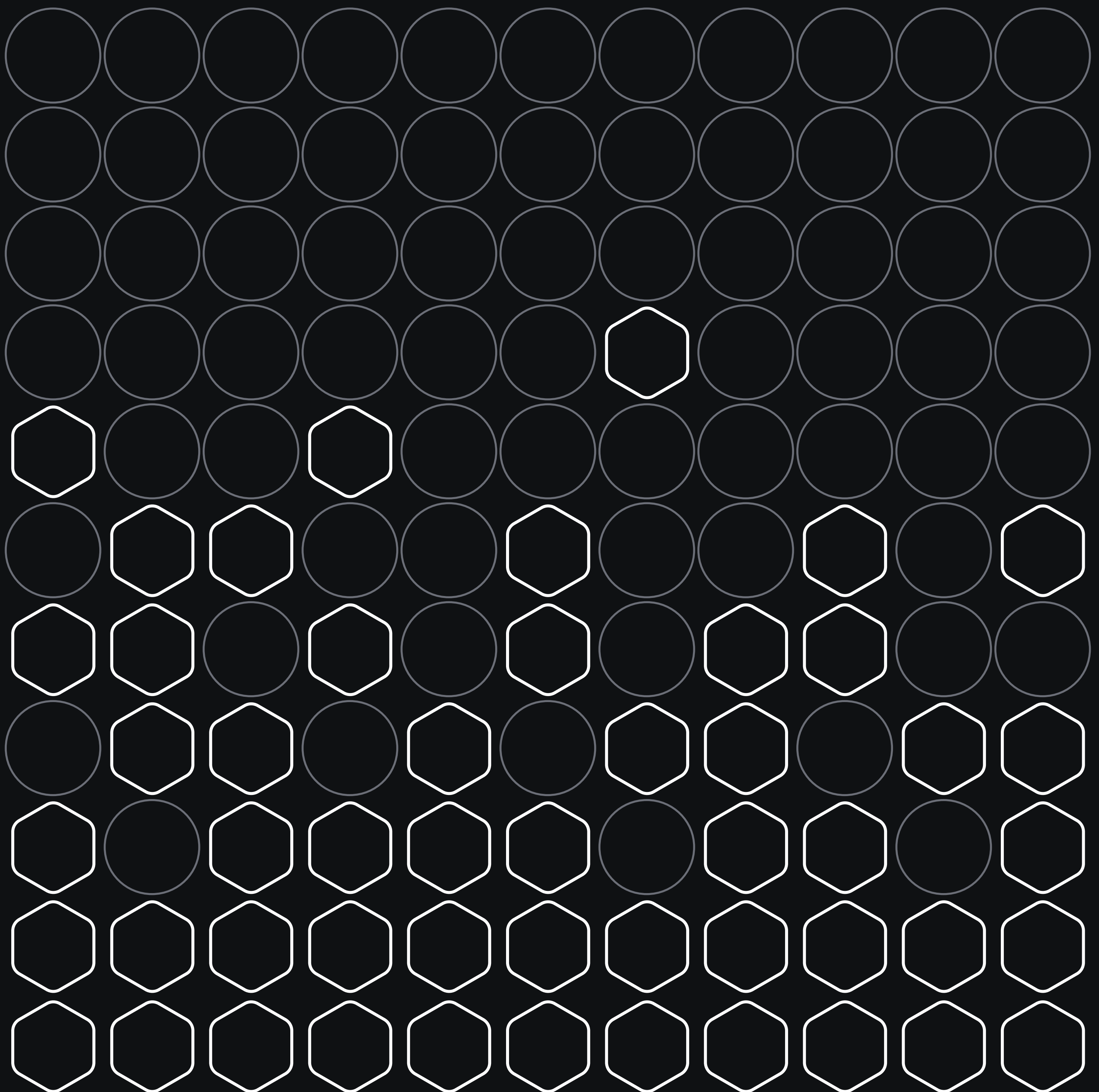


The AI SDLC transformation playbook



Introduction

Teams across the industry are sprinting to reinvent their software development processes with agentic AI. Yet, navigating the fast-changing landscape of tools and opinions can be daunting for even the most AI-forward organizations.

This guide shares Atlassian’s approach and best practices, informed by our own internal AI SDLC transformation and collaborative learnings from top engineering organizations. As our approach continues to evolve, the foundational elements we describe here are the ones we’ve found consistently mattered.

