Intermediate Topics in Mainframe Application Development

Venkat Balabhadrapatruni
venkatu@us.ibm.com

August 4th, 2014
Session: 15478

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

Purpose and Presentation flow

- Purpose … to present the features in Rational Developer for System z that help organizations enhance code quality and streamline the delivery of new functions into existing code.

- Flow
  - Code quality and governance
    - Importance of code quality and governance
    - Tools available
  - Unit testing
    - Why Unit test ?
    - Tools available

DISCLAIMER

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

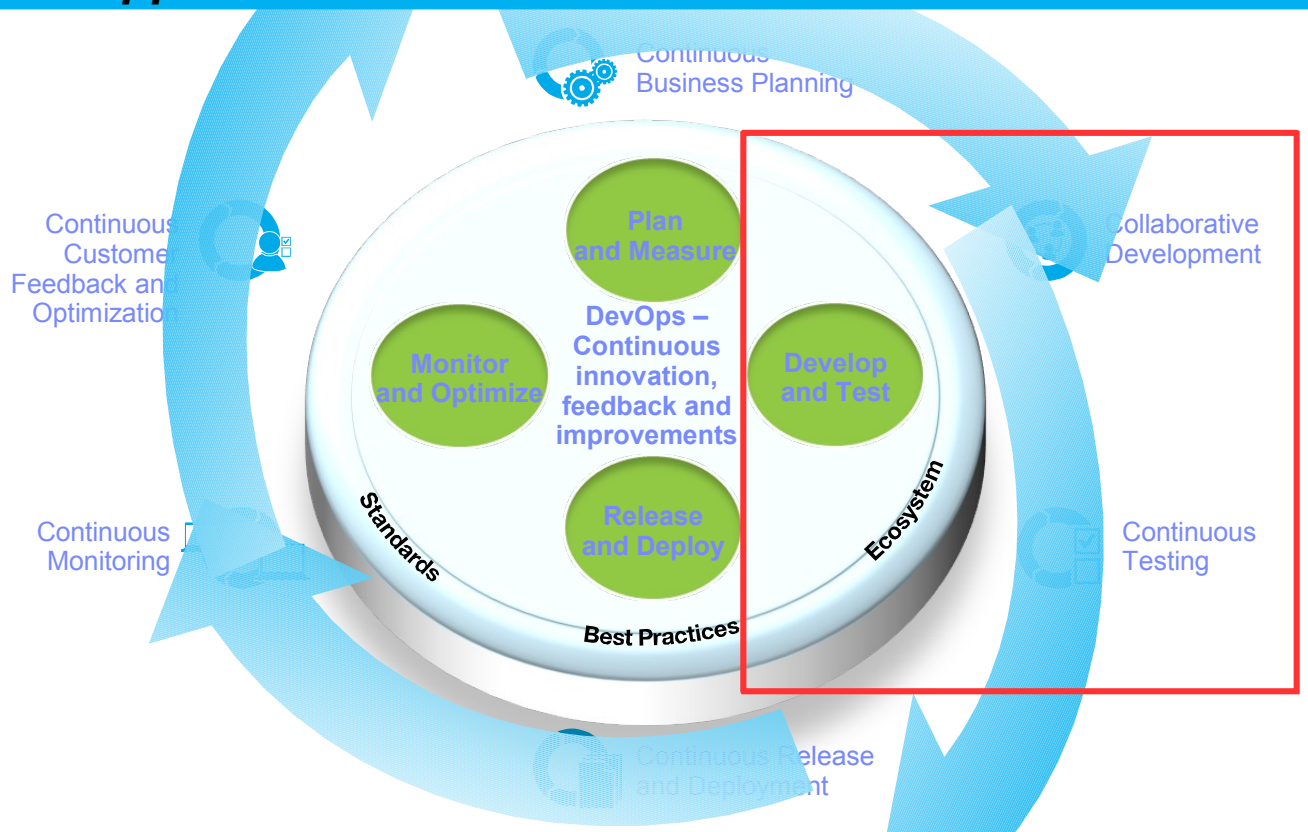
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Information is confidential and must not be shared or redistributed without permission from IBM. Plans are based on best information available and may change in future.

