Agile delivery with AI

How small teams turn business decisions into verified software

Christopher R. Marrs | Executive Technology Advisor

Executive summary

AI can help a small team carry more of the work involved in software delivery. It can draft implementation briefs, produce code and tests, review proposed changes, investigate failures, and maintain documentation. Taking advantage of that capacity requires an operating model that keeps business intent, technical evidence, and decision authority connected.

In a recent engagement with a founder building a service-matching platform, AI participated across those activities. The work rested on multiple introductory, discovery, demonstration, and planning meetings. Those conversations established context that we carried into implementation through written decisions and bounded work packages. Testing continued to change our understanding as the product took shape.

The most revealing moment came when a test passed. The system followed the rule we had defined, but seeing the result made the founder realize that the categories did not reflect how the business needed to operate. That discovery led to a revision across the routing logic, data, and customer experience.

This case illustrates a shift in where delivery needs attention. As implementation becomes easier to generate, clear decisions, current context, and reliable review become increasingly consequential. Leaders should evaluate the full path from business understanding to accepted behavior when deciding how to organize AI-assisted work.

The coordination question

SAFe addresses alignment across multiple teams, shared priorities, dependencies, and delivery obligations. PI Planning creates a common direction and makes those dependencies visible. Its broader planning horizon operates alongside smaller units of execution. SAFe also supports release on demand, so its planning cadence does not require every change to wait for the next interval. [1, 2]

A small team with direct access to the business owner faces a different coordination problem. In this engagement, we could carry decisions through fewer organizational handoffs and use AI across several delivery activities. That combination invites a practical question: which coordination practices does this team need to move work reliably?

Customer collaboration, changing requirements, working software, and reflection remain central Agile principles. AI gives teams additional ways to perform the work around them. The choice of operating model still depends on the product, the people, and the consequences of getting it wrong. [3]

Discovery made the delivery possible

The visible implementation was only part of the engagement. We had already spent time in introductions, business discussions, demonstrations, and planning. The founder supplied domain knowledge and challenged assumptions. I helped translate the operating intent into architecture, decision rules, and a sequence of work we could evaluate.

Those conversations were productive because they surfaced questions that a coding prompt could not settle. What made a provider eligible? How should territory and service type interact? What should happen when the preferred match was unavailable? Which decisions belonged in the platform, and which remained with the business owner?

We recorded the resulting decisions so they could travel with the work. A brief could then identify the desired behavior, the relevant rules, the affected components, and the evidence needed to judge completion. AI tools could use that context when performing an assignment, while humans retained responsibility for resolving ambiguity.

Understanding continued to develop during the build. Demonstrations and tests gave the founder something concrete to react to. The written record needed to follow that learning, including when a new decision replaced an earlier one.

A passing test exposed a business assumption

The platform matched customer requests to eligible providers using service type, territory, and defined fallback rules. During testing, the founder received the result we had designed. The software behaved correctly under the stated rule. The result also showed that a category was too broad for the distinction the business needed to make.

That was useful feedback about the product model. The founder clarified the categories, and we translated the decision into changes across the service, its data, and the public-facing experience. AI assisted with the implementation brief, the proposed changes, and the checks that followed.

An updated service list arrived while implementation was underway. A coding tool could have completed its original assignment correctly and still delivered an outdated answer. We had to reconcile the brief and the proposed change against the newly approved source before treating the work as complete.

Release verification introduced another complication. An automated check failed, prompting investigation of what the deployed system was actually serving. The investigation pointed to a propagation window in which the check encountered the previous version. Checking the live behavior helped establish whether the release or the verification needed attention.

The founder then evaluated the revised experience. The sequence connected a business observation to a decision, implementation, technical verification, and business confirmation. It also exposed the coordination work required to keep that sequence trustworthy.

How the work moved

The operating loop joined business decisions to reviewable changes and then returned evidence to the people accountable for the outcome. AI assisted at several steps, with a defined assignment and a point at which human judgment was required.

Decide and record. Meetings, demonstrations, and messages resolved business questions. The decision record captured the rule and its context. To keep that record authoritative as work changes, each decision should identify its owner, approval status, and any earlier decision it supersedes.

Define the work. A work package translated intent into an outcome, scope, affected components, acceptance examples, and exclusions. The brief gave the implementation tool enough context to work while exposing questions that still needed a human answer.

Build and review. AI implementation tools produced changes and checks for review. AI also helped examine proposed work against the brief and source material. Generated lists and mappings could be compared systematically, while behavior tests addressed whether the system acted as intended.

Authorize and release. Merging code and deploying it were separate decisions. Human authorization remained part of the release process, with staging and production checks. A proposed implementation did not acquire release authority simply because an AI tool considered it complete.

Verify and learn. We inspected live endpoints and public-facing behavior, then asked the founder to evaluate the result. New understanding returned to the decision record. A later requirements trace checked whether accepted decisions had entered the plan and reached an explicit status.

Accountability across the loop

Participant	Responsibility
Business owner	Defines business intent, resolves product questions, and evaluates whether the delivered behavior serves the business.
Technology leader	Owns architecture and technical judgment, evaluates review evidence, and authorizes integration and release.
AI tools	Assist with planning, implementation, review, investigation, verification, and documentation within assigned boundaries.

The distinction matters when something is wrong. An AI review can identify a problem or suggest a correction. It does not transfer accountability for accepting the change. The technology leader must decide what evidence is sufficient and when additional review is warranted.

Release boundaries also need more than a prompt. Instructions tell an AI tool how to work; permissions and release controls limit what it can actually do. Teams adopting this approach should make those boundaries enforceable and preserve a record of authorized changes.

