Towards the Integration of Large Language Models into the Software Development Life Cycle

A Systematic Literature Review

Authors:

Mohammadamin Madani (Presenter)

Ksenia Neumann, Abdulrahman Nahhas

Maria Chernigovskaya, Damanpreet Singh Walia, Klaus Turowski

Agenda

- Motivation
- Research questions
- Methodology
- LLM Roles Per SDLC Phase
- LLM Integration Patterns
- Key Risks
- Actionable Advice

LLMs Are Transforming SDLC, but Create New Uncertainties

Productivity Gains

20–40% improvement in routine coding & testing tasks.

High Error Rates

40–55% of generated outputs contain mistakes, inconsistencies, or vulnerabilities.

Context Limitations

Hallucinations and weak domain grounding reduce reliability.

Early-Phase Vulnerability

Bias and ambiguity in requirements/design easily propagate into later phases.

Our Phase-Aware SLR Builds on Task-Focused Reviews

Study	Year	Focus	Phase-Aware?
Hou et al. [1]	2024	Architecture & tasks	No
Zhang et al. [2]	2024	Code-related tasks	No
Fan et al. [3]	2023	General applications	No
Our Work	2025	SDLC phases + workflow impact	YES

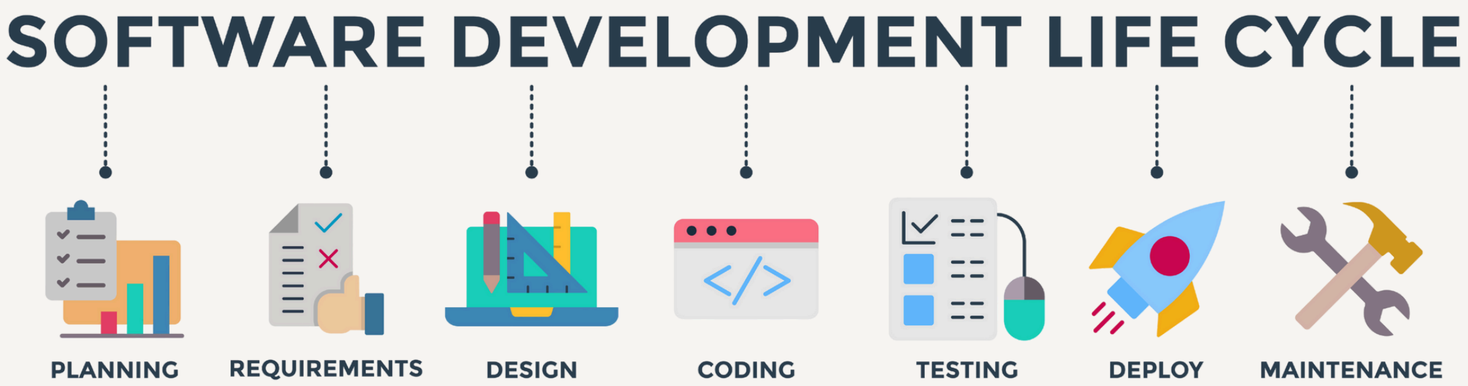
We advance by synthesizing LLM impacts per phase, cross-phase dependencies, and integration patterns—addressing gaps in Agile/DevOps alignment.

Guiding Research Questions

RQ1 How do SDLC phases change with LLMs, and does the traditional model need rethinking?

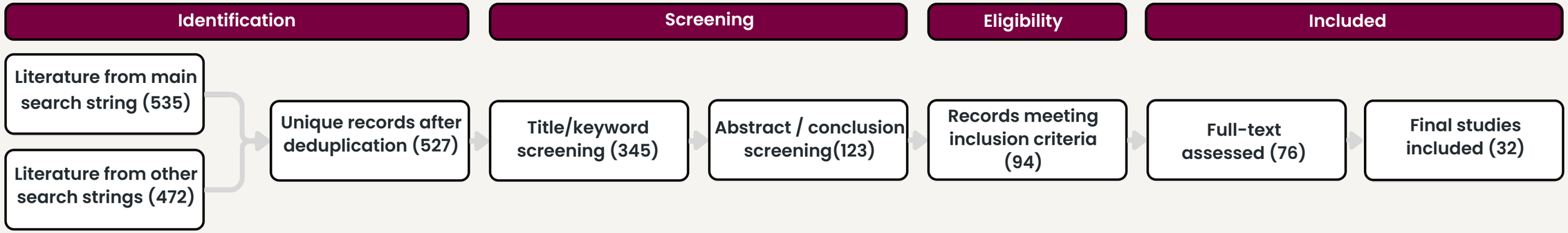
RQ2 What technical, ethical, and operational challenges emerge?