Table of contents

01	Making the shift to AI-native SDLC	2
02	Traditional vs. AI-native SDLC workflows	4
03	Platform investments	6
	CONTEXT GRAPH MEASUREMENT	
04	Workflow shifts	9
	PLANNING DESIGN DEVELOPMENT REVIEW MAINTENANCE	
05	Navigating change	14



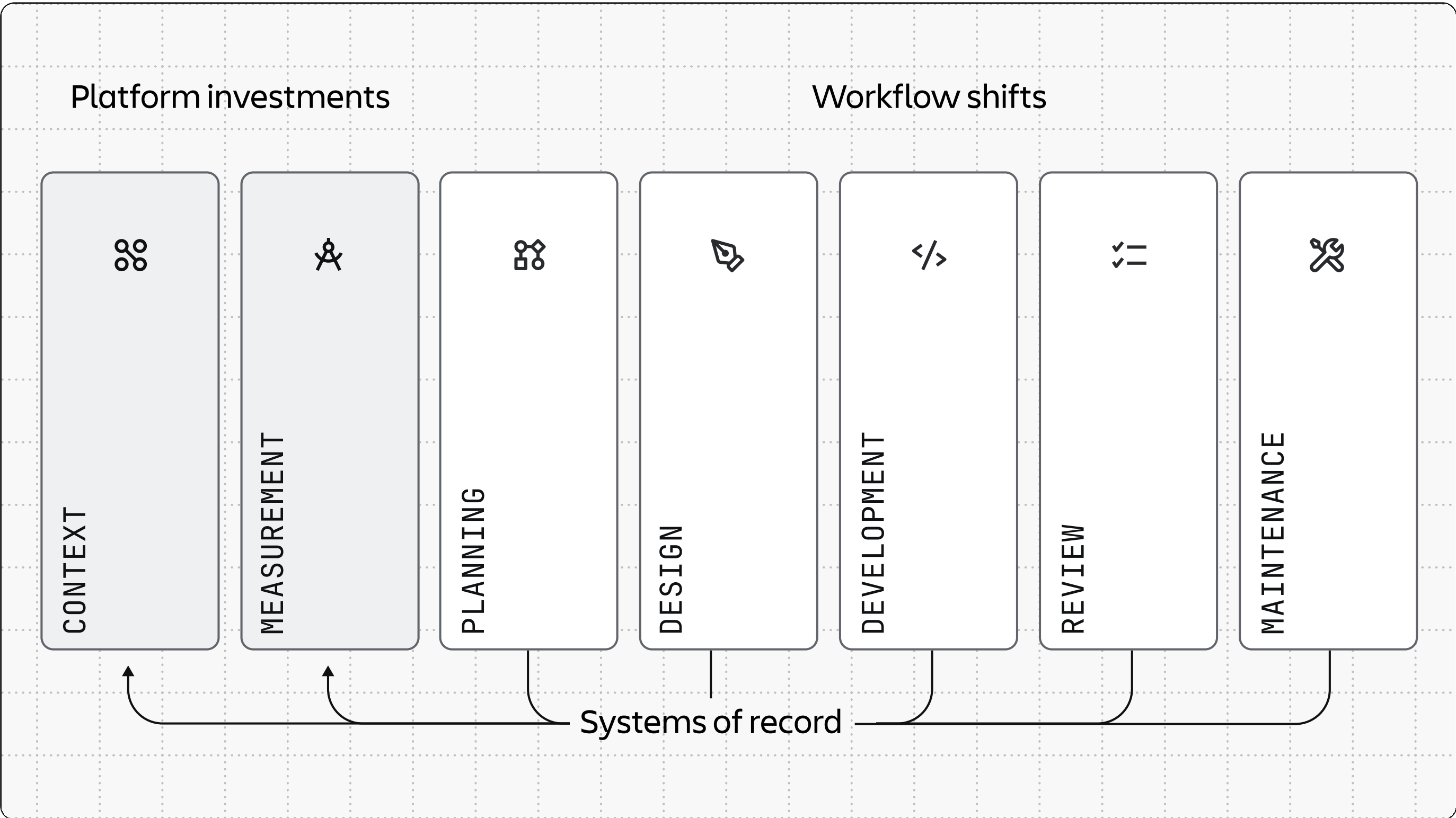
01

Making the shift to AI-native SDLC

The AI-native software development lifecycle (SDLC) goes far beyond having developers adopt coding agents. As less time is spent writing code by hand, coding itself is evolving from a synchronous, single-threaded task into an asynchronous, multi-threaded one. This exposes the processes and steps before and after code (planning, design, review, and maintenance) as the largest bottlenecks and opportunities. Furthermore, team collaboration becomes an even greater bottleneck as the rate of work accelerates.

