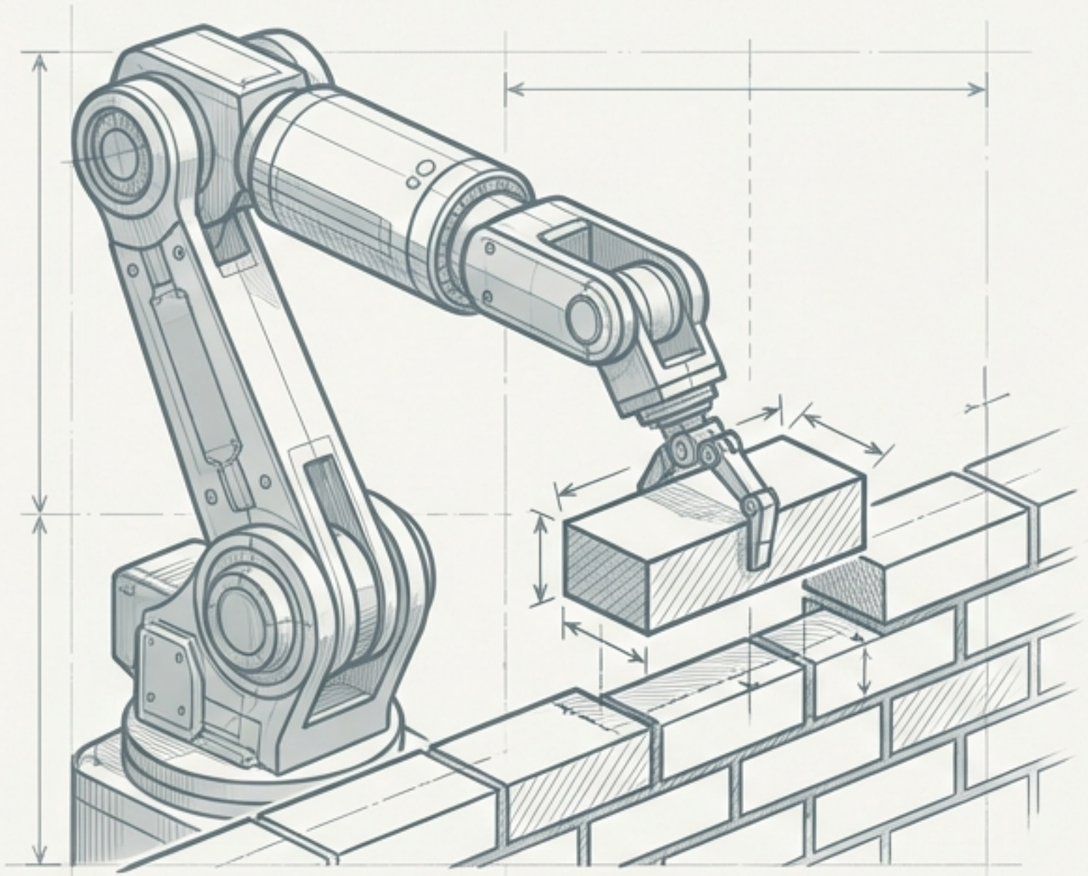


Mission-Driven Teams

A Practical Framework for the Age of Agentic AI

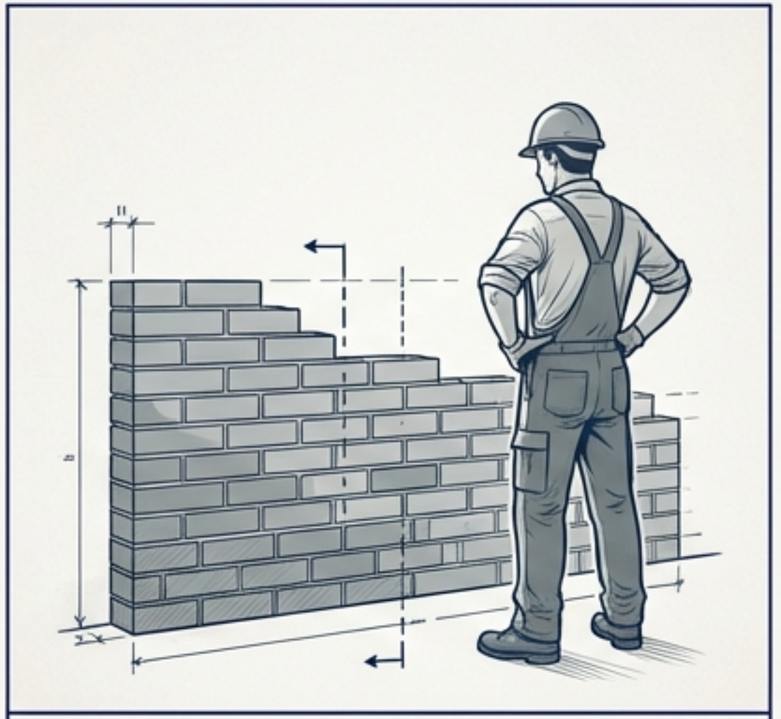
Building human-centered teams that excel at deciding *what* to build and *why*.

A reading deck by Murali Mallina

The Parable of the Three Bricklayers

Bricklayer 1: “I’m laying bricks.”

The Task



Bricklayer 2: “I’m building a wall.”

