International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.2 (2025) pp. 3207-3214
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

An Efficient Solution towards SDLC Automation using Multi-Agent Integration through Crew AI

Nikhil Sagar Miriyala[1*], Bharath Kumar Bandaru[2], Prakhar Mittal[3], Kiran Babu Macha[4], Rishi Venkat[5], Anu Rai[6]

[1]Oracle America Inc., USA
*** Corresponding Author Email:** nmiriya7@gmail.com - **ORCID:** 0009-0005-1985-7543

[2]Fidelity Investments Inc., USA
Email: bandarubk7@gmail.com - **ORCID:** 0009-0003-1151-7961

[3]Principal Analyst, Atricure, USA
Email: Kiranbabu.macha@aol.com - **ORCID:** 0009-0009-0960-0612

[4]Software Engineering, Maximus Inc., USA
Email: Imprakharmittal@gmail.com - **ORCID:** 0009-0004-3278-9381

[5]Principal Product Manager, Walmart Inc., USA
Email: Rishi.x.venkat@gmail.com - **ORCID:** 0009-0001-0155-2929

[6]Technical Product Manager, USA
Email: anurai483@gmail.com - **ORCID:** 0009-0004-5572-3852

Article Info:

DOI: 10.22399/ijcesen.2384
Received : 22 February 2025
Accepted : 12 May 2025

Keywords :

SDLC Automation,
GenAI Tools,
CrewAI,
Multi-Agent Systems,
AI-Orchestrated Integration,
Workflow Engines.

Abstract:

Software Development Life Cycle (SDLC) is the fundamental concept which underlies the systematic development of a software product. It consists of requirements gathering, design of the application, development, testing, deployment & maintenance phases. Due to the increase in software complexity and the demand for fast delivery and continuous improvements, automation has become a key player in developing high-quality applications and maintaining them efficiently. On the other hand, with immense and continuous advancements happening in the Generative AI space, GenAI tools are emerging to be a powerful asset to automate repetitive tasks, enhance accuracy, and accelerate processes. Combining the above, this paper describes the use of GenAI tools in automating different phases of SDLC, the issues and shortcomings associated with the traditional automation approach by depending on a single tool and emphasize the need for using specialized tools for effectively handling various phases of SDLC. We have explored and analyzed a tool named CrewAI, a robust orchestration framework, which has the capability of integrating several AI-powered tools across the different phases of SDLC, while ensuring smooth transition between all the SDLC life cycle phases, maintaining flexibility and efficiency. We have proposed a high-level design towards SDLC automation using CrewAI, along with the challenges that arise with respect to regulatory compliance, data security, and ethical considerations when using GenAI.

1. Introduction

1.1 Overview of SDLC and Its Phases:

The structured process of Software Development Life Cycle (SDLC) is a critical workflow for successful delivery of software products. This process ensures that the software aligns with the business needs while maintaining high quality application and is adhered to the stipulated timelines. The SDLC includes the following phases [1]:

1) Requirement Gathering and Analysis: The first phase of SDLC involves gathering requirements from various stakeholders, clients, customers, business analysts and so on. The information collected forms the building blocks for the software application. The requirements between

the various parties and developers are fulfilled by utilizing the Software Requirement Specification (SRS) documentation. This ensures that everyone has a clear understanding of project expectations.

2) Design: In this phase, the software design and architecture are created. It also includes data flow in the software, interaction between various models, & other system integrations. It typically has two steps, High-level Design (HLD) & Low-level Design (LLD). HLD defines the architecture while LLD defines the workings of each feature and component in the software.

3) Development: In this phase, the design is implemented using code. Developers in organizations typically follow standard development guidelines to ensure consistency and quality. Developers ensure that all the functional requirements & technical specifications are aligned with the planned design.

4) Testing: In this phase, the software is tested rigorously to ensure the software executes smoothly. This phase also aims to identify probable flaws, fix them & retest the application. This ensures that the final product is free from defects.

5) Deployment: In this phase, the product is released in phases as per the organization's strategy. This phase involves configuring servers, ensuring security, and preparing the infrastructure to run the application. Finally, the application is released in the production environment and made available to end users.

6) Maintenance: In this phase, the software enters the maintenance phase. Regular updates are applied to keep the software up to date and secure. Issue that arises are addressed promptly and resolved. Enhancements are made based on the user feedback to continuously improve the user experience.