To successfully transform the entire SDLC, organizations must simultaneously invest in key platform foundations—context and measurement—while also shifting from traditional SDLC processes to AI-native ways of working. Together, these investments and workflow shifts become the modern software factory: faster, more autonomous, more event-driven, and reshaped around critical human judgment.

THE 7 PILLARS OF OF AI SDLC TRANSFORMATION



Platform investments and new SDLC workflows are mutually dependent: new SDLC workflows only work effectively when supported by powerful enterprise context and feedback loops. Inversely, reliable context and measurement are only possible when metadata from new SDLC workflows is consistently captured and accessible. As part of that loop, developer–agent interactions themselves become a new form of context, making the whole workflow smarter over time.

Systems of record tie these pillars together, wherever development happens. They support humans in applying the critical judgment that moves software forward: planning, evolving designs, aligning on goals, acting on customer feedback, and responsibly shipping code. In turn, agents rely on those same systems for the structured, connected data that lets people and AI work effectively together.

02

Traditional vs. AI-native SDLC workflows

The software development lifecycle (SDLC) moves software from idea to production through planning, design, engineering, testing, deployment, and maintenance. Agile and DevOps reshaped how teams work together in the traditional SDLC by tightening feedback loops, breaking down silos, and letting teams own the code they ship and run. Engineering organizations have come a long way, but haven't eliminated the delays that come with every hand-off.

An AI-native SDLC is the next evolution. It builds on the same vision and collapses remaining hand-offs into continuous, AI-supported loops where product, design, and engineering co-create in real time, and where agents help humans do the work at a new pace and scale.

This playbook describes the key SDLC workflow shifts summarized on the following page. Today, most organizations sit somewhere between the traditional and AI-native models. To realize the full benefits of an AI-native SDLC at enterprise scale, it's essential that work activity and knowledge are captured in systems of record in order to provide AI with continuous context and consistent measurement abilities.

Workflow	Traditional SDLC	AI-native SDLC	Atlassian approach
Planning	Requirements are defined manually with incomplete knowledge of the code, causing downstream problems with significant time spent aligning teams.	AI synthesizes customer needs and code context into requirements, captured as versioned spec files in the repo.	Teamwork Graph pulls in relevant context from across the org (Jira, Confluence, third-party sources), so plans reflect what other teams are doing. Jira Product Discovery evaluates ideas against customer feedback; Jira turns them into connected, agent-ready tasks.
Design	Static mockups and manually written briefs are handed off to developers for every change.	Agent sessions are guided by standards that are stored alongside the code. Live prototypes replace static frames; designers can contribute UI code and bug fixes directly.	Design standards are grounded in the Teamwork Graph, so changes fit existing systems, prior decisions, and cross-team dependencies. Agents in Jira let designers and PMs ship UI updates and bug fixes directly, and Loom video feedback drives AI revisions with real customer insight.
Development	Developers pick up tasks, research and plan implementation, then start writing code. Velocity and quality are bottlenecked by individual throughput and skill.	Developers work with agents for planning, generating, and testing code. Quality depends on AI context and velocity relies on effective agent orchestration.	Coding agents use the Teamwork Graph CLI to combine code and organizational context. Each session's decisions flow back to the graph, so knowledge compounds across teams instead of being isolated in each repo.
Review	Humans manually review every change, standards are inconsistent, and feedback is slow.	AI reviews changes first, applying shared standards across new and existing code.	Atlassian's AI Reviewer and third-party coding agents validate changes across teams and services against dependencies mapped in the Teamwork Graph. Standards are centrally defined and governance is auditable in the system of record.
Maintenance	Incidents and errors are manually triaged and resolved; KTLO and reliability work often gets deprioritized due to limited engineering capacity.	Agents help triage, resolve, and document routine work, escalating only when needed. Teams shift more capacity toward new capabilities.	Jira Service Management automates incident triage. Agents trace impact across teams via the graph and feed related work items back into planning. Governed agent loops in Jira continuously scan for well-defined work and turn backlog items into pull requests.

