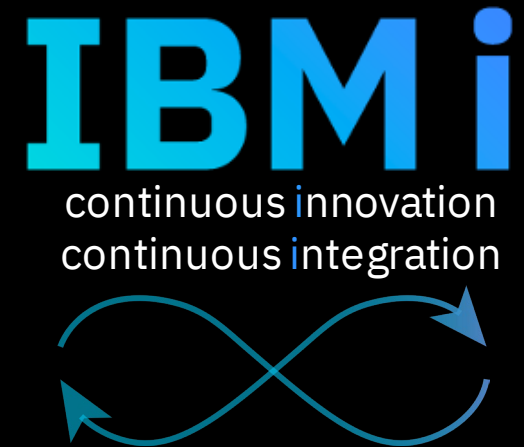


# IBM i & AI – Strategy & Update

Edmund Reinhardt – IBM i App Dev Tools Architect

July 2024



# IBM i + AI

## Db2 Data Analytics

- Trend analysis
- Anomaly detection

## Operations

- Active monitoring / alerting
- Self-healing

## Developer Experience

- Help developer write code
- Understand code

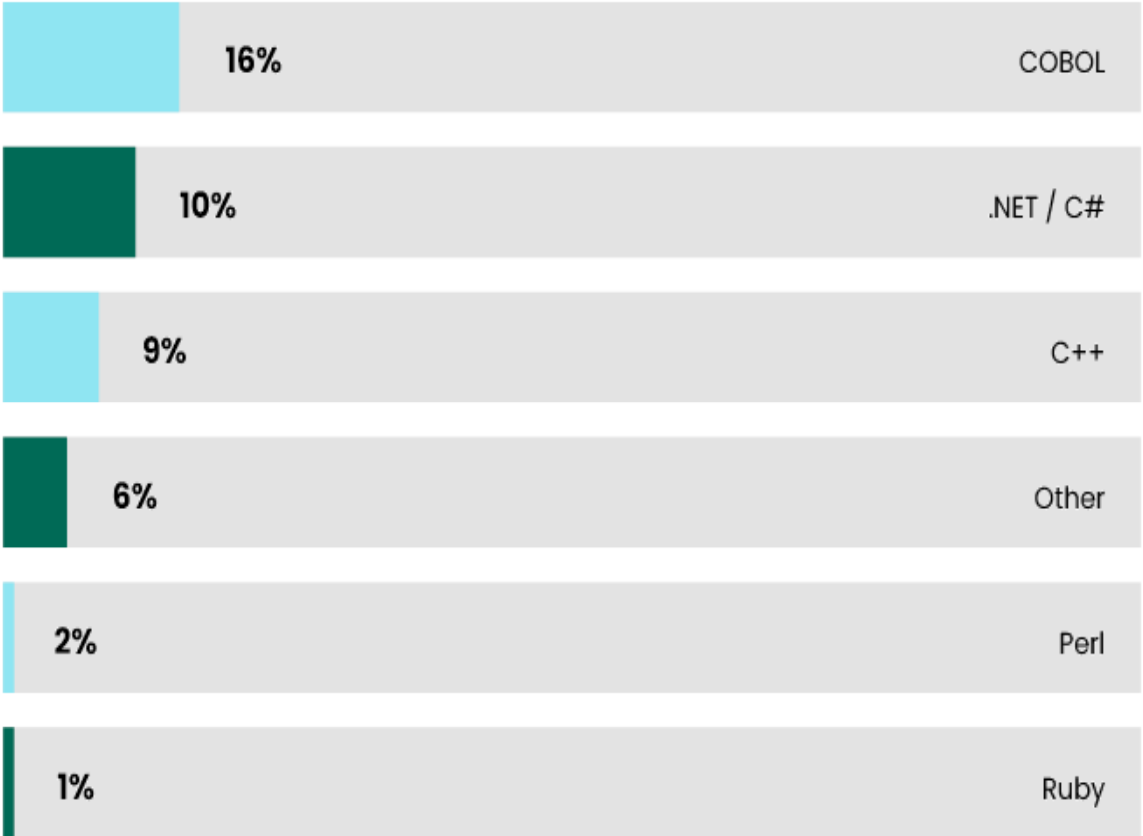
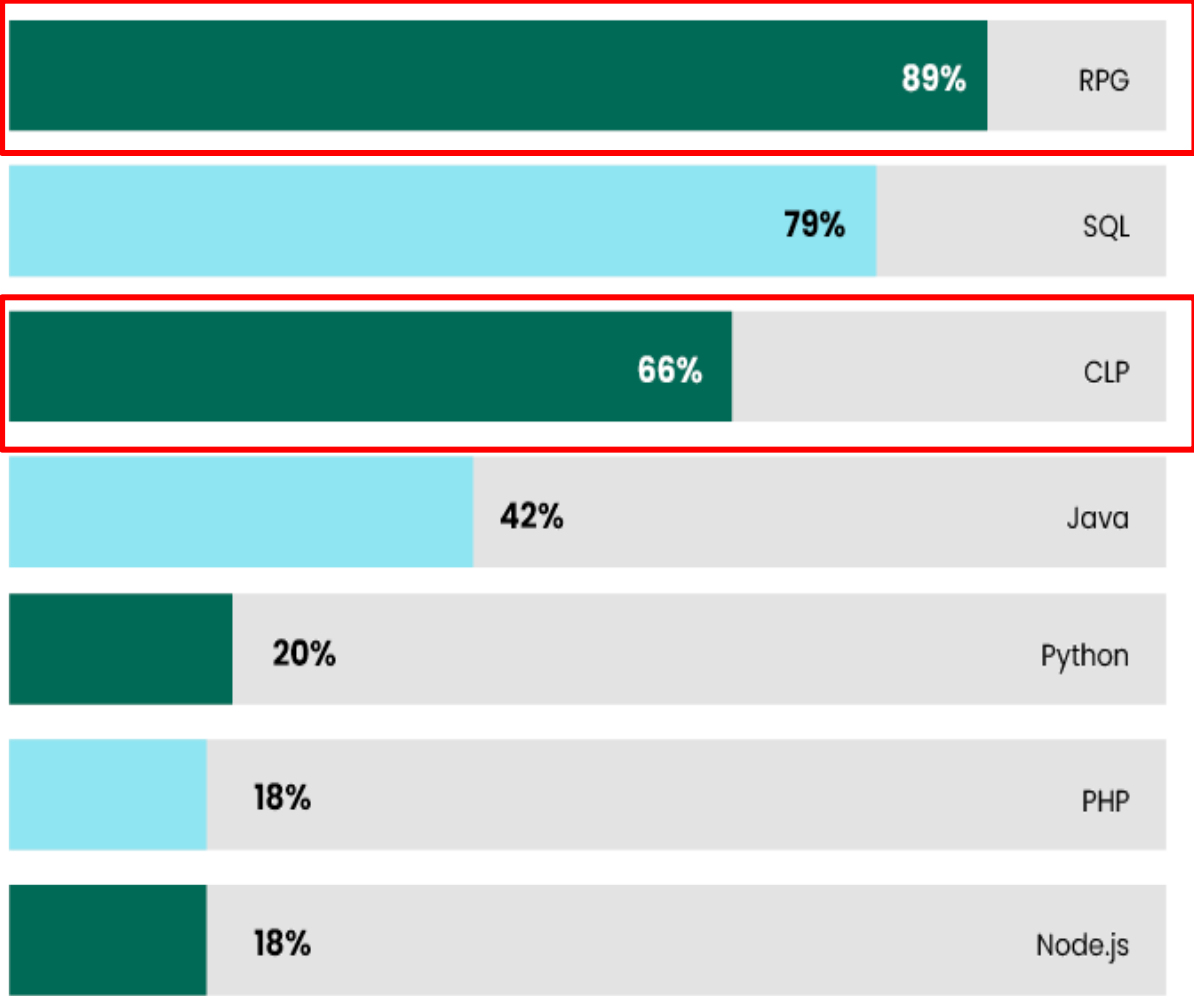
An IBM i Code assist tool should

- Help programmers understand existing RPG
- Answer questions about RPG
- Generate modern free-format ILE RPG based on a description
- Provide content assist to complete the current RPG statement or procedure
- Write test programs for RPG
- ... and, by the way, transform older RPG into modern, ILE-based free-format

It starts with RPG ... most of the time

# Development Languages

Which development languages do you use today for new development on IBM i?



