ISSN - 2456-1142

International journal of humanities and information technology Volume 8, Issue 1, 2026

A Unified Framework for Integrating Generative AI Across the Agile Software Development Lifecycle (SDLC)

Vraj Bharatkumar Thakkar*

PhD Candidate, Westcliff University, California, USA

Abstract

Generative Artificial Intelligence (AI) is now widely adopted across various software engineering practices to assist in coding, testing, planning, and deployment tasks. Many enterprise deployments are still siloed, though, with AI solutions being used in only part of the software development lifecycle, not as a part of a cohesive agile workflow. This research aims at providing a single perspective of integrating generative AI into the entire agile software development lifecycle, while focusing on continuous context sharing between discovery, planning, development, integration, delivery, and feedback. It illustrates the synergy between AI voice agents and context-aware development prompts, automated Jira epic generation, predictive integration models, machine learning telemetry, and ERP-linked analytics in enhancing product execution. The article draws on experiences from enterprise-scale hardware-software environments to showcase how integrated AI workflows can reduce manual research effort, enhance access management, assist with faster engineering decisions, minimize data discrepancies, and boost release velocity. The study suggests that generative AI can bring more value when it's not just a productivity tool but a tool that helps product managers, engineers, and stakeholders across the entire product lifecycle. The proposed AI-SDLC framework can provide a practical basis for better agile coordination, minimize the silos in the operation of the enterprise, and maximize the delivery of software in complex enterprise systems.

Keywords: Generative AI, Agile Software Development, Software Development Lifecycle, AI-SDLC Framework, Product Management, Enterprise Software, Continuous Deployment, Machine Learning Integration, ERP Systems, Context-Aware Development

International journal of humanities and information technology (2026) DOI: 10.21590/ijhit.08.02.02

Introduction

The speed at which generative AI has been incorporated into software engineering has sparked exciting possibilities for enhancing software development processes, such as planning, building, testing, and delivering software products in agile teams. While I think the true power of generative AI is not limited to coding or repetitive tasks, it's important to recognize that the tools for generating code or automating specific tasks are only a small part of the picture. The greater promise it has is that it can link various phases of the software development process together into an intelligent, responsive, and context-aware workflow. With the right use of AI, product managers, engineers, testers, and deployment teams can operate from the same set of requirements, risks, user feedback, and delivery priorities.

But I see that many organizations use generative AI in a disjointed manner. These tools can be used by code developers, predictive defect models by quality assurance teams, and even AI-powered research agents by product teams to gather user insights, but they are used individually, not together. This leaves a space between discovery, planning, development, integration, testing, deployment, and feedback. Consequently, beneficial knowledge created in one phase of the lifecycle is not necessarily passed through effectively to another phase of the lifecycle. This reduces the agile team's ability to make timely, coordinated, and data-driven decisions.

This is particularly relevant in enterprise settings where software products are integrated with hardware systems, ERP systems, legacy databases, and business operations. In such environments, it is very difficult to make product decisions

Corresponding Author: Vraj Bharatkumar Thakkar, PhD Candidate, Westcliff University, California, USA, E-mail: vrajthakkar4142@gmail.com

How to cite this article: Thakkar, V.B. (2026). A Unified Framework for Integrating Generative AI Across the Agile Software Development Lifecycle (SDLC). *International journal of humanities and information technology* 8(1), 11-18.

Source of support: Nil

Conflict of interest: None

alone because the performance of the software, data integration, access control, user requirements, and business results are interrelated. I therefore propose that there should be a single AI-SDLC framework that enables the exchange of context from a generative AI tool, machine learning model, telemetry system, and agile planning platform throughout the lifecycle.

This article is an attempt to provide a common set of ideas for incorporating generative AI into the agile software development lifecycle. The framework is about establishing a continuous thread in context from discovery to planning through development to integration and delivery to feedback. It demonstrates the use of AI voice agents for stakeholder research, understanding how generative models can convert product requirement documents into Jira epics and developer prompts, understanding how predictive models can be used to identify integration bottlenecks, and how machine learning telemetry can return defect insights into future planning cycles. These are some of the things the article highlights as the core of its discussion of the need to keep AI as an intelligence layer throughout the lifecycle and not as a disparate collection of automation tools.

