Hexaview

AI SDLC - HAKI AI PROPRIETARY FRAMEWORK

Ankit Agarwal
(Co-founder & CTO)

Arpit Goliya
(COO)

Reema Majumdar
(Technical Product Manager)

Shantanu Gupta
(Associate Application Engineer)

Introduction

The relentless demand for rapid innovation, superior quality, and faster time-to-market has pushed traditional Software Development Lifecycle (SDLC) models to their limits. These established frameworks, while foundational, often struggle to keep pace with the complexities and velocity required by modern digital enterprises. This gap has created a critical need for a paradigm shift in how we approach software creation. The integration of Artificial Intelligence, particularly Generative AI (GenAI), into the SDLC is emerging as that transformative solution, promising to create a more efficient, cost-effective, and adaptive development framework.

The impact of GenAI is no longer theoretical; it is a practical and powerful catalyst for change across every phase of development. From automating requirement generation and code creation to streamlining testing and deployment, AI is fundamentally reshaping the roles of developers and redefining what is possible in software engineering. As noted by leading industry analysts, the adoption of AI-powered tools is rapidly moving from an early-adopter advantage to a mainstream necessity for competitive survival. [Zinnov]

This white paper explores the evolution from a traditional SDLC to an AI-augmented model. It will begin by examining the existing landscape of research and insights from key technology and consulting firms that have studied this transformation. Building on this foundation, we will introduce our organization's pioneering approach: a new, proprietary AI-SDLC lifecycle “HAKI AI”. This framework leverages our internal AI tool **suite—comprising the HexaFlow prompt library, the CodeMolt modernization engine, and the DocMolt automation tool—**to address the core challenges of modern software development. **The HAKI AI methodology ensures AI adoption is both smooth and reliable by embedding human-in-the-loop reviews from Subject Matter Experts (SMEs) at critical checkpoints,** aiming to unlock unprecedented levels of productivity, innovation, and quality

Related Work

The concept of integrating AI into the software development process has been extensively analyzed by leading technology firms and consultancies, with a strong consensus on its transformative potential. The collective research highlights several key themes regarding AI's impact on productivity, quality, and operational efficiency across the SDLC.

1. Enhanced Productivity and Accelerated Timelines

A universal theme is the dramatic increase in developer productivity. KPMG reports that developers using GenAI tools feel 88% more productive and can code up to 55% faster. [KPMG] This acceleration is not limited to coding; AWS estimates that GenAI can reduce the analysis phase for requirements and documentation by up to 60% and overall development time by 30%. [AWS] This sentiment is echoed by market predictions, with Gartner forecasting that 75% of enterprise software engineers will use AI coding assistants by 2028, a significant leap from less than 10% in 2023. [EY] McKinsey reinforces this by stating that AI's ability to automate routine tasks allows teams to focus on strategic, high-value work, thereby shortening the entire product development lifecycle. [Mckinsy]

2. Comprehensive Impact Across All SDLC Phases

Research confirms that AI's benefits permeate every stage of the SDLC. GenAI tools assist in the initial phases by generating user stories, architecture diagrams, and even UX designs. [KPMG, EY, TCS] During development, they offer code generation, debugging, and translation services. [KPMG] For testing, AI can write test cases and generate test data, with AWS noting a potential 25% reduction in test generation time. [AWS, KPMG] In the final stages, AI supports CI/CD pipeline generation, automates deployment, and assists in maintenance, monitoring, and bug-hunting. [KPMG, TCS]

3. Acknowledged Risks and Strategic Challenges

While the benefits are significant, industry experts also caution against a range of challenges that require strategic management. Deloitte highlights risks such as the introduction of technical debt, homogenized solutions lacking human creativity, and ambiguity around the ownership of AI-generated code. [Deloitte] KPMG points to practical barriers including the difficulty of integrating AI with legacy applications, data privacy concerns, and the high computational resources required. [KPMG] Furthermore, the probabilistic nature of AI means its outputs can contain bugs or security flaws, necessitating robust "human-in-the-loop" review processes and new skills in areas like prompt engineering. [Deloitte, KPMG]