IBM DevOps accelerates enterprise software delivery

Enterprise capability for continuous software delivery that enables you to seize market opportunities and reduce time to customer feedback



Accelerate software delivery – *for faster time to value*

Balance speed, cost, quality and risk – *for increased capacity to innovate*

Reduce time to customer feedback – *for improved customer experience*

Rational Developer for System z:

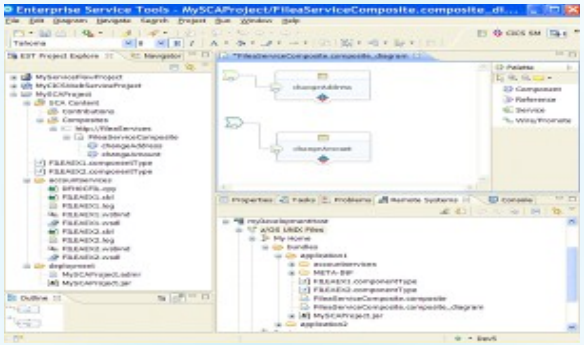
An Integrated Development Environment for System z

Integration with Team Concert for Lifecycle and Source Management

Access to typical System z sub-system functionality in z/OS, CICS, IMS, DB2, WAS

Rational Developer for System z

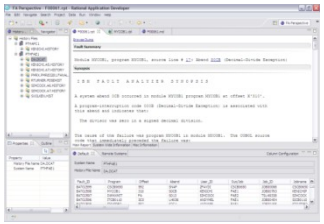
A modern IDE for development of cross-platform applications written in COBOL, PL/I, ASM, Java, EGL or C/C++ in System z CICS, IMS, DB2, Batch applications



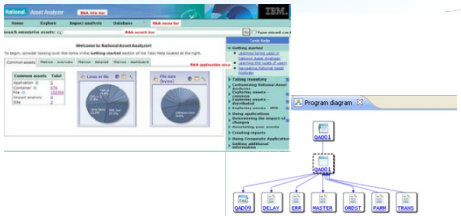
Integration with RD&T for flexible access to System z environment

Robust Mobile Development in conjunction with Worklight

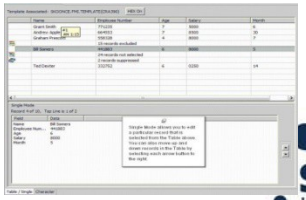
Integration with Fault Analyzer for Dump Analysis



Integration with Asset Analyzer for Application Understanding and Impact Analysis



Integration with File Manager and Fault Analyzer for file and test data handling and Dump Analysis



Why is Code Review capability needed ?

Every organization has standards by which their code must be developed.

- Currently no governance capability when developing and checking COBOL/PLI code into an SCM.
- No tools to help determine if coding best practices were being followed, or if internal guidelines were being followed, etc.
- Most mainframe organizations judge adherence to those standards via time consuming code reviews and manual reporting.

Difficult to report on these non-compliant practices, leaving management with no clear picture of health of source code.

- Companies that obtain code via services contracts have no way to validate the quality of this code being developed for them

Code Review tooling evolution..

RDz 8.5
- Additional COBOL rules provided
- PL/I rules provided

RDz 8.5.0.1
- COBOL Application Model
- Custom COBOL rules

RDz 8.5.1
- Command line invocation
- XML, CSV reports

RDz 8.0.3
- IDE Code Review introduced
- COBOL rules provided
- Select rules
- Run the Analysis
- View results in UI
- HTML, PDF reports

