

From Key-Person Dependency to a Transferable Operating System

How I converted fragmented product and technical knowledge into an operating system another person could actually use.

THE RISK

Critical knowledge about how the product worked, how it should behave and how the technology was operated was distributed across people, tickets, conversations, systems and undocumented working practices.

The real question was: could the company continue to understand, operate and develop the product if the people holding critical knowledge were no longer there?

01

STRUCTURE THE PRODUCT

- Reorganised the company knowledge base
- Prioritised product and technical roadmap
- Mapped launch stages and dependencies
- Documented customer journeys and feature behaviour
- Captured happy and unhappy paths, rules and requirements

02

EXTRACT TACIT KNOWLEDGE

- Identified knowledge at risk of disappearing with the CTO
- Used AI to understand unfamiliar technical areas and prepare questions
- Ran structured knowledge-transfer sessions
- Challenged gaps and ambiguities
- Converted explanations into reusable operating knowledge

03

BUILD THE OPERATING SYSTEM

- Product SSOTs and feature requirements
- Architecture, local setup and source-code continuity
- Build, release, deployment and rollback
- Testing, incidents, monitoring and technical debt
- Security, access, infrastructure, integrations and decision logs



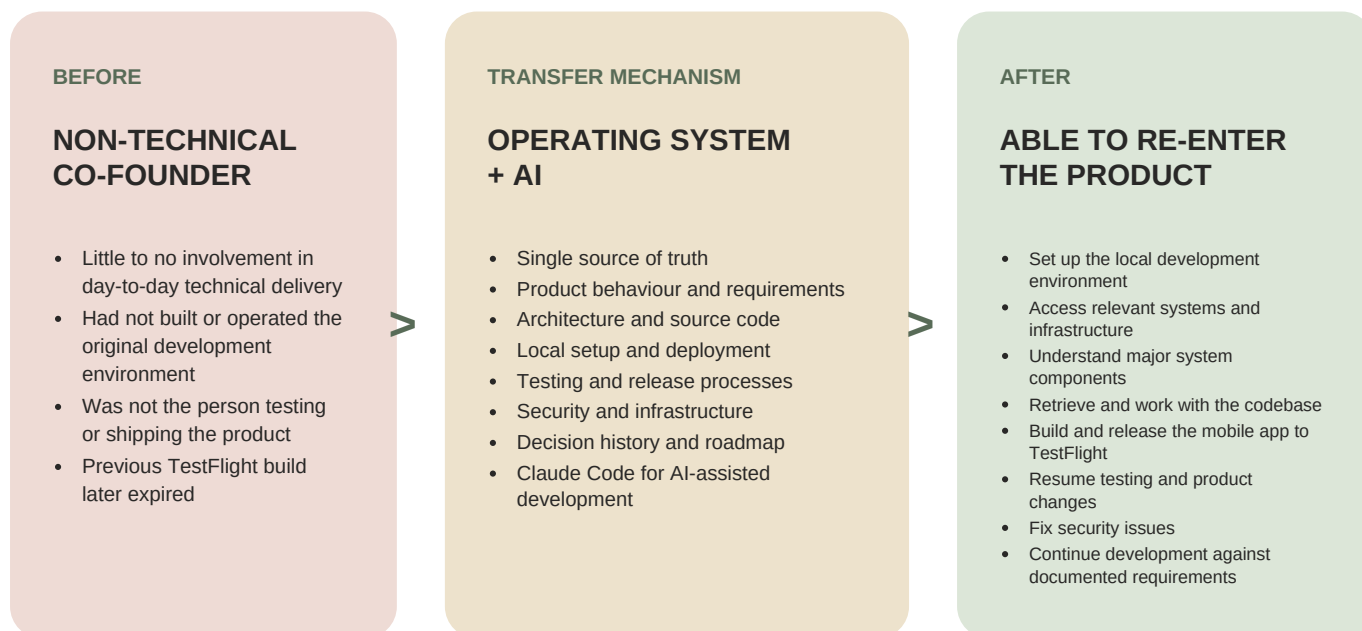
THE DESIGN PRINCIPLE

Documentation wasn't the outcome. Transferability was.

The system needed to let someone unfamiliar with the technical operation orient themselves, understand what existed, know what to do next and act without the original knowledge-holder beside them.

Could Someone Else Actually Use It?

The handover was tested after the original technical knowledge-holder had gone.



THE OUTCOME

A co-founder with virtually no previous involvement in technical delivery was able to take over enough of the environment to understand, build, release, test and continue developing the product using the operating system and AI-assisted development.

He later said explicitly that without the structure, playbooks and knowledge base, he would not have been able to continue the development this way.

A HANDOVER ISN'T COMPLETE WHEN THE INFORMATION HAS BEEN WRITTEN DOWN.

It's complete when someone who didn't hold the original knowledge can use it to understand the system, make decisions and continue the work.

CAPABILITIES DEMONSTRATED

Knowledge Architecture | Operational Resilience | Process Design | Product Operations | Technical Governance | AI-Assisted Knowledge Transfer