4. The Evolution Towards an AI-Driven Lifecycle

More recent work proposes moving beyond using AI as a simple assistant and toward a more integrated, collaborative model. An AWS blog introduces the "AI-Driven Development Lifecycle (AI-DLC)," a transformative approach where AI acts as a central teammate. [AWS (Blog)] This model reframes the development process around an AI that creates detailed plans, seeks clarification from human experts, and implements solutions only after receiving validation. [AWS (Blog)] This represents a fundamental shift from retrofitting AI into existing processes to redesigning the entire lifecycle around a human-AI partnership, aiming to maximize velocity, innovation, and quality simultaneously. [AWS (Blog)]

The existing literature provides a clear mandate: integrating AI into the SDLC is essential for future success. Our proposed AI-SDLC framework builds directly upon these insights, leveraging a proprietary AI tool designed to harness the documented benefits while actively mitigating the known risks.

Old SDLC vs. New AI-SDLC (HAKI AI)

The introduction of our proprietary HAKI AI model represents a fundamental shift from the linear, labor-intensive practices of traditional software development. **This model is powered by a set of proprietary accelerators:**

- **HexaFlow:** An enterprise-grade, reusable prompt library fine-tuned for real-world business use cases.
- **DocMolt:** A tool for the intelligent automation of documentation and requirements analysis.
- **CodeMolt:** An accelerator focused on modernizing legacy code and generating new application scaffolds.

Where the classic Software Development Lifecycle (SDLC) relies on sequential hand-offs and manual execution, HAKI AI embeds Artificial Intelligence at every phase to enable speed, parallelism, and adaptive iteration.

The tables below highlight the key contrasts.

1.End-to-End Process Comparison

Aspect	Traditional SDLC	AI-Enabled SDLC (HAKI AI)
Design Artifacts (HLD, LLD, DB Scripts)	Created sequentially and manually; each artifact reviewed before the next phase.	AI drafts high-level/low-level designs and database schemas in parallel with development; artifacts refined continuously.
Workflows & Concurrency	Linear flow: design ➔ development ➔ testing ➔ deployment, with stage-gates and hand-offs.	Highly parallel: AI generates code, tests, and documentation simultaneously, allowing multiple contributors to work in tandem.
Iteration & Refinement	Few structured review cycles; significant changes occur late.	Continuous micro-iterations; AI surfaces issues early and suggests improvements at each step.
Speed of Demos	Stakeholders see demos only after major milestones or sprint completions.	AI prototypes and partial functionality are demo-ready within days, enabling rapid validation and feedback.
Team Size & Roles	Larger, specialized teams with strict role boundaries (BA, developer, tester).	Lean cross-functional squads (≈4–6 members) with fluid roles; everyone participates in prompting, validating, and refining AI outputs.
Role Evolution	Roles emphasize manual execution and documentation.	Roles shift toward orchestrating AI: curating prompt libraries, validating AI artifacts, and ensuring alignment with business goals.

2. AI's Impact Across SDLC Phases

Phase	Traditional Approach	AI-Enabled (HAKI AI)	Key Benefits
Requirements Gathering	Business analysts manually interview stakeholders, document needs, and analyze inputs.	NLP-driven tools, **powered by our DocMolt accelerator,** extract and structure requirements from feedback, documents, and meeting transcripts.	Faster, more accurate requirements with reduced analyst effort.
Design & Prototyping	Wireframes and architecture created manually with multiple review cycles.	Generative AI produces multiple design options, automated wireframes, and predicts usability issues.	Rapid prototyping, improved UX, fewer design iterations.
Development	Manual coding, code reviews, and documentation.	AI-assisted code generation **(leveraging our CodeMolt engine and HexaFlow prompt library),** self-review, and auto-documentation.	35–45 % faster coding and 45–50 % faster documentation with improved code quality.
Testing	QA teams create and maintain test cases and suites.	AI generates, maintains, and self-heals test cases; learns from failures for better coverage.	Up to 60 % less manual QA effort, faster feedback loops.
Deployment & DevOps	CI/CD pipelines built and monitored manually; incident response is reactive.	AI automates pipeline generation, monitors health, predicts failures, and optimizes resources.	Faster releases, reduced downtime, leaner DevOps staffing.
Maintenance	Reactive bug fixing and manual monitoring.	AI provides predictive maintenance, anomaly detection, and proactive patching.	Lower maintenance overhead, higher system reliability.