RQ3 How can processes (e.g., Agile, DevOps, CI/CD) adapt for sustainable LLM integration?

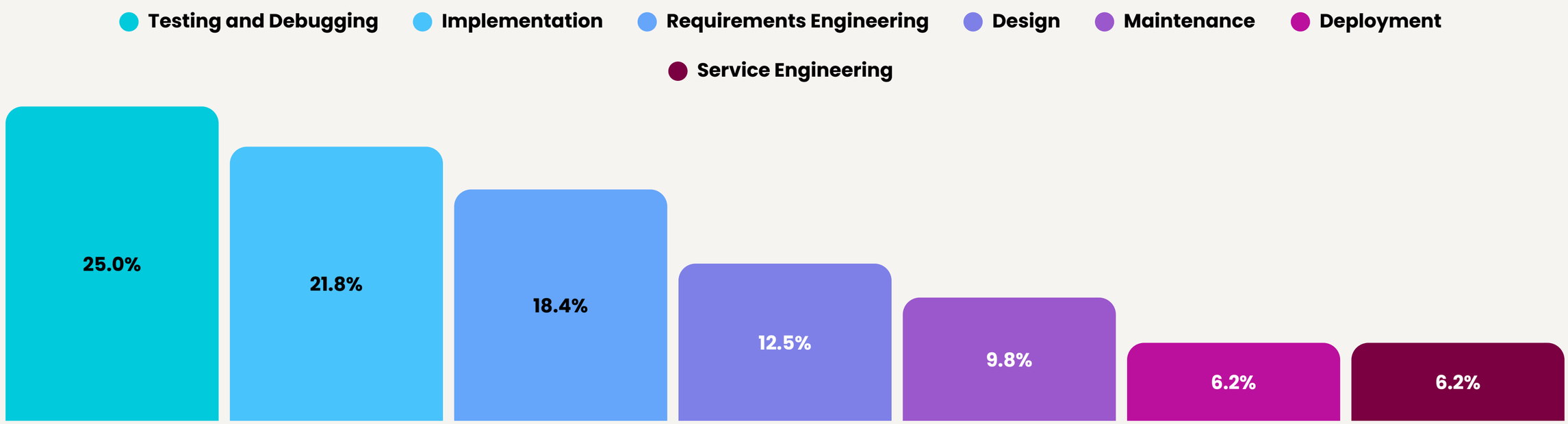


Transparent SLR Protocol (PRISMA-Based)

- Searched IEEE Xplore, ACM DL, SpringerLink, ScienceDirect (Oct 2024–Jan 2025).
- Keywords: Phase-aware terms + LLM/SDLC combos (Table I in paper).
- Inclusion: Peer-reviewed 2019–2025, mapped to SDLC phases. Exclusion: Non-SE, general AI.
- Process: 1,007 records → 527 unique → 76 full-text → 32 included after quality appraisal



