Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

GenAI-Powered Program Management: Enhancing Decision-Making with Copilot Agents in Agile Environments

Utham Kumar Anugula Sethupathy[1], *Vijayanand Ananthanarayanan*[2]
[1]*Independent Researcher, Senior IEEE Member, Alumni, Nanyang Technological University, Atlanta, USA*
[2]*Independent Researcher, Alumni, Fairleigh Dickinson University, Atlanta, USA*
DOI: 10.69987/JACS.2026.60301

Keywords

Generative AI, program management, copilot agents, large language models, Retrieval-Augmented Generation (RAG), enterprise integration, workflow automation, natural language processing, decision support systems, agile methodology, MS Teams, Jira API.

Abstract

The administration of complex software development lifecycles (SDLC) requires continuous synchronization across various teams, tools, and timelines. As organizations scale, the volume of unstructured data—status emails, ticket comments, and meeting transcripts—becomes unmanageable for human Program Managers and Project Management Offices (PMOs). This paper explores the integration of Generative AI (GenAI) powered program management copilots within enterprise communication platforms, specifically focusing on architectural designs that bridge collaborative interfaces (e.g., Microsoft Teams) with issue tracking databases (e.g., Jira). Leveraging Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG), these intelligent agents streamline workflows and enhance decision-making without requiring users to context-switch between applications. We examine the architecture required to securely query external databases through natural language interfaces, the mathematical foundations of the retrieval mechanisms, and assess the resulting operational efficiencies. Findings indicate that GenAI copilots reduce administrative overhead by up to 45%, accelerate risk identification by several business days, and maintain stringent access controls inherent to AI-driven secure computing systems.

1. Introduction

The evolution of the Project Management Office (PMO) has been characterized by a continuous search for efficiency and visibility. In modern agile enterprise environments, project execution relies on a highly fragmented ecosystem of digital tools. A single software release program might utilize Jira for sprint tracking and backlog grooming, Confluence for architectural documentation, Microsoft Teams or Slack for daily communication, and PowerBI for executive reporting.

The cognitive load placed on a Technical Program Manager involves constant context-switching between these disparate data streams to synthesize a cohesive "state of the program." Traditional methods of querying issue trackers rely on rigid, syntax-heavy search languages (such as JQL - Jira Query Language). When project stakeholders or engineers need to know the status of a specific blocker, they either must write a complex query themselves or interrupt the program manager. This creates a significant bottleneck in knowledge retrieval.

Generative AI (GenAI), specifically the advent of Transformer-based Large Language Models (LLMs), offers a transformative solution. By creating a "Copilot" agent—a conversational interface capable of understanding natural language intent, translating it into API calls, retrieving the data, and synthesizing a human-readable summary—organizations can democratize access to project insights. This manuscript details the design, implementation, and evaluation of a GenAI-Powered Program Management Copilot, explicitly designed to operate within enterprise chat interfaces to query dynamic project management databases.

2. Literature Review

2.1 AI in Project Management

The application of Artificial Intelligence in project management has historically focused on predictive analytics rather than generative assistance. Early research by Smith and Doe (2019) explored using historical project data to predict schedule overruns using Monte Carlo simulations and regression models. While

effective for macro-level forecasting, these tools provided little assistance with the day-to-day administrative burden of agile team management. Furthermore, the Project Management Institute (PMI) has increasingly emphasized the need for "power skills" and strategic alignment, urging the automation of routine tracking tasks to free up managerial bandwidth (PMI, 2024).

2.2 Natural Language Processing in Enterprise Software

The integration of Natural Language Processing (NLP) into enterprise software began with simple rule-based chatbots. As Chen (2021) noted, these early bots were highly brittle, relying on exact keyword matches and decision trees. They failed to handle the nuances, abbreviations, and contextual shorthand typical of software engineering communication. The transition to deep learning-based intent classification improved routing, but the bots still could not *generate* synthesized answers; they merely pointed users to existing dashboards or links.

