International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(22s) 2026 pp. 590-603
https://doi.org/10.70917/ijcisim-2026-5464
Article

The AI-Driven Development Lifecycle: Replacing Over-Planning with Rapid Iteration and Agentic Execution

Subbarao Duggisetty

Independent Researcher, India

Abstract: The fast development of AI has provided opportunities to reshape the traditional approaches to software engineering beyond the automation of tasks. The Software Development Lifecycle (SDLC) practices that have existed in the past are still based on massive planning, orderly sequential execution, and decision-making processes, which lead to sluggish delivery and adaptation capabilities. This study presents the AI-Driven Development Lifecycle (AI-DLC), a new approach to software development that substitutes planning-filled processes with fast development and agentic behavior. The suggested model introduces an AI maturity spectrum that includes traditional SDLC, AI-assisted development, AI-driven development and AI-managed development models. Moreover, AI-DLC includes five-stage lifecycle that includes business intent capture, collected contextual knowledge, AI-supported planning, deployment, and continuous improvement. It suggests a multi-agent architecture, which combines design, development, and security, testing, DevOps, and mail industry-specific agents. The framework provides a human-AI operating model wherein human controls are made in strategic decisions but operational processes are run through AI. The assessment system evaluates the speed of delivery, efficiency in engineering, preparedness, software reliability and automation potential. The study is the base of the future self-governing software engineering environments.
Keywords: Artificial Intelligence, AI-Driven Development Lifecycle, Agentic AI, Multi-Agent Systems, Software Engineering Automation, Human-AI Collaboration, Autonomous Software Development

1. INTRODUCTION

Background of Traditional Software Development Lifecycle

The old Software Development lifecycle (SDLC) has been transformed to the sequential waterfall-based models to the iterative Agile and DevOps-based models to enhance flexibility, collaboration and delivery efficiency [1]. Contemporary software engineering still strongly depends on a structured planning, specification of requirements, architectural design and controlled phases of execution [2]. Agile and DevOps have enhanced flexibility and the continuous delivery of software products, but due to the human factor in decision-making and the predetermined lifecycle activities, their usefulness is limited.

Disadvantages of Conventional Agile Development

Although Agile software methodologies enhance the flexibility of software, the traditional ones come with challenges of over-documentation, over-requirement modeling, sprint planning, and numerous approval processes. Such practices embracing the practices of planning it will slow down more in such a quick changes business kind of environment and it will slow down responsiveness [3]. Comprehensive initial choices might become irrelevant as market circumstances, client anticipations, or tech necessities change, leading to greater restart, wasteful resource usage, and longer delivery timings [4].

Disadvantages of the Existing AI-Assisted Development

The existing AIs-assisted development method methods such as intelligent coding assistants and automated code generation tools have greatly increased developer productivity by speeding up programming tasks, debugging, and documentation [5]. Current solutions are still relying on the established workflows where human still have to make planning, architecture, testing, and deployment processes, curbing the possible benefits associated with AI-powered autonomous execution [6]. All these solutions are more useful to optimize single tasks of development than to transform the entire process of software engineering.

AI-Driven Development Lifecycle (AI-DLC)

AI-DLC is a process to shift in software development, replacing human-centered development with AI-driven lifecycle development. As opposed to the traditional AI support, AI-DLC is an entire approach where the intelligent agents ensure the coordination among the planning, development, verification, and ongoing improvement processes [7]. This can help to move towards achieving governance of the results instead of conducting operational work so as to maintain a strategic level of governance through accountability and quality assurance. Amazon Q Developer and Kiro are the examples of technologies which help with this new type of development. Kiro implements AI-DLC's spec-driven workflow through a three-step process Requirements, Design, and Task generation where AI agents produce structured artifacts at each stage while humans verify and approve before proceeding.

Research Aim

The study can yield an AI-Driven Development Lifecycle (AI-DLC) framework that can reshape the conventional software engineering practices with a new approach of agentic execution, quick iteration, and autonomous development managed by humans.