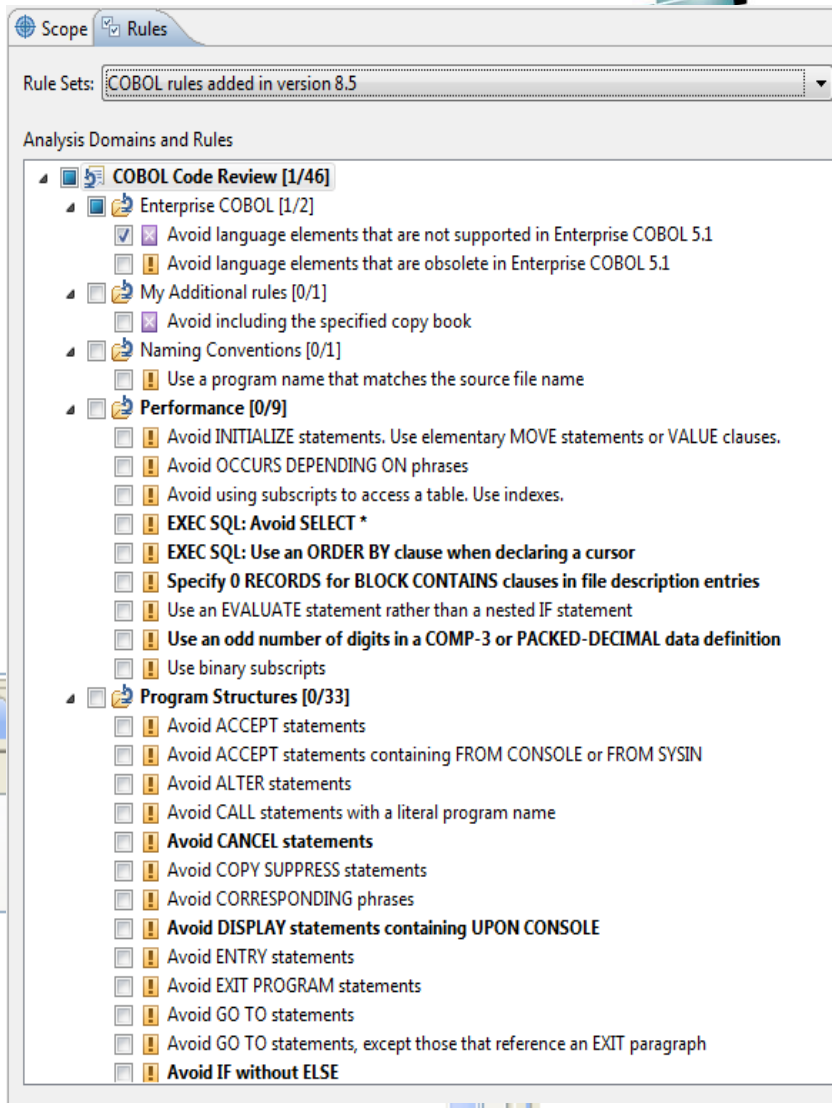
RDz 9.0
- z/OS Batch invocation
- Additional rules provided
- CICS CAM updates

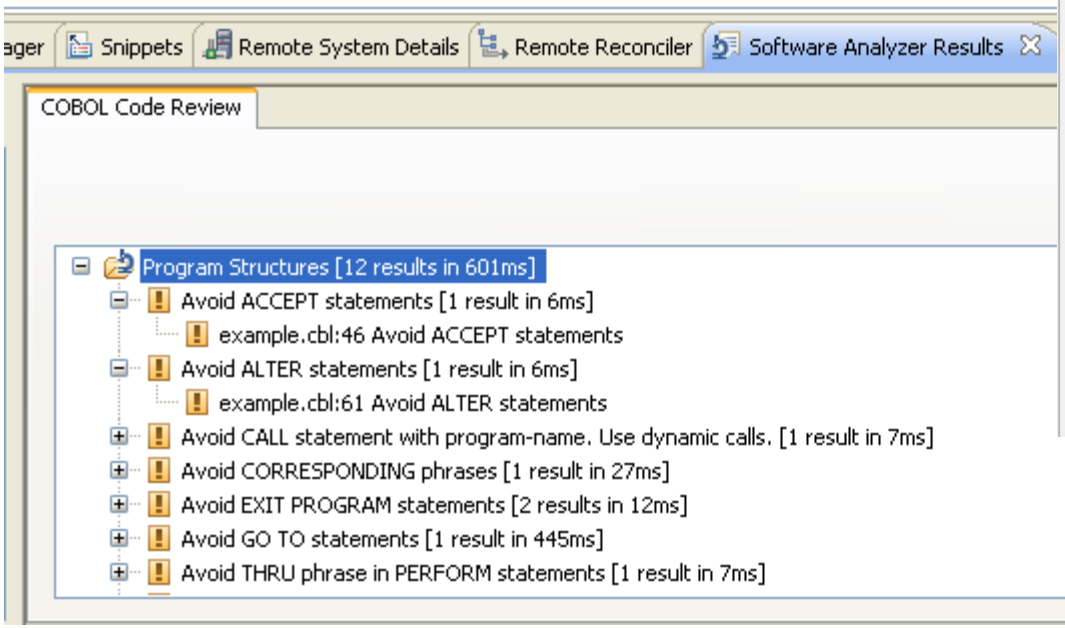
RDz 9.0.1
- z/OS Batch invocation
- Additional rules provided
- Export language specific results

