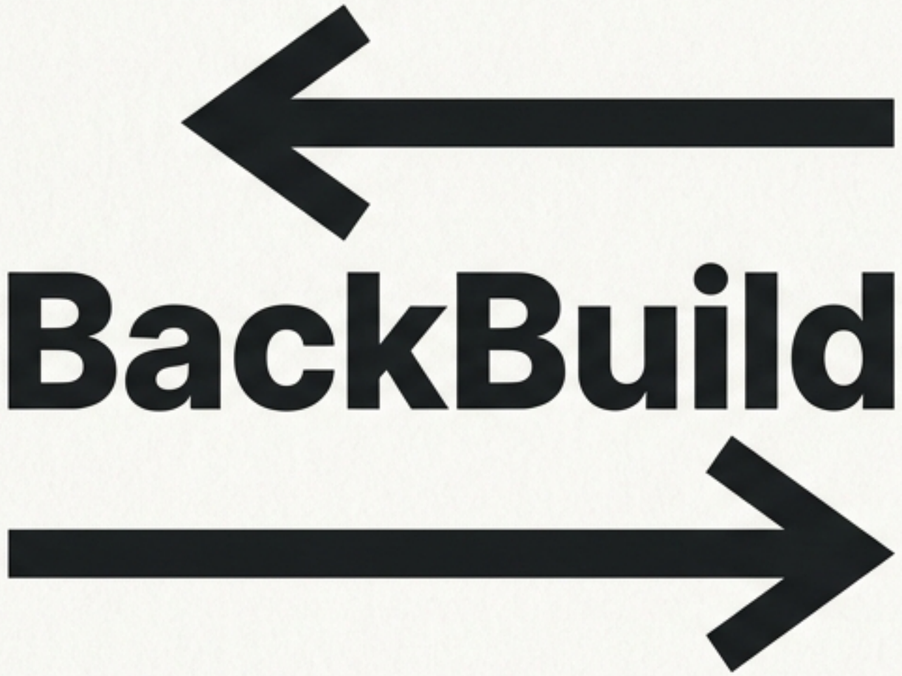


The BackBuild Framework

Start from the end. Work backwards. Build with foresight.

Your most ambitious projects are at risk before a single line of code is written.

Large and complex software programmes—especially multi-feature apps, legacy-tech migrations, and redesigns—burn millions because teams start work before the real order of work is understood. A lack of foresight leads to ballooning costs, project delays, and sometimes, complete failure. No business is immune.

Does this chaos feel familiar?

Design starts in random places

UIs are created without knowing API constraints, data availability, or underlying logic.

Result: Rework, team tension, and wasted months.

Engineering squads start too early

Full teams are hired from day one with nothing ready to build.

Result: Idle engineers burning budgets while everyone else scrambles.

Hidden dependencies appear too late

Critical data gaps, legal blockers, or technical unknowns emerge after designs are done and work is estimated.

Result: Endless rewrites that can kill the entire project.

