Cheat Sheet: Managing Legacy Code with Gen AI

Estimate Time: 15 Minutes

Objectives

After completing this reading, you will be able to:

- Describe legacy code modernization using generative AI
- Explain IBM watsonx Code Assistant for Z

Legacy Code Modernization with Generative AI

Legacy code refers to the existing codebase that has been developed over time and may be outdated or difficult to maintain. However, thanks to advancements in technology, managing and modernizing legacy code is becoming more efficient with the emergence of Generative Artificial Intelligence (AI) techniques.

Legacy code can be characterized by outdated programming languages, lack of documentation, poor software architecture, and dependencies on obsolete technologies. The challenges associated with legacy code include limited scalability, reduced maintainability, and a higher risk of bugs and security vulnerabilities.

The Role of Generative AI:

Generative AI uses machine learning algorithms to generate new content like code, text, or images. When applied to legacy code, Generative AI techniques help to automate various tasks involved in managing and modernizing the existing codebase.

Examples:

Code refactoring: Generative AI models can analyze the structure and patterns in legacy code to suggest refactoring improvements. They automatically identify redundant code blocks, optimize algorithms, or propose alternative coding styles that align with modern best practices. This helps enhance code readability, reduce technical debt, and improve overall software quality. For example, tools like `DeepCode` can identify potential bugs and vulnerabilities in the code, enabling developers to fix them more efficiently.

Document generation: Generative AI models can analyze the codebase and automatically generate documentation, including function descriptions, API references, and usage examples. This saves developers valuable time and effort in manually documenting the codebase, making it easier for new team members to understand and work with the existing code. For instance, tools like `NaturalDocs` can generate comprehensive documentation from comments, variables, and function signatures, making it easier for developers to understand and work with the legacy code.

Bug detection and resolution: Legacy code often contains hidden bugs that may go unnoticed for a long time. Generative AI models can analyze code patterns and identify potential bug-prone areas. By flagging potential issues early on, developers can proactively address them before they escalate into critical problems. Moreover, Generative AI can also suggest fixes or patches for identified bugs based on patterns observed in similar scenarios. For instance, tools like "DeepCode" can generate code snippets or entire functions based on the context of the legacy code, saving developers valuable time and effort.

Migration to modern technologies: Legacy systems often rely on outdated technologies that are no longer supported or efficient. Generative AI can assist in migrating legacy code to modern platforms or languages by automatically translating the codebase into the desired target technology. This significantly reduces the manual effort required for rewriting the entire codebase from scratch. For example, tools like "Kite" use machine learning algorithms to suggest relevant code snippets, function calls, and variable names, enhancing developer productivity while working with legacy code.

IBM watsonx Code Assistant for Z

Over 80% of in-person transactions at U.S. financial institutions use COBOL. IBM has identified 240 billion lines of COBOL running today with an additional 5 billion being written every year.

According to research from the IBM Institute for Business Value, organizations are more likely to leverage existing mainframe assets rather than rebuild their application estates from scratch in the next two years. However, a lack of resources and skills remains the top challenge for these organizations.

There are various approaches to application modernization available today. Some options include rewriting all application code in Java or migrating everything to the public cloud. However, these approaches may sacrifice core capabilities of IBM Z and fail to deliver expected cost reductions.

Tools that convert COBOL applications to Java syntax can produce code that is difficult to maintain and unfamiliar to Java developers. While generative AI shows promise, current AI-assisted partial re-write technology lacks COBOL support and does not optimize the resulting Java code for specific tasks.

IBM has launched watsonx Code Assistant for Z, a new product that utilizes generative AI to facilitate the translation of COBOL to Java and enhance developer productivity on the platform.

Watsonx Code Assistant for Z is a new addition to the watsonx code assistant product family. These solutions will be powered by IBM's watsonx.ai code model, which has knowledge of 115 coding languages and has learned from 1.5 trillion tokens. With 20 billion parameters, it is on track to become one of the largest generative AI models for code automation.

The watsonx Code Assistant product portfolio will expand over time to cover other programming languages, addressing the challenges faced by developers and improving the time to value for modernization efforts.

Watsonx Code Assistant for Z is being developed to assist businesses in leveraging generative AI and automated tools to expedite the modernization of their mainframe applications, all while preserving the performance, security, and resilience capabilities of IBM Z.

By using watsonx Code Assistant for Z on a large scale, developers can selectively and incrementally transform COBOL business services into well-structured, high-quality Java code. With potentially billions of lines of COBOL code as candidates for targeted modernization over time, generative AI can help to quickly assess, update, validate, and test the appropriate code. This allows for more efficient modernization of large applications and enables developers to focus on higher-impact tasks.

Key steps in the journey include refactoring business services in COBOL, transforming COBOL code into optimized Java code, and validating the outcome through automated testing.

The potential benefits for clients include:

- Accelerated code development and increased developer productivity throughout the application modernization lifecycle
- Managed cost, complexity, and risk associated with modernization initiatives, including translation and code optimization
- Expanded access to a wider pool of IT skills and faster developer onboarding
- Improved quality, easily maintainable code through model customization and best practices application

The resulting Java code from watsonx Code Assistant for Z will be object-oriented and optimized for interoperability with the rest of the COBOL application, including CICS, IMS, DB2, and other z/OS runtimes. Java on Z is designed to deliver performance optimization compared to x86 platforms.

Summary

In this reading, you learned that:

- Legacy code refers to the existing codebase that has been developed over time and may be outdated or difficult to maintain.
- Legacy code can be characterized by outdated programming languages, lack of documentation, poor software architecture, and dependencies on obsolete technologies.
- IBM watsonx Code Assistant for Z, is a new product that utilizes generative AI to facilitate the translation of COBOL to Java and enhance productivity on the platform.

Skills Network