This is an effort to show the way in which a product manager can leverage generative AI to enhance agile coordination, minimize operational silos, speed up the release velocity, and improve software delivery in a complex enterprise system. The article also highlights the importance of AI integration that needs to be complemented by strong product strategies, engineering-first thinking, data governance, access control, and stakeholder alignment. As such, this study makes a significant contribution to the ongoing debates about using AI in software development, as it takes a holistic approach to integrating AI throughout the SDLC rather than just into specific phases.

Context and Literature Review

Generative AI has revolutionized the software development lifecycle, impacting planning, development, integration, and delivery, transforming the way agile teams operate. The traditional software development processes were different in each phase of the SDLC with different documentation and different decision-making processes. The iterative planning, sprint development, feedback loops, and adaptive prioritization encouraged more collaboration as a result of agile approaches. When generative AI is applied across the AI lifecycle, however, it can further contribute to these agile principles, bringing an additional layer of automation and intelligence. The biggest hurdle is that a lot of organizations are yet to utilize AI tools as isolated productivity aids and not as part of a comprehensive development system.

However, with the existing usage, there are several use cases that are commonly used with generative AI, such as code completion, generation of test cases, documentation assistance, prediction of defects, and user feedback analysis. While these applications may make one particular stage more efficient, they need to be connected with other lifecycle applications if they are going to have a wider benefit. For instance, a conversation with a stakeholder can be conducted with an AI tool that delivers valuable insights, but those insights don't necessarily translate to product requirement documents, sprint planning, backlog prioritization, or deployment monitoring. Similarly, code generation tools can assist, but not enough context from product discovery or from integration constraints or post-release telemetry data. This leads to a less intelligent workflow and a lack of support for AI throughout the workflow lifecycle.

It is obvious that for enterprise systems where software systems are tied to hardware platforms, operational infrastructure, customer-facing products, and legacy enterprise systems, new requirements are relevant to a more integrated model of AI-SDLC. These are the situations where software development is not autonomous from other tasks. It requires product managers, engineers, data teams, operations, and business stakeholders to work across technical and organizational boundaries. This is particularly insightful in sectors with critical software requirements, like smart mobility, electric vehicles, manufacturing processes, and enterprise resource planning systems, where software architecture decisions may have physical system, supply chain, manufacturing production, and operational cost implications.

Product management is being portrayed as a connection point between AI's capabilities and actual product engineering, and the body of literature is full of focuses on this. AI supports product managers to go beyond being a "stakeholder champion" and turn user requirements into inflexible documents. Product managers can no longer be restricted to a "stakeholder champion" role and can turn user requirements into static documents. They will also be responsible for handling data flow, comprehending the impact of data-based recommendations, checking the data, and giving relevant and contextually applicable data to the technical teams. This means a change from document-driven product management to intelligence-driven product orchestration. This method focuses on using AI to collect user input, identify trends, develop user personas, write Jira epics, help with backlog prioritization, and link development projects to business objectives.

One additional critical element of the literature is the importance of sharing the context throughout the life cycle. Generative AI systems perform better when given the necessary history and technical and business context. Context in agile development can be an interview with stakeholders, user stories, product requirement documents, architecture constraints, histories of sprints, defect logs, integration dependencies, telemetry data, and deployment results. Without these data sources, AI systems are able to make partial or shallow recommendations. Rather, when integrated by a single workflow, AI can aid in more precise planning, improved development prompts, more powerful

integration forecasting, and quicker feedback loops. It fits into the overall message of the article, which emphasizes the importance of embedding AI into the SDLC as a continuum of intelligence rather than any one tool.

The field of integration of machine learning into enterprise delivery systems is another relevant field. Enterprise software development is often complex, involving complex data pipelines, API integrations, access control, and backend infrastructure. It emphasizes the importance of implementing role-based access control, optimizing data management systems, automating ETL pipelines, and centralizing data to enhance processes in delivery and reduce data inconsistencies. The components indicate that AI use isn't just for creating text and code with generative models. It also involves creating reliable data infrastructure that can facilitate the retrieval and manipulation of valuable data within an AI system in a controlled and secure environment.

AI integration is especially advantageous when dealing with ERP environments, where operational performance, inventory management, demand forecasting, and system reliability can impact product and engineering decisions. The potential of AI can be leveraged in ERP systems to identify inefficiencies, predict bottlenecks, and optimize resource use. But, to have any value, such tools need to be integrated into agile workflows. Predictive insights should not only be in dashboards. They are intended to be used for sprint planning, feature prioritization, deployment decisions, and deployment monitoring. This improved relationship of enterprise intelligence and software operation leads to improved business intelligence integration and management.