RDz 9.1
- PL/I Application Model
- Custom PL/I rules
- Baseline comparison

Code Review Scenario

- The code review feature is used to identify violations of coding conventions, which are defined by a set of rules
  - Ensuring code quality and conformance
  - What about established, trusted legacy code?
- Running a code review analysis on legacy code may produce large numbers of results
  - Take corrective action? Or ignore?
  - It depends…

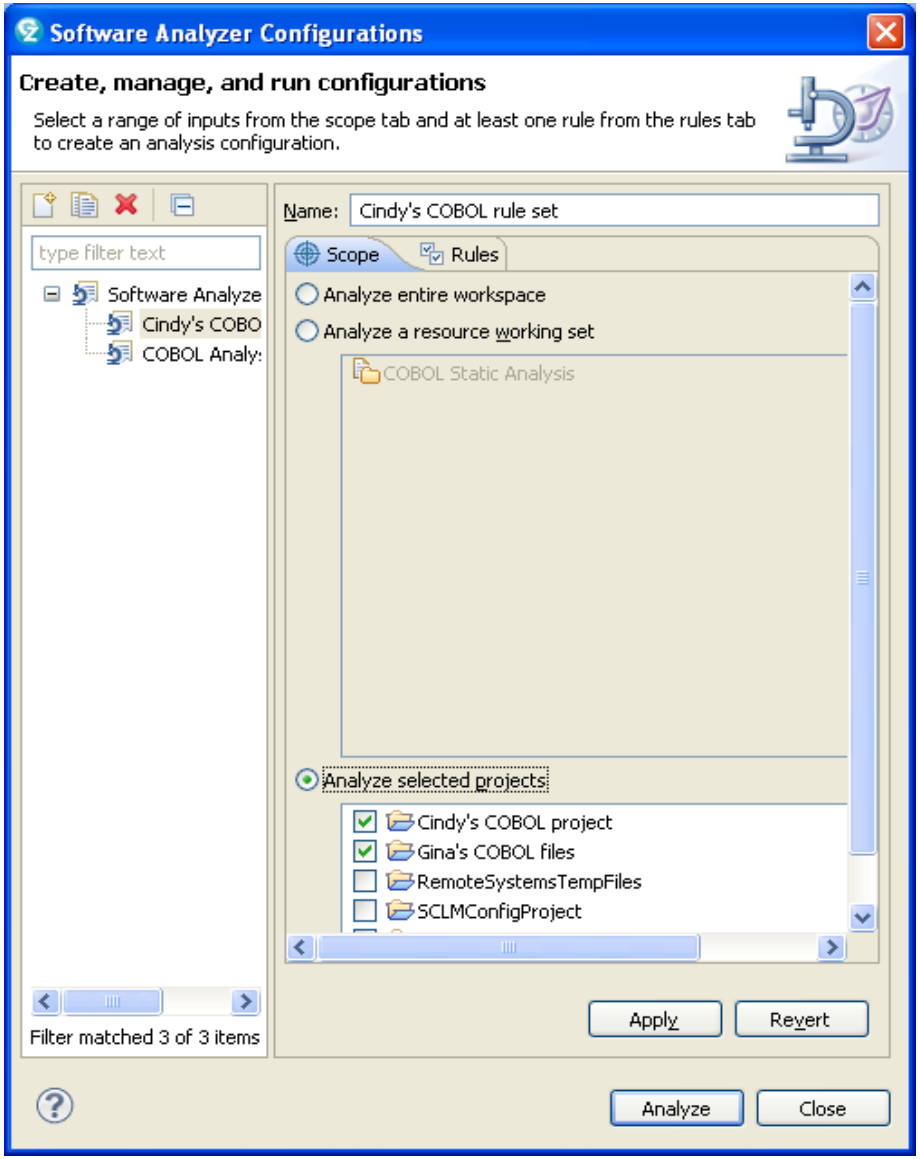


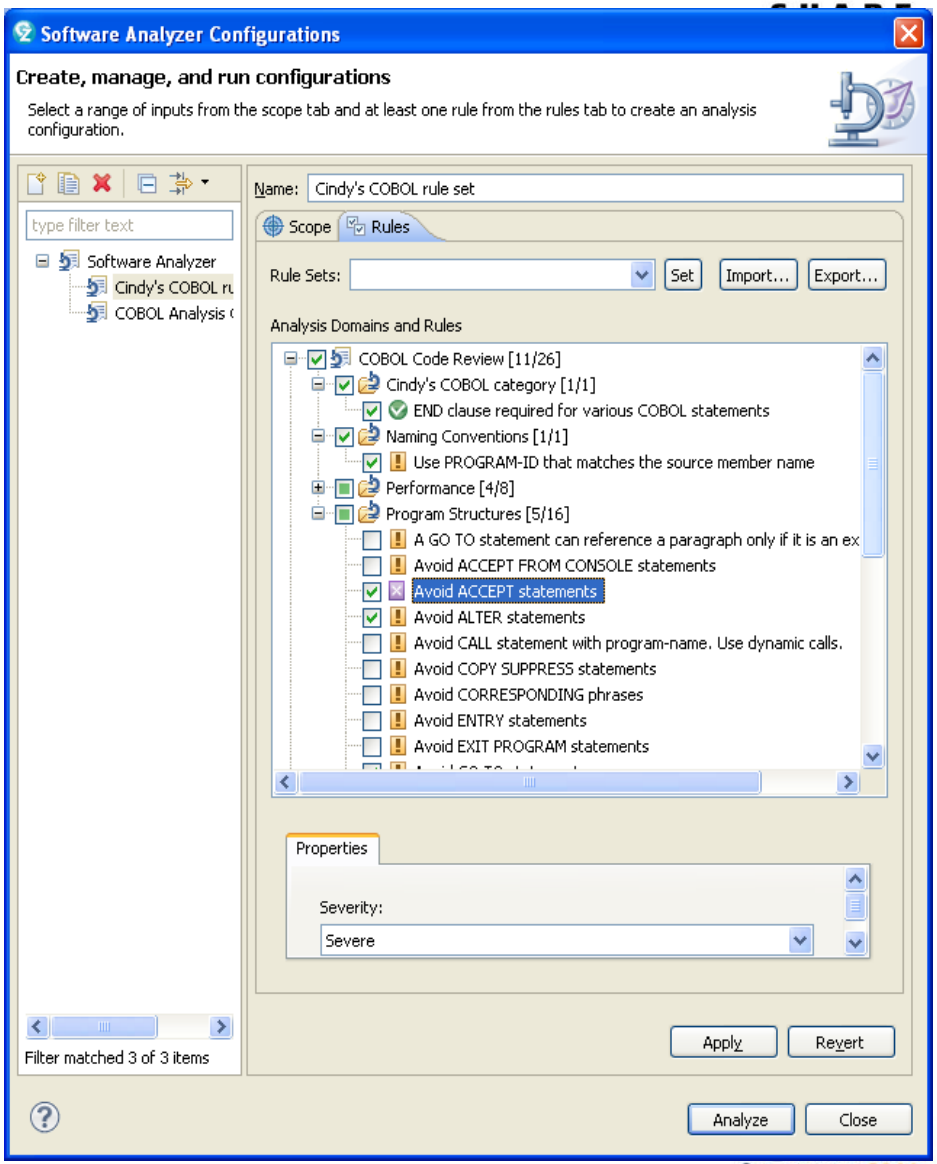


Code review support for COBOL & PL/I

- Rules definition
  - Define analysis configurations, specifying scope and rule set
  - Rules defined/managed using the rules configuration definition editor
    - Select from a list of pre-built COBOL & PL/I rules
    - Select from a list of COBOL &PL/I rules templates and customize parameters
    - Specify severity for each selected rule
    - Scope analysis per file, project, workspace, etc.
- Rules administration (via Push-to-client)
  - Central administration of rule sets
  - Export rules configurations to push to multiple developers

Creating configuration and specifying rules





IDE code review support for COBOL & PL/I

- Analysis
  - Launch the code review from:
    - RDz project (local) - Entire project or on individual file
