

Improving Software Quality in Agile Development Environments Using Artificial Intelligence Tools

Table of Contents

Abstract	1
Introduction	2
Background	3
The Role of AI Across Agile SDLC	3
Challenges to Quality Improvement in Agile	4
Problem Statement	5
Research Objectives	6
Research Scope	7
Literature Review	8
Research Methodology	15
Phase 1: Literature Review	15
Phase 2: Qualitative Research	15
Phase 3: Quantitative Research	16
Phase 4: Framework Development and Validation	16
Ethical Considerations	17
Limitations	17
Expected Outcomes	17
Gantt Chart	19
References	20

Abstract

Agile software development accelerates delivery through short iterations and continuous feedback; however, this cadence can degrade quality unless teams employ robust, automated assurance. This thesis examines how Artificial Intelligence (AI) tools—spanning defect prediction, AI-powered code review, intelligent test generation, natural language processing (NLP) for requirements, and Artificial Intelligence for IT Operations (AIOps)—can measurably improve software quality in Agile environments. We synthesize evidence from recent surveys and empirical studies and propose a practical framework (AI-for-Agile Quality, A4AQ) that maps concrete AI capabilities to Scrum/Kanban ceremonies and DevOps pipelines with quality gates and key performance indicators (KPIs). A mixed-methods design (systematic literature review, semi-structured interviews, and a broad survey) underpins the framework, which we validate through expert review and pilot implementations.

Introduction

Agile practices deliver value incrementally but often face recurring quality risks: regression defects from frequent merges, brittle UI tests, uneven code reviews, and reactive operations. Concurrently, AI has matured to assist across the lifecycle—from requirements analysis and code review to test generation and operations analytics. The central challenge is integrating AI responsibly to produce measurable improvements without disrupting Agile cadence or introducing new risks such as model hallucinations or privacy exposure.

Background

The Role of AI Across the Agile SDLC

In backlog refinement and sprint planning, NLP/LLMs help detect ambiguity and align user stories with acceptance criteria. During implementation, assistants provide context-aware suggestions and highlight code smells. Testing benefits from ML-based prioritization, LLM-generated test cases, and self-healing automation. CI/CD integrates static analysis and quality gates focused on new code, while AIOps predicts incidents and flags anomalies, feeding learning back into sprints.

Challenges to Quality Improvement in Agile

Agile teams confront volatile requirements, fragmented test assets, incomplete coverage on new code, and subjective reviews. AI adds risks: hallucinations, bias from historical repositories, privacy issues when analyzing proprietary code/logs, and evaluation drift as models update. Teams need guardrails—explainability, human-in-the-loop review, data minimization, and governance—to safely realize benefits.

Problem Statement

Existing works focus on single phases—testing, code review, or requirements—but lack an end-to-end, empirically grounded framework tailored to Agile/DevOps that connects AI capabilities to ceremonies, pipelines, and KPIs with governance controls. This thesis addresses the gap by defining and validating A4AQ, offering implementation playbooks and metrics that teams can adopt incrementally.

Research Objectives

- Analyze pain points where AI can lift quality without impeding cadence.
- Assess effectiveness of AI tools across requirements, coding, testing, CI/CD, and operations.
- Design A4AQ mapping capabilities to ceremonies/pipelines with KPIs and quality gates.
- Validate A4AQ via expert reviews and pilots; quantify pre/post improvements.
- Provide governance (privacy, ethics), human-in-the-loop, and rollout roadmap.

Research Scope

Scope: Agile delivery of web/mobile/cloud systems with CI/CD. Quality dimensions: reliability (defects, MTTR), security (vulnerabilities), maintainability (code smells, technical debt), and test effectiveness (coverage on new code, flaky rate). Out of scope: safety-critical formal verification and hardware-in-the-loop testing.

Literature Review

AI in Test Automation

Systematic reviews highlight improved coverage and reduced maintenance (e.g., self-healing, visual testing), while reporting limitations such as false positives and limited domain context.

Defect Prediction

Surveys on ML/DL for defect prediction show value in prioritizing QA on high-risk modules using code/process metrics, improving focus and outcomes.

LLMs in Software Engineering

Recent systematic reviews report broad applicability (code generation, documentation, review) but emphasize the need for dataset curation, governance, and rigorous evaluation.