Effective governance is essential for the successful implementation of AI. The use of generative AI systems can impact planning, coding, testing, and deployment, necessitating the management of data quality, access control, model reliability, and human oversight risks. While an AI-driven suggestion can be a time-saver and a potential great benefit, it's important to test it against your business objectives, engineering limitations, and practical considerations. This is especially critical in enterprise scenarios where the incorrect requirement, integration, or poorly managed access can lead to financial and technical risk. So in the literature, there is a need for a balanced use of AI, where AI facilitates human decision-making but does not replace it.

In general, the current situation indicates that there is a need for research in this area. While there are many use cases for generative AI in specific software development tasks, only a few frameworks articulate how AI can be used throughout the entire Agile SDLC as a unified system. Most of the current methods focus on productivity enhancements and less on lifecycle-wide coordination, feedback in both directions, enterprise data integration, governance, etc. This article addresses this gap by presenting a single AI-SDLC framework to tie together discovery, planning, development, integration, delivery, and feedback. The framework is based on the principle that the most value generative AI is able to provide is when it flows seamlessly between teams, tools, and lifecycle phases to help agile organizations optimize their release cadence, minimize silos, and enhance product decisions.

The Unified AI-SDLC Framework

In this article, I propose a unified AI-SDLC framework designed to address the fragmentation that currently limits the value of generative AI in agile software development. Rather than treating AI as a separate tool used only for coding, testing, documentation, or analytics, I position generative AI as a continuous intelligence layer that connects all major phases of the software development lifecycle. My central argument is that AI becomes more valuable when the outputs generated in one phase of development are carried forward, refined, and reused across later phases.

The main purpose of this framework is to create a continuous contextual thread across discovery, planning, development, integration, delivery, and feedback. In many organizations, product discovery insights remain separated from engineering execution, while defect analytics and user feedback often reach planning teams too late to influence sprint priorities. This separation weakens decision-making and prevents agile teams from fully benefiting from AI-supported workflows. To address this problem, I propose a connected lifecycle model in which generative AI, machine learning, telemetry systems, Jira workflows, product requirement documents, and ERP-linked data systems work together within a shared feedback structure.

In the discovery and planning phase, I use AI to support stakeholder research, interview collection, and persona development. Instead of relying only on manual interviews and static product requirement documents, AI voice agents can collect stakeholder feedback, summarize user needs, and translate qualitative insights into jobs-to-be-done personas. These AI-generated outputs can then be connected directly to product roadmaps, wireframes, backlog items, and sprint priorities. This approach allows product managers to move from passive documentation to active product intelligence.

During the development phase, I extend the framework by connecting product requirements to engineering execution. Generative AI can assist in converting product requirement documents into structured Jira epics, user stories, acceptance criteria, and context-aware developer prompts. In this way, developers do not receive isolated coding instructions. Instead, they receive prompts that are grounded in stakeholder needs, business priorities, system constraints, and sprint objectives. This improves traceability between product strategy and technical implementation.

In the integration and ERP phase, I apply AI to one of the most complex areas of enterprise software delivery: system connectivity. Many organizations still depend on manual data mapping, siloed APIs, and fragmented integration processes. Within the proposed framework, AI models can

predict integration bottlenecks, identify data inconsistencies, and automate ETL pipeline mappings across legacy systems. This is especially important in enterprise environments where software platforms must interact with ERP systems, inventory tools, access control systems, and operational databases.

In the delivery and feedback phase, I emphasize the importance of continuous learning. Instead of treating deployment as the final stage of development, the unified AI-SDLC framework uses machine learning pipelines and telemetry data to monitor system performance, detect defects, analyze user behavior, and return insights to future planning cycles. This creates a bidirectional feedback loop in which post-release data informs backlog refinement, sprint planning, defect prediction, and product roadmap adjustments.

SDLC Phase	*Traditional Process*	*Unified AI-Integrated Process*
Discovery & Planning	Manual stakeholder interviews and static PRD drafting.	AI voice agents automate interview collection and synthesize jobs-to-be-done personas dynamically.
Development	Manual ticket creation and isolated code generation.	Generative models translate PRDs directly into structured Jira epics and context-aware developer prompts.
Integration & ERP	Siloed API connectivity and manual data mapping.	AI models predict integration bottlenecks and automate ETL pipeline mappings across legacy systems.
Delivery & Feedback	Reactive bug tracking and delayed user analytics.	Machine learning pipelines monitor real-time telemetry and automatically update planning models with defect predictions.