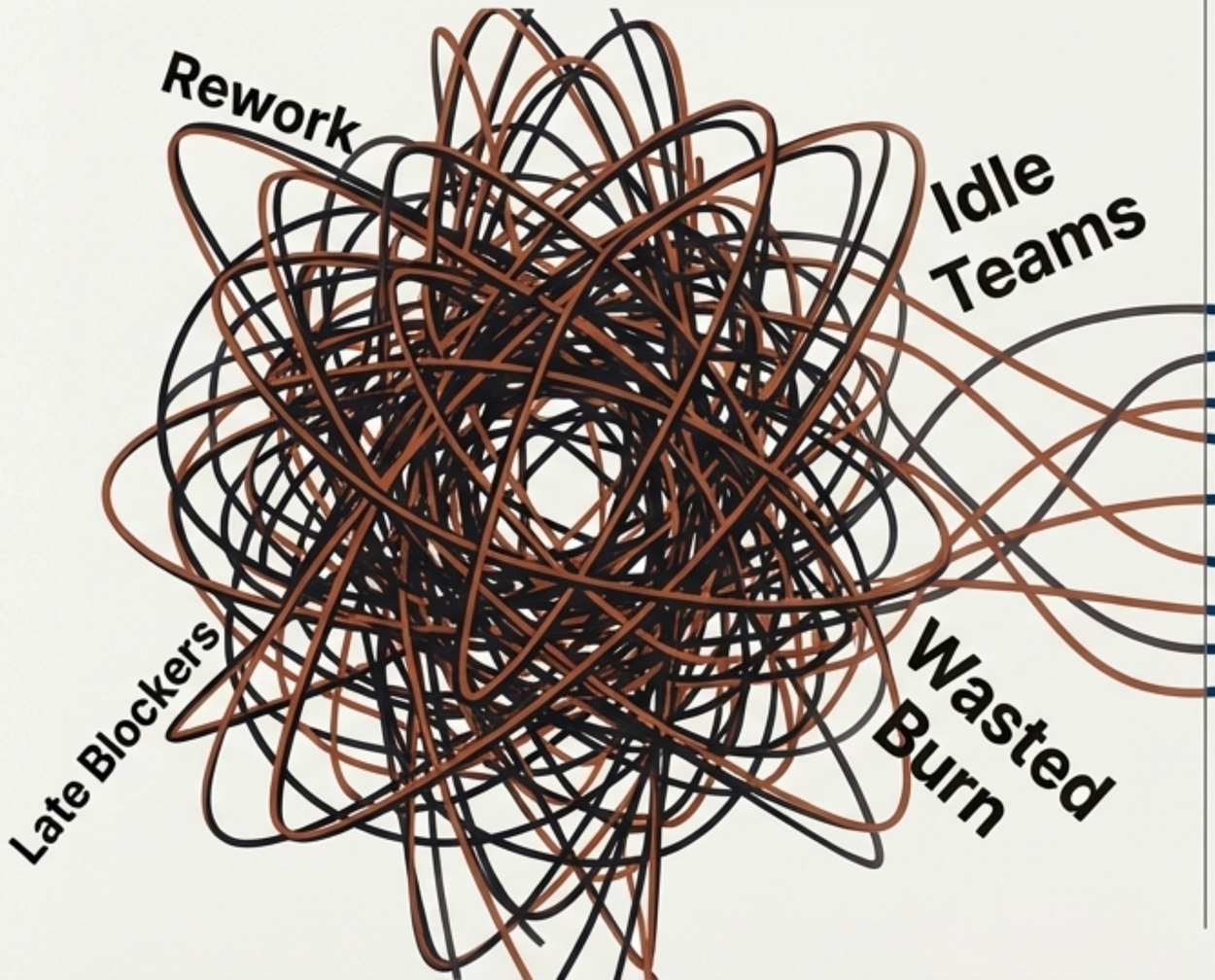
Team shape stays fixed

Rigid squad structures don't match the actual needs of the work phase.

Result: Overloaded designers, idle engineers, and blown timelines.

From Reactive Chaos to Proactive Clarity.