Although these phases are typically handled by separate teams, an overarching goal is to streamline them to reduce time, cost, and human error. By automating in the SDLC, the aim is to reduce manual intervention and accelerate delivery [2].

1.2 The Role of GenAI Tools in Automating the SDLC:

With the rise of Artificial Intelligence, especially Generative AI (GenAI), significant improvements have been made in automating various aspects of software development. GenAI powered tools excel at automating tasks which require human effort.

GenAI tools offer numerous advantages across the stages of SDLC by automating routine tasks, improving productivity and efficiency [3]. The automation includes coding snippets, bug detection, and testing phases, allowing developers to optimize resource management. GenAI tools offer data-driven insights, which allows strategic decisions making, enabling more accurate project estimations and improving software quality. It also helps in optimizing deployment strategies & detect issues during maintenance through automation. These advantages position GenAI tools as a crucial component of modern software development.

Tools like GitHub Copilot use AI models to suggest code completions and generate snippets, reducing the manual coding efforts. During the testing phase, tools like Selenium and Testim automate the execution of test cases, ensuing that software is tested thoroughly. These tools also provide comprehensive test coverage & identify the issues to provide continuous improvement, ensuring that software is more robust and reliable.

Additionally, AI-driven tools that help in requirement gathering, use Natural Language Processing (NLP) models, to analyze user requirements and extract relevant functional and technical requirements. Another tool called Figma, used in design phase, lets designers work in real-time, and makes it easier to create, test and modify UI/UX designs with suggestions from machine learning models.

1.3 Challenges with One-Size-Fits-All Tools

GenAI tools have evolved in such a way that they can solve certain type of problem but not all. So, one-size-fits-all solution for automating the SDLC is not practical. This is because of the unique challenges and requirements that are present in each phase of the SDLC [4]. One tool designed to solve one phase may not solve the other phase efficiently. The crux of the challenge lies in the integration of these AI tools ensuring they operate seamlessly across multiple phases of the SDLC.

To facilitate the seamless integration of multiple GenAI tools across the SDLC phases, a multi-agent platform is required. This is where we will discuss a tool called CrewAI [5], which orchestrates and coordinate these tools effectively. Prior to exploring the CrewAI abilities we will explore the optimal GenAI tools required for each phase in the next section.

2. Best-Suited GenAI Tools for Each Phase

2.1 Requirements Gathering and Analysis:

The requirements gathering and analysis serves as the foundational phase, where the requirements for software design and development are defined. It is important to ensure the requirements in order to meet the final expectations of the stakeholders &

users. A significant portion of this stage involves manual work that heavily relies on direct interactions via meetings, surveys, questionnaires and so on.

GenAI tools can be extensively used in this phase and the reduce the amount of manual work involved in analyzing and understanding the business and functional requirements [6]. Essential information is extracted using Natural Language Processing (NLP) techniques by processing data from documentation systems and communication channels. Generative models enable the system developers to begin the requirement drafting process, ensuring accurate documentation. They also enable the project team to create a visualization of the requirements & enable live simulation access while allowing for prototype testing [7]. Two well-known tools that can greatly optimize & automate this phase are considered below.

1) IBM Watson Assistant: IBM Watson Assistant AI tool offers powerful capabilities that uses natural language to understand and interact with stakeholders, ask to questions & extract relevant information. It also organizes the data into structured formats, streamlining analysis and processing.

2) ThoughtWorks Insights: ThoughtWorks Insights uses AI driven analysis on historical data to predict risk and potential blockers during the requirement gathering phase. By recognizing patterns across the projects, it provides valuable insights into critical factors for consideration.

2.2 Design:

In the design phase, the architecture and the user interface design are developed based on the requirements that are gathered and inter relationships that were defined. These designs visualize the system interactions between the functional requirements. Ideally, this involves human expertise to perform iterative adjustments to refine the design.

GenAI tools offer advanced capabilities to enhance architecture in the design phase. They can automate the architectural designs, propose optimal system design patterns, and simulate potential issues like performance bottlenecks. By enabling predictive design and early risk identification, they improve the scalability and security of the application. Furthermore, they streamline diagramming and documentation, ensuring consistency and efficiency. The tools listed below can be used for automating this phase.

