Software Development at a Crossroads

How AI Agents Are Changing the Way Teams Work — and What It Means for Visionaries, Agility, and the Future of Software Development

IT Kombinat GmbH · Alexander Grein

Executive Summary

Software development is changing. Not at the level of methods: at the question of who develops software and how.

AI-powered development environments today make it possible to orchestrate specialised agents for architecture, backend, frontend, testing, and DevOps — coordinated by a single person. What previously required a team of numerous specialists is now achievable for a few experienced individuals (sometimes even just one) with the right setup.

This has consequences: for team structures, for agile methods, for the question of who will build software in the future. And above all: who makes the decisions.

The Status Quo: Agile Software Development Today

Since the early 2000s, agile software development has established itself as the dominant paradigm. The Agile Manifesto of 2001 was a reaction to cumbersome waterfall processes: too much planning, too little delivery, too long between idea and functioning software.

Scrum, Kanban, SAFe, and their derivatives have since shaped millions of development teams. Faster delivery cycles, closer collaboration with the customer, continuous improvement: in many projects, these promises were fulfilled.

But with growing adoption, the downsides also became visible:

Ceremonies Consume Time

Daily standups, sprint plannings, reviews, retrospectives, refinements... In a typical Scrum team, a developer spends several hours per week in meetings that don't make the code any faster. In large organisations, this is multiplied through multi-layered coordination.

Teams Become Large and Unwieldy

A complete agile team (Product Owner, Scrum Master, several developers, QA, DevOps, UX) requires coordination on many levels. Decision paths lengthen. The original agility mindset is undermined by sheer team size.

The Gap Between Vision and Implementation Remains

The person who has a software idea is rarely the one who builds it. Between the visionary and the finished product stand backlogs, prioritisation conversations, sprints, reviews. The friction is systemic.

Skills Shortage Worsens the Situation

Qualified software developers are scarce and expensive. Small companies, startups, and organisations with limited budgets simply cannot afford complete development teams. Their ideas remain ideas.

These structural tensions are not new. What is new is that a technological answer is emerging: and it is more radical than expected.

The New Paradigm: AI as Development Team (Support)

The public perception of AI in software development has shifted in the past two years. What began as intelligent autocomplete (GitHub Copilot, TabNine, and similar tools) is today something qualitatively different.

Modern AI systems can no longer only suggest individual lines or functions. They analyse requirements, design architectures, write code, generate tests, orchestrate deployments — in coordinated workflows that resemble a division-of-labour team structure.

The step that changes this is the transition from tool to agent.

An AI tool responds to a request. An AI agent pursues a goal, plans intermediate steps, coordinates with other agents, and acts autonomously: within defined boundaries, with human oversight at critical points.

What Is Possible Today

Concepts such as the AI Development OS show where this development is heading. Specialised AI agents take on the roles of a complete development team — from architecture and backend through frontend and UX to testing, code review, and DevOps. In the extreme case, a human orchestrator sets the direction, makes fundamental decisions, and accepts results.

The system does not work according to fixed sprints. Tasks arise and are processed when they are needed. Decisions are made where expertise resides: with the responsible agent, with human approval at the critical points.

An experienced developer or technically versed product owner can, with AI support,

accomplish noticeably more in clearly defined task types and take on tasks that previously required several specialists. For complex architectural decisions, deep domain knowledge, and critical review, human expertise remains indispensable.

This development is underway. For certain task areas, it is already practice, and the distance to what is still considered a limitation today is shrinking faster than most roadmaps have accounted for.

The Power Shift: The Visionary Gets Involved

In traditional software development, there is a fundamental tension: the person who knows what should be built usually cannot build it themselves. The Product Owner writes user stories: the developers decide how they are implemented. The founder has the vision, the team decides on the technical realisation.

This is not a knowledge question. It is a power question.

Whoever writes the code makes a thousand small decisions that no backlog item can capture. Which library is used? How is the data model structured? Where are shortcuts taken because the sprint deadline is approaching? The technical implementation shapes the product: often more strongly than any roadmap.

AI-powered development environments are beginning to change that.

A visionary who previously depended on a development team can today increasingly set the pace themselves. They don't have to write code. But they must know what they want and be able to communicate it precisely. The quality of the result depends less on technical detail knowledge than on conceptual clarity.

This Changes Team Structures

Smaller Teams with More Latitude

An experienced product thinker with AI support can, in certain areas, take on tasks that previously required several specialists.

New Role Models

The classic "full-stack developer" was an attempt to combine breadth with depth. What is emerging now is different: the technically informed visionary who leads AI agents rather than programming themselves. Conceptual competence becomes the differentiator.

Technical Knowledge Remains Valuable, but Differently

Those who understand how systems work can steer AI agents more effectively, critically evaluate suggestions, and recognise limitations. The specialist knowledge of the developer does not become worthless. Its point of application changes. Less execution, more judgement.

Access Opens Up

Startups, small companies, individuals with a good idea and limited capital gain access to development capacity that was previously reserved for well-funded organisations. This shifts the balance of power in many areas. Not someday: now.

This shift is underway. Those who actively shape it build an advantage that latecomers will find hard to close.

What Becomes of Agility?

Scrum is not dead. But it is under pressure from an unexpected direction.