This framework shows that AI integration should not be limited to task-level automation. I argue that the real value of generative AI emerges when it supports lifecycle-wide coordination. For example, stakeholder insights gathered during discovery should influence backlog creation; backlog decisions should guide developer prompts; integration risks should inform sprint planning; and telemetry data should return to product teams as evidence for future decisions. This connected structure allows agile teams to move from fragmented AI adoption to a more intelligent, adaptive, and traceable development process.

The framework also strengthens collaboration among product managers, software engineers, data teams, QA teams, and operations teams. By creating a shared AI-supported workflow, each team can work from a more consistent understanding of product goals, system constraints, and user needs. This reduces duplicated effort, improves communication, and helps teams identify risks earlier in the development lifecycle.

However, I also recognize that this framework requires careful governance. AI-generated outputs must be validated by human experts, especially in enterprise systems where inaccurate requirements, weak access controls, or faulty integrations can create operational risks. Therefore, the proposed AI-SDLC framework does not remove human judgment from agile development. Instead, it strengthens human decision-making by improving context availability, automating repetitive analysis, and supporting faster movement from insight to execution.

Overall, the unified AI-SDLC framework provides a practical model for using generative AI across the full agile lifecycle. By connecting discovery, development, integration, and delivery through continuous feedback, the framework helps organizations reduce operational silos, improve release velocity, strengthen product decisions, and build more adaptive software systems.

Case Studies in Enterprise Implementation

In this section, I examine how the unified AI-SDLC framework can be applied within enterprise software environments where product delivery depends on strong coordination between product strategy, engineering execution, data infrastructure, and stakeholder management. I present these case studies to show that generative AI integration is not only a technical issue but also a product leadership challenge. For AI to create measurable value across the software development lifecycle, it must be connected to real workflows, clear decision points, and accountable human oversight.

Enterprise implementation requires more than the adoption of separate AI tools. In my view, the success of an AI-SDLC model depends on how well AI-generated outputs move across teams and systems. Discovery insights must inform planning. Product requirement documents must support development. Integration risks must be visible before they delay delivery. Post-release feedback must return to the product roadmap. The following case studies show how this connected approach can reduce manual effort, improve delivery efficiency, and strengthen agile decision-making in complex enterprise systems.

Orchestrating Context in Planning and Discovery

The first case focuses on the use of generative AI in product discovery and planning. In many enterprise environments, stakeholder research is still handled through manual interviews, note-taking, transcript review, and static reporting. While this process can produce useful insight, it often creates delays between stakeholder engagement and product execution. Product managers may spend significant time organizing interview data before the findings can be converted into personas, wireframes, backlog items, or roadmap priorities.

To address this limitation, I position AI voice research agents as a practical tool for automating early discovery

work. These agents can support interview collection, summarize stakeholder responses, identify recurring needs, and organize feedback into Jobs-to-be-Done personas. This allows product teams to move faster from raw stakeholder input to structured product intelligence. Instead of treating user research as a separate documentation exercise, I use AI to make discovery outputs more actionable for agile planning.

Deploying an AI voice research agent can reduce manual screening time by approximately 60%, provided that interview collection, transcription, screening, and synthesis workflows are integrated into the discovery phase *Jadczyk, T., Wojakowski, W., Tendera, M., Henry, T. D., Egnaczyk, G., & Shreenivas, S. (2021). Artificial intelligence can improve patient management at the time of a pandemic: the role of voice technology. Journal of medical internet research, 23(5), e22959.* . This is important because enterprise product teams often work with large groups of stakeholders whose feedback must be reviewed, compared, and prioritized. In one implementation context, insights from more than 40 enterprise stakeholders were synthesized into Jobs-to-be-Done personas, helping product teams connect high-impact findings directly to wireframes and strategic product roadmaps.

From my perspective, the main value of this case study is not only time reduction. The deeper value is the creation of a stronger contextual foundation for the rest of the SDLC. When stakeholder insights are structured early, they can flow into product requirement documents, Jira epics, sprint planning, and development prompts. This reduces the risk that engineering teams will work from incomplete or outdated requirements. It also improves traceability between user needs and technical implementation.