Research Objectives

- ***To examine constraints of traditional SDLC and Agile methodologies due to over-planning, sequential processes and late software development.***
- ***To create an AI-DLC process that incorporates agentic AI skills of autonomous planning, development, testing, deployment and continuous improvement.***
- ***To prepare a multi-agent software engineering architecture that can be used to develop intelligent co-operation between specialized AI agents throughout all phases of the development lifecycle.***
- ***To assess how effective AI-DLC was in its ability to enhance the rate of development, flexibility, software quality, and collaboration between human and AI factors in comparison to traditional development method.***

Research Contribution

The study adds a new AI-DLC framework that transforms the traditional software engineering operations with quick repetitive processes. The approach suggested in this case brings about multifaceted cooperation among agents, self-driven lifecycle management, and a model of human-AI governance. It offers a conceptual basis to those organizations that move beyond AI-assisted development to AI-driven and AI-managed software engineering practices.

2. THE AI MATURITY SPECTRUM

Level 1: SDLC without AI (Traditional)

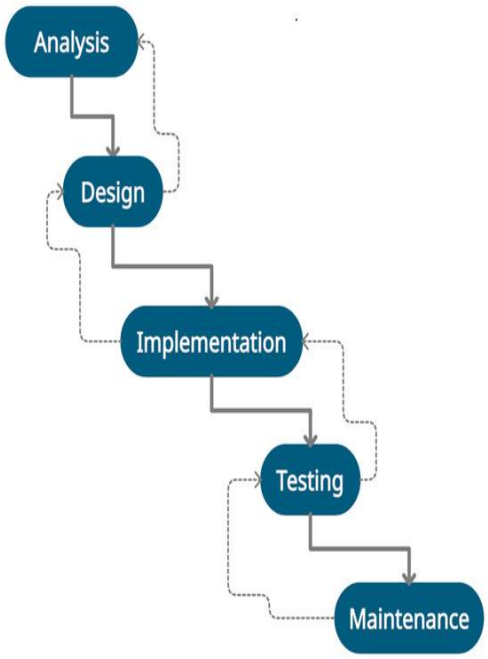


Fig. 1: Traditional SDLC

Traditional SDLC depicts a completely human-faceted approach to software engineering in which all lifecycle events are carried out manually. The development of requirements analysis, project planning, architectural choices, code, testing, and deployment requires knowledge of the developer and a series of established procedures [8]. This model uses a stepwise working process:

$$Requirement \rightarrow Design \rightarrow Development \rightarrow Testing \rightarrow Deployment$$

Key Characteristics:

- Human-only decision-making.
- Manual requirement management.
- Developer-controlled implementation.
- Limited automation capability.

Level 2: AI-Assisted development

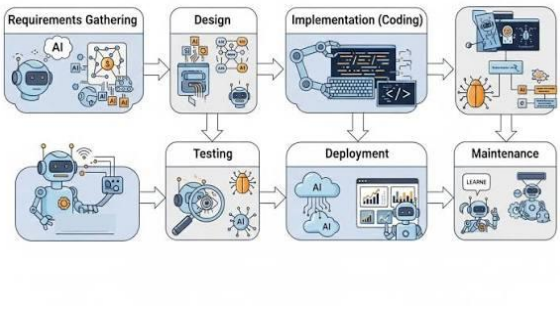


Fig. 2: AI-Assisted development

AI-assisted development proposes smart development tools that would improve both developer productivity by providing automatic suggestions and speeding up their work. It can be systems such as code completion, AI supports

in debugging, and automatic documentation generation [9]. On the other hand, the basic software lifecycle is the same since humans are still in charge of making planning, architecture, and implementation and deployment decisions.

Key Characteristics:

- AI supports developers.
- Code generation assistance.
- Automated recommendations.
- Human-controlled workflow

$$Human + AI\ Support \rightarrow Improved\ Productivity$$

Level 3: AI-Driven Development

AI-inspired evolution is one of various changes where artificial intelligence agents are involved in the management of software lifecycle. The planning, generation of architecture, implementation strategies and continuous optimization is supported by AI systems and strategies and strategic validation checkpoints are provided by humans. The adaptive and context-aware development is possible with the use of technologies like AI agents, Retrieval-Augmented Generation (RAG), and agent-based reasoning [10].

$$Business\ Intent \rightarrow AI\ agents \rightarrow Implementation \rightarrow Human\ Governance$$

Level 4: AI-Managed Development

AI-managed development is the next level of autonomous software engineering in which AI systems autonomously handle entire development processes. Design creation, design implementation, testing, deployment and optimization tasks are all carried out with intelligent agents having minimum human involvement. The main controls within the human sphere include governance, compliance with ethical standards, risk management, and end quality approval.