The agile movement arose as a reaction to rigid processes. Its core was never Scrum or Kanban. Its core was the principle: deliver early, deliver often, learn from feedback, adapt. The ceremonies were a means to an end, not ends in themselves.

With AI-powered development environments, this core re-emerges in a different form.

Sprints Give Way to Continuous Flow

When an AI team delivers in hours rather than weeks, the artificial bundling of work into two-week cycles makes little sense. Tasks arise, are prioritised, processed, and accepted at the pace of actual need.

Ceremonies Give Way to Adaptive Coordination

The daily standup exists because humans need to align. AI agents align continuously. Their status is available at any time. What remains is the human decision: Where do we go next? What is truly important?

The Core Agile Principle Survives — in Purer Form

Customer value over process adherence. Working software over documentation. Responding to change over rigid plans. These values lose none of their validity in an AI-powered environment. They come to the fore more strongly because the friction of implementation decreases.

The Role of Humans Changes

No longer ceremony masters or resource planners. The human becomes the visionary and quality guardian. The question "What do we want to build?" remains human. The question "How is it built?" increasingly less so.

Adaptive Autonomy as a Working Model

Instead of a fixed degree of automation, the visionary decides situationally: for familiar, well-understood tasks, the AI agents work autonomously. For new, risky, or critical decisions, the human is involved at every relevant step. Control remains, but it is applied with purpose.

This is not a departure from agility. It is its evolution.

Risks and Open Questions

Paradigm shifts have downsides. This one is no exception.

Dependency on a Few Platforms

Anyone who builds their development processes on an AI infrastructure becomes dependent on its availability, pricing, and strategic decisions. What is affordable today can be expensive tomorrow. What works today can change with an API update.

One answer to this is the hybrid stack: local language models for data-sensitive or standardised tasks, cloud models where reasoning depth and context length are required. Capable local models (such as Qwen2.5-Coder or DeepSeek) can today handle many coding tasks on suitable hardware without data leaving one's own system. For complex architectural decisions, cloud deployment remains sensible.

This only works with clear governance: who uses which system for which task type, under which data protection requirements. Sovereignty here does not necessarily mean a closed local system, but control over one's own infrastructure and data strategy.

Quality and Responsibility

AI agents make mistakes — sometimes subtle, hard-to-detect ones. Responsibility lies with whoever granted the approval. This requires the ability to assess the result. This capability must be actively maintained. It atrophies quickly when one stops looking.

Knowledge Flattening

When AI agents take over routine tasks, there is a risk that the knowledge embedded in those tasks is no longer built up. The next visionary may simply accept architectural decisions without understanding why they were made that way. Organisational memory must be consciously cultivated.

The Creativity Question

AI combines what it has learned: convincingly, but not originally. New ideas, unusual approaches, the lateral thinking that leads to breakthroughs: that remains human. The real danger is not that AI becomes too creative, but that the convenience of AI use lulls human creativity to sleep.

Social Impact

When a visionary with an AI team does the work of five developers, the question arises: what happens to those five? New roles emerge, old ones disappear. This has been the case with every technological shift. This transition is rarely smooth, and it won't be this time either.

These risks are not an argument for standing still. But whoever ignores them will eventually pay the price.

Outlook: Three Possible Futures

Like every transformative technology, AI in software development is evolving in different directions: depending on how organisations, developers, and society deal with it.

Scenario 1: AI as a Tool

AI becomes a powerful aid, comparable to the transition from the typewriter to the PC. Developers become more productive, teams leaner. The basic structure remains recognisable. AI accelerates but does not replace.

This scenario is the most likely in the short term. It is comfortable — and risky for all who wait too long.

Scenario 2: AI as a Colleague

The boundary between human developer and AI agent blurs. Teams consist of humans and agents with clearly defined roles, mutual expectations, established collaboration models. The human sets the direction and makes decisions. The agent implements and makes suggestions.

This scenario demands new forms of leadership, new competencies, new ethical guidelines: and organisations that want to actively pursue this.

Scenario 3: AI as a Replacement

In certain areas (well-defined, recurring development tasks), AI takes over completely. Not in one big step, but incrementally. What remains is strategy, domain knowledge, conceptual work.

This is not a worst case. For many areas, it is the logical continuation of the current trend.

The reality will be a mix of all three — and probably not in the order that anyone predicts today. Those who assess now which scenario is relevant for their context are better off than those who wait.

The AI Development OS as an Answer

What has been described so far is not a thought experiment.

The AI Development OS puts this power shift into practice: a solo developer or a small team gains, through specialised AI agents, the impact of a complete agile development team: without coordination overhead, without rigid sprint cycles, without the friction losses of large team structures.

The visionary retains control. The system suggests, asks, waits for approval, and then implements. Every critical step is documented, every decision remains traceable.

The goal is not to replace developers, but to empower people with conceptual clarity to build

high-quality software: and to deploy experienced developers where it truly counts: architectural decisions, critical review, domain expertise.

The concept is designed for tech-savvy teams and companies that are actively rethinking software development: not as a universal solution, but as a targeted instrument for those who can and want to use it.

The concept is in active development. Those who are curious — about the architecture, about initial practical experiences, or about pilot projects — are welcome to get in touch.

Author: Alexander Grein, IT Kombinat GmbH

This whitepaper was written on the basis of an ongoing concept and development project. As of: June 2026.

This text was created with the support of AI tools.

IT Kombinat GmbH · Alexander Grein