1) Tools based on LLM like ChatGPT can contribute to the design and architecture of the system design by providing suggestions, automating the documentation, and recommending best practices based on the project requirements. It also helps in streamlining the process of creating and refining the architectural decisions.

2) A visual collaboration tools, Lucidchart, enables teams to create detailed architecture diagrams. With the AI-driven templates and auto-layout functionalities, it simplifies the design and visualization of complex system architectures, improving both collaboration and clarity.

2.3 Implementation:

In the development phase of the SDLC, the developers write the code for the software translating the functional requirements & design into a working application. In this phase, the developers start with setting up the environment, and then writing code based on the design. This phase is iterative, consists of code reviews, debugging issues, and testing. Developers collaborate with front-end and back-end components, integrate various systems and libraries to meet project's technical and business specifications. The aim in this phase is to produce a software which consist of functional requirements and non-functional requirements, such as scalability, maintainability, etc., for subsequent testing and deployment.

GenAI tool elevates the process of development by playing a key role in maintaining the superior code quality and streamlining the coding tasks. Developers benefit from the real-time assistance of these tools in code completion suggestions, function generations, and even recognizing potential bugs or inefficiencies. Developers can use automation to generate code for repetitive tasks, for example in generating boiler plate code, and optimization of complex processes [8]. The tools listed below can be used in development phase.

1) GitHub Copilot, driven by OpenAI's GPT-3 model, assists developers by suggesting lines of code, generating an entire function, and providing documentation. By learning from the existing codebase, it significantly improves developer productivity.

2) Codex, AI tool from OpenAI, extends GitHub Copilot’s capability and generates full code snippets based on the description of the problem. This tool excels in understanding complex requirements and provides an optimized code.

2.4 Testing:

In the testing phase of SDLC, the objective is to deliver a robust, error free product that is ready for

deployment. Testers design a comprehensive suite of tests including unit tests, integration tests, and system tests. These tests aim to identify the defects and validate the functionality of the software application. Testers meticulously execute the tests that cover positive, negative & out of bound scenarios to uncover bugs, identify performance issues, and security vulnerabilities. This is an iterative process, involving testing, identification of bugs, resolution of bugs and retesting, to ensure the software's reliability and performance across various environments.

The testing phase can be improved significantly through automation by adoption of GenAI tools, which substantially reduces the time and manual effort required to execute the test suites and identify bugs. These tools help in generating automated test cases, performing regression testing, and identifying potential issues proactively. Testing tools that leverage AI can adapt to changes in the application's UI or code, ensuring tests remain up to date without requiring manual intervention [9]. Below are two commonly used tools in the testing phase:

1) Selenium, an open-source tool, facilitates the automation of web browser interactions, enabling testers to conduct functional and regression testing across various browsers and platforms. Selenium scripts adapt to changes in the application's interfaces and ensure that web applications function as expected.

2) Testim employs AI to automate the generation and maintenance of automated test suites, keeping the test scripts up to date by detecting changes made in the UI. This simplifies the test creation process and reduces the maintenance effort, allowing teams to focus on more complex testing scenarios.

2.5 Deployment:

The Deployment phase begins when the software application passes all tests in the testing phase and is ready for users. This phase involves the configuration of the environment, including servers, databases, and networks, to ensure the application functions properly in the production environment. Processes such as data migration, security audits, and performance tuning may also be executed as required. Using Continuous Integration (CI) and Continuous Deployment (CD) pipelines, the deployment process is often automated to provide smooth, efficient, and frequent releases, with the aim of minimizing downtime and disruptions to the user and providing a stable version of the application.

GenAI tools can mitigate the challenges that occur in software deployment by performing automatic configuration and handling environment installation tasks. These technologies help in identifying and resolving deployment issues, ensuring seamless system-wide execution. AI-powered monitoring tools implement pre-defined corrective measures automatically upon detection of abnormal patterns during deployment, thereby reducing user disruptions and decreasing system outages.

GenAI tools are being employed in the deployment phase to automate and optimize deployment pipelines, thereby ensuring faster and reliable releases. These tools help in the automation of environment setup, configuration, and release management, minimizing the manual effort required for deployment tasks [10]. Through the integration of existing CI/CD pipelines, GenAI tools enable teams to achieve continuous delivery of updates to production, ensuring application stability and security. Below are two tools commonly used in the deployment phase.