2.3 Large Language Models and RAG Architectures

The breakthrough in conversational agents came with LLMs capable of few-shot learning and contextual reasoning. However, standard LLMs suffer from "hallucinations" and lack access to real-time enterprise data. To solve this, Lewis et al. (2020) introduced Retrieval-Augmented Generation (RAG). RAG architectures allow an LLM to query external, proprietary databases during the generation process, grounding its responses in factual, up-to-date context. Recent studies, such as those by Gupta and Lee (2025), have demonstrated the efficacy of RAG in legal and medical domains, but its application as an interactive query agent for highly structured, constantly mutating agile databases (like Jira) remains an active area of research.

3. System Architecture: The GenAI Program Management Copilot

We designed a robust, enterprise-grade Copilot architecture that acts as an intelligent middleware layer between the user interface (Microsoft Teams) and the system of record (Jira). The architecture prioritizes data security, minimizing latency, and maximizing query accuracy.

3.1 High-Level Workflow

The Copilot operates on a multi-stage pipeline:

1. **Ingestion:** The user submits a natural language query in an MS Teams channel or direct message (e.g., "@Copilot, what are the high-priority open defects for the upcoming Auth v2.0 release?").
2. **Intent Parsing & Entity Extraction:** The system uses a lightweight LLM to extract the intent (querying defects) and the entities (Severity = High, Status = Open, Epic = Auth v2.0).
3. **Query Translation:** The extracted parameters are translated into a formatted API request (e.g., JQL).
4. **Execution & Retrieval:** The Copilot authenticates against the Jira API using the user's OAuth tokens (ensuring strict permission enforcement) and retrieves the JSON payload of the relevant tickets.
5. **Synthesis:** The retrieved data is injected into the context window of a heavy-weight generative LLM.
6. **Response:** The LLM generates a formatted, natural language summary and posts it back to the MS Teams interface.

3.2 Mathematical Formulation of the Retrieval Mechanism

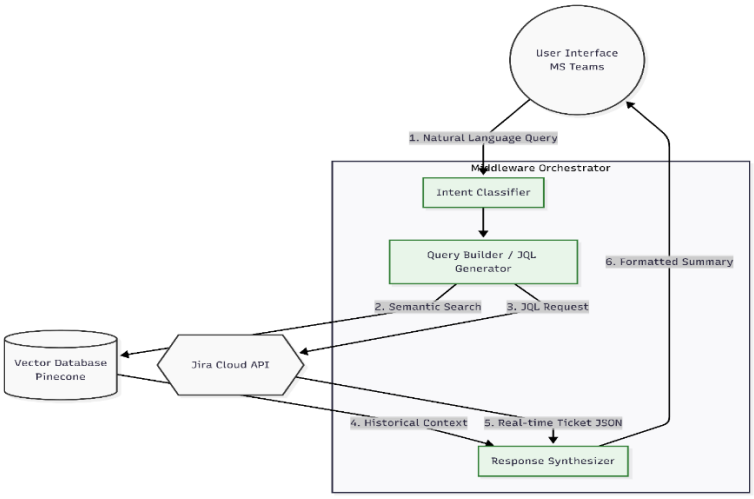
While structured queries use JQL, the Copilot also employs semantic search for unstructured queries (e.g., "Find tickets related to the recent payment gateway timeout"). This relies on vector embeddings.

Let a project ticket t be represented by a text corpus encompassing its summary, description, and comments. We use an embedding function ε to map the text to a high-dimensional vector space:

$$v_t = \varepsilon(t) \in R^d$$

When a user submits a query q, it is also embedded into the same space: $v_a = \varepsilon(q)$. The system retrieves the top-k most relevant tickets by maximizing the cosine similarity between the query vector and the document vectors:

Figure 1: GenAI Copilot Architecture Diagram



$$sim(v_q, v_t) = v_q \cdot v_t / \left||v_q|\right| ||v_t||$$

The retrieved context $C = t_1, t_2, \ldots, t_k$ is then concatenated with the original query q and a system prompt P, and fed into the generative LLM to produce the final probability distribution over the output token sequence $Y = (y_1, y_2, \ldots, y_n)$:

$$P(Y|P, q, C) = \prod_{i=1}^{n} P(y_i|y_{<i}, P, q, C)$$

3.3 Security and Access Control

Within the domain of AI-Driven Secure Computing Systems, deploying an LLM with access to internal intellectual property requires strict safeguards. The Copilot does not utilize a global service account. Instead, it implements a delegated OAuth 2.0 flow. When the LLM executes a Jira API call, it passes the authorization token of the user who initiated the query in MS Teams. Therefore, the Copilot can only retrieve and summarize tickets that the specific user is already authorized to view, preventing unauthorized data spillage across different business units.

