Bringing generative AI to z/OS Application Modernization with *IBM watsonx Code Assistant for Z* Wildfire Workshop

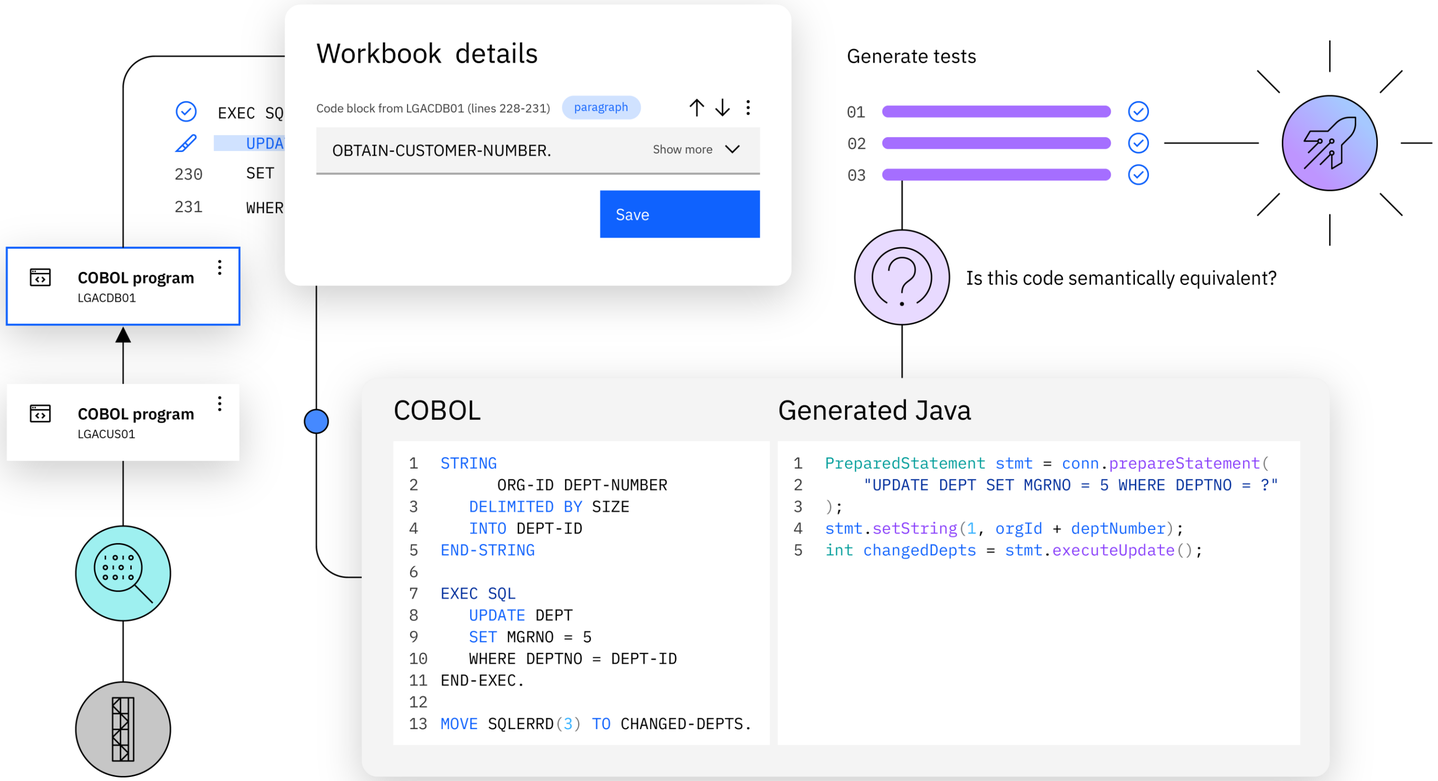
September 19, 2024

Barry Silliman
IBM Washington Systems Center
silliman@us.ibm.com

Matt Mondics
IBM Washington Systems Center
matt.mondics@ibm.com

Joel Moss
IBM Washington Systems Center
jmoss@us.ibm.com

Garrett Woodworth
IBM Washington Systems Center
garrett.lee.woodworth@ibm.com



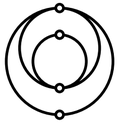
One principal challenge in organizations today is accelerating mainframe application modernization

230+ billion

lines of COBOL are estimated to be actively running by enterprises[1]

1. Source: Reuters Survey (2020)

Mainframe application modernization challenges



Agility

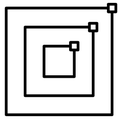
With enterprise DevOps, go from code releases quarterly (per calendar year) to quarterly (per hour)

Skills

Reduce the talent gap with common tools and operating models across platforms

Costs

Leveraging consumption-based pricing on the mainframe (Tailored-fit-pricing) to add new apps to the mainframe



How

What are the proven patterns and best practices for modernizing mainframe applications?

IBM is accelerating application modernization with generative AI

Build the right foundation

Optimize existing applications
Manage the efficiency, cost, and performance of running current applications.

IBM watsonx Code Assistant for Z

Enhance and extend applications
Understand, refactor, and transform applications leveraging an AI-assisted cloud-native experience

Integrate across hybrid cloud
Leverage open APIs and event-driven architecture to integrate hybrid applications.

Simplify information sharing and data access
Optimize and secure data access and information sharing across the enterprise.

Increase business agility

Adopt enterprise DevOps and observability
Leverage enterprise DevOps with an integrated CI/CD pipeline and full application observability.

Make AI-driven decisions at scale
Achieve AI-driven insight at scale to help make decisions in real time.

Automate and standardize IT
Standardized (AI-assisted) enterprise capabilities to automate and manage the application and IT life cycle

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed

Accelerate your journey

Application Discovery / co-creation

Patterns

OpenTools & Languages

IBM Z DevOps is recommended for IBM watsonx Code Assistant for Z

Unlock successful application modernization with watsonx Code Assistant for Z with end-to-end DevOps pipeline

watsonx Code Assistant for Z

Understand ⟷ Refactor ⟷ Transform ⟷ Validate

Integrate and automate delivery of z/OS applications

Continuous Integration

Edit (IDE), Debug, Build [1]

Continuous Delivery

*Automated Application Deployment [1]

GitLab Jenkins JFrog Artifactory Nexus

Continuous Testing

Unit Testing [1]

Integration Testing [2]

On Demand Test Environments [3]

IBM Test Accelerator for Z

Number	Product Mapping
1	IBM Developer for z/OS Enterprise Edition
2	IBM Z Virtual Test Platform, Galasa
3	IBM Z Virtual Dev and Test[On-Prem] OR IBM Wazi aaS [IBM Cloud Service Offering}

- Continuous improvement with a DevOps pipeline is fundamental to iterative application modernization using IBM watsonx Code Assistant for Z
- 20 to 50% productivity improvement with IBM Z DevOps modern development tools based on clients' use cases**

* IBM Urbancode deploy is an option for clients that already use it.