1) Jenkins, an open-source automation platform, enables continuous integration and continuous deployment. It automates the stages of the deployment process, including build, test, and deployment, allowing teams to deploy code changes quickly and reliably across environments.

2) Docker platform uses containerization and encapsulates the software and its dependencies into a single, portable container. This facilitates consistent deployment across various environments, thereby minimizing the risk associated with environment discrepancies.

2.6 Maintenance:

The maintenance phase involves continuous monitoring of application, resolution of the bugs, resolution of bugs, providing updates and enhancements to maintain software functionality, security and alignment with evolving user needs. Activities such as adding new features, optimizing performance, and ensuring compatibility with newer technologies are integral part of this phase. This is ongoing process throughout the software's lifecycle, ensuring the system reliability and effectiveness. The aim is to address the issues efficiently, maintain system stability and fulfill user expectations for continuous enhancement.

GenAI tools significantly refine the maintenance phase by automating monitoring tasks and suggesting actionable insights through advanced system performance analysis. While traditional tools like Grafana and Datadog are used for metrics and identify issues via logs, this process is often time-consuming and prone to errors. The

integration of GenAI enables the platform to detect anomalies automatically, trigger alerts, and prioritize issues based on real-time data analysis. This integration minimizes manual intervention, generates solutions, and helps teams to swiftly address system health concerns to improve software stability. Below are two tools commonly used in the maintenance phase:

1) New Relic is an AI powered monitoring tool that provides real-time tracking of application and infrastructure health. This platform helps developers and operations team to actively detect performance bottlenecks, resource consumption issues, and other anomalies and enable quick resolution and continuous optimization.

2) Sentry is another popular tool for error tracking that enables developers to quickly identify and resolve issues in real time. It automatically captures errors, exceptions, and crashes, providing detailed, actionable insights into the root cause of the issue. Sentry's AI-driven analysis helps prioritize issues based on their impact on the user experience, enabling development teams to focus on the high-priority problems first. It integrates well with both Grafana and Datadog, allowing teams to correlate error data with metrics for more comprehensive monitoring.

3. Using CrewAI for Integration

3.1 What is CrewAI?

CrewAI is an advanced, open-source orchestration framework that aims to integrate various AI agents into a single, automated workflow. It allows organizations to streamline complex workflows by enabling different AI agents to work together and execute tasks throughout the different stages of a process that. The fundamental idea behind CrewAI is the use of a multi-agent system, in which each agent is responsible for a particular task within the workflow. These agents operate independently, are intelligent, and can execute tasks based on established instructions, data inputs, or pre-defined triggers from other agents [11].

3.2 Core Features

1) Agent Specialization: Each agent in CrewAI is specifically designed to handle a particular task within the workflow, enhancing efficiency through specialized abilities tailored to each task.

2) Autonomy and Collaboration: Although each agent can function independently, CrewAI can enable collaboration, allowing for data sharing and triggering actions across phases without manual input.

3) Centralized Control and Monitoring: CrewAI provides a central control panel for monitoring agent activity, tracking progress, and ensuring smooth workflows across multiple agents.

4) Task Coordination and Dependency Management: CrewAI manages dependencies between tasks, ensuring that agents perform their duties in the correct order and trigger subsequent actions based on task completion.

5) Intelligent Decision Making: CrewAI uses AI to make decisions and adjust workflows based on real-time data, ensuring the system adapts to emerging issues or changes.

6) Scalability: CrewAI can scale to accommodate larger projects or additional agents, maintaining performance and reliability even as the SDLC grows in complexity.

3.3 The Process of Multi-Agent Integration in CrewAI

1) Configuration and Setup: The setup of a workflow on CrewAI involves choosing the appropriate and efficient agent for each task and defining the instructions, triggers and conditions for each of them to form an automated process.

2) Inter-Agent Communication: Agents use a standardized protocol to ensure seamless data exchange and transitions between tasks within the workflow.

3) Task Execution: Each agent independently carries out its designated task according to established instructions, triggering the next agent once its task is complete.

4) Orchestration and Workflow Management: CrewAI's orchestration layer is used for optimizing task order, identifies bottlenecks, and oversees agent coordination to ensure efficient workflow operations.

5) Feedback and Adaptation: CrewAI extracts feedback from agents, analyzing the information to modify workflows or task allocations as necessary, enabling the system to adapt to changes and improve over time.