03

Platform investments

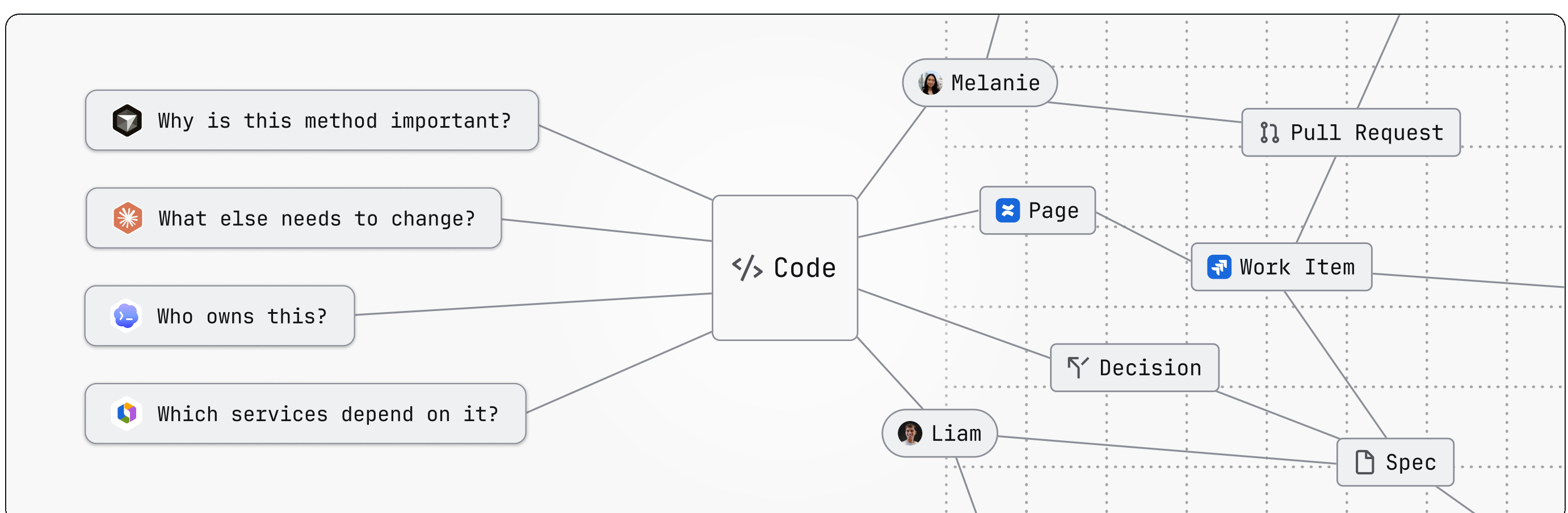
CONTEXT GRAPH

WHY IT MATTERS

AI agents may have the intelligence to write, refactor, and review code, but they face the same challenge development teams always have: making sense of the context surrounding each task. Humans continuously gather that context, whether its understanding of code, product strategy, architectural decisions, standards, and constraints, accumulated over years of working in the same system. Agents need the same context supplied to them for every task in order to produce high-quality output.

WHAT GOOD LOOKS LIKE

To provide effective context to agents, organizations need a context graph containing interconnected data spanning code repositories, work items, documentation, technical standards, meeting recordings, and chat. This data must then be made accessible to both humans and agents through a permissions-aware layer, so that AI can act as a knowledgeable teammate rather than guessing at the shape of the system.



ATLASSIAN APPROACH

To tackle this problem, we've developed a context engine called the Teamwork Graph. It combines data from 50+ connected sources with Jira, Confluence, and Loom: a living map of work, code, people, decisions, and dependencies that helps agents understand not just the task, but the system around it. Governance and permissions are enforced at the data layer, so what a person can see is exactly what an agent can act on. The Teamwork Graph is reachable from wherever teams work: Atlassian apps, terminals, IDEs, or any agent surface via MCP or CLI. As agents and humans work in these surfaces, their activity flows back into the graph, so the system of record maintains itself autonomously and context compounds every time the graph is used.

RESEARCH AT A GLANCE

In internal benchmarks, agents with access to the Teamwork Graph delivered 44% more accurate results while using 48% fewer tokens compared to agents operating without it. Because every agent run generates more structured context, each subsequent run becomes higher quality and lower cost. Learn more about Teamwork Graph at teamworkgraph.com.

MEASUREMENT

WHY IT MATTERS

Amidst such rapid change to how teams work, it's difficult to know what's working and what's not without data. With the right feedback loops and metrics, organizations can confidently track the impact of AI tools and workflows alongside overall cost, productivity, and ROI while enabling teams to continuously improve their own processes.