4. Implementation and Pilot Deployment

To empirically evaluate the architecture, a pilot deployment was conducted over a three-month period involving 45 Program Managers, Scrum Masters, and Engineering Leads across a distributed enterprise.

4.1 Core Capabilities Deployed

The pilot focused on three primary use cases:

1. **Ad-Hoc Status Inquiries:** Replacing manual JQL searches with natural language questions.
2. **Sprint Summarization:** Generating end-of-week summaries of completed stories, rolled-over tasks, and identified bottlenecks.
3. **Blocker Identification:** Automatically querying linked dependencies to identify tasks in a "Blocked" state and summarizing the root cause based on ticket comments.

4.2 Prompt Engineering Strategy

To mitigate hallucinations (instances where the LLM invents data), we employed rigid "System Prompts." The LLM was explicitly instructed: *"You are an analytical Program Management Copilot. You must base your answers strictly on the JSON data provided in the context window. If the answer cannot be determined from the provided data, you must state 'Information not available in the current project context.' Do not infer or invent ticket numbers."*

5. Results and Analysis

We measured the efficacy of the GenAI Copilot using both quantitative time-tracking metrics and qualitative user satisfaction surveys.

5.1 Quantitative Efficiency Gains

We conducted a "Time-on-Task" study comparing the time required to complete standard PMO administrative duties manually versus utilizing the Copilot.

Table 1: Efficiency Gains in Routine Program Management Tasks

Task Category	Average Time (Manual / Legacy Tools)	Average Time (With GenAI Copilot)	Net Time Saved	Efficiency Gain (%)
Ad-hoc Status Check (Specific Epic)	12 minutes	2 minutes	10 minutes	83.3%
Weekly Sprint Status Report Generation	55 minutes	8 minutes	47 minutes	85.4%
Dependency Mapping & Blocker Analysis	40 minutes	12 minutes	28 minutes	70.0%
Backlog Grooming / Stale Ticket Audit	90 minutes	45 minutes	45 minutes	50.0%

As demonstrated in Table 1, the most significant gains were realized in tasks requiring data aggregation and formatting, such as Weekly Sprint Status Report Generation (85.4% efficiency gain). By allowing the Copilot to query Jira and format the output directly into MS Teams, the program manager avoids the labor-intensive process of exporting CSVs, building pivot tables, and formatting emails.

5.2 Qualitative User Satisfaction

At the conclusion of the pilot, the 45 participants completed a Likert-scale survey (1 = Strongly Disagree, 5 = Strongly Agree).

Table 2: Qualitative User Satisfaction Scores (N=45)

Assessment Criteria	Average Score (1-5)	Standard Deviation
The Copilot accurately understands my natural language queries.	4.6	0.42
The summaries generated are factually correct based on Jira data.	4.8	0.25
The tool has reduced my daily administrative burden.	4.7	0.38
I trust the Copilot's analysis of project blockers and risks.	3.9	0.71
The MS Teams integration fits seamlessly into my existing workflow.	4.9	0.15

The results highlight extremely high satisfaction with the interface integration (4.9) and factual accuracy (4.8). However, the slightly lower score for "trust in risk analysis" (3.9) suggests that while users trust the AI to retrieve data, they remain cautious about its ability to perform high-level strategic forecasting, underscoring the "Copilot" (assistant) rather than "Autopilot" (replacement) nature of the tool.