3. AI-DLC METHODOLOGY: FIVE PHASES

Fig. 3: Methodology Phases of AI-DLC

Phase 1: Business Intent

The initial step in AI-DLC is aimed at comprehending business goals and translating strategic goals into coordinated AI commands. This step sets the stage of the agent-driven implementation, determining outcomes required, anticipated functionality and organizational priorities [11]. The important steps are intent extraction, requirement interpretation and goal decomposition to make sure that AI systems know what is expected in business.

Aim of Business → AI Learning → Action Requirements

Phase 2: Construct and Gather Context

The second stage forms a contextual intelligence through acquiring and examining appropriate organizational knowledge. To enhance the accuracy of information retrieval in making decisions and to provide context-centered solutions, AI systems access various sources of information [12]. The phase guarantees that AI agents can know the prevailing environments, technical dependencies, as well as business constraints, prior to undertaking development actions.

Key Sources:

- Technical documentation.
- Business reports.
- Knowledge repositories.
- Existing software codebases.

$$Data + Knowledge + Context \rightarrow Intelligent\ AI\ Decisions$$

Phase 3: Define and Execute Plan

The third stage facilitates the conversion of the business intent and gathered context into the actionable development plans by AI agents. The intelligent agents conduct requirement study, architecture design, development planning and implementation generation. The phase signifies the shift in the model of traditional human-executed implementation to the option of AI-oriented software engineering where agents plan lifecycle activities.

AI Activities:

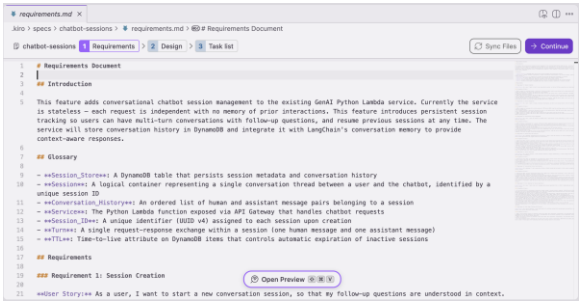
- Automated requirement analysis.
- AI-generated architecture design.
- Development strategy creation.
- Code implementation generation.

$$Intent + Context \rightarrow AI\ Planning \rightarrow Software\ Execution$$

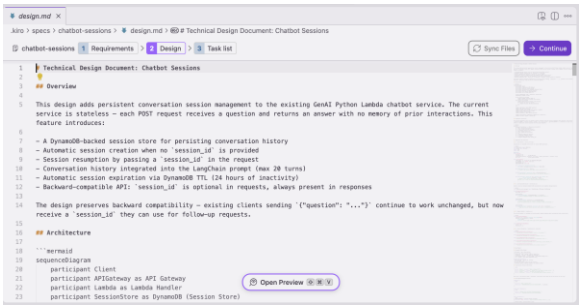
The following screenshots demonstrate AI-DLC's "Define & Execute Plan" phase using Kiro, an AI-powered IDE that implements the spec-driven development workflow. Starting from a business requirement (adding persistent session management to a GenAI Python Lambda service), the AI agent autonomously generates structured artifacts at each stage requiring only human verification at each checkpoint.

The following screenshots demonstrate AI-DLC's "Define & Execute Plan" phase using Kiro, an AI-powered IDE that implements the spec-driven development workflow. Starting from a business requirement (adding session management to a GenAI Lambda service), the AI agent autonomously generates: (a) structured requirements with glossary and user stories, (b) technical design with architecture and sequence diagrams, and (c) decomposed implementation tasks with acceptance criteria — all requiring only human verification at each checkpoint.

Requirements generation:



Design generation:



Task decomposition:

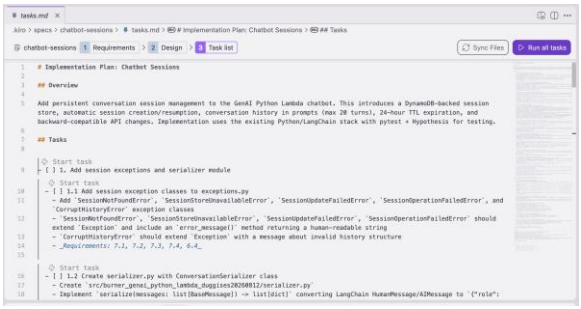


Fig. 4: AI-DLC Phase 3 in Practice — Kiro Spec-Driven Development

Phase 4: Ship to Production

The fourth stage is about automated deployment of software, through combining AI-driven development products and continuous delivery settings. AI-DLC facilitates the efficient release of production by using automated deployment pipelines, infrastructure management, and automated operations. AWS CodePipeline, AWS CodeBuild, and AWS CodeDeploy are the technologies that provide the reliability and scalability of delivering software [13].