Generative AI helps address modernization challenges

30%

Reduction in time to complete coding tasks through the combination of human & AI assistants working in tandem by 2028

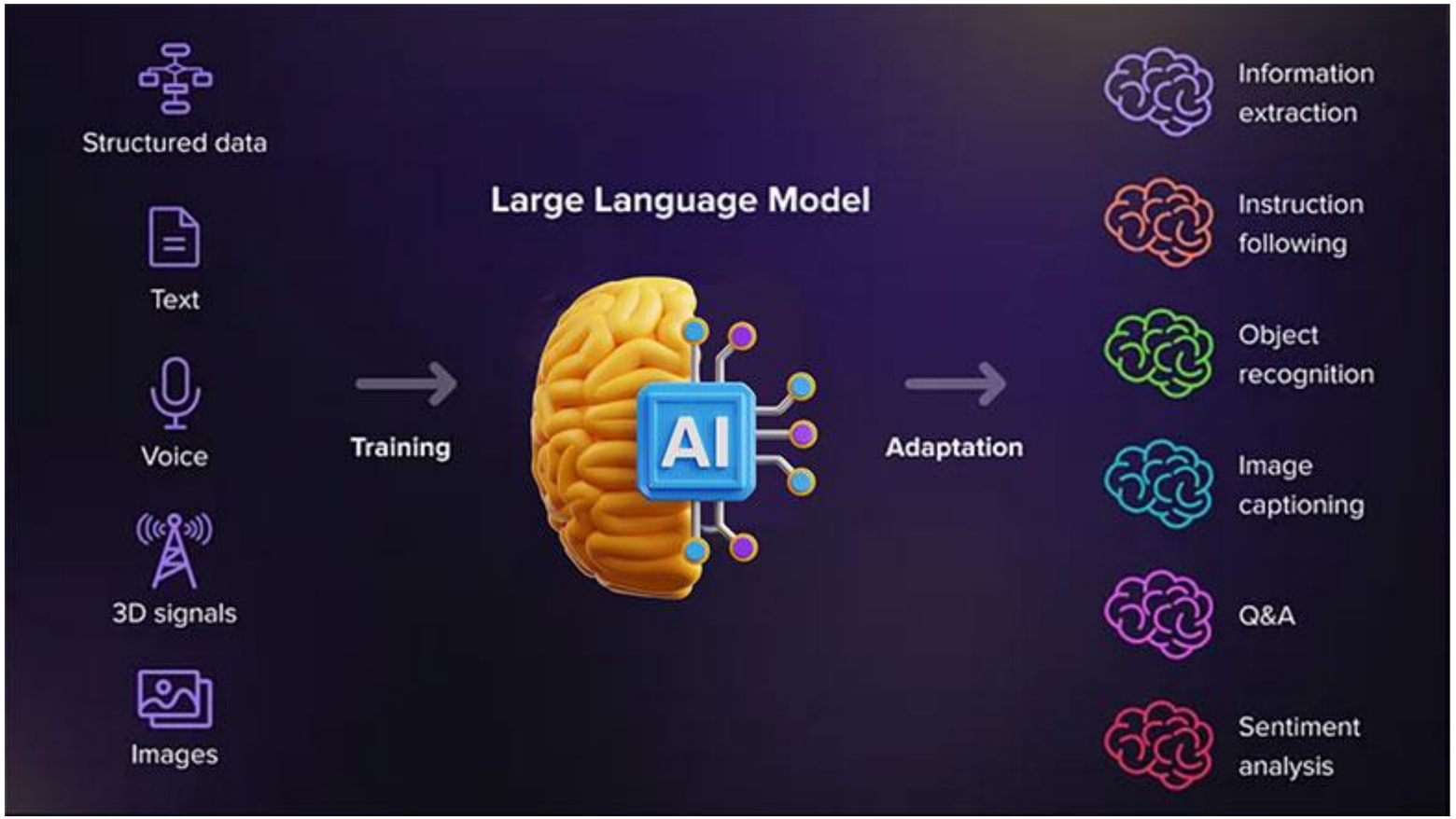
80%

Of the product development lifecycle will make use of generative AI code generation by 2025

Source: Gartner, Emerging Tech: Generative AI Code Assistants Are Becoming Essential to Developer Experience, ID G00790320, May 2023

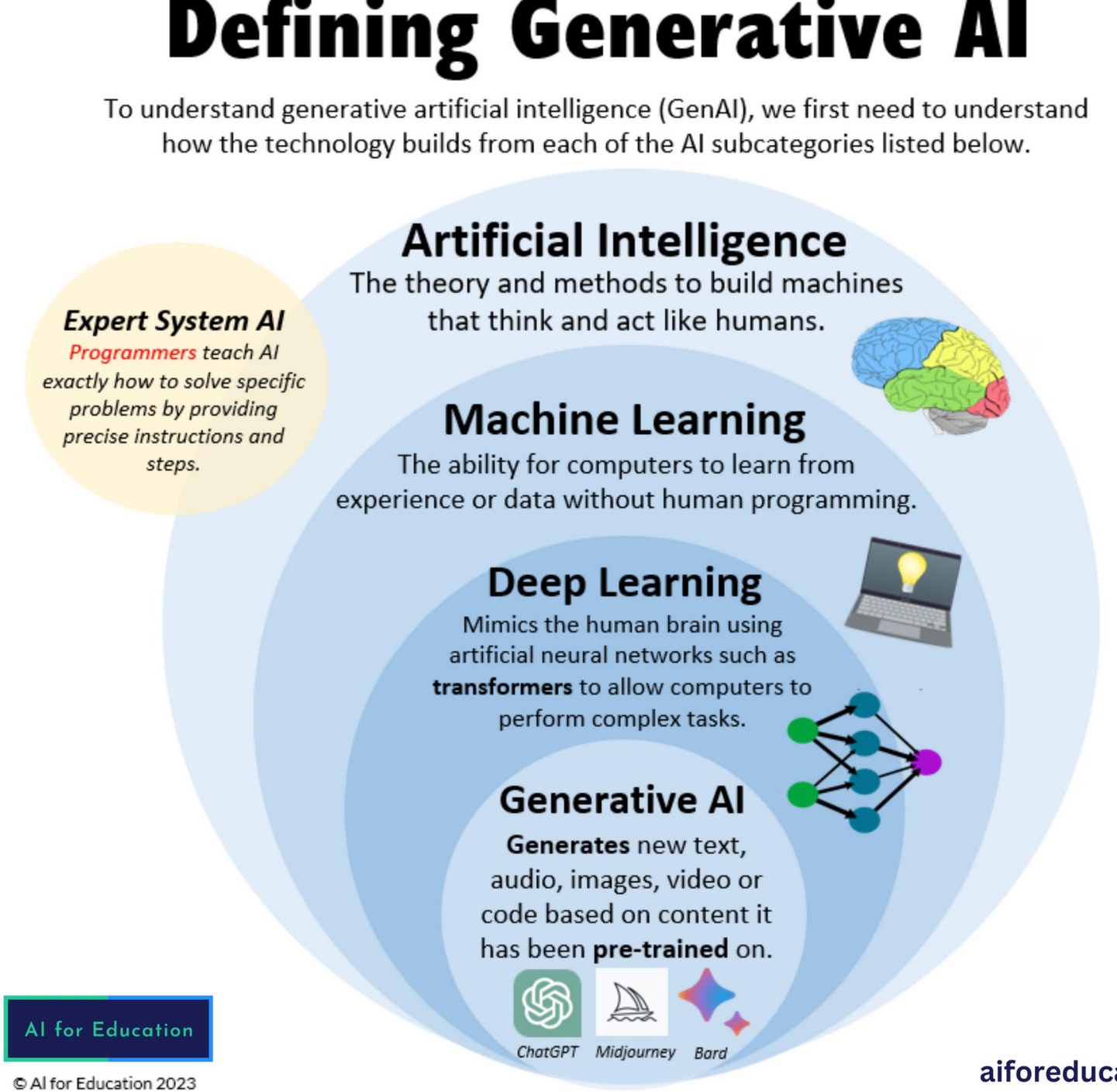
Large Language Models (LLMs)

