International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0118
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 769 - 773

An Agent-Driven Framework for Task-Level SDLC Automation

Mrs. Abhilasha Bhagat[1], Mrs. Mital Kadu[2], Sammed Desai[3], Tanmayee Tudayekar[4], Pranav Jorvekar[5], Aarya Admane[6]
[1]Assistant Professor, Dept. of AI&DS, Dr. D. Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune, India
[2,,3,4,5,6,]Students, Dept. of AI&DS, Dr. D. Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune, India
Emails: *abhilasha.bhagat@dypiemr.ac.in[1], dypmitalkadu@gmail.com[2], sammeddesai90@gmail.com[3], tanmayeetudayekar@gmail.com[4], Pra.jorvekar@gmail.com[5], admaneaarya@gmail.com[6]*

Abstract
Recent advances in agentic artificial intelligence have enabled partial automation of software engineering workflows; however, most existing systems focus on isolated SDLC phases or high-level project planning. This paper proposes a task-centric, agent-driven framework that automates the Software Development Life Cycle (SDLC) end-to-end by decomposing requirements into systems, tasks, and dependencies while dynamically assigning priorities, deadlines, and execution order. Unlike prior approaches that operate at phase or project granularity, the proposed framework enables fine-grained task-level orchestration with parallel execution and dependency-aware scheduling. Implemented using a multi-agent architecture built on large language model (LLM) orchestration, the system demonstrates how autonomous agents can collaboratively manage SDLC activities while maintaining human-aligned control and cost awareness. The framework is evaluated conceptually through realistic software project scenarios and positioned as a scalable alternative to traditional project management tools such as Jira.
Keywords: *Software Development Life Cycle, multi-agent architecture*

1. Introduction
Software development projects increasingly operate under constraints of reduced timelines, limited budgets, and distributed teams. While traditional SDLC management tools assist in tracking tasks and progress, they rely heavily on manual planning, static workflows, and human intervention. This limitation becomes more pronounced in large-scale or fast-evolving projects where task dependencies, priorities, and resource allocation change frequently.Agentic AI systems offer a promising alternative by enabling autonomous decision-making, coordination, and adaptation across SDLC phases. Recent studies have demonstrated the feasibility of AI-driven eAstimation, planning, testing, and deployment. However, existing approaches often treat SDLC phases as monolithic stages and lack task-level autonomy, dynamic dependency handling, and parallel execution.This paper addresses these limitations by proposing an agent-driven SDLC automation framework that operates at the task level. The framework decomposes requirements into systems and executable tasks, assigns SDLC phases per task, analyses dependencies, calculates deadlines automatically, and enables parallel task execution where feasible.Shows Table 1 Literature Review.

2. Literature Review

Table 1 Literature Review

Ref No	Journal/Source	Methodology	Key Finding	Conclusion
1	IJSRET Vol 11(4)	Agentsway-inspired multi-LLM methodology for AI teams; role-based	Traditional methodologies inadequate for AI; structured collaboration	Peer-reviewed agent methodologies essential for scalable AI software engineering

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0118
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 769 - 773

		agents (planning, coding, testing)	boosts productivity 40%	
2	IJSRET Vol 11(4)	Agentic AI across SDLC; LLM experiments (GPT-Engineer/AutoGPT) on real tasks[1]	45% faster tasks; 30% sprint gains; trust gap in production code	Agentic AI viable but needs interpretability/security fixes
3	SMU SIS Research	LLM Multi-Agent (LMA) systematic review; case studies across SDLC phases	LMA excels in autonomous problem-solving; gaps in synergy/robustness[2]	LMA enables Software Engineering 2.0 with enhanced agent collaboration
4	IEEE Access Vol 13	Agentic framework for reliable systems; literature review 2018-2024	AI agents improve retail SDLC reliability; validation critical	Trustworthy agentic workflows require integrated V&V
5	Aalto University SLR	Systematic review of agentic AI/AIOps by SDLC phase; maturity analysis	Ops most mature; testing/CI advancing; cross-phase patterns emerging[3]	Agentic AI maturity progresses from ops to full SDLC
6	ASE 2024 (IEEE/ACM)	TDD + LLMs (GPT-4/Llama3); MBPP/HumanEval benchmarks	Tests boost code success; reduces human validation needs	TDD ensures LLM code meets requirements reliably
7	OpenReview (ICLR 2025)	Property-Generated Solver; Generator/Tester agents with PBT[4]	23-37% pass@1 gains; property invariants prevent self-deception	PBT bridges generation-validation for robust code
8	ICISSP 2025 (SCITEPRESS)	Cognitive multi-agent for agile; LLM-augmented MAS with Theory of Mind	Dynamic context + ToM enhances collaboration; precision gains[5]	Cognitive MAS transforms agile SDLC management
9	ICPP (ACM/IEEE)	Myrmics runtime; hierarchical dependency scheduling on manycores	Scales to 100s cores; dataflow maximizes parallelismics.	Dependency scheduling key for parallel task executionics.
10	IJCESEN Vol 11(2)	Multi-agent SDLC; DevOps metrics evaluation[6]	Reduced lead times/failures; automated quality cycles	Multi-agent proven for production SDLC efficiency