Key Components:

- Automated deployment processes.
- CI/CD pipeline integration.
- Infrastructure automation.
- Production environment management.

$$Development\ Output \rightarrow Automated\ Deployment \rightarrow Production\ System$$

Phase 5: Continuous Improvement

The last stage ends up creating consistent optimization by tracking, feedback series as well as repetitive improvement. AI-DLC, lets software systems keep getting upgraded as they detect the performance and get improvement opportunities and also develops adaptive solutions. This step generates a feedback-based lifecycle such that AI agents constantly improve software capabilities in response to evolving business needs.

$$Feedback \rightarrow AI\ Analysis \rightarrow Continuous\ Improvement$$

AI-DLC Lifecycle Representation

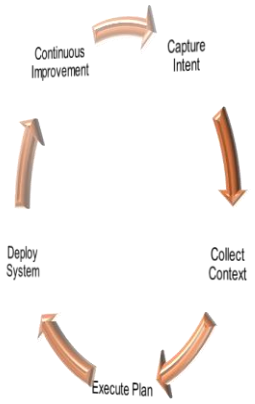


Fig. 5: AI-DLC Lifecycle

The five-step process-based approach is a way of changing software engineering to be an agent-oriented, iterative and more adaptive development paradigm, instead of a lifecycle driven by planning.

4. THE PARADIGM SHIFT: AMBIGUITY ACCEPTANCE AND RAPID ITERATION

Traditional SDLC Approach

In a traditional SDLC, development is a structured and sequential process in which altogether certainty of requirements is anticipated prior to starting the implementation process. The lifecycle moves towards the planning phase, designing, development, and testing process [14]. Nonetheless, such reliance on predetermined information inhibits experimentation, slows innovation down and it is more likely to come up with solutions that are no longer relevant to the changing business needs.

Traditional Flow:

Requirement Certainty → Planning → Design → Development → Testing

AI-DLC Approach

Traditional code-driven software development paradigms do not accommodate early uncertainty and allow refinement at any stage of the lifecycle, a feature of AI-DLC [15]. As opposed to having a solid design in place to be implemented, AI agents continuously improve solutions as they learn, analyze the context, and incorporate feedback. By doing so, organizations can flexibly adjust but also keep the software deliverables on par with business goals.

Core Principles:

Early ambiguity acceptance

Incremental solution refinement

Continuous AI-driven learning

Original Intention→ Repetitive Learning→ Better Solution

Rapid Prototyping Advantage

Rapid prototyping in AI-DLC can help organizations test concepts swiftly to ensure that a lot of money is not committed before such concepts are proven. The creation of AI-driven development facilitates experimentation, creates first software iterations, and delivers first business responses [16]. This eliminates uncertainty, enhances decision-making, and the financial and operational cost of requirements changes or wrong assumptions at late stage.

Continuous Iteration Model

The continuous iteration model takes a new turn in software development, and it may lessen the degree of upfront planning. The cost of the change is reduced by having AI agents introduce modification by analyzing, generating and optimizing the modifications. This leads to an iterative enhancement process being more successful than trying to predefine all those requirements at the start and then allows the software to evolve according to the needs of the organization.

Continuous Iteration Model:

Feedback → AI Processing → Refinement → Improved System

5. AGENTIC EXECUTION: MULTI-AGENT ARCHITECTURE

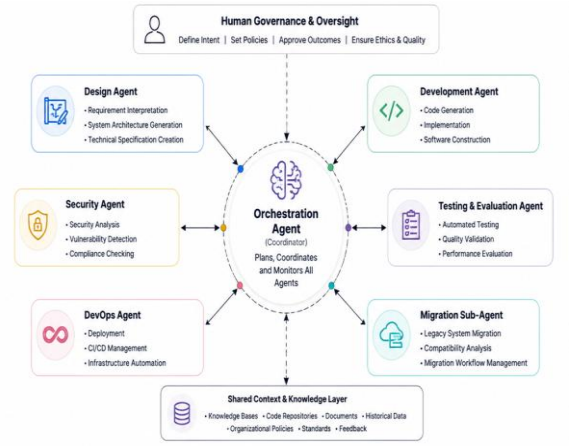


Fig. 6: Multi-Agent Architecture

Multi-Agent Architecture Overview

The multi-agent system enables an ecosystem of collaboration among specialized agents that undertake unique software engineering tasks. All the agents communicate between themselves in order to complete lifecycle operations providing domain-specific intelligence. This decentralized method enhances efficiency, flexibility, and automation through coordinated implementation of planning, development, security, and testing and deployment phases.

Design Agent

Design Agent is directed to changing the business requirement into software architecture and technical specifications [17]. It examines the functional objectives, finds system components and constructs appropriate design strategies. Integrating contextual knowledge with architectural rationale, this agent contributes to making the decision-making process more efficient and preconditions the further development and implementation processes.

Responsibilities:

Requirement interpretation

Architecture generation

Technical specification creation

Development Agent

The Development Agent conducts software construction tasks which involve transforming approved specifications into working work output. It helps in code generation, program development as well as component development but at the same time it keeps a check with functionality of specific architectural requirements [18]. This

agent saves on the time required to create software manually and shortens the delivery time in the context of the AI-driven development space through the capabilities of autonomous execution.

Responsibilities:

Automated code generation

Software implementation

Component construction

Security Agent

The Security Agent makes sure that the AI-guided software development is highly secure during the lifecycle. It evaluates security requirements, detects vulnerabilities, assesses possible risks and assists in verifying compliance [19]. The agent supports active risk management and enhances the overall reliability of the software by incorporating a security analysis at the level of its development, as opposed to its post-deployment.

Responsibilities:

Security assessment

Vulnerability identification

Compliance validation

Testing and Evaluation Agent

The Testing and Evaluation Agent is used to conduct automated software validation, which involves the creation of test scenarios, analysis of system behavior and the evaluation of performance results. It constantly checks the quality of software with a predefined set of codes and gives the feedback to improve it. Reliability is promoted since this agent allows completing the testing cycles in a shorter period and maintaining quality assurance in the process of development.

Responsibilities:

Automated test generation

Quality verification

Performance evaluation

DevOps Agent

The DevOps Agent deals with operational functions needed to ensure a successful software deployment and maintenance. It makes the release processes automated, coordinates CI/CD pipelines, and aids in configuring infrastructure [20]. This agent optimizes the efficiency of deployment, lessens the need to use people and enables continuous delivery of software across cloud-based settings by integrating both the development and the operational processes.

Responsibilities:

Deployment automation

CI/CD management

Infrastructure optimization

Migration Sub-Agent

The Migration Sub-Agent is specialized in software transformation and migration of legacy systems. It evaluates the current applications, compatibility conditions and creates migration procedures that would facilitate

smooth transition between platforms [21]. This agent enables migration planning and evaluation to be automated, which minimizes operational risks and promotes modernization of existing software systems efficiently.

Responsibilities:

Legacy application migration

Compatibility assessment

Migration workflow management

6. HUMAN-AI OPERATING MODEL: WHO OWNS WHAT

Human Responsibilities

The human elements retain their role in providing strategic direction, defining business intent, and making ultimate decisions in the AI-DLC operating model [22]. Outcomes are confirmed and critical choices made, and expert judgment is available through human validation of the outcomes.

AI Responsibilities

The AI systems handle execution of operations throughout the software development lifecycle by carrying out planning activities, implementation assistance, optimization as well as feedback processing. The intelligent agents examine the needs, create solutions, orchestrate activities, and never-stop advancing development activities [23]. It allows quicker implementation and helps to better minimize redundant human effort and enhance flexibility in software development processes.

Governance Framework

The AI-DLC system of governance forms regulated cooperation of humans and intelligent agents with the help of a system of checks and balances, confidence limits, and responsible AI practices. Human beings oversee the spheres of important choices, observe the actions of AI, and add control over the adherence to security and ethical standards [24]. The underlying principle is that human beings control strategic activities and implement operational tasks, where AI plays a role.