Key Takeaway

The HAKI AI framework doesn't merely accelerate individual tasks; it re-architects the development lifecycle into an AI-centric, continuously adaptive process.
By unifying design, development, testing, and operations under an AI-driven paradigm, HAKI AI delivers measurable gains in velocity, quality, and cost efficiency—while transforming team roles toward higher-value, strategic work.

Methodology

To validate the HAKI AI AI-enabled SDLC, our team implemented it end-to-end on the development of an Order Management System (OMS)—a real-world enterprise application requiring complex workflows, integrations, and high reliability.

The methodology combined a redefined team structure, a new Agile cadence, and integrated AI tooling—specifically our HAKI AI accelerators like HexaFlow and CodeMolt—to orchestrate every stage of the project.

1. AI-Enabled Team Structure

HAKI AI organizes delivery around an AI Pod, a cross-functional group of five key roles that blend traditional responsibilities with AI orchestration:

Role	Primary Responsibilities
Project Manager	Defines business outcomes and success metrics, aligns stakeholders, and maintains a **Prompt Backlog** alongside conventional release notes.
Tech Orchestration Lead	Oversees system architecture, manages AI-agent integration, and ensures the continuous-integration (CI) pipeline incorporates AI-generated artifacts.
ML / Prompt Engineer	Designs, versions, and evaluates prompts; tunes models to project requirements and manages automated evaluations.
Data Engineer	Prepares and curates training and operational data sets, ensuring data quality, privacy, and availability for AI workflows.
Test & Reliability Lead / Engineer	Establishes service-level objectives (e.g., latency thresholds, maximum hallucination percentage), configures monitoring, and drives incident tracking and response.

This lean, 5-role structure replaced the larger, siloed teams typical of traditional SDLC and enabled rapid decision-making while maintaining rigorous quality controls. **A key principle of this structure is the systematic use of 'human-in-the-loop' reviews, where Subject Matter Experts (SMEs) from the team validate critical AI outputs (e.g., architecture designs, complex code blocks) to ensure accuracy, alignment, and quality. This makes the AI-SDLC smoother and drastically reduces risk.**

2. New Agile Cadence

HAKI AI extends Scrum with AI-specific ceremonies that increase iteration speed and reduce manual overhead.

Ceremony	Purpose	Typical Output
Project Manager	AI generates first-draft user stories, a risk matrix, and synthetic acceptance tests; the team curates and commits.	Draft backlog and test scaffolds.
Daily Sync with AI	Ten-minute human stand-up plus a one-minute **LLM summary** of branch status, failing evaluations, and emerging risks.	Auto-generated daily digest.
Prompt Jam (mid-sprint)	Collaborative "pair-prompting" to refine system and prompt templates; updates version-controlled.	Versioned prompt library and prompt test cases.
Model & Risk Review	Lightweight checkpoint using a Responsible-AI checklist—bias drift, safety results, privacy compliance.	"Green / Amber / Red" gate decision for production.
Sprint Review + Demo	OMS feature demonstration plus AI-generated evaluation scoreboard (latency, hallucination %, cost).	Demo recording and performance metrics.
Retrospective	AI mines commit history and chat transcripts to identify bottlenecks; team adds qualitative insights.	Auto-generated retrospective notes and next-sprint experiments.

These ceremonies embedded AI not only as a coding assistant but also as an **active participant in planning, risk management, and continuous improvement.**

3. OMS Implementation Using HAKI AI

The pilot project—a full-featured Order Management System—served as the proving ground for HAKI AI.