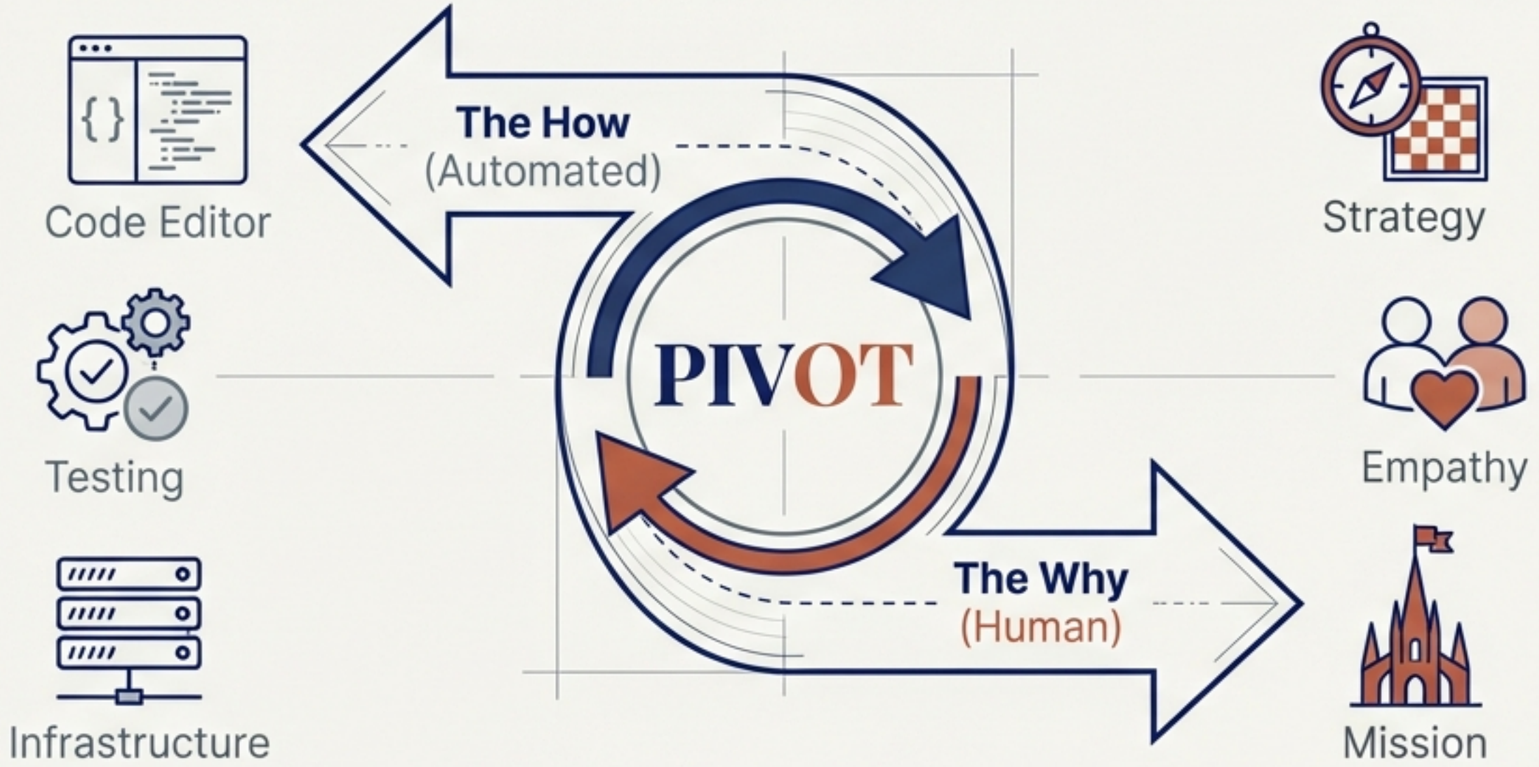
The Output

Bricklayer 3: “I’m building a cathedral.”

The Mission

The Thesis: Teams that only see **bricks** (tickets/tasks) **miss the cathedral (impact).**
In 2026, AI lays bricks faster than ever. Human ingenuity must focus on the cathedral.

The Landscape of 2026: The Rise of Agentic AI