Before BackBuild



With BackBuild

Sequenced Plan

Right-Sized Teams

Risks Front-Loaded

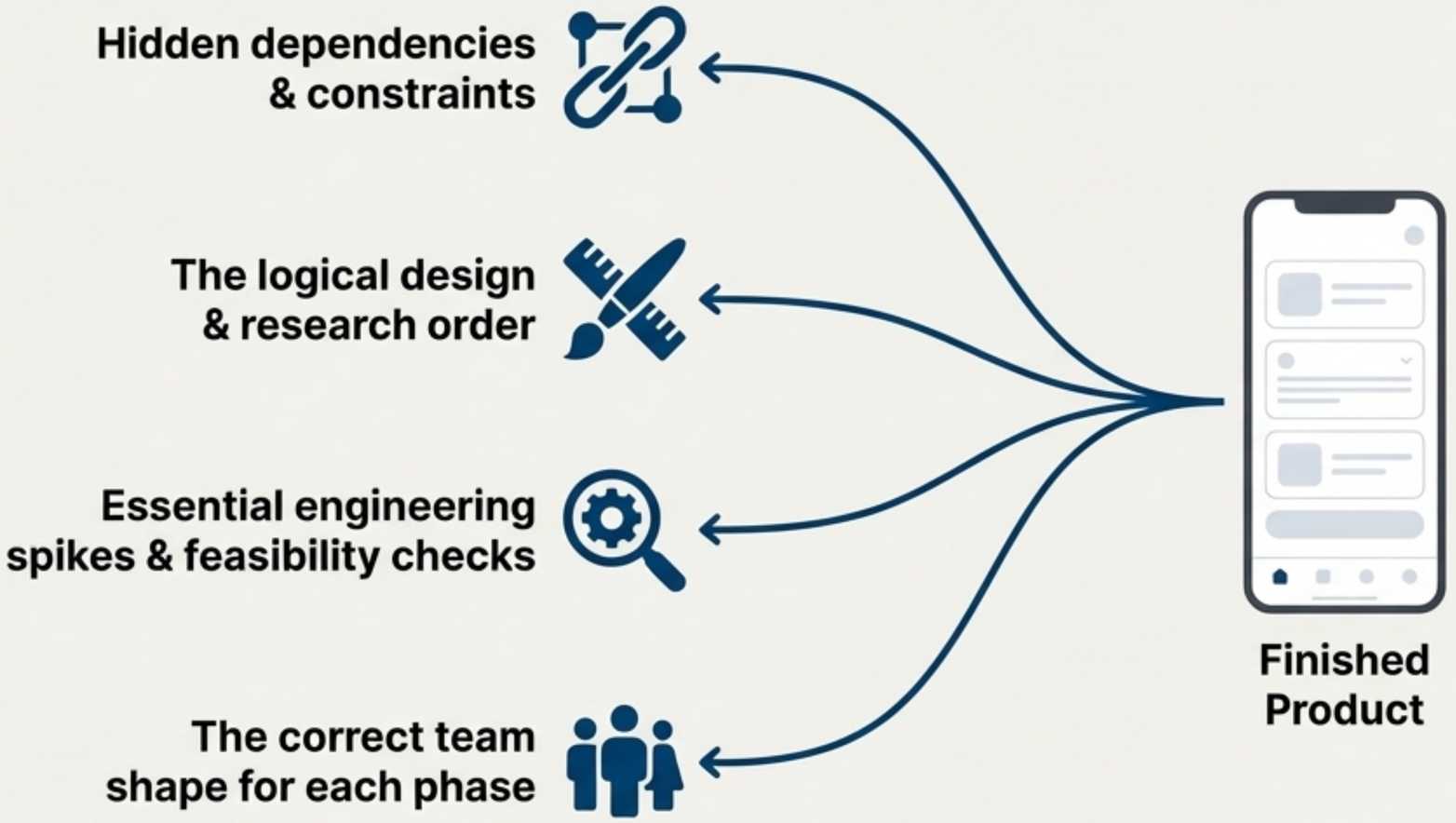
Efficient Burn

BackBuild brings order to chaos by starting from the end.

****Core Concept**:**

BackBuild is a framework that slots into Agile to prevent the biggest delivery risks. By imagining the finished product and working backwards, it reveals what *must be true* for that future to exist.

****The Outcome*:** It turns this backward map into a low-waste, low-risk forward delivery plan grounded in reality, not guesses.



A 10-Step Process in Three Strategic Phases.

Phase 1: Map the Future
(Discover & Align)

1. Sketch the End Picture
2. Work Backwards from Each Journey
3. Identify Preconditions

Phase 2: Create Order
(Sequence & De-Risk)

4. Connect the Logic (3-5 Levels Deep)
5. Determine Design Order
6. Define Engineering Spikes & Feasibility

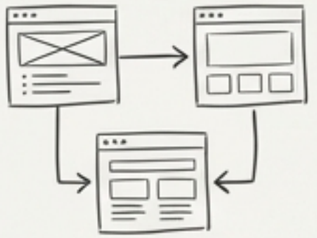
Phase 3: Build Forward
(Plan & Adapt)

7. Shape Teams and Ramp Them Up
8. Create the Forward Plan & Parallel Tracks
9. Isolate Fragile Spots (Critical Assumptions)
10. Establish the BackBuild Loop

Phase 1: Map the Future

This phase is about creating a shared, low-fidelity vision of the end state and using it to uncover the high-level needs and foundational requirements.

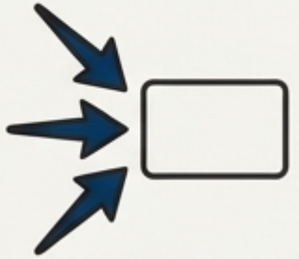
1. Step 1: Sketch the End Picture



Create a rough, conceptual map of core screens, flows, and features.

Output: High-level future-state sketches and an initial feature inventory.