    - RDz remote file opened in editor session
    - Toolbar option across scope defined in configuration
  - Review results and fix problems
    - Double-click error, source file opened, and line containing violation is highlighted
  - Generate HTML or PDF or CSV report from these results

Code Review example

Software Analyzer Configurations

Create, manage, and run configurations

Select a range of inputs from the scope tab and at least one rule from the rules tab to create an analysis configuration.

type filter text

- Software Analyzer
  - COBOL Performance rules
  - COBOL Program Structure rules
  - My Favorite Coding Conventions

Filter matched 6 of 6 items

Name: My Favorite Coding Conventions

Scope | Rules

Rule Sets: Set Import... Export...

Analysis Domains and Rules

- COBOL Code Review [8/46]
  - Enterprise COBOL [2/2]
  - My Additional rules [0/1]
  - Naming Conventions [1/1]
  - Performance [0/9]
  - Program Structures [5/33]
    - Avoid ACCEPT statements
    - Avoid ACCEPT statements containing FROM CONSOLE or FROM SYSIN
    - Avoid ALTER statements
    - Avoid CALL statements with a literal program name
    - Avoid CANCEL statements
    - Avoid COPY SUPPRESS statements
    - Avoid CORRESPONDING phrases
    - Avoid DISPLAY statements containi
