10 Proven Approaches to Reinsurance IT Projects That Actually Deliver.

Toucanberry Tech

Dataflow Design.

When we sit down with reinsurers struggling through a ”transformation,” the pattern is almost always the same. It’s not that the technology was bad. It’s that they solved for the wrong problems.

At Toucanberry, we take a different approach.

We focus on **Dataflow Design** - mapping how information, decisions, and outcomes actually move through the business. Then we design systems that accelerate those flows, not block them.

Here are 10 frameworks we use to help clients deliver projects that don’t just launch - they compound operational advantage...

Jobs To Be Done (JTBD)

Forget features. Start with real-world jobs users are trying to accomplish. If you can't describe the "when, I want to, so I can" moment for your system, you don't have a system - you have a wishlist.

EXAMPLE: Instead of specifying "a treaty pricing system," define the jobs: "When pricing a new treaty, actuaries need to access historical loss experience alongside current treaty terms, so they can price accurately within a 24-hour turnaround window."

Interview 3-5 users and ask "What are you trying to accomplish?" rather than "What features do you need?"

Use the format: "When [situation], I want to [motivation], so I can [outcome]"

Look for emotional responses that signal important jobs

Resources:
[JTBD Framework]
[Intercom on JTBD]

Marty Cagan's Discovery Principles

Most projects fail at the first hurdle because they don't validate real user pain. If you haven't spent serious time with frustrated users before designing anything, you're just guessing, which can prove very expensive.

EXAMPLE: We spent three days shadowing settlement operations before proposing any solution. What we found contradicted the project brief entirely - the problem wasn't data quality but workflow handoffs.

Schedule "ride along" sessions with actual users

Ask: "Show me what breaks," not "Tell me what you want"

Identify the riskiest assumptions in your project and test them first

Resources:
[Inspired by Marty Cagan](#)
[Silicon Valley Product Group](#)

Continuous Discovery Habits

Discovery isn't a one-time workshop. It's a weekly conversation with users - spotting shifting needs before they derail six months of work.

EXAMPLE: Regular 30-minute sessions with actuaries revealed that a proposed valuation dashboard wasn't solving their real problem: they needed better error detection during data imports, not better visualisation after the fact.

Schedule bi-weekly 30-minute conversations with key users

Focus on recent challenges they've faced, not feature requests

Use a simple opportunity solution tree to track insights

Resources:
[Continuous Discovery Habits by Teresa Torres](#)
[Product Talk](#)

User Story Mapping

Most Request For Proposals (RFPs) assume a user journey that doesn't exist. By gathering representatives from all teams involved in a process, User Story Mapping forces you to visualise how people actually move through tasks - and where they really get stuck.

EXAMPLE: Mapping the entire treaty onboarding process revealed that what executives saw as a "system problem" was actually a coordination issue between legal, pricing, and finance teams.

Map the process left-to-right in chronological order

Place activities vertically from "must have" to "nice to have"

Look for handoff points and information gaps

Resources:

[User Story Mapping by Jeff Patton Quick Guide]

Charlie Munger's Inversion Principle

Instead of dreaming about success, start by imagining failure. Ask: "How will this definitely fail?" You'll find your biggest risks faster - and design harder, smarter systems.

EXAMPLE: By starting with "How would users work around this system?" we identified critical integration points that weren't in the original plan but were essential for adoption.

Run a pre-mortem workshop: "It's one year later and this project has failed. Why?"

List all the reasons for failure, from technical to organisational

Prioritise the most likely failure points and address them first

Resources:
Farnam Street
James Clear

Critical Path Mapping

Not every feature is equally important. Critical Path Mapping forces you to see what must work first for anything else to matter.

EXAMPLE: A reinsurer had planned a complex data lake project, but critical path mapping showed that cleaning up their treaty data intake process had to happen first - saving months of rework.