7. RESULTS AND ACCELERATION METRICS

Delivery Acceleration

The analysis of AI-DLC shows that software delivery time is greatly improved because it does not rely on a lot of planning and manual performance. Workflows that are agent-driven are easier to develop, can coordinate automatically and keep on refining. The framework aims at achieving compounded like results of 4x to 15x better than traditional development styles.

Delivery Acceleration Metric:

$$DA = \frac{Traditional\ SDLC\ Delivery\ Time}{AI - DLC\ Delivery\ Time}$$

Where:

DA > 1 is the sign of better delivery efficiency.

The higher the numbers, the more acceleration

Improvement of the efficiency of engineering

AI-DLC enhances efficiency in engineering, as it automatizes monotonous development tasks, allows intelligent task-sharing between specialized agents and minimizes the number of people involved. Efficiency

improvement is quantifiable by the amount of effort required to develop things and the percentage of features completed. The collaboration powered by AI will enable the engineering team to be innovation-driven, validate-driven, and strategic-driven.

Engineering Efficiency Formula:

$$EE = \frac{Feature\ Output}{Development\ Effort}$$

Where:

Increased EE translates to increased productivity.

Capabilities such as completed software are referred to as feature output.

Qualification Framework

Team Readiness

Team readiness assesses the level of AI capabilities, engineering maturity that organizations have to implement AI-DLC successfully. It is a scale of developer knowledge of AI technologies, level of organizational learning and readiness to use AI-enabling workflows. Greater preparedness reflects enhanced ability to work cooperatively with intelligent agents along the software lifecycle.

Team Readiness Index:

$$TRI = w1(AI\ Skill) + w2(Engineering\ Maturity) + w3(Collaboration)$$

Technical Readiness

Technical preparedness will define the presence of the necessary infrastructure, computational facilities, data availability, and integration facilities in the organization to develop AI-driven activities. Effective AI-DLC requires effective technical settings that facilitate execution of the agents, knowledge retrieval, automation pipelines and continuous software delivery.

Technical Readiness Index:

$$TechRI = w1(Infrastructure) + w2(Data\ Availability) + w3(System\ Integration)$$

Business Case Clarity

The clarity of business cases makes the adoption of AI-DLC based on a clear set of goals, quantifiable results, and strategic priorities. Before implementing an organization needs to define certain development challenges, anticipated gains, and the success criteria. Clear goals enhance decision-making in AI agents and guarantee a match-to-business business needs with the capabilities of the technology.

Business Alignment Score:

$$BAS = \frac{Defined\ Objectives}{Required\ Business\ Goals}$$

Organisational Alignment

Organisational alignment estimates how prepared an enterprise is to embrace AI-based change, in terms of cultural acceptance, leadership support and adaptation of operations. This can only be adopted effectively through working in collaboration with the technical teams, business stakeholders and governance groups. Near-perfect fit guarantees that AI agents will be sustainable in the established software engineering practice.

Adoption Readiness Formula:

$$ARI = w1(Leadership) + w2(Culture) + w3(Process\ Adaptability)$$

ADDL Practical Implementation Model

The actual execution of AI-DLC is done using a stet-by-step adoption plan to bring about a preliminary workshop to map out appropriate use cases. An initial trial confirms AI functionality, workflow integration and performance gains. Effective pilots facilitate enterprise-level implementation via widened capability of agents, organizational training and ongoing optimization.

Table 1: AI-DLC Evaluation Matrix

Evaluation Dimension	Evaluation Criteria	Measurement Parameters	Expected Outcome
Delivery Acceleration	Measures improvement in software delivery speed.	Development time, frequency of release, lifetime.	Quick service delivery with shortened development time.
Engineering Efficiency	Measures productivity of the developer.	Characteristic completion, intensity of resource use.	More efficient work with the aid of AI.
Planning Effectiveness	Evaluates the ability of AI to produce adaptive plans.	Velocity Requirement analysis Accuracy in planning.	Less reliance on high up-front planning.
Team Readiness	Appraises the organizational capability to adopt AI.	IT expertise, engineering experience, teamwork.	Better human-AI synergies and uptake.
Technical Readiness	Assesses the infrastructure appropriateness to AI-DLC.	Availability of data, computing resources, system integration.	Stable platform to execute AI agents.
Business Alignment	Measures relate AI implementation with organizational endpoints.	Clarity of objectives, anticipated results, strategic priorities.	Increased harmony among technology, business needs.
Software Quality	Measures improvements in reliability and performance.	Reduction of the defect rates, effectiveness of testing, security validation.	Higher quality of software with constant AI analysis.
Automation Capability	Measures AI agent automated lifecycle activities.	Automation of planning, coding, testing and deployment.	Enhanced independent software engineering.

