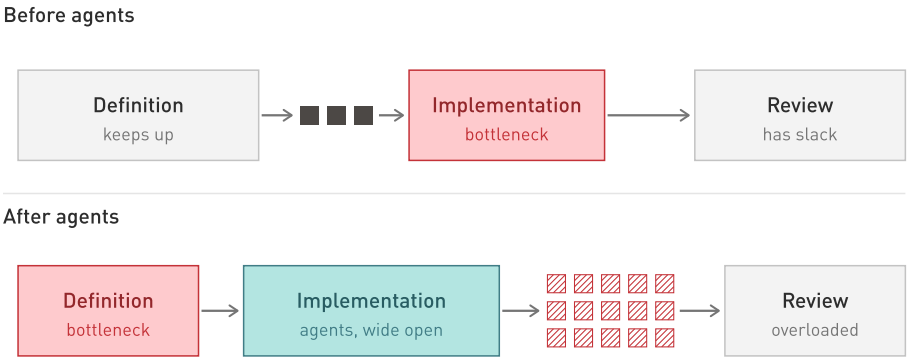
Agentic Engineering for Teams

The operating model at a glance

Coding agents really can make one developer much faster at producing code. But serious software gets built by teams, and team working patterns haven't caught up with the tools.

When implementation took longer, developers resolved many missing details while writing the code. Agents have made writing code the fast part. Now vague direction can turn into a week's worth of bad code, implemented in one afternoon. And the answers that used to live in one developer's head never reach the teammates and agents building in parallel.

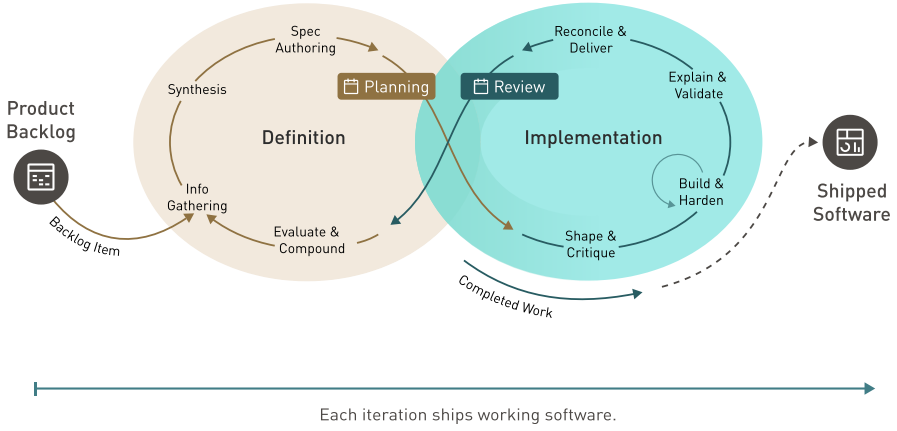
The bottleneck has moved from writing code to deciding what to build, so agents either wait on decisions or run ahead of them. Atomic teams have shipped more than 300 products over 25 years. This is the operating model we recommend for serious agent-assisted software work. It keeps agile's short feedback loops, with working software as the measure of progress. It adds explicit Definition, tighter team checkpoints, and feedback systems that let agents verify more of their own work.



The bottleneck has shifted.

What it looks like in practice

Atomic's default cadence is two iterations a week, each opened by Planning and closed by Review. Planning and Review are working sessions: the team reviews what is ready, chooses what comes next, demonstrates what landed, and probes risks. Between sessions, some people define upcoming features while others build the work defined last iteration. The exact schedule can change by project; the short loop, explicit Definition, and whole-team checkpoints remain.



Each iteration ships working software.

Teams have always defined work. What is new is making Definition an explicit phase that runs ahead of Implementation. An agent told to build will settle open questions unless the team reserves them for people. Those decisions live in specifications and the Knowledge Base, the durable team context kept in the repository, where the whole team and its agents can act on them.

01 Planning

The whole team selects ready work, scopes the iteration, and names the risks that need human attention before agents produce code.

02 Definition

Part of the team turns product owner decisions, stakeholder input, research, design exploration, and technical spikes into specifications: written, testable direction an agent can build from without guessing.

03 Implementation

Developers direct agents through the defined work, keep the written record current, run checks, and decide what is safe to merge.

04 Review

The whole team demos and probes what landed, tests the integrated product, and turns findings into fixes and follow-up work.

How the team keeps ownership

Planning and Review keep unfamiliar work in batches the team can understand and expose unclear direction within days. They are also where the product owner and delivery team decide what to build, what counts as good enough, and how the pieces fit together.

On an Atomic engagement, the client-side product owner joins Planning and Review and owns product direction, priority, and acceptance. Atomic's delivery team owns execution quality: the product manager coordinates stakeholders, the designer leads research and design, the tech lead drives architecture, and developers own Implementation and merge decisions. Assignments can move with the bottleneck: a developer can move into Definition, while strong checks may let a designer or product manager carry a small fix through Implementation.

When a correction or manual check repeats, the team turns it into a test, agent instruction, example, or tool. The next agent starts with the answer.

What this means for your projects

The aim is to deliver more useful software within the same budget. Teams spend more time defining work and building checks so that faster implementation does not create an equally large review and repair queue. Legacy systems may need groundwork before the speed-up appears. Judge the gain over several weeks by what reaches users and how much rework it needs.

Teams also vary how much they resolve before building. Hard-to-reverse data models and contracts deserve more Definition than a screen the team can safely build, show, and revise.

The full guide covers phase playbooks, a worked example, and failure modes. Read it free at atomicobject.com/agentic-engineering. If you want to apply the process on a real project, Atomic can work alongside your team to deliver the product.