Identify the absolute minimum viable product for user value

Map dependencies between components and features

Sequence development to tackle the critical path first, constantly reassessing as you learn more

Resources:
[Project Management Institute]
[Critical Chain Project Management]

Service Blueprinting

Great user experiences fail when backstage systems break down. Service Blueprinting maps both frontstage (user) and backstage (ops/data) flows together - because your users don't care whose fault it is when it doesn't work.

EXAMPLE: Mapping the entire settlement process revealed that front-end improvements would be useless without fixing how data flowed between actuarial and finance systems.

Map the process from the user's perspective (frontstage)

Add behind-the-scenes activities (backstage)

Identify supporting systems and data flows, looking for front and backstage disconnects

Resources:
[Service Blueprint Guide]
[Service Design]

The Iceberg of Ignorance

Most operational problems are invisible to the exec team. Real discovery starts at the coalface: underwriters, pricing teams, ops managers.

EXAMPLE: Executive interviews suggested a pricing efficiency problem; frontline interviews revealed it was actually a data access issue that could be solved without building new systems.

Interview users at ALL levels, asking the same questions to compare perspectives

Look for patterns that senior leaders might miss

Pay attention to workarounds and "unofficial processes"

Resources:

[The Iceberg of Ignorance Explained]

[Toyota Kata by Mike Rother]

Risks Before Requirements

Every project has killer risks: Bad data, no adoption, wrong integration points.If you don't surface and attack those risks first, the rest doesn't matter.

EXAMPLE: We identified data quality as the top risk for a reporting platform. By building a quality assessment tool first, we prevented months of building dashboards that would display unreliable information.

List all assumptions that could derail the project if wrong

Rank risks by impact and uncertainty, tackling high-impact, high-uncertainty risks first

Build experiments to test assumptions before full development

Resources:
[The Lean Startup](#)
[Hypothesis Prioritization Canvas](#)

Amazon's Working Backwards Process

If you can't write the press release for your platform before you spec it, you're not ready to build it. Working Backwards forces brutal clarity: who benefits, why it matters, and what actually changes.

EXAMPLE: Writing a future announcement for a reinsurer's valuation system revealed they couldn't articulate tangible benefits. This sparked a necessary reset of project objectives before coding began.

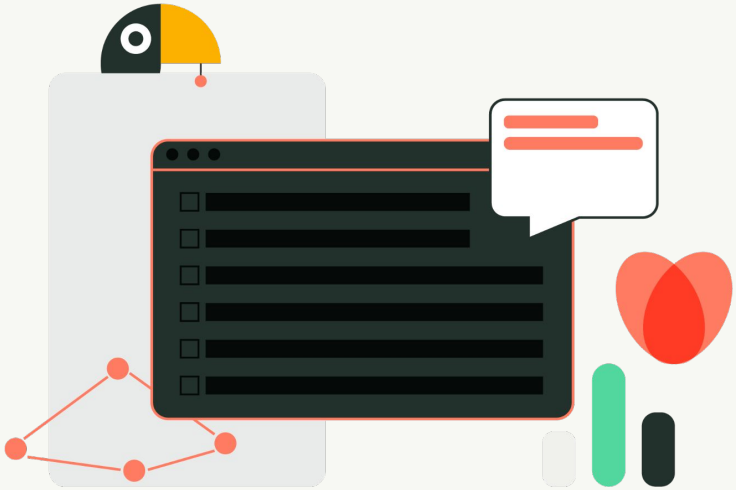
Write a one-page press release announcing the completed project

Include specific customer quotes and measurable outcomes, listing FAQs that address concerns & objections

If you struggle to write this, you don't understand the project well enough

Resources:

[Working Backwards How to Implement Amazon's "Working Backwards" Approach]



The Toucanberry Approach.

We don't build software for its own sake. We **design dataflows** that create your competitive advantage.

That's why every project we run follows the same three steps: **Advise. Design. Build.**

- Sharp discovery.
- Ruthless prioritisation.
- Systems people adopt because they make work better.

Want to build systems users actually want to use?

Let's talk about how we apply these frameworks inside real reinsurance workflows, and how we help teams scale smarter, not just bigger.

Get in touch: laith@toucanberry.com