- Models used for human tasks like recognizing and generating text
- Use a deep learning neural network called a “transformer”
- Trained on massive (i.e., large [think thousands or millions of gigabytes]) sets of data
- Refined for specific tasks such as translating COBOL to JAVA with “tuning”
- Use probabilities from their data to predict the next part of the sequence (e.g., JAVA code corresponding to COBOL code)



Generative AI

- Generative AI uses a variety of AI models to generate content
- Many of the popular generative AI solutions use LLMs as the backbone of Generative AI
- watsonx Code Assistant for Z, watsonx Code Assistant for Red Hat Ansible Lightspeed, and ChatGPT are examples of generative AI using an LLM foundational model (IBM Granite Model, IBM Granite Model, and Open AI's GPT Model respectively)

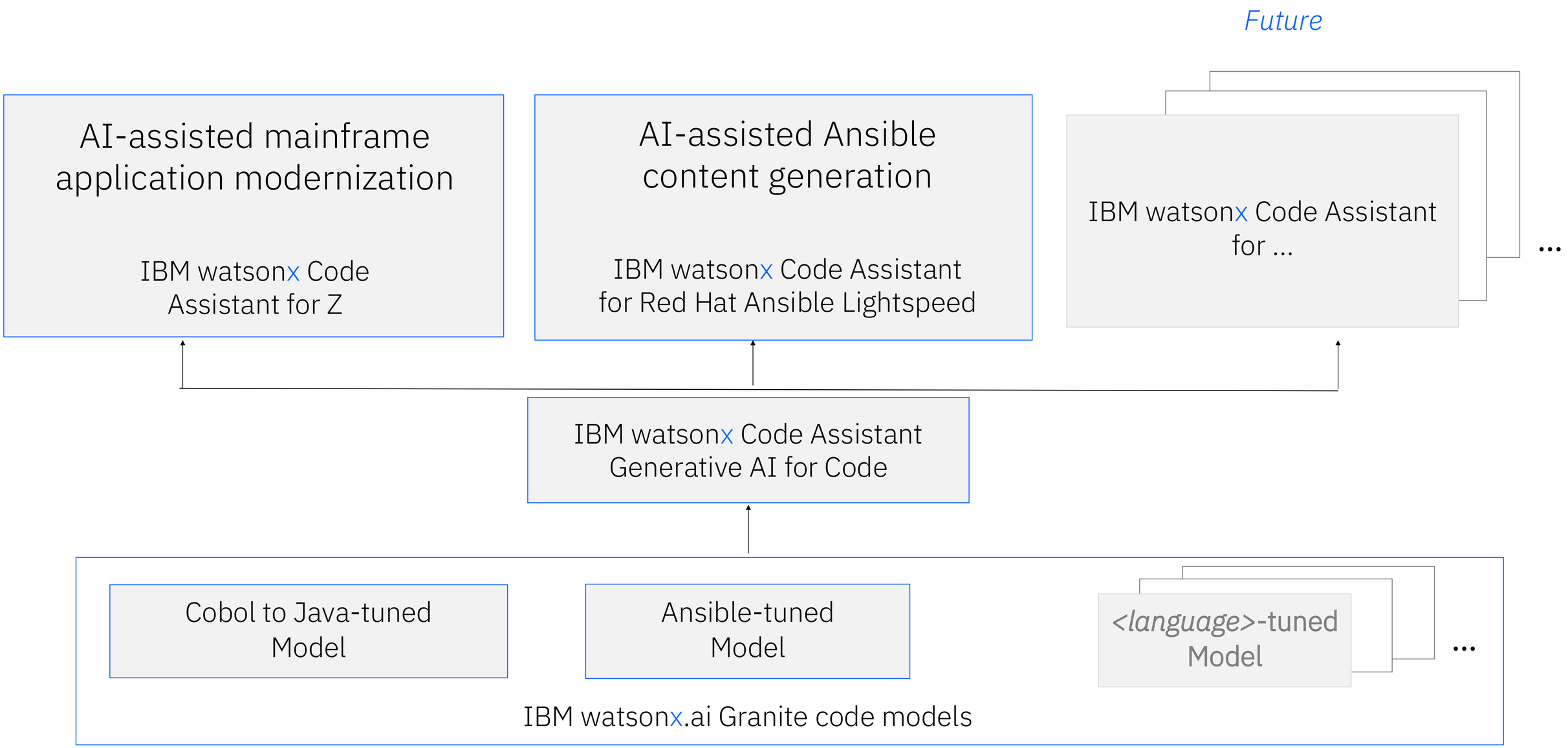


AI for Education

© AI for Education 2023

IBM watsonx Code Assistant

Built for targeted use cases

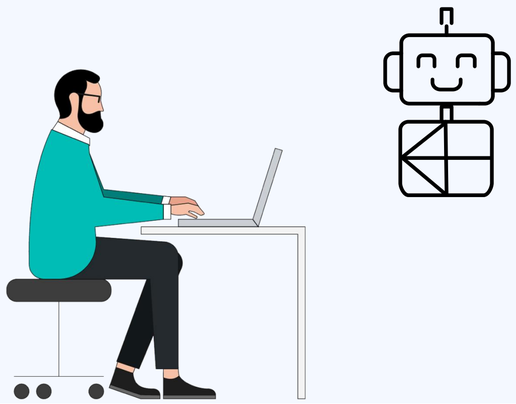


Generative AI is transforming the way users experience and interact with IBM Z

Mainframe AI assistant for operations

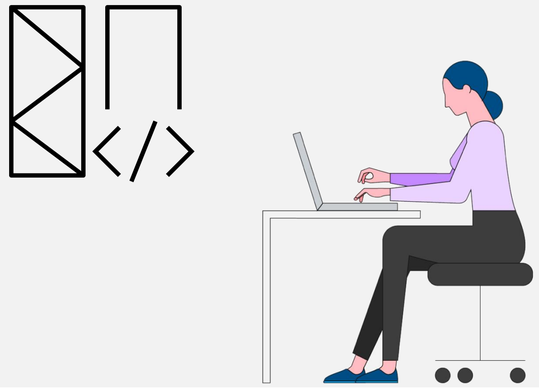
- Quick and accurate answers to questions.
- Execute automation initiated through AI conversation
- Personalize based on job and experience

IBM watsonx Assistant for Z



Mainframe AI assistant for application development

- Supports end-to-end application lifecycle
- Code explanation, optimization, & transformation
- Increased flexibility, interoperability, and quality



IBM watsonx Code Assistant for Z

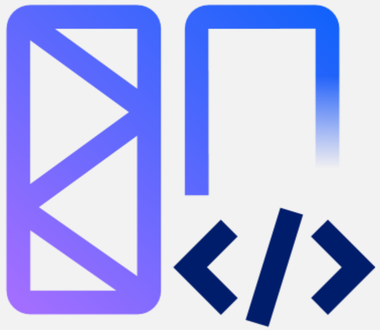
Benefits:

Increase productivity	Reduce learning curve	Increase agility	High quality outcomes

© 2024 IBM Corporation

IBM watsonx Code Assistant for Z

AI-assisted mainframe application modernization



Accelerated application lifecycle

New automated and AI-assisted capabilities to support end-to-end application modernization lifecycle with auto discovery, refactor, and test.

Fine-tuned generative AI for mainframe code

Leverage the power of generative AI to make it easier for developers to explain applications and selectively transform them into well architected, high-quality Java code.