3.4 System Design Incorporating CrewAI

Figure 1 shows a high-level system design for SDLC Automation using CrewAI. Before reviewing the design, below are some of the assumptions made with respect to the enterprise applications that could be used throughout the automation process:

1) JiraAlign is a tool used by product teams to provide a detailed description of the product or feature to be delivered.

2) Outlook and Slack are generally used for communication between product and engineering teams, and the data retrieved from these

applications can provide important clarifications in addition to the description retrieved from Jira Align.

3) Existing source code from Bitbucket can be used for understanding the coding principles adhered to by the organization and for any boilerplate code that can used by the application to be developed.

4) Confluence is a place for maintaining all the internal documentation, which can be extensively helpful in understanding the design principles adhered to, the testing technologies and any such information that can be utilized in every phase of the SDLC process.

5) Jira for creating tasks if needed and for analyzing any open issues that could affect the delivery of the application to be developed.

6) IntelliJ IDE for plugging in the GitHub Copilot Agent and developing the application code

7) Selenium for creating and running tests for validation.

8) Grafana, Datadog and OpenSearch for integration with Sentry agent for monitoring and maintenance purposes.

9) Jenkins, Ansible, Terraform and Docker, for creating infrastructure and building and deploying the application.

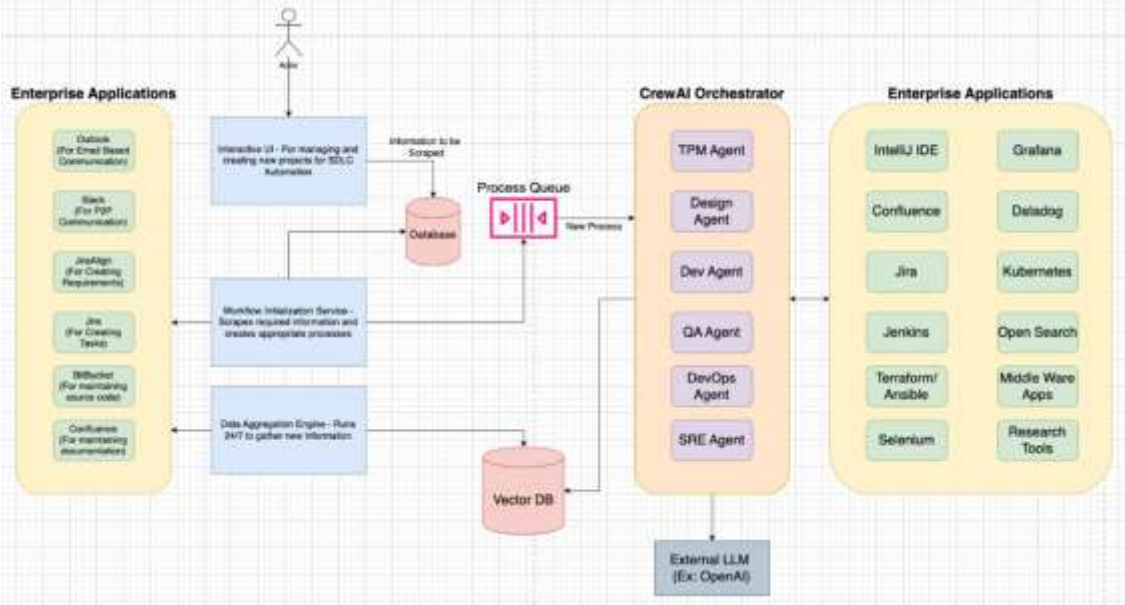

Figure 1. *High Level Architecture for SDLC Automation using Crew AI*

Here are the high-level steps involved in SDLC automation using the above architecture.

1) User Interface (UI):

- The User Interface acts as an entry point of the system where users can initiate a new project or feature for automation.
- It captures essential project details like tags, descriptions, and relevant requirements, which are crucial for creating processes or workflows.
- The UI acts as the control center for users to provide input and define automation parameters, such as what information to scrape, the scope of tasks, or additional manual tasks.