8. CONCLUSION AND FUTURE DIRECTIONS

The study introduces AI-DLC as a radical software engineering procedure that relocates the development focus of a human execution to an agentic and adaptive lifecycle overall management. Offering a combination of multi-agent collaboration, continuous iteration, and human governance, AI-DLC overcomes the drawbacks of classic SDLC and provides a basis of faster, larger-scale, and clever software development practices.

Future Research Directions

Further studies to prove AI-DLC by the large scale implementation in the industrial sector and its empirical proof against current development methodologies are required in future. Future research is possible on autonomous agent coordination, ethical governance, self-improving software systems, and intelligent AI-supervised development

environments to increase reliability, transparency, security, and adoption of autonomous software engineering systems.

REFERENCES

1. Yas, Q.M., ALazzawi, A. and Rahmatullah, B., 2023. A comprehensive review of software development life cycle methodologies: Pros, cons, and future directions. *Iraqi Journal for Computer Science and Mathematics*, *4*(4), p.14. https://ijcsm.researchcommons.org/cgi/viewcontent.cgi?article=1114&context=ijcsm
2. Gurung, G., Shah, R. and Jaiswal, D.P., 2020. Software development life cycle models-A comparative study. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *6*(4), pp.30-37. https://www.academia.edu/download/65274994/CSEIT206410.pdf
3. Gumiński, A., Dohn, K. and Oloyede, E., 2023. Advantages and disadvantages of traditional and agile methods in software development projects-case study. *Organization Manag Ser*. http://dx.doi.org/10.29119/1641-3466.2023.188.11
4. Al-Ashmoery, Y., Nasser, N., Chaabi, Y., Haider, H., Haider, A. and Alwesabi, K., 2023. A systematic study on Traditional software development models and Agile Software Development Methodologies: Yahya Al-Ashmoery, Najran Nasser, Youness Chaabi, Hisham Haider, Khaled Alwesabi, and Adnan Haider. *AlRazi University Journal for Computer Sciences and Technology*, *1*(1), pp.1-13. http://rujms.alraziuni.edu.ye/index.php/RUJCST/article/download/198/200
5. Borger, J.G., Ng, A.P., Anderton, H., Ashdown, G.W., Auld, M., Blewitt, M.E., Brown, D.V., Call, M.J., Collins, P., Freytag, S. and Harrison, L.C., 2023. Artificial intelligence takes center stage: exploring the capabilities and implications of ChatGPT and other AI-assisted technologies in scientific research and education. *Immunology and cell biology*, *101*(10), pp.923-935. https://onlinelibrary.wiley.com/doi/pdfdirect/10.1111/imcb.12689
6. Semerikov, S.O., Striuk, A.M. and Shalatska, H.M., 2021. AI-assisted language education: critical review. *Educational Dimension*, *4*, pp.1-7. https://journal.kdpu.edu.ua/ed/article/download/8213/7760
7. Soni, A., Kumar, A., Arora, R. and Garine, R., 2023. Integrating AI into the software development life cycle: best practices, tools, and impact analysis. *Tools, and Impact Analysis (June 10, 2023)*. https://papers.ssrn.com/sol3/Delivery.cfm?abstractid=4918992
8. Umeugo, W., Lowrey, K. and Pandya, S., 2023. Effect of SDLC Models on The Perception of SSDLC Innovation Characteristics and SSDLC Adoption Intention. *International Journal of Security (IJS)*, *14*(1), pp.1-16. http://mail.cscjournals.org/manuscript/Journals/IJS/Volume14/Issue1/IJS-166.pdf
9. Bailey, L., 2024. The Impact of AI on Software Development. *PhD diss., Doctoral dissertation, Worcester Polytechnic Institute*. https://digital.wpi.edu/downloads/qj72pc43b
10. Singh, A., Ehtesham, A., Kumar, S., Khoei, T.T. and Vasilakos, A.V., 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*. https://arxiv.org/pdf/2501.09136
11. Cepeda-Zapata, K.A., Loughran, R., Ward, T. and McCaffery, F., 2025, August. A Review of AI Life Cycle-Related Standards to Address AI-Enabled Medical Device Development. In *European Conference on Software Process Improvement* (pp. 289-307). Cham: Springer Nature Switzerland. https://eprints.dkit.ie/id/eprint/982/1/%28PrePrint%29%20A%20Review%20of%20AI%20Life%20Cycle-related%20Standards%20to%20Address%20AIeMD.pdf
12. Aluri, Y.S., 2025. Agentic AI Frameworks for Autonomous Enterprise Software Development Workflows. *International Journal of AI, BigData, Computational and Management Studies*, *6*(1), pp.217-226. https://ijaibdcms.org/index.php/ijaibdcms/article/download/574/563
13. Bagai, R., Masrani, A., Ranjan, P. and Najana, M., 2024. Implementing continuous integration and deployment (CI/CD) for machine learning models on AWS. *International Journal of Global Innovations and Solutions (IJGIS)*. https://ijgis.pubpub.org/pub/y9higdqa/release/1
14. Teja Swaroop, A., 2025. The Transformative Impact Of Artificial Intelligence On Professional Software Development: A Comprehensive Analysis. *Title of Paper: The Transformative Impact Of Artificial Intelligence On Professional Software Development: A Comprehensive Analysis published in Page No*, *13*(8), pp.b74-b90. https://papers.ssrn.com/sol3/Delivery.cfm?abstractid=5384403
15. Mandulapalli, S., Hernandez, E., Hall, W.J., Chakeri, A. and Jaimes, L., 2025, June. Development of Agentic Workflows with LangGraph for Software Development Life Cycle Automation. In *North American Conference on Industrial Engineering and Operations Management-Computer Science Tracks* (pp. 45-54). Cham: Springer Nature Switzerland. https://ieomsociety.org/proceedings/orlando2025/216.pdf
16. Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Yau, S., Lin, Z., Zhou, L. and Ran, C., 2024, May. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations* (Vol. 2024, pp. 23247-23275). https://proceedings.iclr.cc/paper_files/paper/2024/file/6507b115562bb0a305f1958ccc87355a-Paper-Conference.pdf
17. Shen, M., Li, Y., Chen, L., Fan, Z., Li, Y. and Yang, Q., 2025. From mind to machine: The rise of manus ai as a fully autonomous digital agent. *arXiv preprint arXiv:2505.02024*. https://arxiv.org/pdf/2505.02024?
18. Canese, L., Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M. and Spanò, S., 2021. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, *11*(11), p.4948. https://www.mdpi.com/2076-3417/11/11/4948