Incremental approach provides faster value

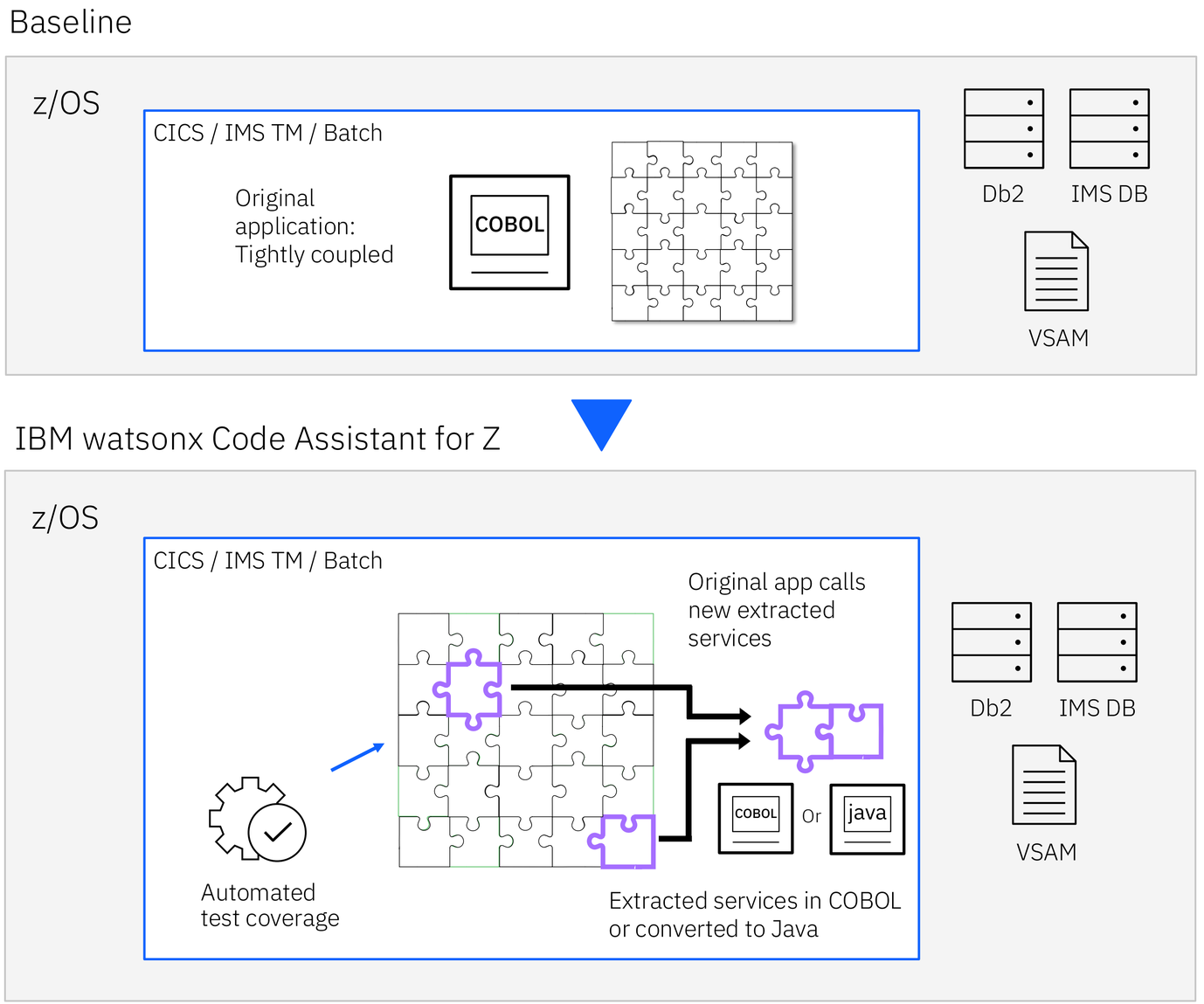
Modernize using an incremental approach that is faster, lower cost and less risk and supports full mixed language interoperability.

Incremental approach provides faster value

Objectives:

- Minimize risk with an incremental approach to modernization
- Selectively modernize based on technical advantages and business needs
- Maintain flexibility to leverage mixed language and architecture with complete interoperability

Transformation with a best-fit approach

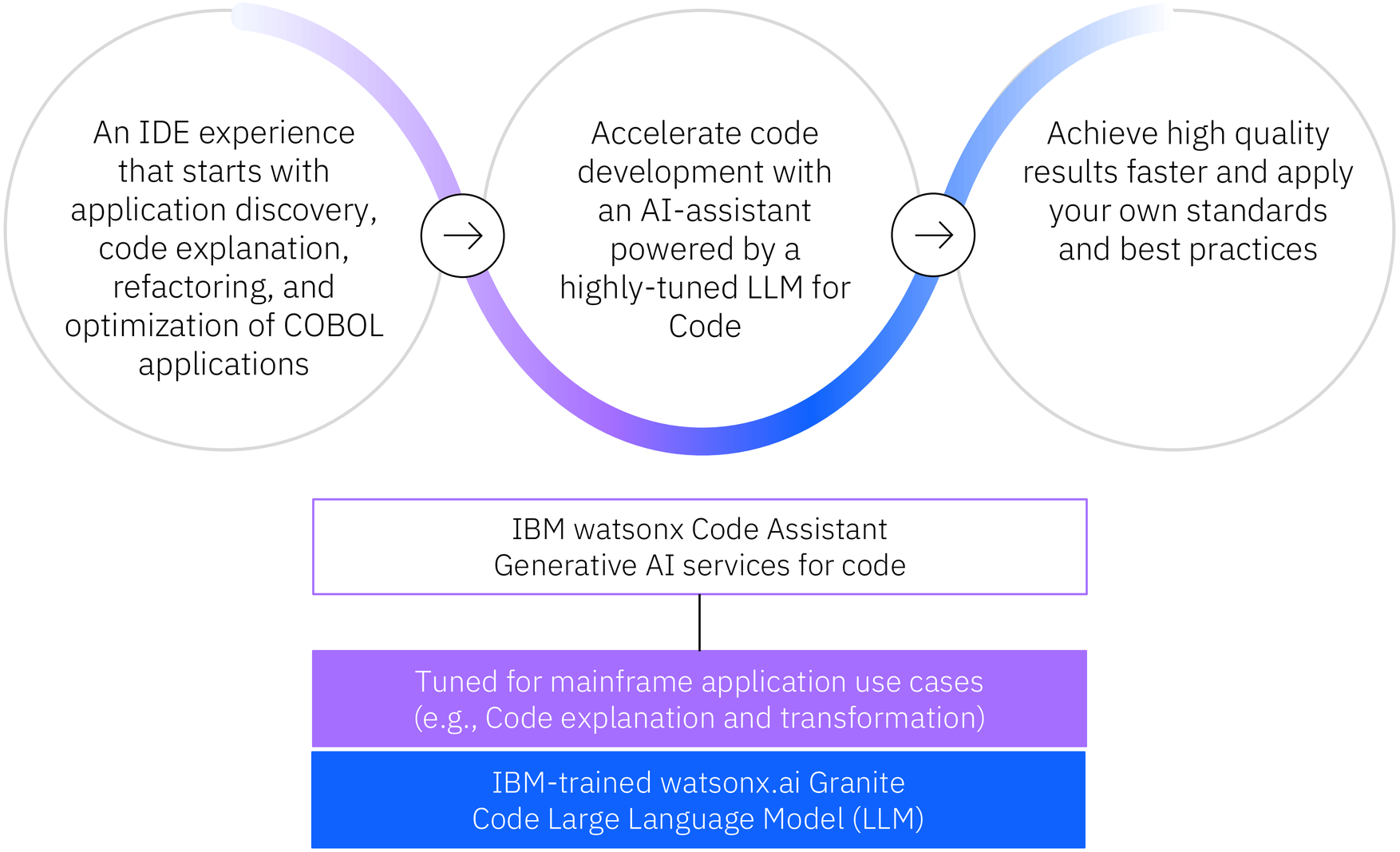


Fine-tuned generative AI for mainframe code

Objectives:

- Finely-tuned model improves understanding, accuracy, and code quality
- Well-architected AI-generated code
- Easy to maintain code that can be enhanced with your standards and best practices

Leverage the power of generative AI to make it easier for developers to modernize code with AI-generated recommendations



Accelerated application lifecycle

Objectives:

- Address skills and productivity challenges with automation and AI
- Ensure IBM Z qualities of service with mixed language interoperability
- Align with industry standard DevOps approaches

IBM watsonx Code Assistant for Z modernization experience

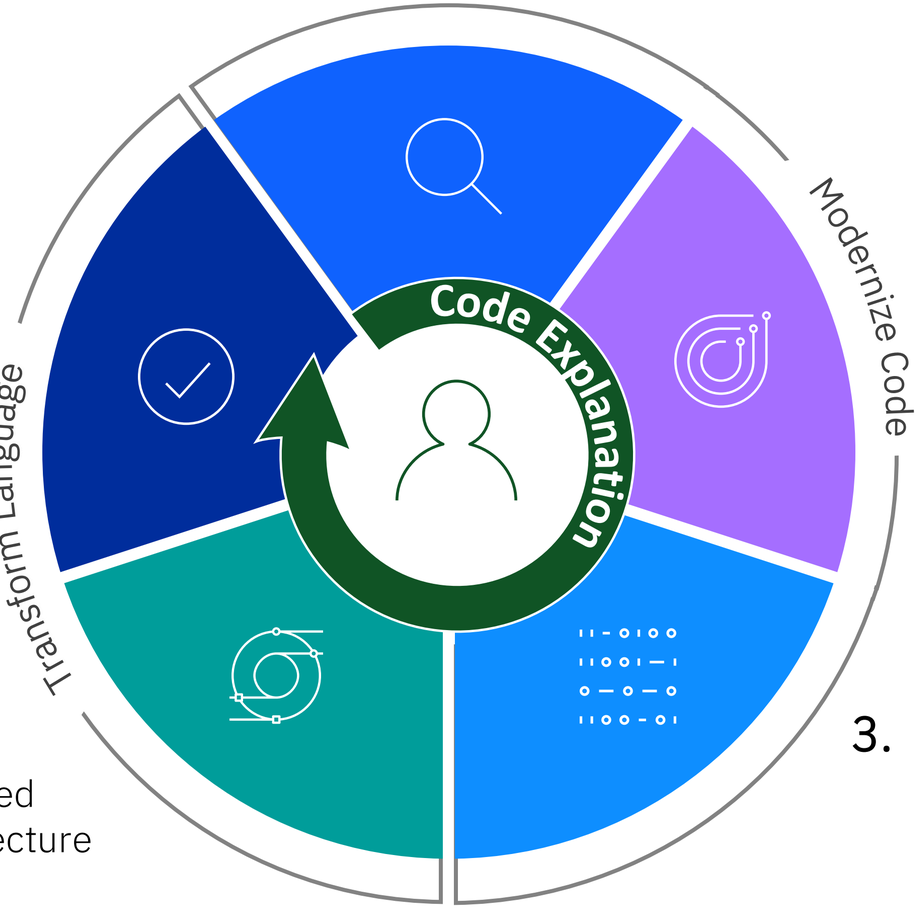
1. **Understand**
   application landscape with auto-discovery

2. **Refactor**
   to modular business services

3. **Optimize**
   code to increase performance

4. **Transform**
   code with optimized design and architecture

5. **Validate**
   outcome with auto-generated tests



Tailor your journey based on your application modernization and development needs

IBM watsonx Code Assistant for Z deployment models

Base software

On-premises

- Understand
- Refactor
- Optimize
- Validate

Add-on capabilities

SaaS in IBM Cloud

- Transform

or

On-premises

- Transform

SaaS in IBM Cloud

- Code Explanation

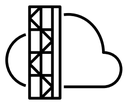
or

On-premises (Coming 3Q)

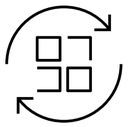
- Code Explanation

Business Value Drivers

Enhance developer productivity



Increase business agility



Increase operational efficiency

Lower risk

Large European Bank

2-5x Productivity Gain
in Code Isolation

Financial Company

66% reduced effort
for understanding and refactoring

Westfield Insurance

2-2.5x developer productivity gain

- 80% less time for application understanding
- reduced change management and onboarding costs

Read the Case Study

Global Logistics Company

14-47% reduced effort for understanding & refactoring

60% reduced effort Code Transformation

COBOL Modernization use case

	Understand your application	Code explanation	Code refactoring	Code optimization
Steps	Deep analysis to capture and document program understanding and relationship. Creating an application “blueprint”	Leverage Gen AI to explain code in natural language that can be inserted as comments or downloaded for documentation	Gain agility by decomposing (refactoring) your application into more modular business services	Improve COBOL code by obtaining insights and recommendations for performance improvement
Outcomes	• 2-5X productivity gain in code isolation • 80% less time for application understanding	• Improved developer onboarding time	• 66% reduced effort for understanding and refactoring	• Improve COBOL performance by up to 30%

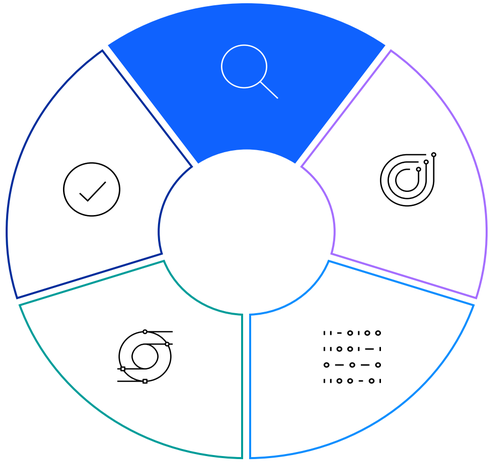
COBOL to Java Transformation use case

	Understand your application	Code explanation	Code refactoring	Transform COBOL to Java	Validate Java
Steps	Deep analysis to capture and document program understanding and relationship. Creating an application “blueprint”	Leverage Gen AI to explain code in natural language that can be inserted as comments or downloaded for documentation	Gain agility by decomposing (refactoring) your application into more modular business services	Leverage Gen AI to transform the refactored and optimized COBOL code into object-oriented Java code	Ensure semantic equivalence between refactored COBOL code and transformed Java code. Assist Java developer with leave behind test asset
Outcomes	• 2-5X productivity gain in code isolation • 80% less time for application understanding	• Improved developer onboarding time	• 66% reduced effort for understanding and refactoring	• Best fit language – Bring the tooling and ecosystem benefits of enterprise Java • Expand mainframe developer talent pool	• Accelerate unit testing • Increase code quality