This case also shows how generative AI can strengthen the role of the product manager. Rather than manually processing every piece of discovery data, the product manager can focus on validating AI-generated patterns, resolving conflicting stakeholder priorities, and aligning product decisions with business goals. In this way, AI supports judgment rather than replacing it. Human oversight remains necessary, especially when stakeholder feedback includes ambiguous needs, competing priorities, or context that requires domain expertise.

Machine Learning Integration for Agile Delivery

The second case focuses on machine learning integration within agile delivery environments. While generative AI is often discussed in relation to code generation or documentation, enterprise implementation requires a broader technical foundation. AI-supported development depends on reliable data structures, secure access controls, automated data pipelines, and integration with existing business systems. Without this foundation, AI tools may generate useful outputs, but those outputs may not be reliable, secure, or actionable within real enterprise workflows.

In this case, I emphasize the importance of customized data management platforms supported by role-based access control. RBAC is especially important in enterprise environments because different users require different levels of access to product data, engineering data, operational systems, and analytics outputs. A unified AI-SDLC framework must therefore ensure that AI systems can access relevant context without exposing sensitive data or weakening governance.

Agile tools such as Jira and structured product requirement documents also play a central role in this implementation. In my framework, Jira is not simply a task management tool. It becomes a bridge between product intelligence and engineering execution. AI-generated insights from discovery, defect prediction, ERP analytics, and integration monitoring can be translated into epics, user stories, sprint tasks, and acceptance criteria. This makes agile delivery more responsive to real system conditions and business priorities.

The case also includes machine learning-powered ERP tools, such as Demand Engine 2.0. According to the source paper, orchestrating roadmaps for such ERP tools can produce a 15% efficiency improvement and approximately $13 million in inventory savings. These outcomes show that AI-SDLC integration can influence more than software delivery speed. It can also improve operational performance, financial efficiency, and enterprise resource planning.

A key lesson from this case is that AI-generated insight must be connected to delivery mechanisms. Predictive analytics alone cannot improve agile performance if the insights remain inside dashboards or reports. For AI to support delivery, its outputs must feed into backlog decisions, integration planning, release prioritization, defect management, and deployment monitoring. This is where the unified AI-SDLC framework becomes valuable: it creates a structured pathway for moving machine learning insight into agile execution.

Automated ETL pipelines and centralized databases also play a major role in this case. Enterprise systems often suffer from data discrepancies caused by fragmented databases, manual updates, inconsistent mappings, and disconnected tools. By using centralized databases and automated ETL pipelines for parts management, organizations can reduce discrepancies and limit financial write-offs. In my framework, this kind of integration is essential because AI models need consistent and reliable data to generate useful predictions, recommendations, and planning outputs.

Overall, these case studies show that enterprise AI implementation must be treated as a connected product and engineering strategy. AI voice agents can improve discovery and planning, while machine learning systems, RBAC platforms, ERP analytics, Jira workflows, and ETL pipelines can strengthen agile delivery. Together, these cases support my argument that the greatest value of generative AI emerges when organizations move beyond isolated automation and build a unified intelligence layer across the full software development lifecycle.

Discussion

In the above article, I have said that the usefulness of generative AI in agile software development does not lie in each new tool, but rather in how those tools can be integrated throughout the entire software development lifecycle. The integrated AI-SDLC process depicted in the proposed unified framework illustrates that the use of generative AI is not restricted to specific tasks, like code generation, supporting documentation, summarizing interviews, predicting and identifying defects, or monitoring deployments. Rather, I see AI as an ongoing layer of intelligence that is used for discovery, planning, development, integration, delivery, and feedback, all in one stream.

This is one of the most significant implications of this framework: It alters the way agile teams work with context. Traditional agile environments tend to be populated with lots of notes on stakeholders' backs, product requirements, Jira tickets, sprint boards, code repositories, testing platforms, ERP systems, and telemetry dashboards. Each tool could be helpful, but the absence of a steady information flow can hinder the coordination process. One of the primary reasons many organizations are not realizing the value of AI implementation is because they are not seeing it on the job. For me, this is one of the main reasons that many organizations aren't getting the full value of AI adoption. No, they're not seeing it on the job. If the insights generated by AI are confined to a single phase of the SDLC, they generate local efficiencies, but not intelligence across the SDLC.