Uneven Focus Across SDLC Phases

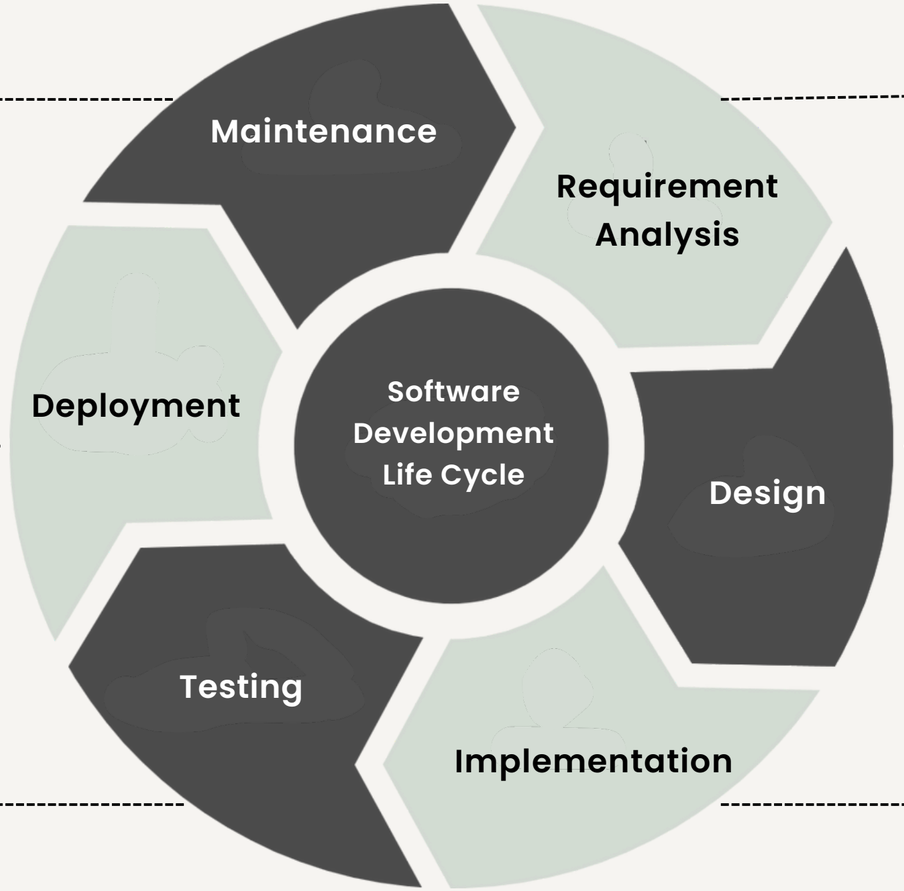


Mid-phases dominate due to code/test automation evidence; early/late phases understudied, limiting holistic insights—calls for more balanced research.

LLM Roles and Benefits Per SDLC Phase

- Predictive Maintenance
- Impact Analysis
- Performance Optimization

- Ambiguity Detection
- Requirement Extraction
- Specification Drafting



- CI/CD Pipeline Generation
- Infrastructure as Code
- Cloud Configuration

- Architecture Recommendation
- Iterative Design
- UML Diagram Generation

- Test Case Generation
- Edge Case Identification
- Bug Diagnosis

- Code Generation
- Code Translation
- Documentation

Four Recurring LLM Integration Patterns

Task-Specific Assistance: Discrete aids like unit-test gen in IDEs—low-risk entry point.

Phase-Oriented Augmentation: Augments one phase, e.g., auto-tests on PRs, modular, reversible.

Cross-Phase Orchestration: Bridges phases, e.g., requirements -> tests -> feedback loops—needs traceability.

LLM-Centric Pipelines: AI drives full lifecycle with human gates—max automation, high risks.

Key Risks and How to Address Them

Hallucination/Inaccuracies

Syntactically valid but wrong outputs.
Multi-step validation, human review.

Over-Reliance,Skill Loss

Attenuates judgment, esp. for juniors
We need Mentoring, review culture.

Bias,Ethics,Opacity

Lack of transparency/IP issues
Audit logs, licensed models, and explainability practices.

Context Misalignment

Domain gaps, esp. regulated sectors
RAG, prompt chaining, hybrid pipelines.

Resources

High compute overhead
Offline/smaller models.

Traceability

Outputs hard to track across phases
Provenance metadata
versioned prompts.

Actionable Advice for Teams

1. Start with Patterns 1-2 for quick wins in implementation/testing.
2. Build human-in-the-loop for advanced patterns to mitigate risks.
3. Address gaps: Focus on design/deployment for untapped opportunities.
4. Adopt LLMOps: Prompt versioning, monitoring, rollback for governance.
5. Adaptive processes like AI-orchestrated pipelines, with ethical safeguards.

Wrapping Up

LLMs unlock efficiencies but necessitate rethinking SDLC phases, patterns, and risks.

RQ1 Phases shift toward automation (strong in implementation/testing), but traditional models need tweaks for cross-phase flows.

RQ2 Challenges like hallucinations, bias, and traceability persist—mitigate with governance and human oversight.

RQ3 Adapt Agile/DevOps with LLMOps for sustainable integration; future focus on understudied phases (design/deployment).

Thank You

Contact: Mohammadamin.madani@st.ovgu.de