Modernize JCL

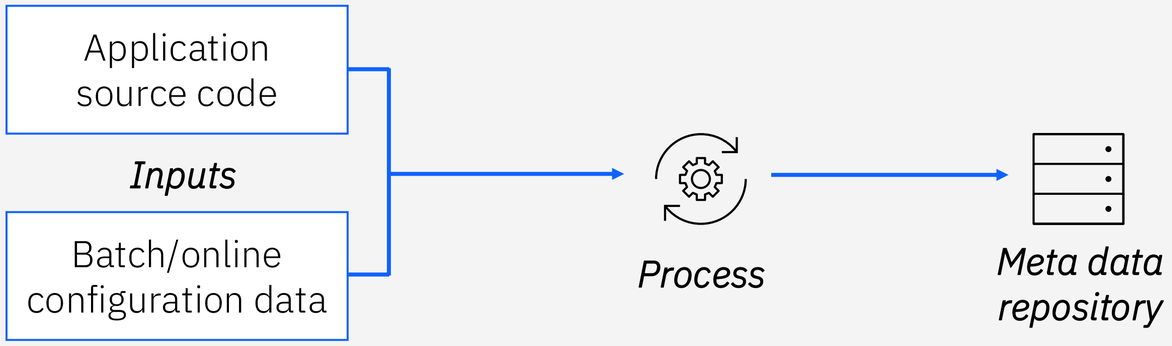
	Understand your JCL jobs	Explain your JCL steps
Steps	Gain insights and understanding of JCL jobs with graphs to map the dependencies, datasets, executed procedures, and programs. • Create Job Flow graphs to easily gain an understanding of dependencies within the job steps and view the analysis report for a display of all the job steps. • Visualize structure of JCL jobs with Job Usage Inventory to understand the jobs, datasets defined in JCL, procedures, and programs that are executed in the application. • Understand relationships between JCL jobs and other components of the application with Job Call Graph	Generate natural language explanations of JCL steps with watsonx Code Assistant for Z. These explanations can then be added as comments or saved as documentation. The user can also choose to add those as annotations to the Job Flow graph or Dataset Flow graph as appropriate.
Outcomes	**Reduce disruption on SMEs time** empowering new team members to accelerate their understanding of JCL by leveraging graphs to visualize JCL job dependencies and relationships. **Reduce risk of erroneous actions** through better understanding of JCL jobs and its functionality.	**Boosting efficiency and productivity for system programmers** as they can quickly understand JCL steps with AI-generated explanations, reducing the need for manual research and improving task documentation. **Enhance documentation, knowledge sharing and accelerate onboarding** Generated explanations of the selected JCL steps can be added directly to the code, or saved as other documentation, making them accessible for the entire team and minimizing the need for repeated consultations.

Understand: Begin continuous modernization of your tightly coupled applications



Visualize and auto-document your COBOL application at the enterprise level

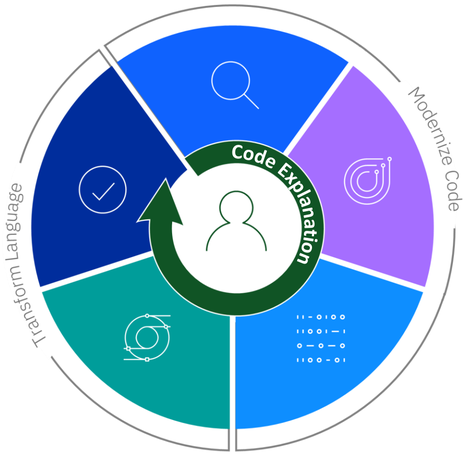
- Start of your application modernization journey with an inventory of applications, resource usage, and dependencies. Leverage COBOL explanation to improve understanding
- Build business alignment and confirm that your understanding of the application is valid – ensuring modernization efforts achieve expectations
- Mitigate the challenge of lack of application SMEs with automated analysis & visualized application flows to enable accelerated application understanding



Application Discovery is the starting point for z/OS application modernization

- Deep enterprise application analysis
- Auto discovery of data and program relationships
- Enable incremental refactoring of business services

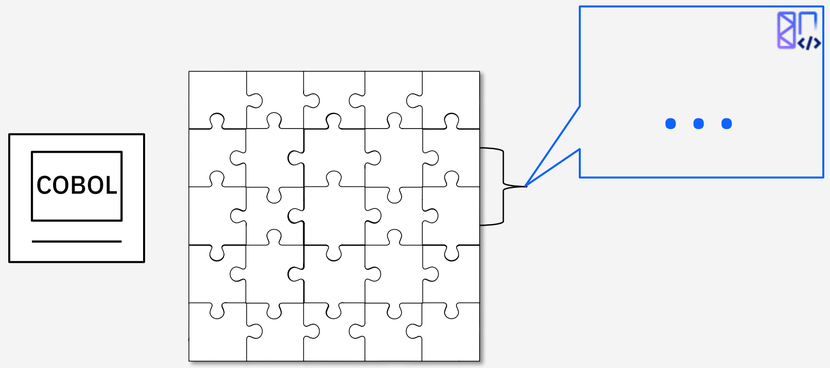
Code explanation: Understand and document your application faster



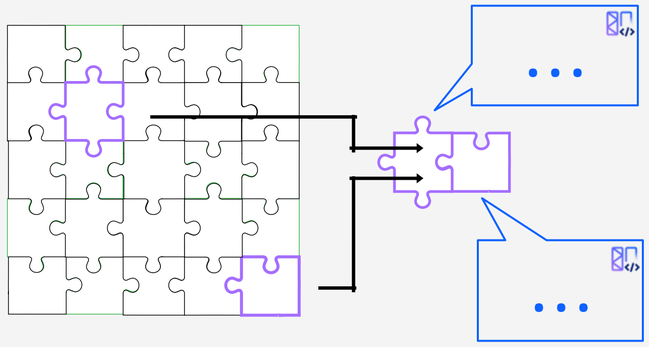
Leverage Generative AI for a natural language explanation of your COBOL code

- **Narrow the knowledge gap**: Real-time COBOL code explanations aid developers, accelerating development or modernization efforts
- **Free up SMEs:** Less reliance on senior experts frees them for advanced work, reducing knowledge bottlenecks via real-time code explanations
- **Streamline documentation**: Utilize code explanations to update application knowledge, reducing manual efforts
- **Facilitate modernization strategy**: Architects gain deeper insights into COBOL programs, aiding in identifying optimal modernization approaches

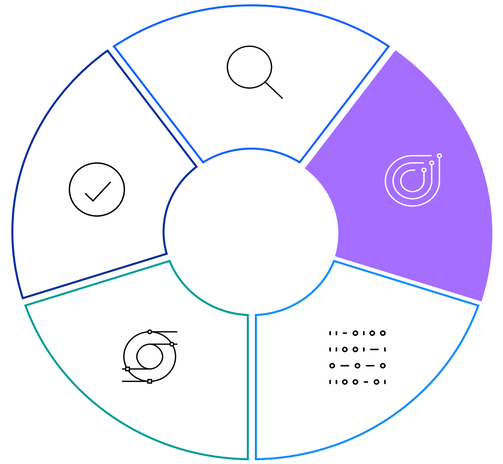
Monolithic application



Mainframe Application with Business Services



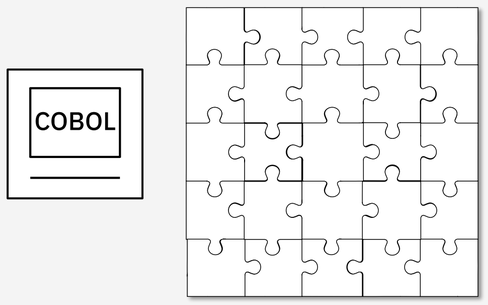
Refactor: Automated tooling to identify services within an application to modernize



Discover programs and data needed for a refactored business service within a large application