19. Gronauer, S. and Diepold, K., 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, *55*(2), pp.895-943. https://link.springer.com/content/pdf/10.1007/s10462-021-09996-w.pdf
20. Kale, P., 2024. A Multi-Agent AI Framework for Distributed DevOps Automation and Collaborative Decision-Making in Software Pipelines. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *5*(1), pp.274-282. http://ijaidsml.org/index.php/ijaidsml/article/download/601/563
21. Wang, Y., Guo, S., Pan, Y., Su, Z., Chen, F., Luan, T.H., Li, P., Kang, J. and Niyato, D., 2025. Internet of agents: Fundamentals, applications, and challenges. *IEEE Transactions on Cognitive Communications and Networking*. https://arxiv.org/pdf/2505.07176
22. Shneiderman, B., 2020. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *10*(4), pp.1-31. https://dl.acm.org/doi/pdf/10.1145/3419764
23. Katangoori, S. and Katangoori, A., 2023. Intelligent ETL Orchestration With Reinforcement Learning and Bayesian Optimization. *International Journal of Emerging Research in Engineering and Technology*, *4*(4), pp.208-219. https://ijeret.org/index.php/ijeret/article/download/649/624
24. Müller, V.C., 2020. Ethics of artificial intelligence and robotics. https://plato.stanford.edu/ENTRiES/ethics-ai/
25. Kiro IDE, "Spec-driven development with AI agents," Amazon Web Services, 2025. [Online]. Available: https://kiro.dev
26. AWS Labs, "AI-DLC Workflows: Adaptive workflow steering rules for AI coding agents," GitHub, 2026. [Online]. Available: https://github.com/awslabs/aidlc-workflows