This is why the unified AI-SDLC framework has been created to ensure that information from one phase of SDLC affects the following phase. For example, stakeholder feedback gathered via the AI voice agent can feed into Jobs-to-be-Done personas that can be used to create product requirement documents, Jira epics, developer prompts, and sprint planning. Similarly, data acquired from machine learning-based insights from telemetry and defect prediction can be fed back to the planning teams and inform future prioritization efforts. This two-way exchange of information makes agile development more adaptive, as product decisions are made on the basis of immediate feedback rather than reporting.

In my opinion, one of the greatest assets of this framework is its applicability to enterprise systems where software systems are connected to hardware platforms, ERP systems, legacy software, access control systems, and operational databases. In these environments, software delivery is more than just writing and releasing code. It also includes the management of data consistency, integration risk, requirements of the stakeholders, compliance expectations, and the impact on the operation. The real-world examples in the article demonstrate how AI-powered discovery can enhance software application performance, how RBAC can streamline access to business applications, how automated ETL pipelines can simplify the data integration process, how Jira can complement workflows, and how machine learning can drive enhanced ERP software performance. The source paper contains examples of reduced manual screening time. improved access management efficiency, efficiency gains, inventory savings, and reduced data discrepancies.

A second implication is that the product manager is a role that is also evolving. With a fragmented AI, product managers can still do a significant amount of the work, first through translation of stakeholder needs into requirements, structuring of discovery data, and communication between disconnected teams. The product manager in a unified AI-SDLC model is the orchestrator of AI-powered intelligence. This role is focused on verifying AI-generated results, linking the insights to engineering tasks, ensuring that recommendations are aligned to business agendas, and holding accountability throughout the lifecycle. I don't think that AI is about to replace product leadership. Instead, I view it as being able to make quicker, more informed, and more traceable decisions as a product manager.

The framework also holds special significance for engineering teams. In larger organizations, the product discovery process, system architecture, and delivery operations are often managed by different teams, and developers are given tasks with limited business context. Generative AI can enhance the quality of developer input by converting product requirement documents into structured Jira epics and context-aware development prompts. This can in turn minimize ambiguity, enhance and refine sprint execution, and enable the developer to see why the feature is important, how it relates to the needs of the stakeholders, and what constraints must be applied when it is to be implemented.

But at the same time, I know that the implementation of a unified AI-SDLC framework poses a few challenges. The first one is data governance. AI systems rely on data product information, user feedback, operational history, telemetry logs, engineering designs, and more, so organizations need to establish guidelines for data access, storage, validation, and use. In the absence of robust governance, AI tools can generate recommendations based on incomplete, outdated, and ill-structured data. This may result in poor planning decisions, the wrong number of defects predicted, or poor development results.

Access control is the second challenge. Enterprise systems typically hold valuable business, technical, and operational data. Access to sufficient context is essential for AI systems to be valuable and support a unified AI-SDLC framework without compromising role-based permissions or security boundaries. This is why RBAC and secure data architecture are crucial. Weak access control can make the integration of AI with a system even more dangerous than it is without it.

The third challenge is integration of legacy systems. In many enterprise environments, older ERP systems, disparate databases, manual data mappings, and inconsistent API structures continue to be the norm. Such constraints hinder AI systems from obtaining proper and timely information. Organizations need to create centralized databases and automated ETL pipelines and monitor integration before

generative AI can provide a lifecycle-wide value. Poor system architecture is not something that AI can rectify. It needs to be backed up with a solid infrastructure.

The fourth challenge is the reliability of the models and human supervision. With generative AI, outputs can be fluent and appear to be helpful but still need to be validated. A poor set of requirements, bad acceptance criteria, bad code suggestions, or bad integration predictions can impact the quality of delivery in agile software development. Therefore, I believe that Human in the Loop (HWIL) governance of AI-SDLC should be kept in the mainstream. AI-generated outputs need to be reviewed and validated by product managers, engineers, QA teams, and data specialists before they impact decisions on releasing products or changes.

A potential issue is relying too heavily on AI-generated suggestions. AI can speed up research synthesis, create backlogs, drive development prompting, and predict defects but should not be used in lieu of domain expertise. Judgment, negotiation, prioritization, and accountability are all essential parts of agile development. These are obligations of humans. AI isn't a tool that should take away the professional responsibility from the development process; it's a tool that can assist, enhance, and support the process.