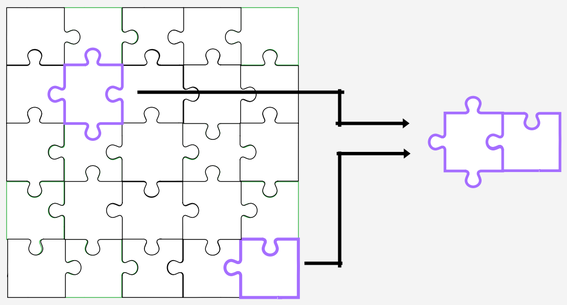
- Separate code needed into a refactored service which will be easier to maintain and reuse
- Automate the service creation process to improve accuracy and reduce time and skill required for manual developer analysis
- Unlock modernization development agility and ease of integration

Monolithic application

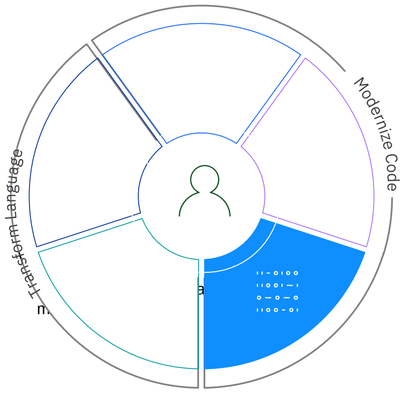


New automated refactoring capability

New Refactoring Assistant can quickly identify parts of an application to refactor and extract into modular, reusable services via deep functional analysis of the source code.



Optimize: Optimize your COBOL code with prioritized performance insights



Performance analysis and recommendations Conducts in-depth analysis of COBOL modules through static and dynamic analysis, providing actionable recommendations to optimize performance.

Source-code matching
Offers line-to-line analysis for targeted fixes and enhancements, ensuring precise improvements, all within your IDE.

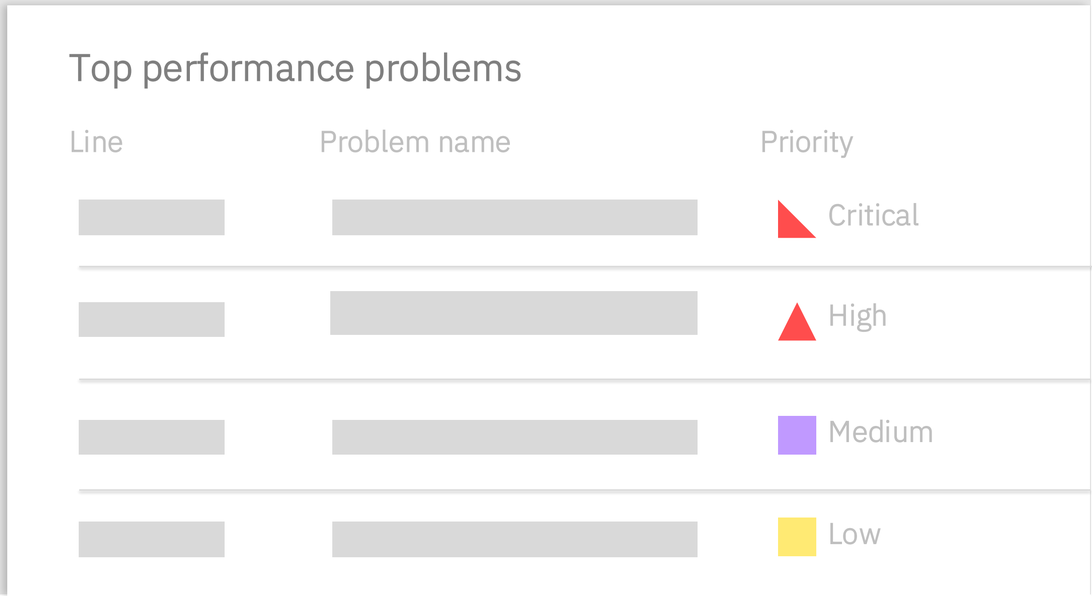
Ranking and prioritization
Ranks performance issues based on impact, enabling developers to focus on high-priority tasks for maximum efficiency.

Benefits of Performance Insights

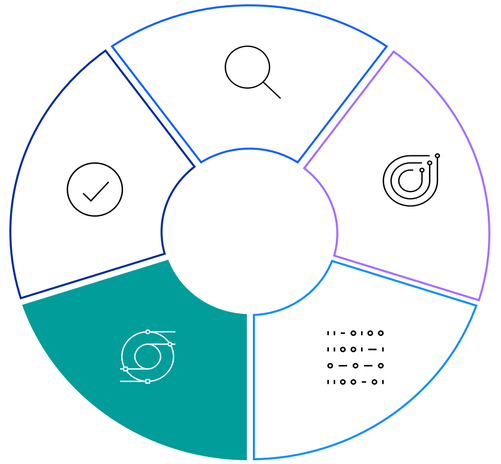
Save time, money, and resources through early detection and problem resolution

Reduce skill gap by allowing entry level developers to fix performance issues independently.

Deliver robust and efficient COBOL applications by quickly detecting and fixing issues.



Transform: Leverage generative AI to accelerate COBOL to Java conversion

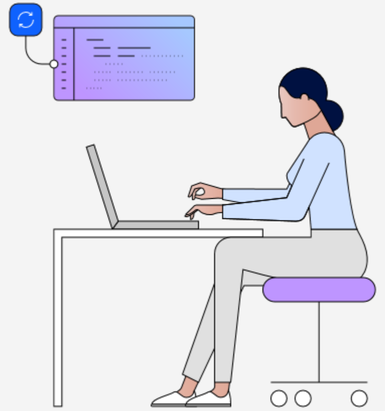


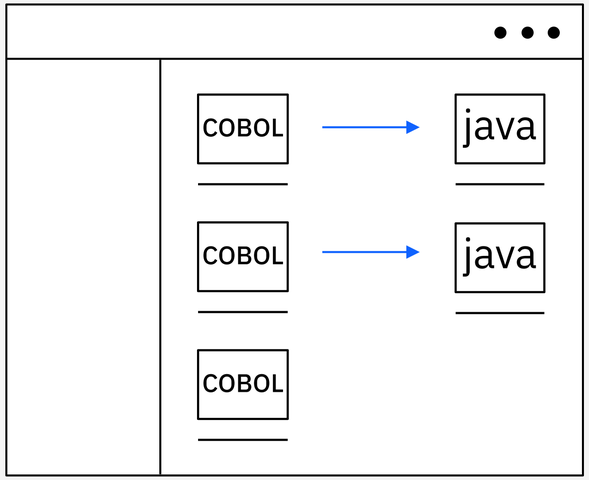
AI assistant to generate Java code in minutes, not months

- Generative AI to build data structures and business logic in Java from your refactored COBOL code
- Well-architected object-oriented Java – not JOBOL
- Maintains IBM Z runtimes and qualities of services with interoperability, integration, and enterprise standardization

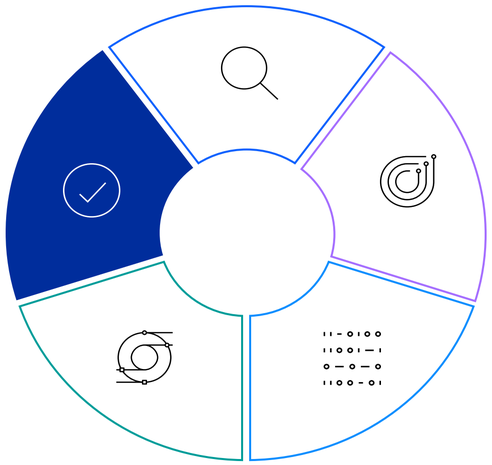
IBM watsonx Code Assistant for Z

- State of the art granite.20b.code large language model with a 32k token context window
- Trained with 1.6T tokens across 115 programming languages
- Tuned for Cobol to Java use case