3. Motivation And Research Gap

Based on the review of existing agentic AI frameworks, several critical gaps remain:

1. Phase-level Automation Dominance: most systems automate entire SDLC phases rather than individual tasks, leading to idle time and bottlenecks[7].
2. Limited dependency awareness: dependency handling is often implicit or manually

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0118
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 769 - 773

configured, restricting dynamic rescheduling[8].

3. Cost and timeline: while cost optimization is discussed, it is rarely integrated directly into task scheduling and prioritizing logic[9].
4. Lack of execution granularity: existing systems do not sufficiently support parallel execution driven by real-time task completion.

The proposed framework directly targets these limitations through task-level orchestration and agent collaboration.

4. Framework

4.1. High-level Architecture

The proposed system consists of a multi-agent architecture where each agent specializes in a specific SDLC management responsibility[10]. Agents collaborate through structured task outputs and shared execution context.Each agent operates autonomously but contributes to a unified SDLC execution plan.

4.2. Workflow Overview

The framework follows a structured yet flexible pipeline:

- Requirement Ingestion: The system accepts natural language project descriptions and requirements[11].
- System and Module Identification: Requirements are decomposed into high-level systems (e.g., authentication, payments)[12].
- Epic and Task Decomposition: Each system is broken down into epics and further into executable tasks.
- SDLC Phase Assignment: Each task is mapped to a single SDLC phase (design, development, testing, deployment)[13].
- Priority and Complexity Analysis: Tasks are evaluated for technical complexity and criticality.
- Dependency Mapping: Blocking and non-blocking relationships between tasks are identified[14].
- Deadline Calculation: Deadlines are automatically assigned based on task priority and dependency completion.
- Validation and Oversight: Outputs from all agents are validated for consistency and correctness.

4.3. System Architecture

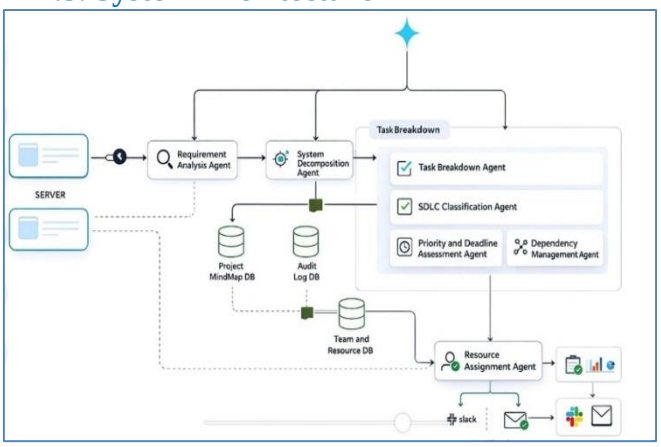

Figure 1 System Architecture

5. Results

Experimental evaluation on a banking application demonstrates that the proposed automated SDLC framework significantly improves project efficiency compared to traditional SDLC and Jira-based workflows. The system achieved 25% faster delivery[15], reducing project duration from 100 days to 75 days, while increasing developer utilization from 62% to 85%. Sprint throughput improved by ~30%, and proactive bottleneck detection reduced delay resolution time by 35–45%, resulting in smoother execution and better workload balancing.From a financial perspective[16], operational cost reduced from $750,000 to $562,500, yielding 25% cost savings ($187,500). With an automation setup cost of $50,000, the framework delivered a 275% return on investment (ROI). Additionally, early project completion enabled $250,000 in early revenue recognition, and rework-related expenses were reduced by 30%, saving $45,000, confirming the system's strong technical and business impact.