2. Step 2: Work Backwards from Each Journey



For each feature, ask "What needs to be true for this to exist?" and map the data needs, APIs, logic, etc.

Output: A broad dependency list for each journey.

3. Step 3: Identify Preconditions

Capture foundational blockers like regulatory approval, data availability, or 3rd-party contracts.

Output: A clear list of red flags to resolve early.

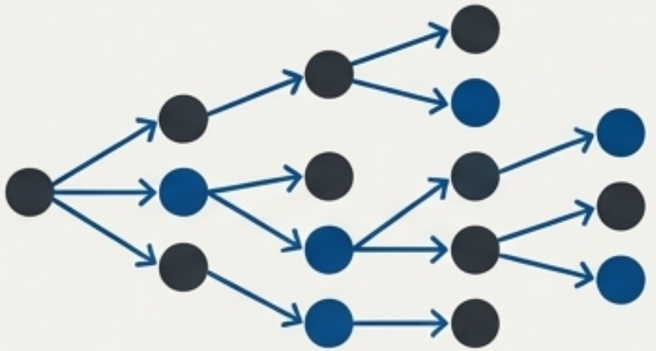
Phase 2: Create Order

Here, we transform the raw map into an ordered sequence of work. We connect the dots to reveal the true critical path and proactively tackle technical demons.

Step 4: Connect the Logic

Map dependencies 3-5 levels deep until you hit a root, revealing the true sequence and effort.

Output: A visual dependency map showing all hidden work.



Step 5: Determine Design Order

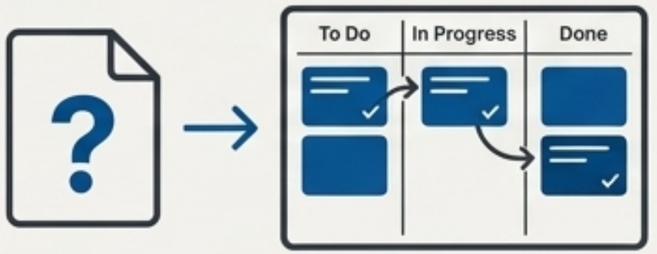
Use the dependency map to create a rational, unblocking order for design and research.

Output: A sequenced design plan that enables engineering.

Step 6: Engineering Spikes

Every technical unknown (‘Can the API do this?”) becomes a prioritized task for engineering to investigate immediately.

Output: A prioritized spike backlog and early feasibility results.



Phase 3: Build Forward

With a clear, de-risked map, we now shape the team and the plan for forward execution, building in mechanisms for continuous adaptation.

Step 7: Shape Teams

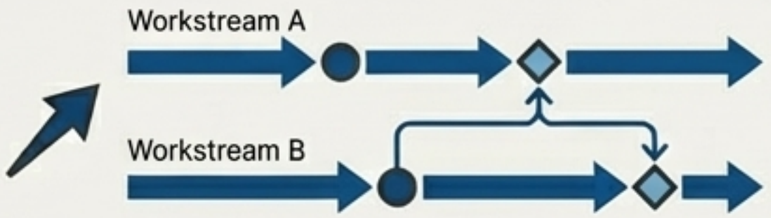
Use the map to determine exactly which roles are needed and when, ramping teams up and down to eliminate waste.

Output: A right-sized team structure over time.

Step 8: Forward Plan

Flip the backward map to create a realistic, sequenced roadmap with parallel workstreams.

Output: A forward delivery roadmap with dependencies resolved.



Steps 9 & 10: Isolate Fragility & Loop

Explicitly identify critical assumptions to test in Sprint 1, and establish a rhythm to update the map based on new learnings.

Output: A living strategic plan that adapts to reality.

The Result: A Roadmap Grounded in Reality, Not Hope.

- A clear understanding of what the final product **really** requires.
- Early discovery of dependencies and hidden demons.
- A logical, unblocking order for design work.
- Engineering spikes identified **before** sprinting begins.
- Correct team shapes at every phase of the project.
- No idle engineers, no burnt-out designers.
- Lower cost, higher speed, and dramatically less waste.
- Confidence in a delivery plan that is achievable.

BackBuild is a specialized tool for complex challenges.