The Shift:
Agentic AI systems now design interfaces, write code, generate tests, and deploy infrastructure. Implementation is being commoditized.

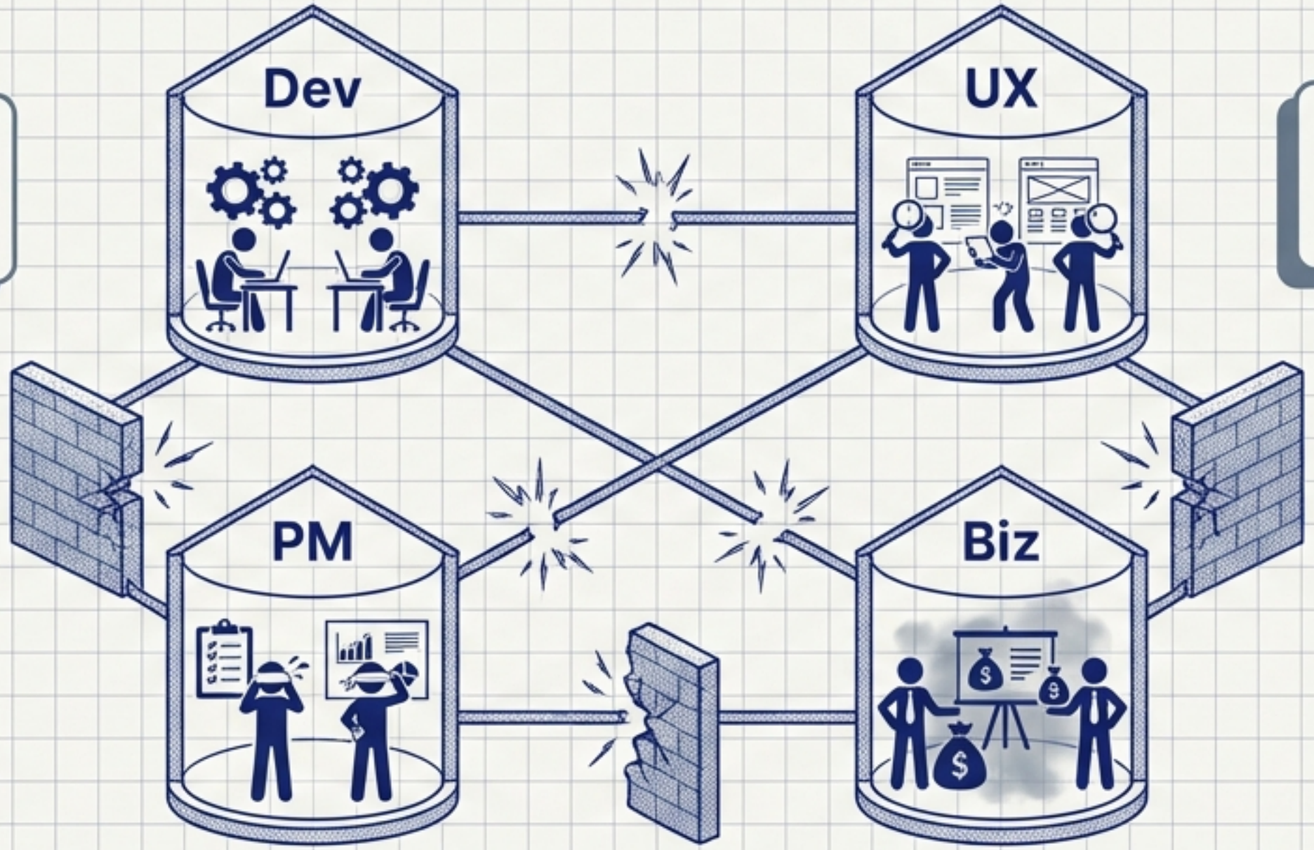
The Risk: "Decisions in the Dark." AI can build the wrong things efficiently if humans lack context.