2) Workflow Initialization Service:

- The Workflow Initialization Service is responsible for orchestrating the initial phase of task creation by taking the data provided by the UI (tags, project details, etc.) and automatically fetching the necessary information from the enterprise applications (e.g., Outlook, Slack, Jira Align).
- The service is responsible for creating new workflows or processes based on the fetched data, which are then add them to the Process Queue for processing by the CrewAI Orchestrator

3) Data Aggregation Engine:

- The Data Aggregation Engine is responsible for continuously learning from internal sources like Jira, Bitbucket, and Confluence. It runs 24/7, gathering valuable insights and feedback from the historical data of ongoing and completed projects.
- The learnings from these tools are aggregated in a Vector Database (Vector DB), allowing the system to continuously update itself with new principles, patterns, and methodologies used across different projects.

4) CrewAI Orchestrator:

- The CrewAI Orchestrator serves as the central controller for managing the workflow and task execution. It monitors and processes the new workflows created by the Workflow Initialization Service and distributes tasks to the appropriate agents for each SDLC phase.
- Once the new processes or workflows are added to the Process Queue, the CrewAI

Orchestrator kicks off the agents to perform specific tasks, each based on the project requirements and the learning acquired by the system.

- Each agent (e.g., Design Agent, Dev Agent, QA Agent, TPM Agent) uses the information stored in the Vector DB to align their actions with existing coding standards, project goals, and best practices learned from Bitbucket, Confluence, and Jira.
- The orchestrator ensures that these agents run in the correct sequence, depending on the dependencies between tasks (e.g., design needs to happen before development, and development before QA). It ensures efficient handoffs and task execution.

5) External Large Language Model (LLM) Integration:

- While the Vector DB stores historical and organizational knowledge, the LLM can access external data sources and learnings, allowing the agents to tap into real-time trends, best practices, or research outside the organization's historical context.
- Enhanced Decision Making: For tasks that require deep contextual understanding or creative solutions (e.g., code generation, document generation, design suggestions), the LLM enhances the agents' capabilities by providing insights or performing specific tasks that go beyond the internal data available in the Vector DB.

6) Enterprise Tools Integration:

- The agents integrate with appropriate tools (eg: Grafana by SRE Agent) to execute the tasks assigned to them.

4. Challenges and Considerations

1) Data Privacy and Security Concerns: The usage of GenAI tools to automate the development and maintenance of software can introduce potential risks and concerns with respect to data privacy and security, especially when the project involves processing of sensitive information, PII data like account numbers for example. Thus, it is important to make sure that the agents being used can incorporate strict access controls, encrypt data and ensure compliance with privacy regulations (such as GDPR), in order to protect data from unauthorized access [12].

2) Ensuring Model Interpretability and Transparency: The GenAI models that the agents use mostly operate as "black boxes", which makes it difficult to understand why certain decisions are taken. Thus, in order to gain trust in the system, it is essential to incorporate necessary explainability features and allow developers to analyze AI-generated outputs. "A system that includes human oversight can help ensure that AI-generated decisions are justifiable" [13].

3) Continuous Learning and Adaptation: To ensure that the GenAI models being used remain aligned with changing project needs, new frameworks and libraries, and evolving industry practices, they must continuously learn from new information. This requires implementation of strong feedback mechanisms within the system, that would enable it to upgrade its models and modify their behavior over time based on real-world insights.

4) Resource Consumption and Scalability: Certain GenAI tools that would require a large language model in the background, can end up being resource heavy. In such scenarios, it is essential that the infrastructure behind these tools is scalable and flexible to handle varying workloads. In addition, continuously monitoring and analyzing the resource consumption is helpful, in order to maintain cost efficiency.

5) Human Supervision and Accountability: While AI tools can help with complete automation, there might be scenarios where human intervention is necessary, particularly to verify and approve critical decisions such as design choices, code integrations, or deployment strategies, to make sure that they align with business objectives, preserving accountability and oversight [14].

5. Future Trends and Improvements

The integration of AutoML within SDLC automation flow will further accelerate this transformation, minimizing the challenges with respect to model selection and tuning [15]. In addition, we can incorporate efficient Predictive AI systems to our design, that would allow organizations to make informed, data-driven decisions tied to resource allocation, risk management, and software scalability, resulting in more flexible and optimized development approaches. As GenAI technologies continue to improve, their use in automating the software development life cycle (SDLC) is set to transform the process of software creation and maintenance. In addition, Quantum computing is expected to significantly enhance GenAI's capabilities by offering extraordinary computational power, thereby overcoming current limitations.