NLP for Requirements

NLP/LLMs reduce ambiguity and improve traceability in user stories and requirements; however, reproducibility and hallucination risks persist.

AI Code Review

Industrial-scale deployments indicate faster PR cycles and consistent standards; early academic results caution against unguided reliance and highlight education effects.

Quality Gates in CI/CD

Platforms like SonarQube operationalize continuous quality via Clean-as-You-Code and gates focused on new/changed code.

AIOps & DevOps

Empirical studies on ML DevOps adoption show slower uptake but tangible benefits—higher productivity, better code sharing, and faster issue resolution when integrated.

Research Methodology

We adopt mixed methods: (1) systematic literature review (2019–2025) across IEEE, ACM, arXiv, Springer, MDPI; (2) semi-structured interviews with POs, developers, SDETs, DevOps/SRE, and security engineers; (3) survey ($n \approx 200$) measuring escaped defects, PR cycle time, flaky-test rate, coverage on new code, MTTR, and vulnerability trends; (4) framework development and validation through expert panels and two pilot teams—analyzing pre/post KPI deltas with statistical tests.

Phase 1: Literature Review

Systematic search and thematic coding of improvements and risks.

Phase 2: Qualitative Research

Semi-structured interviews across roles to map pain points and AI opportunities.

Phase 3: Quantitative Research

Survey on AI adoption in Agile—collecting KPIs and perceptions about effectiveness and risk.

Phase 4: Framework Development and Validation

Develop A4AQ; validate via expert review and two pilot teams; measure improvements on KPIs.

Ethical Considerations

Informed consent, anonymization, secure handling of code/logs, privacy-by-design, bias testing, and audit trails for AI interventions.

Limitations

External validity and tool-version drift; mitigations include triangulation and sensitivity analyses.

Expected Outcomes

Validated A4AQ with playbooks (requirements NLP, PR AI review, CI quality gates, AI testing, AIOps), governance checklists, and an adoption roadmap; anticipated reductions in escaped defects, PR cycle time, and MTTR; higher coverage on new code.

Gantt Chart

Task	Start Date	Duration (Days)
Literature Review	2024-09-01	30
Formulate Research Questions	2024-10-01	15
Develop Methodology	2024-10-16	16
Data Collection	2024-11-01	61
Data Analysis	2025-01-01	31
Framework Development	2025-02-01	28
Drafting Thesis	2025-03-01	45
Revisions	2025-04-16	15
Final Submission	2025-05-01	15
Presentation Preparation	2025-05-16	16

References

- Trudova, A., Dolezel, M., & Buchalceva, A. (2020). Artificial Intelligence in Software Test Automation: A Systematic Literature Review. ENASE. DOI:10.5220/0009417801810192.
- Garousi, V., et al. (2025). AI-powered software testing tools: a systematic review and empirical assessment. arXiv:2409.00411.
- Omri, S., & Sinz, C. (2020). Deep Learning for Software Defect Prediction: A Survey. DeepTestConf.
- Akimova, E. N., et al. (2021). A Survey on Software Defect Prediction Using Deep Learning. Mathematics, 9(11), 1180.
- Hou, X., et al. (2024). Large Language Models for Software Engineering: A Systematic Literature Review. arXiv:2308.10620.
- Zheng, Z., et al. (2024). Towards an understanding of LLMs in software engineering tasks. Empirical Software Engineering (2025).
- Cheng, H., et al. (2024). Generative AI for Requirements Engineering: A Systematic Literature Review. arXiv:2409.06741.
- Necula, S.-C., et al. (2024). Using NLP in Software Requirements Engineering: A Systematic Literature Review. Electronics, 13(11), 2055.
- Rasheed, Z., et al. (2024). AI-powered Code Review with LLMs: Early Results. arXiv:2404.18496.
- Engineering@Microsoft (2025). Enhancing Code Quality at Scale with AI-Powered Code Reviews. Microsoft Dev Blogs.
- SonarSource (2024). Integrating Quality Gates into Your CI/CD Pipeline. SonarSource Library.
- Elhaq Rzig, D., et al. (2022). An Empirical Study on ML DevOps Adoption Trends, Efforts, and Benefits. IST 2022 Preprint.