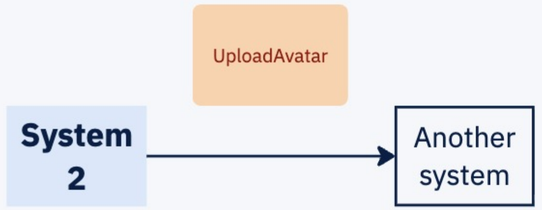
Ask a sub-agents to create a pre-mortem

Give it more rules

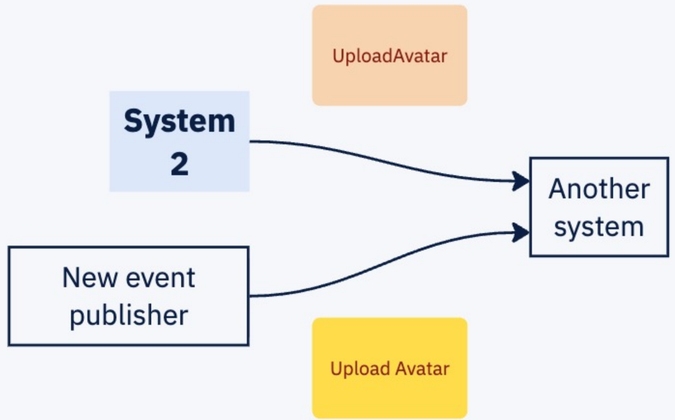
Improve the agent generating the plan

AI proposed a hard cutover... we baked in Expand and Contract

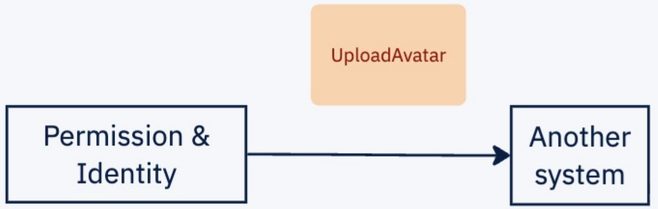
Current State



Transitional phase
(listen for both events)



Target state
(stop listening for old event)



Encode your expertise into the harness with prescriptive instructions.

Goal-oriented language empowers the AI to decide how to achieve the goal.

Algorithmic-like instructions encourage it to think how you want it to

Part 6:

Does AI eliminate dependencies on other teams and collaboration?

Inner sourcing

We created a skill to do the migration work on other teams repo

Knowledge of each endpoint replacement

Adapting itself to the target repository preferences

Collaborating with other teams

Listing all the dependencies from other teams

This is actually much harder than we anticipated

Other team had their own priorities

Release cycle

They have their own infra, ci, processes and ways of working

You have to learn their codebase & conventions

Every team is responsible for the migration

Shared the skill

CTO support



Dependencies between teams is not going away thanks to AI... And the value of collaboration skills is increasing

When you are interacting with others, keep in mind that they have other priorities and preferences than yours.

You'll have to deal with delays.
You must try to make there life easier.
You need global alignement on why that transformation is important.

So you're thinking about building a modernization harness to replace a legacy system?

We advise....



You still have to understand the business goals your optimising for

Build the harness your project needs... driven by business goals, system architecture and people

Having AI + someone who knows the existing system is better than just AI

Offload all the admin to AI, it's good at it and frees you up to focus on interesting things

Built custom tools… insane cheat code

AI doesn't make cross-team dependencies go away. You will need their help.

Collaboration skills = even more vital