Perfect For... ✅

- New multi-feature apps
- Major redesigns
- Legacy migrations
- Fintech / banking systems
- Products with heavy backend logic
- High-risk integrations

Not Suitable For... ❌

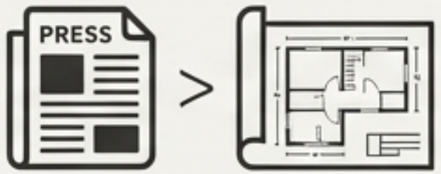
- Zero-to-one discovery products without a clear end vision.

Key takeaway: BackBuild provides structure when the destination is known but the path is complex and **full** of unknowns.

How BackBuild is different from what you already use.

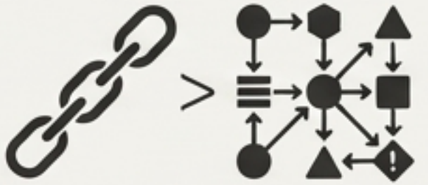
vs. Amazon's Working Backwards

Amazon validates the *idea*. BackBuild structures the *build* after the idea is validated.



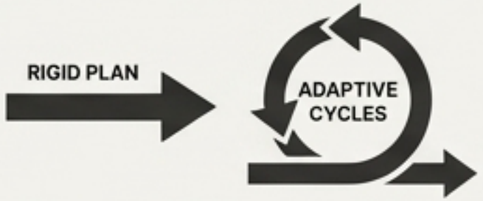
vs. Dependency Mapping

Traditional mapping shows task order. BackBuild integrates this with product strategy, design sequencing, team shaping, and risk testing into a single, cohesive framework.



vs. Traditional Waterfall

Waterfall locks in a rigid upfront plan. BackBuild's end picture is deliberately provisional and adapts via mandatory feedback loops.