# RPG Assistant Concept: Generate RPG

Given the table PAYROLL defined in the SQL below, generate a free form ILE RPG procedure that would calculate the amount of due to a employee with the given employee\_id who has worked the given amount of hours. Use the employee's pay rate and tax rate that can be retrieved from the PAYROLL table.

```
CREATE TABLE payroll (  
  employee_id INT NOT NULL,  
  pay_rate DECIMAL(7, 2) NOT NULL,  
  tax_rate DECIMAL(7,2) NOT NULL,  
  PRIMARY KEY (id)
```

Press ⌘+Enter to apply

Generate

Clear Input

Clear Output

# RPG Assistant Concept: Generate RPG

## Output:

```
1  **free
2
3  dcl-proc calculate_pay export;
4      dcl-pi calculate_pay packed(12:2);
5          employee_id int value;
6          hours packed(5:2) value;
7      end-pi;
8
9      dcl-s pay_rate packed(7:2);
10     dcl-s tax_rate packed(7:2);
11     dcl-s gross_pay packed(12:2);
12     dcl-s net_pay packed(12:2);
13
14     select pay_rate, tax_rate
15     from payroll
16     where employee_id = :employee_id;
17
18     gross_pay = pay_rate * hours;
19     net_pay = gross_pay - (gross_pay * tax_rate);
20
21     return net_pay;
22 end-proc;
```

## Code-Text. Instruction: Summarize the next RPG code

```
D wMsgSta 1A
D wMsgStaTxt 10A

C GetMsgStatTxt BEGSR
C SELECT
C WHEN wMsgSta = 'D'
C EVAL wMsgStaTxt = 'Deleted'
C WHEN wMsgSta = 'R'
C EVAL wMsgStaTxt = 'Read'
C WHEN wMsgSta = 'U'
C EVAL wMsgStaTxt = 'Unread'
C ENDSL
C ENDSR
```

The RPGLE code defines a subroutine called "GetMsgStatTxt" that translates a message status code into a textual description. There are two variables declared at the start of the program: 'wMsgSta' which is a one-character variable to hold the message status, and 'wMsgStaTxt' which is a ten-character variable intended to hold the textual description of the status.

The 'GetMsgStatTxt' subroutine uses a 'Select' block to determine what the content of 'wMsgStaTxt' should be based on the value of 'wMsgSta'. If 'wMsgSta' is 'D', 'R', or 'U', the corresponding text ('Deleted', 'Read', or 'Unread', respectively) is assigned to 'wMsgStaTxt'. There are no other actions or alternative cases specified for other potential values of 'wMsgSta'.



# RPG Assistant Concept: Generate Unit Test

Given the procedure and DDL below, please write a unit test that will exercise a number of edge cases for this procedure. In the unit include the SQL DML to get the SQL into the desired state.

# RPG Assistant Concept: Generate Unit Test

```
// Test case 1: Employee with ID 1, hours worked = 10  
employee_id = 1;  
hours_worked = 10;  
calculate_pay(employee_id: hours_worked);
```

Exec SQL

```
SELECT pay_date, gross_pay, tax_paid, net_pay  
INTO :pay_date, :gross_pay, :tax_paid, :net_pay  
FROM payroll  
WHERE employee_id = :employee_id;
```

```
AssertEqual(pay_date: %Date(): 'Pay date should be today');  
AssertEqual(gross_pay: 100.00: 'Gross pay should be 100.00');  
AssertEqual(tax_paid: 20.00: 'Tax paid should be 20.00');  
AssertEqual(net_pay: 80.00: 'Net pay should be 80.00');
```

And it keeps going...

# RPG Assistant Concept: Generate Unit Test

```
Dcl-Proc calculate_pay_test;  
  Dcl-PI calculate_pay_test;  
  End-PI;  
  
  Exec SQL  
    DELETE FROM employee;  
  Exec SQL  
    DELETE FROM payroll;  
  
  Exec SQL  
    INSERT INTO employee (employee_id, pay_rate, tax_rate)  
    VALUES (1, 10.00, 0.20);  
  
  Exec SQL  
    INSERT INTO employee (employee_id, pay_rate, tax_rate)  
    VALUES (2, 15.00, 0.15);  
  
  Exec SQL  
    INSERT INTO employee (employee_id, pay_rate, tax_rate)  
    VALUES (3, 20.00, 0.10);  
  
  Exec SQL  
    INSERT INTO employee (employee_id, pay_rate, tax_rate)  
    VALUES (4, 25.00, 0.05);  
  
  Exec SQL  
    INSERT INTO employee (employee_id, pay_rate, tax_rate)  
    VALUES (5, 30.00, 0.00);
```

# IBM i Approach – Involve the Community

- IBM i is launching a project to train a “ready for prime time” AI language model-based code assistant
  - Using the RPG code IBM has developed
  - Using code donated by experts & Champions
    - Susan Gantner, Jon Paris, Scott Klement, Jim Buck, Paul Tuohy, Niels Liisberg
    - Yvonne Enselman, Mats Lidström, Koen DeCorte, Steve Bradshaw
    - ... and we hope to have many more!
- Soliciting code for training from the entire IBM i community
  - That could mean YOU!
- The project for a community-trained Large Language Model has begun!