What the failures revealed

The useful lessons came from the points where information, implementation, and business meaning fell out of alignment. AI helped investigate and correct issues. It did not prevent them from arising.

A shared document can still contain the wrong assumption

Implementation exposed a uniqueness constraint problem in a brief that had also been used for review. The specification itself needed correction. This is a reason to test the intended data behavior and allow implementation findings to challenge the brief. A second pass through the same assumptions, even by another AI tool, can preserve the same mistake.

A local change can be incomplete across the business flow

A connected system retained an older category definition in a dropdown. Updating routing logic and a public interface did not automatically update every downstream field. The gap reinforced the need to follow a change from the customer's selection through storage, mapping, and routing. Each boundary had its own definition and owner.

A related identity issue showed why terminology matters. An identifier represented a person in one system and an assignment in another. Matching text did not mean matching concepts. We documented the distinction rather than treating a cosmetic alignment as a safe correction.

Documented agreement does not guarantee planned work

A deliberate requirements trace found accepted items that had not reached the work plan. The document hub preserved information, but it did not guarantee that every decision became an assignment. Reconciliation needed to be an explicit activity: connect each accepted requirement to planned work, delivered behavior, or a documented deferral.

Parallel activity still requires isolation

Concurrent AI-assisted work in a shared working copy allowed unfinished changes to cross into another branch. Separating working directories and assigning clear branch ownership addressed the immediate coordination problem. The general lesson is to make concurrent work independently reviewable before integration.

The leadership implication

These failures changed what deserved attention. Keeping context current, testing assumptions, checking system boundaries, and reconciling scope were essential parts of delivery. They were easy to underweight when the visible implementation was moving quickly.

The same applies to review capacity. If a team generates changes faster than it can evaluate them, unfinished judgment accumulates even when coding appears complete. Work should remain bounded by the team's ability to verify and accept it. The cost of finding a mistake also matters: an error caught in a brief has different consequences from one discovered after release.

What carries into larger organizations

This engagement involved a small team, experienced technical leadership, direct access to the business decision-maker, and a bounded product scope. Those conditions contributed to the delivery loop. The case does not isolate a measured AI productivity effect or establish a percentage reduction in cost, effort, or elapsed time.

Larger organizations retain dependencies that a small team may not encounter. Shared platforms, multiple business owners, established data contracts, and operational obligations continue to require coordination. Adding AI does not remove those relationships. It can make the quality of the information crossing them more consequential.

Practices that transfer across team sizes

Practice	Why it remains useful
Approved decisions with clear ownership	Gives people and tools a common basis for resolving scope and interpretation.
Bounded work with acceptance examples	Makes the assignment and the evidence of completion explicit.
Review across system boundaries	Checks the whole business flow rather than relying on one component's tests.
Controlled release and live verification	Connects authorization to deployed behavior and an observable result.
Requirements reconciliation	Shows which commitments are delivered, deferred, or still without planned work.

Scale changes the supporting structure

As the number of participants grows, a decision record needs more explicit ownership, discoverability, and change notification. More concurrent implementation requires disciplined integration. Higher-consequence releases call for stronger evidence and controls proportionate to the environment. A small team's reliance on immediate access to one senior reviewer can become a bottleneck or continuity risk.

SAFe or another established framework can provide part of that structure. The practical move is to examine the work inside it: where decisions wait, where context is repeatedly translated, where dependencies are real, and where teams already have enough authority and evidence to proceed.

Useful meetings remain useful. A demonstration that changes the business owner's understanding can be one of the highest-value events in the delivery process. Meeting count alone is a weak measure of improvement. The more relevant question is whether the interaction resolves uncertainty, enables a decision, or improves the result.

That is also the opportunity for leadership. AI can expand the work a team can attempt. Leaders determine whether that capacity is directed toward the right problem and whether the organization can absorb and operate what the team delivers.

How leaders should evaluate the approach

Start with a bounded workflow whose owner can define the intended behavior and evaluate the result. Establish how the work is delivered today, including discovery, meetings, implementation, review, rework, and release. Then assign AI a specific role and measure the entire delivery loop.

Measure	What to include
Decision to verified behavior	Elapsed time from an approved change to technical verification and business acceptance; record waiting time separately.
Total effort and cost	Human time across the full engagement, AI costs, rework, review, and operational support.
Quality and completeness	First-pass acceptance, defects after release, reopened decisions, and accepted requirements without planned work.
Business effect	Whether the changed workflow improves the intended customer or operating outcome after it is used.

Code volume, test count, and completed prompts describe activity. Business value depends on what the delivered capability does and what it costs to achieve and maintain that result. A fast build can

still leave the organization with incomplete requirements, an unsupported process, or a larger review burden.

Use the findings to adjust the operating model. If decisions are the bottleneck, clarify ownership. If review is the bottleneck, reduce concurrent work or improve the evidence. If downstream mismatches recur, strengthen interface ownership and acceptance checks. Expand the approach when the complete loop supports it.

The executive decision

In this engagement, AI helped us carry business decisions into working software. The conversations and demonstrations supplied context. Written decisions and work packages carried it forward. Review, controlled release, and business testing established whether the result was acceptable.

The opportunity is to organize delivery around that complete loop. Faster implementation raises the value of a clear decision, and it raises the cost of allowing an unclear one to move unchecked.

For leaders evaluating AI in technology delivery, the starting point is a practical question: where does the path from business intent to verified behavior break down today?

Start the conversation

References

[1] SAFe PI Planning

[2] SAFe Release on Demand

[3] Principles behind the Agile Manifesto