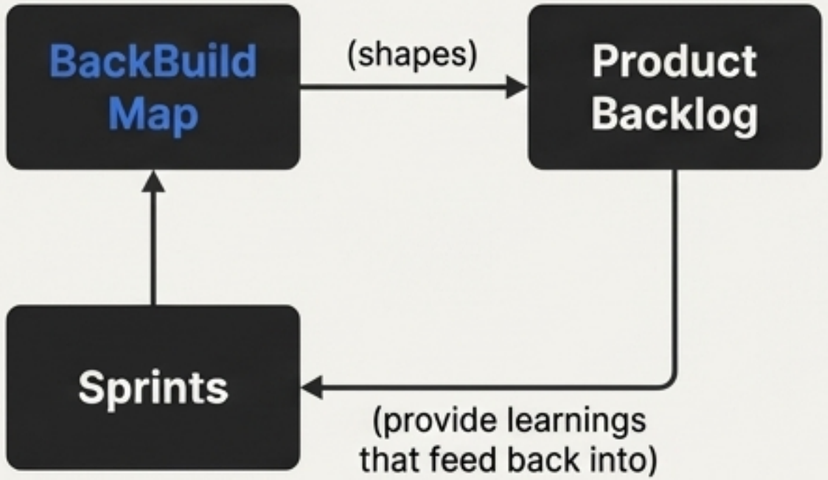


vs. Agile Backlogs

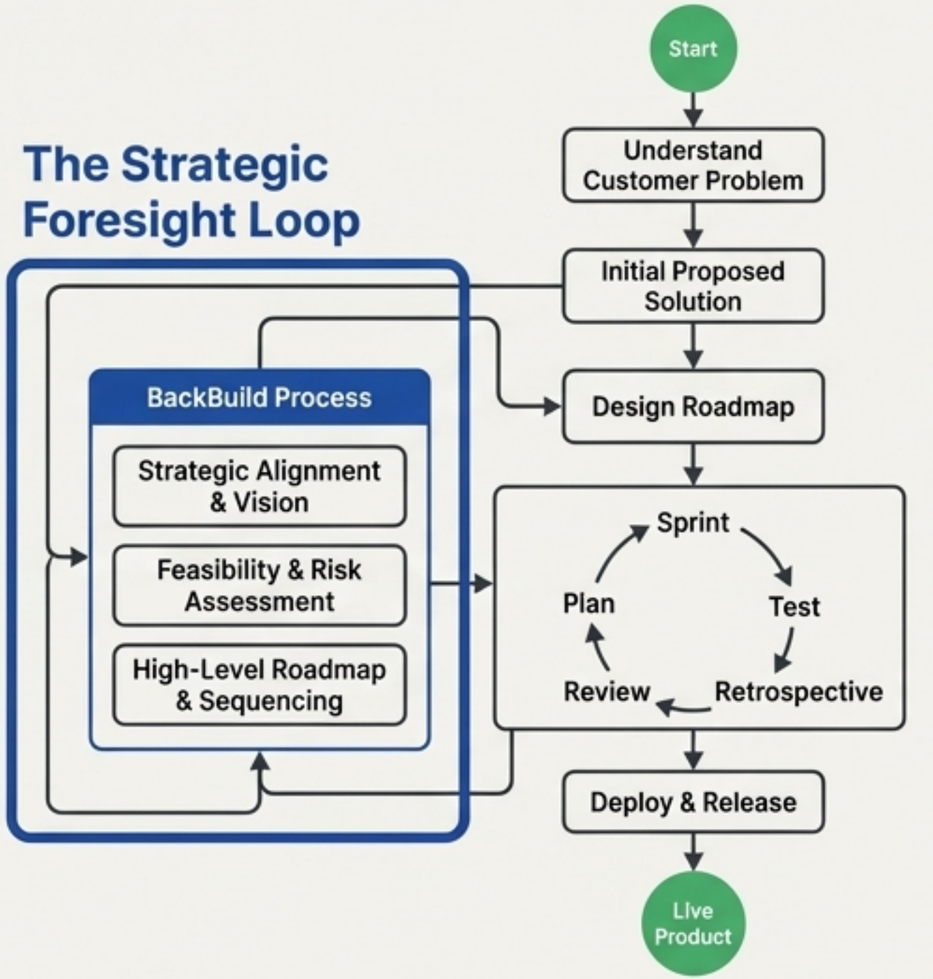
Backlogs manage sprint-level work. BackBuild is a living strategic layer *above* the backlog, shaping what goes into it to ensure a coherent, efficient flow.

BackBuild is the strategic layer that feeds Agile delivery.

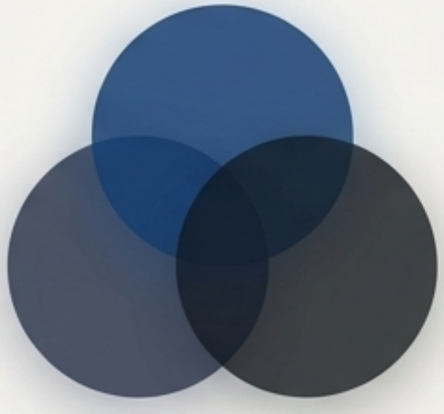
BackBuild is not a one-off plan. It's a living map that runs in parallel with delivery. It provides strategic foresight, while Agile sprints provide empirical learning. They reinforce each other.



Example End-to-end Software Delivery Lifecycle



	Outputs	Who?
Understand Customer Problem	Customer Needs Analysis, Persona Definitions	Product Manager, UX Researcher
Initial Proposed Solution	Concept Models, Value Proposition	Product Manager, Architect
Design Roadmap	Strategic Plan, Risk Register, Sequenced Roadmap	Lead Architect, Product Lead, Engineering Lead
Sprint (Plan, Test, Review, Retrospective)	Detailed Specifications, Wireframes	Designer, Product Owner
	Increment of Working Software, Test Results	Development Squad (Dev, QA, PO)
Deploy & Release	Released Feature/Product, Deployment Notes	DevOps, Squad
Live Product	User Feedback, Usage Data, Performance Metrics	Product Manager, Data Analyst



A framework is only as good as the team that uses it.

Success with BackBuild requires handling the human side well.

1.
Getting Buy-in

Frame it as a way to save money and protect team focus. Prove its value with a small, 2-hour pilot on a single feature.

2.
Managing Resistance

For Engineers: “This protects your time from rework and ensures you always have meaningful problems to solve.”

For Designers: “This clarifies the ***order*** of work, not the creative content. It ensures your focus is on what matters most, now.”

3.
Creating Psychological Safety

Celebrate the discovery of “fragile spots.” They aren’t criticisms; they are opportunities to prevent future failure.

Stop building on hope. Start building with foresight.

The BackBuild Framework is a lightweight, repeatable process for de-risking complex programmes, saving money, and keeping delivery flowing.

If you want to learn more about how to use and implement the BackBuild framework within your business, we can help.

Get in touch: contact@arcaned.co