ThoughtFox

THOUGHT LEADERSHIP

The Whirlpool SDLC How Software Teams Plan in the Agentic Era

Why going back to waterfall is the wrong frame and what the right one looks like

February 2026

The Paradox of Agentic Development

Something unexpected is happening in organisations furthest along in adopting AI agents for software development. After two decades of moving away from upfront planning, teams are rediscovering that planning matters. A lot.

Engineering leaders who built their careers on agile principles are watching their most productive teams spend more time on specification, decomposition, and acceptance criteria than they have in years. It feels like regression. It feels like waterfall.

It isn't. But it takes a new mental model to understand why.

AI agents can generate code at extraordinary speed. They can scaffold entire features, write tests, refactor modules, and resolve bugs in minutes rather than hours. But they have a critical limitation: they execute exactly what you specify. Not what you meant. Not what you intended. What you specified.

In the human-led SDLC, ambiguity was absorbed by developer judgment. A ticket that said 'add filtering to the reports page' would be interpreted by a developer who understood the domain, the existing patterns, the team's conventions, and the product manager's unstated preferences. The developer would fill in the gaps, make reasonable assumptions, and ask questions when something felt off.

Agents don't do this. They don't fill gaps with judgment. They fill gaps with plausible-looking code that may or may not reflect what you actually needed. At speed, a poorly specified task doesn't just produce a wrong result. It produces an elaborate, confidently constructed wrong result that takes longer to review and fix than it would have taken to build correctly in the first place.

> **The teams seeing the biggest gains from agentic development aren't the ones with the best AI tools. They're the ones with the best specifications.**

This is the paradox: to move fast with agents, you must slow down on planning. But the way you slow down matters enormously.

This Is Not Waterfall

> The obvious comparison is to waterfall. Big upfront design. Comprehensive specifications. Detailed requirements documents. It's tempting to conclude that the agentic era is simply validating what waterfall advocates said all along.

But this reading misses what made waterfall fail. Waterfall didn't fail because planning is bad. It failed because of three specific properties of how it approached planning.

Linearity

Waterfall planning was a one-pass, sequential process. You planned, then you built, then you tested. Learning from implementation didn't flow back into the plan because the plan was considered complete.

Comprehensiveness

Waterfall attempted to specify everything before building anything. This meant the planning phase was slow, expensive, and generated enormous documentation that was outdated before the first line of code was written.

Rigidity

Once the plan was set, deviating from it was treated as a failure of process rather than an adaptive response to new information.

What agentic development demands is none of these things. It demands planning that is iterative, focused, and adaptive. It demands a fundamentally different shape.

Enter the Whirlpool

> The right mental model is not a waterfall. It's not even a spiral, though Boehm's spiral model comes closer. The right metaphor is a whirlpool.

Convergence, Not Sequence

A whirlpool pulls everything toward a centre. Each orbit is tighter than the last. This is exactly how agentic planning works. You start with a broad understanding of the problem, and each iteration narrows the specification. First pass: what are we solving and why? Second pass: what are the constraints, edge cases, and boundaries? Third pass: what does 'done' look like in terms an agent can execute against? You're not moving through sequential phases. You're converging on precision.

Multiple Passes, Increasing Depth

Water in a whirlpool crosses the same space repeatedly, but at a different depth each time. The same is true of agentic planning. You revisit the same problem statement, the same feature, the same requirements, but each cycle adds specificity. Waterfall is wide and flat: cover everything at the same shallow depth. The whirlpool is narrow and deep: cover less ground with much greater precision on the things that matter.

Slow at the Edges, Fast at the Centre

The outer rings of a whirlpool move slowly. The inner rings move at extraordinary speed. This mirrors the agentic workflow perfectly. The early planning iterations are slow, deliberate, and intensely human. But as the specification tightens, the agent can execute with remarkable velocity.

> **The teams that struggle with agentic development try to skip the whirlpool: throw a one-line ticket at an agent and expect magic. The ones who succeed have learned to orbit the problem three or four times before letting the agent dive.**

The Depth Dimension

> This is where the whirlpool metaphor distinguishes itself from the spiral. A spiral is flat, a two-dimensional loop. A whirlpool has depth. You're not just going around. You're going down: toward implementation, toward the concrete, toward the executable.

Each orbit isn't a repetition at the same altitude. It's a descent into greater precision. And this matters because the entire point of the planning process is to reach a level of specificity where the agent can execute without ambiguity.

A whirlpool also pulls surrounding material into its vortex. Good agentic planning does the same. As you iterate through the planning cycles, you naturally draw in edge cases, dependencies, acceptance criteria, architectural constraints, and domain knowledge that would otherwise have been discovered during development.

