ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/aishwarya-babu/574

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science
https://urfpublishers.com/journal/artificial-intelligence

Vol: 3 & Iss: 2

Research Article

Rethinking AI Integration in Software Teams: A Framework for Treating AI as a high context scriptable Intern

Aishwarya Babu*

Citation: Babu A. Rethinking AI Integration in Software Teams: A Framework for Treating AI as a high context scriptable Intern. *J Artif Intell Mach Learn & Data Sci 2025* 3(2), 2725-2726. DOI: doi.org/10.51219/JAIMLD/aishwarya-babu/574

Received: 03 June, 2025; **Accepted:** 28 June, 2025; **Published:** 30 June, 2025

***Corresponding author:** Aishwarya Babu, USA, E-mail: babu.aishwarya@gmail.com

ABSTRACT

As AI tools grow more capable and integrated into software development workflows, their role is evolving. This paper presents a design-driven thought experiment: What if AI were treated not merely as a tool, but as a high-context scriptable intern. An intern who is highly capable and efficient but still requiring supervision? We explore a hypothetical sprint in which an AI agent is assigned tasks, delivers pull requests, participates in retrospectives, and receives feedback just like any other teammate. This model proposes a new mental framework for AI integration in software teams, outlines its potential benefits and constraints, and anticipates the cultural and operational shifts needed to support such a change. This scenario also surfaces critical questions about trust, accountability, and the redefinition of work in AI-augmented engineering.

Keywords: Artificial intelligence, software development, engineering, scriptable intern, AI agent, sprint, AI augmentation

1. Introduction

Today, most Artificial Intelligence (AI) tools in software engineering are positioned as assistants to individual engineers i.e. copilots, code generators, or autocomplete utilities for individuals. What if we extended that and positioned AI as an intern to a software engineering team? AI as an intern with speed and pattern-based insight but without human-like intuition or contextual awareness.

This paper explores the idea of integrating AI into the software development lifecycle (SDLC) as a structured participant. The idea is that the AI is not necessarily a peer, but a *high-context scriptable intern* embedded in sprint cycles. This thought experiment encourages engineering leaders to imagine new collaboration patterns and review processes that support effective AI augmentation.

2. Thought Experiment: AI as a Sprint Participant

To explore how AI might function as a software team member, we propose a sprint-based thought experiment as follows:

Team Composition: A typical cross-functional software engineering team with one virtual contributor: an AI agent powered by a large language model (LLM).

Assigned Scope for the AI Intern: The AI agent receives its own set of tickets in a project management tool (e.g. Jira), just like a junior engineer or intern. Task categories include:

- Generating boilerplate or scaffolding code
- Writing unit and integration tests
- Drafting first-pass internal documentation (e.g. README files, architecture notes)
- Reviewing pull requests (PR) for stylistic consistency, common security issues, and known anti-patterns[1].
- Proposing low-risk refactoring opportunities

Working Protocols:

- For each assigned task, the AI must submit work through the version control system.
- Human engineers review AI-generated PRs i.e. no auto-

merges are allowed as would be expected of any team member.

- For code and document reviews initiated by human team members, the AI provides structured feedback using preset review guidelines or past project examples.

Sprint Retrospective Participation:

- The AI compiles insights based on prompt logs, commit history, and review activity.
- It outputs a summary of its own contributions and observations.
- It flags repeated issues (e.g. ambiguous ticket descriptions, redundant patterns).
- It suggests backlog tasks or process adjustments based on trends across sprints.

AI is a contributor that is consistent, fast, and reliable in structured tasks, but still reliant on human engineers for oversight, judgment, and nuance[1]. This approach replicates the expectations of an intern with their outputs reviewed and refined by experienced engineers to ensure quality and reliability. The table below lists the abilities and limitations of AI along with the expected validation to be conducted by engineers across the various tasks assigned to the AI intern:

Table 1: Abilities and Limitations of AI vs Human Responsibility.

Domain	AI Responsibility	Human Responsibility
Code Generation	Generate scaffolding and tests	Review architecture design, debugging, edge-case logic
Documentation	Create initial drafts, markdown formatting	Validate context, fill narrative gaps, nuanced context, historical decisions
PR Reviews	Flag style, security, duplication	Understand and assess business intent and risk
Retrospective Input	Log pattern analysis, suggest backlog tasks	Evaluate team dynamics and sentiment, subjective experience, priority setting

3. Managing the AI Intern

In typical AI-augmented development workflows, AI tools function as personal assistants. They are helpful in supporting individual engineers in tasks like code generation, test scaffolding, and refactoring. Responsibility for interpreting, validating, and incorporating AI output generally lies with the individual developer. In contrast, our proposed model positions the engineering manager (EM) as the central coordinator of the AI intern's involvement. The EM is accountable for task delegation, feedback orchestration, and ensuring alignment between human engineers and the AI agent.

Effective integration of AI into the development lifecycle requires some organizational adaptations:

- **Delegation vs. Autonomy:** Just as interns are not expected to own full systems, neither should AI. Tasks should be atomic and reviewable.
- **Feedback Structure:** Just like mentoring new hires, engineers and team leads must actively provide corrective feedback, such as prompt refinement and labeling, to help tune AI performance. If neglected, the AI may regress and/or produce irrelevant outputs thus eroding trust and creating downstream friction. These risks emphasize the need for well-scoped tasks, structured prompts, and feedback loops that are consistent and measurable.
- **Team Sentiment:** Managers or leaders must proactively address questions about role clarity, fairness, and collaboration. Does AI free engineers from repetitive tasks or does it introduce ambiguity and overhead?
- **Process Transparency:** AI contributions should be auditable. Dashboards or tooling enhancements may be required to track prompt history, output deltas, and PR impact. Metrics such as prompt iteration count, PR rejection rates or average turnaround time can help evaluate effectiveness.
- **Infrastructure Integration:** Successful operationalization requires tight integration of the AI intern into the team's tools (e.g., CI/CD pipelines, ticketing systems, project management system). EMs must work with DevOps or platform teams to ensure seamless interaction between AI systems and team infrastructure.

This managerial model for AI oversight builds on emerging research in human-AI teaming, where clear role definitions and guided workflows improve collaborative outcomes[3]. It is also important to emphasize reliability, safety, and accountability especially when AI-generated outputs have the potential to impact production code or operational workflows[4].

4. Conclusion

This paper offers a new lens for thinking about how AI can be integrated into software teams as scriptable interns embedded into sprint cycles and not as passive assistants. Through a structured thought experiment, we explored how an AI agent could be assigned, reviewed, and supported like any junior contributor on an engineering team. By positioning AI as a bounded and accountable contributor, we aim to retain the human-centered nature of engineering while still harnessing the scale and efficiency benefits that AI can provide.

With the right supervision, feedback loops, and organizational infrastructure, these AI interns can help reduce cognitive load, streamline repetitive tasks, and uncover patterns that might otherwise be missed. Ultimately, this approach is not about delegating everything to AI, but designing teams that are ready to collaborate with it.

5. References

1. M Li, Y Liu, T Wang, et al. Assisted Assessment of Coding Practices in Modern Code Review. *Proc. ACM/IEEE 46th International Conference on Software Engineering (ICSE)*, 2024, 1-12.
2. A Klein, M Allamanis, C Sutton. Large Language Models in Software Engineering: A Survey, *arXiv preprint*arXiv:2408.08333, 2024.
3. Seeber I, Bittner E, Briggs Ret al. (2020). Machines as teammates: A research agenda on AI in team collaboration. *Information & Management*, 2020; 57: 103174.
4. Shneiderman B. Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human–Computer Interaction*, 2020; 36: 495-504.