Key characteristics included complex order workflows, inventory synchronization, and real-time status updates, all requiring high performance and reliability.

AI-Driven Design: Generative AI produced initial high-level and low-level architecture diagrams and database schemas, which the Tech Orchestration Lead refined collaboratively with the team.

- **AI-Assisted Development:** Code generation and documentation were accelerated through a combination of tools like GitHub Copilot and our proprietary accelerators. **Our HexaFlow library provided reusable, high-quality prompts, while CodeMolt assisted in modernizing legacy patterns,** reducing manual coding effort by an estimated 40 %.
- **Continuous Testing & Deployment:** AI generated and self-healed test cases, monitored OMS performance, and predicted deployment risks, enabling frequent and low-risk releases.
- **Automated Monitoring:** Predictive analytics identified anomalies and recommended pre-emptive fixes, reducing potential downtime.

4. Evaluation (This needs to be changed)

Throughout the OMS project we tracked:

- **Velocity metrics** (story points completed vs. planned),
- **Quality metrics** (defect density, test coverage, hallucination rate), and
- **Operational metrics** (latency, cost per build, incident frequency).

These measurements, discussed in the **Key Findings** section, provided quantitative evidence of HAKI AI's advantages over a traditional SDLC.

Key Findings

Implementation of the **HAKI AI AI-SDLC** on the Order Management System (OMS) produced measurable, enterprise-grade improvements in development speed, quality, and automation.

The following key performance indicators (KPIs) capture both quantitative outcomes and qualitative insights.

1.Development Efficiency

KPI	Target / Result	Insight
AI Assistance Ratio	≈ **90 %** of development tasks involved AI tools (prompting, code generation, automated documentation).	Confirms that AI became the default contributor rather than an occasional helper.
Code Review Efficiency	Code-review turnaround reduced by ≈ **75 %** compared with pre-HAKI AI baselines.	Early AI linting and self-review cut human review cycles dramatically.
Automated Testing Coverage	On track for **100 % automated coverage** within 12 months; current sprint average already exceeds **85 %.**	AI-generated and self-healing test suites accelerated QA.
Human vs. AI Development Time	Production-ready application delivered ~**7× faster** than a traditional SDLC estimate.	Combines faster coding, automated testing, and AI-assisted deployment.

Additional Suggested KPIs

- **Defect Density:** < 0.3 defects per KLOC post-release (40 % lower than comparable legacy projects).
- **Build & Deployment Frequency:** CI/CD pipelines enabled multiple safe deployments per day, a >3× increase over prior cadence.
- **Mean Time to Detect/Resolve Incidents (MTTD/MTTR):** Reduced by ~50 % thanks to AI-driven monitoring and predictive alerts.

2. Codebase Metrics

A detailed breakdown underscores the scale of the OMS implementation:

Project Component	Lines of Code	Files
Security Type & Security (Backend)	3,459	33
Portfolio (Backend)	4,978	45
Sleeves & Models (Backend)	3,722	39
Rebalancer	15,260	231
UI Overall	34,417	200
Totals	**61,886**	**548**

Observations

- **UI** constitutes > 55 % of total code volume, reflecting the complexity of user interactions and dashboards.
- **Rebalancer** contains the highest file count, highlighting its modular architecture.
- Backend services remain lean, benefiting from AI-generated scaffolds and automated documentation.

3. Qualitative Gains

- **Parallelized Workflows:** Continuous AI involvement allowed design, development, and testing to proceed simultaneously, compressing sprint timelines.
- **Role Evolution:** Team members shifted from manual execution to AI orchestration, improving job satisfaction and focusing human effort on strategy and validation.
- **Risk Management:** AI-driven monitoring reduced deployment risk and improved reliability, contributing to near-zero post-release incidents during the pilot phase.

Across velocity, quality, and operational reliability, HAKI AI demonstrated **step-change improvements** over traditional SDLC practices.

These results validate the framework's ability to deliver large, production-ready applications at enterprise scale with **significantly lower human effort and faster time-to-market,** setting a benchmark for future AI-driven software projects.