Int Journal of Engineering and Computer Science

Open access

*Review Article*

# Applications of Generative AI Across the Software Development Life Cycle

**Sarah Chen***

**School of Artificial Intelligence, International Research University, Singapore*

***Correspondence:** Sarah Chen, School of Artificial Intelligence, International Research University, Singapore, E-mail: slrchens@gmail,com;
DOI: 10.1042/JCTCS.2.1.0004

## Abstract

Generative Artificial Intelligence (GenAI) has transformed software engineering by supporting developers throughout the Software Development Life Cycle (SDLC). Unlike earlier automation tools that focused on isolated tasks, modern large language models can assist in requirements analysis, software design, implementation, testing, deployment, maintenance, and documentation. Their ability to understand both natural language and source code makes them valuable collaborators rather than simple programming assistants. This section reviews the principal applications of GenAI across the SDLC.

**Keywords**: Generative artificial intelligence, Development life cycle, Software design

**Received date:** Sep 02, 2024; **Accepted date:** Sep 12, 2024; **Published date:** Sep 31, 2024

**Citation:** Sarah Chen (2024) Applications of Generative AI Across the Software Development Life Cycle. Int J Eng Comp Sci. 2: 1.



## Introduction

### Requirements engineering

Requirements engineering is the foundation of successful software projects. Poorly defined requirements often lead to cost overruns, delays, and software defects. GenAI can assist analysts by summarizing stakeholder interviews, converting informal descriptions into structured requirements, identifying ambiguities, and suggesting missing functional or non-functional requirements.

Typical applications include:

- Converting natural language requirements into structured specifications.
- Summarizing meeting transcripts.
- Detecting inconsistent or conflicting requirements.
- Generating user stories and acceptance criteria.
- Assisting with requirements traceability.

These capabilities reduce manual effort while improving consistency, although domain experts should validate AI-generated requirements.

## Software architecture and design

System architecture determines the maintainability and scalability of software. GenAI can recommend design patterns, propose software architectures, generate UML-style descriptions, and explain architectural trade-offs.
Applications include:

- Suggesting layered, microservices, or event-driven architectures.

- Recommending design patterns.
- Drafting component descriptions.
- Producing architectural documentation.
- Identifying potential design bottlenecks.

Human architects remain responsible for validating performance, scalability, and security decisions.

## Code generation

Code generation is currently one of the most widely adopted applications of GenAI. Modern AI assistants can generate functions, classes, APIs, and boilerplate code from natural language prompts.

Benefits include:

- Faster implementation.
- Reduced repetitive coding.
- Automatic code completion.
- Multi-language support.
- Improved developer productivity.

Limitations include the possibility of logically incorrect or insecure code, reinforcing the need for code review and testing.

## Code review and refactoring

Maintaining software quality requires systematic review and continuous improvement. GenAI can analyze source code and suggest improvements such as:

- Removing duplicate code.
- Improving readability.
- Renaming variables.
- Simplifying complex logic.
- Identifying potential bugs.
- Suggesting coding-standard compliance.

AI-generated recommendations should complement, not replace, peer review.

## Software testing

Software testing is another area where GenAI provides substantial support. AI models can generate test cases, recommend edge cases, explain failed tests, and assist in regression testing.

Common applications include:

- Unit test generation.
- Integration test suggestions.
- API test creation.
- Boundary-value testing.
- Regression testing assistance.
- Test documentation.

These capabilities help reduce manual testing effort while improving test coverage.

## Debugging and defect localization

Debugging often consumes a significant portion of software development time. GenAI assists by:

- Explaining compiler errors.
- Suggesting possible fixes.
- Identifying probable root causes.
- Recommending debugging strategies.
- Interpreting stack traces.
- Generating corrected code snippets.

Although these suggestions accelerate troubleshooting, developers must verify that proposed fixes address the underlying issue without introducing new defects.

## Documentation generation

Comprehensive documentation is essential for software maintenance but is frequently neglected. GenAI can automatically generate:

- API documentation.
- Function descriptions.
- README files.
- Installation guides.
- User manuals.
- Technical summaries.

Keeping documentation synchronized with code changes remains an important responsibility for development teams.

## DevOps and continuous integration

GenAI also supports DevOps by assisting with infrastructure configuration, deployment scripts, and operational documentation. Representative applications include:

- CI/CD pipeline templates.
- Infrastructure-as-Code assistance.
- Deployment script generation.
- Log interpretation.
- Configuration recommendations.
- Incident summarization.

These features streamline deployment processes while requiring operational oversight.

## Software maintenance

Software maintenance accounts for a large proportion of software lifecycle costs. GenAI contributes by:

- Explaining legacy code.
- Assisting with code migration.
- Recommending refactoring opportunities.
- Detecting deprecated APIs.
- Updating documentation.
- Supporting impact analysis for code changes.

This assistance is particularly valuable for large, long-lived software systems.

**Table 1:** Applications of Generative AI Across the SDLC.

| SDLC Phase | Typical GenAI Applications | Expected Benefits |
|---|---|---|
| Requirements Engineering | Requirement extraction, user stories, ambiguity detection | Improved requirement quality |
| Software Design | Architecture suggestions, design patterns | Better maintainability |
| Implementation | Code generation, code completion | Increased developer productivity |
| Code Review | Refactoring, quality analysis | Improved code quality |
| Testing | Test-case generation, regression support | Higher test coverage |
| Debugging | Error explanation, bug localization | Faster defect resolution |
| Documentation | API documentation, technical writing | Reduced documentation effort |
| DevOps | CI/CD support, deployment scripts | Streamlined operations |
| Maintenance | Legacy code analysis, migration support | Lower maintenance effort |

**Figure 1:** Generative AI integration across the software development life cycle.