# So How Do You Get Involved?

- Let us know you want to help
  - e-mail [AlforIBMi@ibm.com](mailto:AlforIBMi@ibm.com)
- Agree to the license
  - We'll send it to you – it says we can use your submitted material to train the model
- Submit code and/or training data
  - Information on how is found at: <https://ibm.github.io/rpg-genai-data/#/>
  - You decide if others (outside IBM) can see & use your code or not
- At some point in the future, we will likely ask for volunteers to actually use the submissions in training & evaluating



Welcome

**Overview**

How to contribute training data

Training Data Format

Task Description and Examples

Explain Existing RPG

Generate new RPG

Generate Unit Tests for Existing RPG

Ask questions about RPG

Fixed to Free Format

Generate a more modern version of RPG

How to contribute RPG source

How to manually contribute to IBM Repo

How to use UI to contribute training data

## Training RPG Generative AI

IBM would like to train generative AI models to understand RPG and IBM i application development. We recognize that many customers have valuable assets and expertise and we want to collaborate on building an AI that will help us all.

### How to contribute

1. Email [AlforIBMi@ibm.com](mailto:AlforIBMi@ibm.com) to sign up to contribute to training this AI
  2. IBM will reach out and ask you to sign the appropriate legal agreements
  3. Once you have signed the appropriate agreements, please provide your github id to the same email [AlforIBMi@ibm.com](mailto:AlforIBMi@ibm.com). You can create one [here](#)
  4. IBM will add you as a contributor to the git repository <https://github.com/AlforIBMi/rpg-genai-data> where we gathering all of the training data.
  5. Now it is time to unleash your expertise. Follow the instructions below
- We would prefer contributing actual question and answer pairs to train the AI following [these instructions](#). We can start fine tuning our model with this data right away.
  - It will also be useful to contribute RPG source code following [these instructions](#). If we get sufficient source we can do a general unlabeled training of our model.
  - In either case, contributing to the IBM git repository for this data should follow [this process](#)

<https://ibm.github.io/rpg-genai-data/#/>



Welcome

Overview

How to contribute training data

Training Data Format

Task Description and Examples

Explain Existing RPG

Generate new RPG

Generate Unit Tests for Existing RPG

Fixed to Free Format

How to contribute RPG source

How to make contribution IBM Repo

# Training RPG Generative AI

IBM would like to train generative AI models to understand RPG and IBM i application development. We recognize that many customers have valuable assets and expertise and we want to collaborate on building an AI that will help us all.

<https://ibm.github.io/rpg-genai-data/#/>

## Use cases and examples

The scenarios we are initially focused on are

1. [Explaining RPG in natural language](#)
2. [Prototyping new RPG code from a natural language description](#)
3. [Generating RPG unit tests](#)
4. [Asking questions about RPG in natural language](#)
5. [Converting Fixed form RPG to free format](#) The latter is not something we expect to use the AI for as there are excellent deterministic tools to do so. But we feel that this is a very effective way to train the AI.
6. [Modernize old RPG to new RPG](#) In addition we are very interested in modernizing RPG. While this is not going to be our first deliverable, we are eager to start collecting data to train the AI on this high value transformation.



## How to contribute

We would prefer contributing actual question and answer pairs to train the AI following [these instructions](#)

It will also be useful to contribute RPG source code following [these instructions](#)

In either case, contributing to the IBM git repository for this data should follow [this process](#)

# LIVE DEMO!

# There is so much potential after the first functions!

Future, more advanced versions of the LLM could be trained to address

- Converting S/36 RPG, RPG II, etc. into Free-Format ILE RPG
- Modularizing large programs
- Transforming old data definitions and access into modern Db2 & SQL
- Helping modernize when other languages are the starting point
- ... and more!!!

# Questions

# IBM i

continuous innovation  
continuous integration