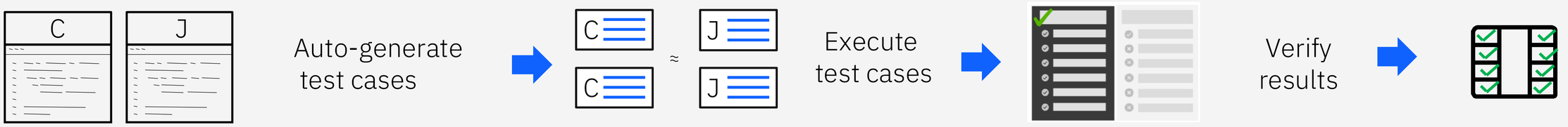
Validate: Automated testing capability



Streamlined and accelerate testing of new code

- Auto generated testing to compare semantic equivalence of new Java service to, providing confidence in a successful Java translation and de-risking the process
- Accelerate developer productivity
  - Enables incremental testing if the Java code is working vs waiting to test broader code path flows in a later test cycle where it's harder to determine errors
  - Tool automation automating tests and enabling them to run in isolation without requiring the middleware to execute the test
  - Junit tests generated can be reused and integrated in the DevOps pipeline per standard practice as the application evolves

Validation Scenario: Tests compare COBOL paragraph and Java method verify equivalence



Uses AI to generate the tests using the same input/output data for both the COBOL and Java tests

Automatic mocking enables unit tests to run on z/OS in isolation without middleware!

Vision and roadmap

Vision:

Code generation

Generate an object-oriented Java equivalent service from an enterprise COBOL service

Code validation

Generate test cases to validate a new service & surrounding application

Code explanation

Generate natural language explanations of COBOL or JCL

Code optimization

Review a COBOL or Java service and help make it better

Roadmap:

2024

Planned Highlights

- PL/I support
- Ongoing z/OS subsystem support

IBM watsonx Code Assistant for Z: Anticipated roadmap highlights

Delivered Capabilities

- Application Understanding
- Application Refactoring: COBOL
- Integrated VS Code Experience for Refactor, Transform, and Validate
- COBOL to Java Transformation (including subsystem support) available hybrid or fully on-premises
- COBOL to Java Transformation Validation
- COBOL Code Explanation (SaaS)

3Q '24

Intended Capability	Outcome
Code Explanation: JCL (SaaS)	Generative AI capability to summarize and explain JCL as written English to enhance understanding, aid skills transfer and enable more efficient hand-over between System Programmers.
Code Optimization: COBOL	Provide a developer with a prioritized set of recommendations to optimize performance of their COBOL application.
Code Explanation: COBOL (on premises)	Clients can deploy LLM for COBOL Code explanation on their premises.

4Q '24

Intended Capability	Outcome
Code Explanation: PL/I (SaaS)	New generative AI capability to summarize and explain PL/I program source code as written English to enhance skills transfer, application understanding, and documentation
Code Explanation Eclipse support	Code explanation capabilities for all supported languages can be accessed in IDz and ADDI interfaces
Simplified User Experience	Consolidation of VS Code plugins and simplification of user experience resulting in a more efficient modernization journey
Code Refactor: Dynamic analysis	Leverage dynamic trace data to incorporate business logic insights when determining refactoring scope, and reduce reliance on application SMEs

Targeted for 2025

- Additional Generative-AI infusion within delivered capabilities (integrated chat, customization)
- Code generation for mainframe languages starting with COBOL
- Code transformation for additional mainframe languages (e.g. PL/I to Java, JCL to Ansible)
- PL/I Application Refactoring
- Assembler support
- Improved user experience and simplification
- Cloud location expansion (SaaS)
- And more...

Submit or vote on new requirements here:

https://ibm-data-and-ai.ideas.ibm.com/

Continuous mainframe-specific model enhancements / fine tuning plus ongoing improvements to delivered capabilities

© 2024 IBM Corporation

Roadmap is subject to change

Next steps

Show me → Initial Project → Scale Delivery

Briefing & demos

- Overview
- Demo
- Next steps

1 Hour

Solution workshop(s)

- Use case alignment
- Pilot scope
- Define success criteria

2-8 Hours

Velocity pilot

- Deliver pilot scope
- Prove the value
- Knowledge transfer

2-4 weeks

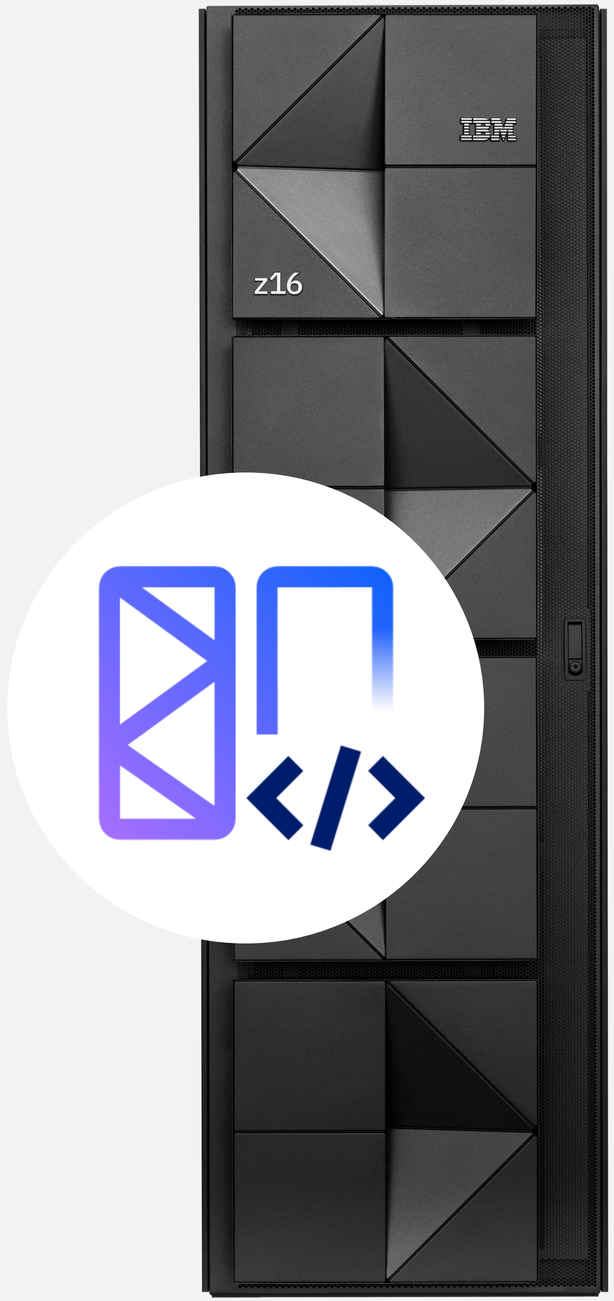
Delivery

- Accelerate and scale
- IBM expertise
- Solution delivery

Get ready to accelerate your application modernization journey

Learn more :

- Read the Accelerate Mainframe Application Modernization with Hybrid Cloud (IBM Redpaper)
- Visit the watsonx Code Assistant for Z webpage
- Request a briefing and demo
- Learn more about IBM Consulting



Notices and disclaimers

© 2024 International Business Machines Corporation. No part of this document may be reproduced or transmitted in any form without written permission from IBM.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. This document is distributed “as is” without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity. IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts.

In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used products and the results they may have.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer’s responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer’s business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.