6. Conclusion

In this paper, we have discussed an efficient solution towards automating the Software Development Life Cycle (SDLC) using CrewAI, a cutting-edge multi-agent integration framework. By combining and coordinating top-tier GenAI tools

that can efficiently handle each phase of software development and maintenance activities, CrewAI serves as an efficient tool towards SDLC automation, while maintaining flexibility and seamless transitions between stages. In addition, CrewAI framework helps with continuous adaptation with respect to evolving project requirements, by providing real-time feedback, leading to continuous enhancements. Further, CrewAI's ability to scale-in or scale-out as needed, makes it applicable for both straightforward and complex projects, ensuring efficient automation of each phase. While the issues with respect to data privacy, security, and model interpretability still exist, CrewAI is designed to allow transparency and human supervision, enabling informed decision making as needed. And, as GenAI technologies continue to improve over time, CrewAI is well-positioned to leverage emerging innovations such as AutoML and Quantum Computing to further improve SDLC automation. In summary, SDLC automation using the combination of integrated GenAI tools and multi-agent systems leads to a transformative approach towards software development.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] S, S. (2017). A Study of Software Development Life Cycle Process Models. *Social Science Research Network*. https://doi.org/10.2139/SSRN.2988291

[2] Anthony, A. R. V., Prasad, G. D., Randunuge, S. U., Alahakoon, S. R. A. M. P. A., Wijendra, D. R., & Krishara, J. (2020). Software development automation: An approach to automate the processes of SDLC. *International Journal of Computer Applications*, *175*(37), 44-51.

[3] Şimşek, T., Gülşeni, Ç., & Olcay, G. A. (2024). The Future of Software Development With GenAI: Evolving Roles of Software Personas. *IEEE Engineering Management Review*.

[4] Raghi, K. R., Sudha, K., & Sreeram, A. M. (2024, December). Software Development Automation Using Generative AI. In *2024 International Conference on Emerging Research in Computational Science (ICERCS)* (pp. 1-6). IEEE.

[5] CrewAI, "Introduction," Accessed: Feb 2025. [Online]. Available: https://docs.crewai.com/introduction

[6] Lakhamraju, M. V. (2024). The importance of data analytics in business process optimization: A focus on predictive process monitoring. African Journal of Biomedical Research, 27(3S), 6937–6941. https://doi.org/10.53555/AJBR.v27i3S.6645

[7] Khan, I. A., & Kumari, D. (2021). The role of analysis phase of SDLC for small scale business application-a review. *International Journal of Humanities, Engineering, Science and Management*, *2*(01), 63-75.

[8] Jackson, V., Vaz Pereira, G., Prikladnicki, R., van der Hoek, A., Fortes, L., Araújo, C., ... & Ramos, D. (2025). Exploring GenAI in Software Development: Insights from a Case Study in a Large Brazilian Company.

[9] Maharana, T., Agrawal, N., Sharma, V., & Alkhayyat, A. (2024, November). An Intelligent Hybrid GenAI Model for Software Testing. In *2024 International Conference on Intelligent Computing and Emerging Communication Technologies (ICEC)* (pp. 1-5). IEEE.

[10] Tembhekar, P., Devan, M., & Jeyaraman, J. (2023). Role of GenAI in Automated Code Generation within DevOps Practices: Explore how Generative AI. *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)*, *2*(2), 500-512. https://doi.org/10.60087/jklst.vol2.n2.p512

[11] Joshi, S. (2025). Review of autonomous systems and collaborative AI agent frameworks. *International Journal of Science and Research Archive*, *14*(2), 961-972.

[12] Tomassi, A. (2024). *Data Security and Privacy Concerns for Generative AI Platforms* (Doctoral dissertation, Politecnico di Torino).

[13] Williams, J. J., & Teal, T. K. (2017). A vision for collaborative training infrastructure for bioinformatics. *Annals of the New York Academy of Sciences*, *1387*(1), 54-60.

[14] Seghier, M. L. (2025). AI-powered peer review needs human supervision. *Journal of Information, Communication and Ethics in Society*, *23*(1), 104-116.

[15] Karmaker, S. K., Hassan, M. M., Smith, M. J., Xu, L., Zhai, C., & Veeramachaneni, K. (2021). Automl to date and beyond: Challenges and opportunities. *Acm computing surveys (csur)*, *54*(8), 1-36.