In spite of these difficulties, I think that the advantages of having a unified AI-SDLC framework are great. Through the integration of discovery, development, integration, and feedback, organizations can overcome operational silos and boost release speed. They can also enhance tracking of the needs and requirements of the stakeholders and the features delivered. This is particularly useful in complex enterprise environments where delays, integration issues, inconsistencies, and ambiguous requirements can cause significant cost and delivery risks.

The conversation also demonstrates the need to consider generative AI in the context of a wider enterprise transformation. Its value is dependent on organizational readiness, data maturity, process alignment, and leadership commitment. If the AI is used to augment current workflows that are already siloed, there might be limited impact. On the contrary, companies that rethink their SDLC with a view to sharing context continuously tend to be more coordinated, faster, and more successful in the creation of products.

In conclusion, I believe the integrated AI-SDLC model offers a viable path for the future of agile development with the help of AI. It moves from single automation to integrated intelligence. It also emphasizes the need for product leadership, secure infrastructure, human validation, and continuous feedback. If implemented with care, this model can enable enterprise teams to break out of the silo mentality of using AI and create a software delivery system that's faster, more adaptive, and more in tune with the needs of real stakeholders and operations.

Conclusion

This article has explored the importance of adopting a shared approach to the integration of generative AI throughout the agile software development lifecycle. I've stated that while generative AI is getting used in software engineering with great frequency, the impact is often minimal because tools are used as standalone phases of the software development lifecycle (SDLC) in discovery, coding, testing, integration, or deployment. This silo mentality can cause individual tasks to be done better, but it isn't sufficient to solve the larger coordination issues of agile product delivery in enterprise environments.

The proposed unified framework, AI-SDLC, addresses this drawback by proposing that AI should be extended as an intelligence layer throughout the discovery and planning, development, integration and ERP, and delivery and feedback stages. This framework will enable stakeholder feedback to flow straight from discussions into product requirement documents, Jira epics, developer reminders, integration planning, defect forecasting, and post-release enhancements. I think this is the information that flows from one to the other, which is crucial to increase release speed, diminish operational silos, reinforce product decisions, and make agile workflow more responsive to real-time feedback.

For enterprise AI to be effective, there needs to be a robust product plan, a dependable data infrastructure, an adequate access control system, and human oversight, as evidenced in the case studies featured in this article. AI voice research agents can help cut down on manual discovery work and enhance stakeholder synthesis; machine learning-integrated ERP systems, automated ETL pipelines, RBAC, and telemetry systems can contribute to the improvements of agile delivery and operational performance. The examples demonstrate that AI can be most effective when it is linked to real-world processes and tangible business results.

Meanwhile, I am aware that generative AI shouldn't be used to replace a product manager, engineer, QA team, or business stakeholders. Rather, it should complement their efforts by enhancing context sharing, automating repetitive analysis, detecting risks early, and providing traceability of decision-making. Despite all the advancements in AI, human oversight is still crucial for the validation of AI-generated content, governance concerns, and stakeholder priorities and to ensure that software products meet technical, operational, and business needs.

The overall aim of the integrated AI-SDLC approach is to offer a realistic path for companies aiming to transcend the disjointed approach to AI adoption. Enterprise teams can create a more intelligent, coordinated, and adaptive software delivery system by integrating and delivering discovery, development, delivery, and feedback with AI support. In my view, the potential of generative AI for agile software development isn't just about the advancements in model capabilities but also about the organizations' capacity to embed the models into secure, context-aware, and life-cycle-embracing software development processes.

References

[1] Jadczyk, T., Wojakowski, W., Tendera, M., Henry, T. D., Egnaczyk,

G., & Shreenivas, S. (2021). Artificial intelligence can improve patient management at the time of a pandemic: the role of voice technology. *Journal of medical internet research*, *23*(5), e22959.

[2] Parthiban, M., Priyanka, M. H., Teja, P., Kumari, R., & Narim, S. (2026, February). Systematic Review of Cloud Enabled AI Framework for Optimizing the Software Development Life Cycle. In *2026 IEEE International Conference on Intelligent Systems, Smart and Green Technologies (ICISSGT)* (Vol. 1, pp. 1-5). IEEE.

[3] Njuguna, L. W. (2024). AI-Assisted Digital Forensics for National Security Investigations. International Journal of Technology, Management and Humanities, 10(01), 125-146.