WHAT GOOD LOOKS LIKE

To produce actionable insights, work and agent activity must be observable and measurable. By integrating data from systems of record, third-party APIs, and agent sessions, organizations can build real-time visibility into AI effectiveness, impact, and ROI, and pair it with qualitative signals from the developers doing the work.

ATLASSIAN APPROACH

At Atlassian, we use DX (getdx.com) to measure and report on how AI is impacting our SDLC and overall productivity. DX combines quantitative data from tools like Jira, Cursor, and Claude Code with qualitative developer feedback. Analyzing and benchmarking this dataset helps us set targets, guide investment, and ensure organizational excellence.



"When an agent starts in the wrong place, everything slows down and costs more. Developers end up explaining where to look, why the code works the way it does, and what else it touches. Teamwork Graph puts all of that in front of the agent from the start, so it can focus on getting the work done."

SPOTON, MARK WALZ, CHIEF TECHNOLOGY OFFICER

04

Workflow shifts

PLANNING

THE FRICTION

In the traditional SDLC, planning relies on manual effort and committee-driven consensus. Product managers and analysts spend weeks drafting requirements and aligning stakeholders. Because those defining the work often have incomplete knowledge of the underlying codebase, specifications frequently miss architectural constraints and technical debt, resulting in friction and costly rework during implementation.

THE SHIFT

An AI-native approach transforms this phase by using agents to synthesize customer needs and organizational context into requirements and specs, drafting from a real understanding of both the business and the codebase. Instead of human teams writing specifications from scratch, they focus on applying judgement: refining, deciding, and ensuring plans are grounded in reality before engineering work begins.

ATLASSIAN APPROACH

Atlassian operationalizes this vision through the Teamwork Graph, which brings together people, goals, code, and knowledge across Atlassian and connected third-party apps, making it available to any agent. Building on this rich context, teams evaluate ideas against customer feedback in Jira Product Discovery, then use Jira to co-create specs, PRDs, and technical blueprints. Once finalized, these plans can be instantly converted into actionable, tracked work items.

DESIGN

THE FRICTION

In the traditional SDLC, design is an isolated phase between requirements gathering and implementation. Designers spend significant time crafting static mockups and manually written briefs, then hand them off to engineering. Because designers lack direct access to the underlying codebase, even minor visual tweaks or layout adjustments require developer intervention, creating recurring bottlenecks and dragging out release timelines.

THE SHIFT

An AI-native SDLC reimagines design as a continuous thread integrated with planning and development, rather than a distinct step. AI-assisted tooling empowers designers to build live, interactive prototypes, generate UI code and commit bug fixes independently, bridging the gap between interface concept and production without relying on developers for every layout change.

ATLASSIAN APPROACH

Atlassian brings design artifacts into the Teamwork Graph, connecting design intent to code and work in flight. Designers, PMs, and developers can generate rapid prototypes and design updates directly from shared context, feeding them into code repositories and treating the resulting code as a design spec. To keep collaboration fluid, teams use Loom to capture feedback, where AI ingests the transcripts and summaries as prompts, making revisions a part of the handshake between crafts.

DEVELOPMENT

THE FRICTION

In the traditional SDLC, work is scoped and logged in backlogs before any code is written. Code quality is constrained by individual developer skill and domain expertise, and velocity is bounded by human throughput, the physical limit of how quickly engineers can write, test, and refactor code.

THE SHIFT

An AI-native SDLC turns development into a fluid, automated process. Tasks are spawned from operational signals and captured automatically as work happens. In this model, code quality is no longer determined by raw developer skill, it depends on the quality of context supplied to AI. And velocity depends on effective agent orchestration: coordinating multiple AI agents working alongside engineers.

ATLASSIAN APPROACH

Atlassian executes this approach by seamlessly embedding work tracking and rich context into the developer's environment. Engineers connect their coding agents to the Teamwork Graph, which gives agents built-in skills and tools to efficiently search and understand the objects in it. AI models then receive more relevant context and generate higher-quality output with fewer tokens spent. Those same developer-agent interactions also feed back into the system of record, creating a compounding context effect that makes the system smarter over time.

RESEARCH AT A GLANCE

AI-native development processes at Atlassian have cut issue cycle time by nearly half, from 11.5 days to 6.7 days. The gain comes from tighter feedback loops, faster reviews, and agents handling routine work in parallel.