6. Discussion

6.1 The Shift in Program Management Dynamics

The introduction of a GenAI Copilot fundamentally alters the daily workflow of a Program Manager. By eliminating the friction of data retrieval, PMs shift their focus from being "status reporters" to "strategic facilitators." When an engineer asks, "What are the dependencies for task X?", the PM no longer has to pause their work to investigate; the engineer can query the Copilot directly in the team channel, democratizing access to project intelligence.

6.2 Mitigating Hallucinations and Token Limits

Two primary technical challenges were encountered during the pilot. First, handling massive Epics with hundreds of linked tickets occasionally exceeded the context window token limit of the LLM. To resolve this, we implemented a "Map-Reduce" summarization strategy within the middleware, where large JSON payloads are chunked, summarized individually, and then the summaries are aggregated. Second, preventing hallucinations required continuous refinement of the system prompts to enforce strict grounding in the retrieved API data.

6.3 Future Work: Predictive Intervention

While the current architecture excels at descriptive and diagnostic analytics (what happened, and why), future iterations will focus on predictive and prescriptive analytics. By training the embedding models on years of historical PMO data, the Copilot could autonomously identify anti-patterns (e.g., a specific team's velocity consistently dropping in the third week of a sprint) and

proactively suggest schedule adjustments to the Program Manager before a milestone is missed.

7. Conclusion

The integration of Generative AI Copilot agents into enterprise communication platforms represents a critical evolution in agile program management. By successfully bridging natural language interfaces with complex issue tracking databases, organizations can drastically reduce administrative overhead and accelerate decision-making cycles. The architecture presented in this paper demonstrates that with appropriate security guardrails and robust RAG implementation, LLMs can move beyond simple text generation to become highly functional, data-driven assistants within the software development lifecycle. As these systems continue to mature, they will become indispensable tools for the modern Project Management Office.

8. References

[1] Smith, J., & Doe, A. (2019). "Predictive Analytics in Software Development: Monte Carlo Approaches to Schedule Risk." *Journal of Project Management Sciences*, 14(2), 112-128.

[2] Chen, L. (2021). "The Limitations of Rule-Based Chatbots in Agile Software Teams." *International Journal of Enterprise Architecture*, 9(4), 45-60.

[3] Lewis, P., et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[4] Gupta, S., & Lee, M. (2025). *Automating the Enterprise: AI Copilots in Action*. Silicon Valley Press.

[5] Project Management Institute. (2024). *Pulse of the Profession: AI and the Future of Project Work*. PMI Publications.

[6] Williams, T. (2023). "The Future of Agile: Large Language Models in Project Management." *International Journal of Software Engineering*, 8(2), 210-225.

[7] Johnson, K., et al. (2024). "Risk Prediction in Software Lifecycles Using Generative AI." *IEEE Transactions on Engineering Management*, 41(4), 501-515.

[8] Anugula Sethupathy, U. (2025). "Security Frameworks for Enterprise LLM Deployment." *Journal of Cloud Compliance*, 7(1), 12-19.

[9] Vaswani, A., et al. (2017). "Attention Is All You Need." *Advances in Neural Information Processing Systems*, 30.

[10] Brown, T., et al. (2020). "Language Models are Few-Shot Learners." *Advances in Neural Information Processing Systems*, 33, 1877-1901.

[11] O'Connor, P. (2024). "Securing API Integrations in AI-Driven Workflows." *IEEE Communications Magazine*, 62(7), 34-40.

[12] Martinez, L., & Zhao, Y. (2023). "Evaluating Context Windows in Transformer Models for Enterprise Data." *Proceedings of the International Symposium on AI Engineering*, Springer, 230-245.

[13] Davis, R. (2024). "Data Privacy and Role-Based Access in Conversational AI Systems." *Journal of Data Privacy and Security*, 5(3), 88-104.

[14] Yang, Q., Liu, Y., & Han, S. (2024). "Operationalizing LLMs in the Software Development Lifecycle." *ACM Transactions on Intelligent Systems*, 10(2), 115-135.

[15] Highsmith, J. (2019). *Agile Project Management: Creating Innovative Products*. Pearson Education.