[4] Soni, A., Kumar, A., Arora, R., & Garine, R. (2023). Integrating AI into the software development life cycle: best practices, tools, and impact analysis. *Tools, and Impact Analysis (June 10, 2023)*.

[5] Chintagunta, S. K. (2025). The Role of Artificial Intelligence in Software Engineering: A Review of Frameworks, and Impact on the Software Development Life Cycle. *International Journal of Emerging Research in Engineering and Technology*, *6*(4), 72-79.

[6] Njuguna, L. W. (2024). National Cyber Workforce Development Strategies for Addressing the Cybersecurity Skills Gap. International Journal of Humanities and Information Technology, 6(04), 101-123.

[7] Malladi, N. V., & Reddy, S. (2025, October). Generative AI in Agile Software Development: A Comprehensive Survey. In *2025 7th International Conference on Innovative Data Communication Technologies and Application (ICIDCA)* (pp. 1199-1208). IEEE.

[8] Mohapatra, H., Pramanik, S., & Ranjan Mishra, S. (2025). Revolutionizing software development: The transformative influence of machine learning integrated SDLC model. In *Boosting Software Development Using Machine Learning* (pp. 41-69). Cham: Springer Nature Switzerland.

[9] Prakash, M. (2024). *Role of generative AI tools (GAITs) in software development life cycle (SDLC)-waterfall model*. Massachusetts Institute of Technology.

[10] Phadke, S. U. A. 23. A Framework for AI-Assisted Software Development Life Cycle for Generative AI Applications. *n, &JfA*, 321.

[11] Wanjiru, L. (2025). Securing IoT Devices: AI and Blockchain as a Dual Defense Mechanism. Algora, 2(2), 53-78.

[12] Mohamed Iqbal, F. F. (2025). Framework for sustainable integration of artificial intelligence in to software development life cycle: insights from state of art and state of practice.

[13] Imam, a. (2024). Integrating ai into software development life cycle.

[14] Pham, V. T. N., & Nguyen, Q. V. (2025, July). Accelerating Software Development Cycle with a Multi-agent Generative AI Approach: A Case Study with OpenAI's GPT. In *Conference on Information Technology and its Applications* (pp. 187-202). Cham: Springer Nature Switzerland.

[15] Mazumder, P.T. Explainable and fair anti-money laundering models using a reproducible SHAP framework for financial institutions. Discov Artif Intell 6, 262 (2026). https://doi.org/10.1007/s44163-026-00944-7

[16] Thakkar, V. B. (2025). Defining Success in Probabilistic Products: Key Performance Indicators and Lifecycle Management for Generative AI Applications in Enterprise. *International Journal of Technology, Management and Humanities*, *11*(04), 66-79.

[17] Mazumder, P. T. (2025). Blockchain in trade finance: reducing fraud and improving efficiency through digital ledger technology. Digital Finance, 7(4), 1043-1063.

[18] Thakkar, V. B. (2026). From Linear Logistics to Neural Supply Chains: Predictive Machine Learning and the Rise of Autonomous Supply Chain Intelligence. *International Journal of AI, BigData, Computational and Management Studies*, *7*(1), 153-161.

[19] https://www.outlookindia.com/hub4business/the-hardware-software-hybrid-thakkars-engineer-first-approach-to-product-leadership

[20] Mazumder, P. T. (2023). Data Driven Detection of Trade Based Money Laundering (TBML): A predictive Analytics Framework for Securing US supply chains and Financial Integrity. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, *15*(04), 423-432.

[21] https://techbullion.com/driving-the-digital-shift-how-vraj-thakkars-product-management-ai-expertise-is-redefining-software-development-in-electric-vehicle-space/

[22] Arshad, N., Butt, T. A., & Iqbal, M. (2025). A comprehensive framework for Intelligent, Scalable, and Performance-Optimized software development. *IEEE Access*, *13*, 74062-74077.

[23] Anderson, M. J. (2026). End-to-End AI-Augmented Software Development Lifecycles (SDLC) in Intelligent Enterprises.

[24] Jana, A. K., Saha, S., & Dey, A. DyGAISP: Generative AI-Powered Approach for Intelligent Software Lifecycle Planning.

[25] Martins, D. D. O. B. (2024). *A framework for leveraging artificial intelligence in software development* (Master's thesis, Universidade NOVA de Lisboa (Portugal)).