 Outreach

"It isn't just about writing code faster. It's about improving the entire software development lifecycle, shortening cycles, raising quality, and giving us real leverage from AI."

OUTREACH, ABHI ABHISHEK, SVP GLOBAL ENGINEERING

REVIEW

THE FRICTION

In the traditional SDLC, manual reviews create persistent bottlenecks and slow feedback cycles: pull requests often sit unattended for hours or days, and standards vary in scrutiny from reviewer to reviewer. Code is validated for correctness, but the intent and requirements aren't validated until it reaches staging or production.

THE SHIFT

An AI-native SDLC puts agents on the critical path for review, so pull requests move within minutes instead of days. Engineering standards are encoded into shared context, enabling AI to continuously apply them across new and existing code. Human reviewers still hold the judgment call, but they arrive better informed, with routine feedback already handled.

ATLASSIAN APPROACH

Atlassian powers context-aware reviews by consolidating the intent, requirements, and discussions behind every change, so reviewers have end-to-end traceability from PR back to the context that shaped it. Platform leads centrally define, enforce, and monitor engineering standards across the org, and Atlassian's AI Reviewer works alongside third-party tools like Cursor to catch gaps between what was asked and what was built.

MAINTENANCE

THE FRICTION

In the traditional SDLC, maintenance is reactive and labor-intensive. Engineers manually triage production errors, analyze stack traces, and route incident tickets across teams for resolution. Because engineering capacity is finite, KTLO work and reliability improvements are routinely deprioritized in favor of new features, leading to accumulated technical debt and operational risk.

THE SHIFT

An AI-native SDLC turns maintenance into a proactive, automated operating model. Agents classify incoming errors and generate root cause analyses before humans engage, escalating only when needed. Routine maintenance, dependency updates, and recurring fixes are increasingly delegated to autonomous coding agents, preventing backlog buildup and freeing engineers for higher-value strategic work.

ATLASSIAN APPROACH

Atlassian combines intelligent incident management with proactive code maintenance. Jira Service Management streamlines triage and resolution, while governed agent loops continuously scan the backlog and codebase to complete well-defined KTLO work and augment engineering capacity. Because every loop writes its actions back into the graph as it runs, the system of record captures a live audit trail of what agents did and why, feeding it with compounding context.

RESEARCH AT A GLANCE

At Atlassian, agent loops in Jira automatically resolve 51% of potential vulnerabilities, freeing engineers to focus on the higher-value tasks that require human judgment.

05

Navigating change

AI-native SDLC is an operating-model transformation. It changes how work is scoped, how quality is achieved, how decisions get made, and how teams learn from every release. Leaders who treat it primarily as a procurement exercise, selecting agents, licensing models, or adopting a new IDE, will see incremental gains at best. The organizations pulling ahead are those building a strong foundation for governed agent execution, rooted in connected context and continuous measurement.

To move deliberately, we recommend that engineering leaders focus on five moves:

- 1 **Anchor on context.** Treat connected, permissioned organizational context as the foundation everything else rests on. Without it, every workflow shift underperforms.

- 2 **Invest in measurement early.** Establish quantitative and qualitative feedback loops before scaling adoption, so you can distinguish real productivity gains from hype.

- 3 **Choose a starting workflow.** Pick the SDLC stage where AI-native ways of working will unlock the most value for your teams and go deep before broadening.

- 4 **Preserve accountability.** Keep humans clearly in charge of judgment, quality, and governance while AI handles execution and enforcement. Governed agent loops (scoped, standards-aware, and captured in systems of record) are the ones that scale.

- 5 **Build the operating model, not just the toolkit.** Update rituals, roles, and success metrics alongside the tooling; the biggest gains come from the changes AI enables in how teams work together.

The AI-native SDLC will keep evolving faster than any playbook can capture. But the organizations that make these foundational investments now will not only realize the benefits of AI sooner, they will build delivery muscles that compound over time. That is the real promise of AI-native SDLC.

About Atlassian



Atlassian unleashes the potential of every team. A recognized leader in software development, work management, and enterprise service management software, Atlassian enables enterprises to connect their business and technology teams with an AI-powered system of work that unlocks productivity at scale. Atlassian's collaboration software powers over 85% of the Fortune 500 and 350,000+ customers worldwide - including NASA, Rivian, Deutsche Bank, United Airlines, and Bosch - who rely on our solutions to drive work forward.

