Agentic AI in Software Engineering

Tools or Team members?

Expert Validation Survey

Background

Agentic Artificial Intelligence (Agentic AI) introduces autonomous, goal-driven agents capable of adapting, learning, and making decisions independently. In software engineering (SE), these agents go beyond traditional tools and generative AI by reshaping development workflows, roles, and human collaboration models.

Objective

- The objective of this study is to contribute a **taxonomy on Agentic AI in Software Engineering**, with a specific focus on its impact on humans and society.

- We aim to investigate the **scope**, **categories**, and **implications** of Agentic AI in SE, focusing on its **technical, ethical, and socio-technical impact** on developers, teams, education, and governance.

- We distinguish Agentic AI from prior generations of AI-based tools, to clarify its role in reshaping human-AI collaboration, and to surface emerging **risks** and **opportunities** tied to its integration across the software development lifecycle.

Our Taxonomy for Agentic AI in SE

Agentic AI is classified along two dimensions:

- **Proactivity level**: **Passive → Reactive → Proactive**
- **Organizational role**: **Questioning → Team-member → Manager**

Combining these dimensions yields **8 agent types**

The taxonomy captures **modes of participation**, not specific tools or products

For each agent type, we describe:

- reported **benefits**
- reported **risks**
- reported **changes** to SE practices

These are **plausible socio-technical trajectories**, not deterministic claims

Effects may vary depending on context, team practices, and governance

Reference slide (1)

Proactivity Level

Passive agents

- Respond only when explicitly invoked
- Do not monitor the environment
- No autonomous initiative
- Examples: on-demand code or documentation generation

Reactive agents

- Respond to events, errors, or predefined triggers
- Act based on policies or rules
- No long-term planning
- Examples: CI agents reacting to failures or changes

Proactive agents

- Observe, plan, and act autonomously
- Initiate actions without explicit triggers
- Adapt behavior over time
- Examples: self-adaptive testing or project orchestration agents

Organization Role

Questioning agents

- Interact through dialogue
- Ask clarifying questions under uncertainty
- Support inquiry and sense-making
- Example: agents eliciting requirements or explaining options

Team-member agents

- Collaborate with humans as part of the team
- Support coordination, learning, and shared understanding
- Adapt communication to team practices
- Example: agents supporting requirements, debugging, or documentation

Manager agents

- Supervise, coordinate, and allocate work
- Assign tasks and manage resources
- Escalate decisions to humans when needed
- Example: agents managing workflows, sprints, or quality signals

Reference slide (2)

Classification of Agentic AI by behavioral capabilities and organizational role

	Questioning Role	Team-member Role	Manager Role
Passive	**Passive & Questioning Agents.** Execute user commands or hardcoded behavior when invoked and engage in interaction with humans to provide a better service. Example: Code or documentation generators that establish a dialogue to better fit user needs.	**Passive & Team-member Agents.** Execute user commands when invoked and can understand and engage in complex social and interpersonal dynamics of a human team. Example: Passive support agents triggered to clarify requirements or summarize discussions, adapting communication to team needs without acting autonomously.	**Passive & Manager Agents.** Not applicable: managerial roles require initiative beyond passive, on-demand interaction.
Reactive	**Reactive & Questioning Agents.** Respond to environmental stimuli using predefined rules or behaviors, engaging with humans to provide a better service. Example: Reactive CI agents monitoring repository events or build results and applying predefined policies to adjust configurations or trigger corrective actions.	**Reactive & Team-member Agents.** Interactive collaborators that respond to stimuli, ask clarifications, and react to team needs through context-aware communication. Example: Requirements engineering agents monitoring discussions or issue trackers and reacting to changes by asking clarifying questions and identifying ambiguities or conflicts.	**Reactive & Manager Agents.** Supervisory agents that respond to stimuli by assigning tasks, monitoring progress, and reacting to project events. Example: Coordination agents that detect delayed tasks or failing builds, reallocate tasks, update sprint plans, notify team members, and escalate issues when needed.
Proactive	**Proactive & Questioning Agents.** Autonomous agents that observe, plan, and act without explicit triggers; they can also ask clarifying questions when uncertainty arises. Example: Self-adaptive testing agents that continuously monitor changes and autonomously adapt testing strategies, generating or modifying tests and querying developers when needed.	**Proactive & Team-member Agents.** Fully autonomous teammates that proactively collaborate, adapt to context, and support shared team goals through continuous contributions. Example: Autonomous development partners that monitor repositories, issues, and documentation, propose design alternatives, contribute code or tests, update documentation, and initiate discussions to align decisions.	**Proactive & Manager Agents.** Advanced manager agents that plan, coordinate, allocate resources, oversee quality, and manage projects end-to-end. Example: Project management agents integrating issue trackers, version control, CI/CD, testing, and communication tools to plan iterations, assign or reassign tasks, enforce standards, monitor quality signals, and coordinate releases with escalation to humans.

Reference example: Passive & Questioning Agents

Passive & Questioning Agents

Definition: Respond only when invoked and interact through dialogue to improve task outcomes.

Example: Code or documentation generators that establish a dialogue with humans to better fit user needs.

Aspect	Summary
Benefits	• Increased productivity and time savings • Support for learning and skill development • Improved accessibility for less experienced developers • Reduced anxiety and higher motivation • Greater user satisfaction and well-being
Risks	• Over-reliance and reduced independent reasoning • Shallow or incorrect learning • Security, IP, bias, and data-leakage risks • Reduced code quality • Higher operational and energy costs
Changes	• Greater reliance on AI-mediated inquiry • Human focus on high-level design decisions • Prompt-based requirements engineering • Need for new practices in auditability, accountability, fairness, and transparency

What we ask you to validate

The questionnaire asks you to evaluate:

- **correctness** of **benefits, risks, and changes**
- **research** and **practical relevance**
- **severity** and **likelihood** of **risks**
- expected **time horizon**

- Of course, there are **no right or wrong answers**
- We are interested in your expert judgment, perception and experience
- Open questions will allow you to highlight missing, unclear or overlooked aspects

Thank you. Now we are ready to start!

Agentic AI in Software Engineering

Tools or Team members?

Expert Validation Survey