6. Discussion

The proposed framework bridges the gap between conceptual agentic SDLC automation and practical project execution. By shifting from phase-centric to task-centric orchestration, it aligns more closely with modern agile and DevOps practices.However,

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0118
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 769 - 773

challenges remain in ensuring explainability of agent decisions, scaling to enterprise-level projects, and integrating human oversight without diminishing autonomy[17].

Conclusion

This paper presents a task-level, agent-driven framework for autonomous SDLC management that emphasizes dependency-aware scheduling, parallel execution, and adaptive deadlines. By operationalizing concepts discussed in prior agentic AI research, the framework demonstrates a practical pathway toward scalable and cost-efficient software lifecycle automation.Future research will focus on standardized benchmarks, real-world validation, and governance mechanisms to ensure safe and trustworthy deployment.

References

[1]. Jennings, N. R. (1996). Coordination techniques for distributed artificial intelligence. Artificial Intelligence, 85(1–2), 275–307.AN-Agent-Driven Framework-For-Task-Level-Sdlc-Automation.docx
[2]. Wooldridge, M., & Jennings, N. R. (1998). Pitfalls of agent-oriented development. Proceedings of the Second International Conference on Autonomous Agents. ACM.papers
[3]. van der Aalst, W. M. P., ter Hofstede, A. H. M., & Weske, M. (2003). Workflow management: Models, methods, and systems. MIT Press.linkedin
[4]. Bandara, G., et al. (2025). Agentsway - Software Development Methodology for AI Agents-based Teams. International Journal of Scientific Research in Engineering and Technology (IJSRET), 11(4), 1-15.microsoft
[5]. Yete, R., & Aakre, A. (2025). Agentic AI Systems for Software Development Automation. International Journal of Scientific Research in Engineering and Technology (IJSRET), 11(4), 16-28.ijsret
[6]. Wang, J., et al. (2025). LLM-based Multi-Agent Systems for Software Engineering: Literature Review, Vision and the Road Ahead. Singapore Management University School of Information Systems Research, SIS-2025-01.github
[7]. Khan, M. A., et al. (2025). An Agentic AI Framework for Building Reliable Retail Systems With Large Language Models. IEEE Access, 13, 10930-10951.atlassian
[8]. Lyberis, C., et al. (2016). Myrmics: Scalable, Dependency-aware Task Scheduling on Heterogeneous Manycores. Proceedings of the International Conference on Parallel Processing (ICPP), 1-10.arxiv
[9]. Multi-Agent SDLC Research Team. (2025). An Efficient Solution towards SDLC Automation using Multi-Agent Systems. International Journal of Computer Engineering and Sciences (IJCESEN), 11(2), 45-58.ijcesen
[10]. Faruqui, N., Thatoi, P., Choudhary, R., et al. (2024). AI-Analyst: A business cost optimization SDLC analysis structure assisted by AI. IEEE Access, 12, 14567-14582.forecast
[11]. Zota, R. D., Bărbulescu, C., & Constantinescu, R. (2025). An action-oriented way of establishing a framework towards creating an agentic AIOps system. Electronics, 14(3), 567-589. MDPI.arxiv
[12]. Mathews, N. S., & Nagappan, M. (2024). Test-Driven Development and LLM-based Code Generation. Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), 1234-1245.engineering.peerislands
[13]. He, L., Chen, Z., Zhang, Z., et al. (2025). Use Property-Based Testing to Bridge LLM Code Generation and Validation. OpenReview (ICLR 2025), Paper ID: L5aXjVPHmN.thedigitalprojectmanager
[14]. Silva, R., et al. (2025). Agile Software Management with Cognitive Multi-Agent Systems. Proceedings of the 20th International Conference on Information Systems Security and Privacy (ICISSP 2025), SCITEPRESS, 112-123.mckinsey
[15]. Catal, C., & Mishra, D. (2013). Test case prioritization: A systematic mapping study. Software Quality Journal, 21(3), 445-467.

International Research Journal on Advanced Engineering and Management
https://goldncloudpublications.com
https://doi.org/10.47392/IRJAEM.2026.0118
e ISSN: 2584-2854
Volume: 04 Issue: 03
March 2026
Page No: 769 - 773

Springer.techcommunity.microsoft
[16]. Yoo, S., & Harman, M. (2012). Regression testing minimization, selection, and prioritization: A survey. Software Testing, Verification and Reliability, 22(2), 67-120.workspace.google
[17]. Arrieta, A. B., et al. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion, 58, 82-115. ScienceDirect