The Necessary Pivot:
From technical implementation (The Bricks) to strategic reasoning and empathy (The Cathedral).

When AI handles implementation, human ingenuity, empathy, and strategic reasoning become the competitive advantage.

The Problem: The "Fragmented Team" Trap

"I just need to finish my tickets. QA will catch any issues."



"I'm not sure how it fits into the user's actual workflow."

"Making it we're artual probiems."

"I feel like we're making decisions in the dark."

Patterns of Failure

1. **Fragmented Understanding:** Individuals optimize for functions, not goals.

2. The Telephone Game: Filtered feedback strips away context.

3. Diluted Ownership: "Not my job" mentality prevails.

The Solution: The Mission-Driven Teams Framework

A unified system to transform fragmented groups into units that understand the “Why”.

Principle 1: Shared Mission & Vision

The “North Star.” Everyone must know ***why*** the product exists.

Before: Fragmented

- Developers focus on completing tickets
- Decisions require constant escalation
- Distinct interpretations of goals per role

After: Unified

- Developers understand user problems
- Autonomous decision-making
- All roles share definition of success

Practical Approaches

- Mission Kickoffs for every project
- Visible Artifacts (Mission Posters/Canvas)
- OKRs linked directly to user outcomes

The AI Pivot

When AI generates code, the critical question shifts from “can we build it?” to “should we build it?” AI requires human mission alignment to avoid efficient failure.

Principle 2: Break Down Silos

Eliminate ‘over-the-wall’ handoffs. Operate as one cohesive unit.

✕ Before: Siloed

- Sequential hand-offs (Analyst -> Design -> Dev -> QA)
- Separate stand-ups per function
- Information lost at every step

✓ After: Collaborative

- Unified stand-ups with all roles
- Simultaneous collaboration (Swarming)
- Feature Teams that succeed or fail together

Practical Approaches

- Unified Backlogs (One list for all work)
- Feature Teams (Cross-functional squads)
- Cross-Functional Pairing (Dev + Design)

The AI Pivot

Cross-functional human review is the new quality gate. Designers must review AI-generated code for UX; Developers must review AI designs for feasibility.

Principle 3: Direct User Engagement

Empathy cannot be outsourced or automated. It must be firsthand.

✕ Before: Filtered

- Feedback flows through PMs or Analysts
- Insights are “watered down” by intermediaries
- Building based on guessed requirements

✓ After: Direct

- Developers observe usability tests directly
- Shadowing users in natural environments
- Support ticket rotation for engineers

Practical Approaches

Job Shadowing

Sprint Reviews with real users

User Interviews involving Dev/QA

The AI Pivot

AI can process user data, but only humans can feel user frustration. Direct engagement prevents “hallucinated” user needs.