In the old model, these discoveries happened when a developer hit a question they couldn't answer, raised a hand, and waited for a product manager to respond. In the whirlpool model, they're surfaced and absorbed during the planning process itself, before the agent ever writes a line of code.

The Great Inversion

The Agile Manifesto famously prioritised 'responding to change over following a plan.' In the agentic era, this dichotomy dissolves.

The whirlpool model responds to change through planning. The planning cycle itself is the adaptive mechanism. Each orbit of the whirlpool is a response to what you learned in the previous orbit. You're not choosing between agility and planning. The planning process itself is agile.

For twenty years, planning and agility were treated as opposing forces on a spectrum. More planning meant less agility. More agility meant less planning. The agentic era reveals this was never actually true: it was an artefact of the human bottleneck.

When agents are the execution layer, the economics invert. Replanning is cheap because re-execution is cheap. If your third orbit reveals the approach is wrong, you adjust the specification and the agent rebuilds in minutes, not weeks.

> **In the agentic era, planning doesn't constrain agility. Planning is agility.**

What This Means in Practice

If the whirlpool model is the right frame, several practical implications follow for how engineering teams organise and operate.

Tickets Become Specifications

The two-sentence user story that relies on developer judgment is an anti-pattern in the agentic world. Not because brevity is bad, but because implied context is invisible to agents. Teams adopting agentic development find themselves writing tickets that look more like mini-specifications: explicit acceptance criteria, stated constraints, identified edge cases, and clear definitions of done.

This isn't overhead. It's the outer orbits of the whirlpool. The time spent writing a precise ticket is returned many times over when the agent executes cleanly on the first attempt rather than producing something that requires three rounds of human correction.

The Planner Role Elevates

In many agile teams, detailed planning was seen as low-status work. The real engineering happened in the code. In the whirlpool SDLC, planning is the highest-leverage activity. The person who can take an ambiguous business requirement and decompose it into an agent-executable specification is the most valuable member of the team.

This doesn't mean dedicated 'planners.' It means senior engineers spending more time on specification and review, and less on implementation. The skill set shifts from 'can you build this?' to 'can you specify this precisely enough that an agent can build it?'

Review Becomes the Bottleneck

In the waterfall model, the bottleneck was analysis. In agile, it was development throughput. In the whirlpool SDLC, the bottleneck moves to review. Agents can generate code faster than humans can evaluate it. Organisations that don't explicitly design for this end up drowning in pull requests, with review queues growing faster than they can be processed.

Feedback Loops Tighten

The traditional agile feedback loop runs on a sprint cadence, typically two weeks. The whirlpool model operates on a much tighter cycle. Each orbit might take minutes or hours, not days or weeks. Stand-ups, sprint reviews, and retrospectives were designed for a two-week cadence. The whirlpool demands something more continuous.

The SDLC Shift at a Glance

SDLC Model	Planning Style	Bottleneck	Feedback Cycle
Waterfall	Comprehensive upfront	Analysis & requirements	Months
Agile / Scrum	Just-in-time	Development throughput	2 weeks (sprint)
Whirlpool (Agentic)	Iterative convergence	Review & verification	Hours to days

Starting the Whirlpool

> If this resonates, the question is how to begin. The answer is deceptively simple: start paying attention to your specifications.

Take a ticket from your backlog. Before handing it to an agent or even a developer, ask: 'If I gave this to someone with zero domain context and no ability to ask clarifying questions, would they build the right thing?' If the answer is no, you've identified a gap that the whirlpool is designed to close.

Then iterate. Write the specification. Have the agent attempt it. Observe where it deviates from what you intended. Refine the specification. Repeat. Each cycle will teach you something about where your implicit assumptions live and how to make them explicit.

This is the whirlpool forming. It feels slow at first, the outer orbits always do. But the speed at the centre, when the agent executes cleanly against a tight specification, is unlike anything the industry has experienced before.

> **The agentic era is not a return to waterfall. It is not the end of agility. It is the discovery that planning and agility were never opposites: they were partners waiting for the right conditions to emerge.**

The whirlpool is that emergence. The organisations that learn to plan this way will build faster, with higher quality, and with greater confidence than those still throwing vague tickets into the void and hoping the AI figures it out.

About ThoughtFox

We help organisations make AI work for their business, not just exist in it. We give your teams the understanding, the capability, and where it matters, the working systems to turn strategy into results.

We believe AI adoption isn't a technology problem; it's a people opportunity. That's why we focus on building internal capability rather than dependency. Your teams end up owning the knowledge and the systems, long after our engagement ends.

Ready to discuss your AI transformation?

thoughtfox.ai

This article draws on ThoughtFox's real-world experience with AI transformation in software engineering. The frameworks described are based on observed patterns across multiple agentic development engagements.