Principle 4: Outcomes Over Outputs

Stop measuring velocity; start measuring impact.

✕ Before: Activity

- Success = Story points completed
- Celebrating shipping features that go unused
- Focus: “Did we build it?”

✓ After: Impact

- Success = User behavior change
- Celebrating solved problems
- Focus: “Did it work?”

Practical Approaches

- Outcome-based User Stories (focus on “so that”)
- Post-Launch Reviews (Did metrics move?)
- Defining Success Metrics Upfront

The AI Pivot

AI creates infinite output. Humans must filter for outcome. Resist measuring AI success by lines of code generated.

Principle 5: Continuous Domain Learning

You can’t solve problems you don’t understand.

✕ Before: Surface Level

- Technical focus only
- Missing the forest for the trees
- Reliance on PMs for all domain answers

☑ After: Expertise

- Deep industry knowledge distributed across team
- Understanding upstream/downstream impacts
- Innovative solutions born from context

Practical Approaches

- Ecosystem Mapping
- Service Design (Front/Back stage)
- Immersion Activities (Conferences, Visits)

The AI Pivot

AI creates technically correct but contextually wrong solutions. Domain-aware humans are the firewall against valid code that violates business reality.

Principle 6: The Art of Storytelling

The bridge between technical work and human impact.

✕ Before: Jargon

"I'm updating the microservice API"
Disconnected stakeholders
Lack of pride in the work

✓ After: Narrative

- "I'm reducing patient wait times"
- Emotional connection to work
- Clear value translation for stakeholders

Practical Approaches

The Dinner Table Test (Explain it to family)

"Before & After" Narratives

Impact Storytelling Sessions

The AI Pivot

Storytelling builds the emotional connection that data (and AI) cannot. It connects the 'what' to the 'why'.

Getting Started: A Practitioner's Guide

Keep it simple. Don't replace Agile/Scrum; enhance it.

Developer

- Join a support call this week.
- Ask "why" on the next ticket you pick up.
- Pair with a designer on implementation.

Designer

- Invite a developer to your next user interview.
- Review code output for UX intent.
- Create "Before/After" visuals for the team.

Product Manager

- Share a real customer story at the next stand-up.
- Define success metrics *before* building.
- Bring the team into the problem space early.

Toolkit Available: Refer to Role-Specific Workbooks for detailed exercises.

The ROI of Mission-Driven Teams

1

Autonomous Decision Making

Teams don't wait for permission. They act effectively because they understand the **'North Star'** and can mw trade off decisions locally.

2

Higher Velocity

Less rework. Because the problem was understood correctly the first time, the team avoids building the wrong thing efficiently.

3

Deep Engagement

Meaningful work retains talent. Connecting code to **human impact** creates a sense of purpose that salary alone cannot match.

Good developers don't need requirements: they connect to customers and own outcomes over following strict requirements.

AI Context AI is a powerful tool, but the human connection to the mission is the true driver of ROI.

Summary Checklist

- [ ] **Mission:** Do we have a “North Star” visible to everyone?
- [ ] **Silos:** Do we have a unified backlog and stand-up?
- [ ] **Users:** Have we watched a user struggle this month?
- [ ] **Outcomes:** Are we measuring behavior change or just shipping code?
- [ ] **Domain:** Do we understand the ecosystem beyond our feature?
- [ ] **Storytelling:** Can we pass the “Dinner Table Test”?

Not a finished product, but a starting point.

Read it, discuss it with your team, and make it your own. Experiment with the principles. Refine them based on what works in your context.

AI Context

Final Thought: In the age of AI, the team that understands the **context** wins.

About the Author: Murali Mallina

- CTO with 27 years of experience building teams.
- Expertise in Federal, Fortune 500, and Startup environments.
- Focus: AI/ML, DevSecOps, and Scalable Architectures.