    - Avoid ENTRY statements
    - Avoid EXIT PROGRAM statements
    - Avoid GO TO statements
    - Avoid GO TO statements, except th
    - Avoid IF without ELSE
    - Avoid NEXT SENTENCE phrases
    - Avoid PERFORM, except PERFORM
    - Avoid RESERVE clauses in FILE-CON
    - Avoid STOP RUN and STOP literal st
    - Avoid THRU phrases in PERFORM st
    - Avoid using level-88 entries in data
    - Avoid using more than one EXIT sta
    - Avoid using SECTION in the proced
    - Avoid XML PARSE statements
    - EXEC CICS: Check EIBRESP after NO
    - EXEC CICS: Use DFHRESP to check t
    - EXEC CICS: Use the RESP option

Properties

```
183             PERFORM 1050-SEND-MAP
184
185           WHEN OTHER
186             PERFORM 1010-COPY-COMMAREA
187             MOVE 'Invalid key pressed.' TO MESSAGEOUT
188             PERFORM 1050-SEND-MAP
189
190           END-EVALUATE.
191
192       1000-GET-DATA.
193             EXEC CICS LINK PROGRAM('GAM0VSI')
194                  COMMAREA(INPUTS-OUTPUTS)
195             END-EXEC.
196                                  -COPY-COMMAREA.
197
198
199             MOVE LOW-VALUES TO INVO.
200             MOVE "___" TO SELECTFIELD.
201             MOVE '1' TO ARRAY-INDEX.
```

EXEC CICS: Use the RESP option

Press 'F2' for focus

Analyze | Close

Custom rules support

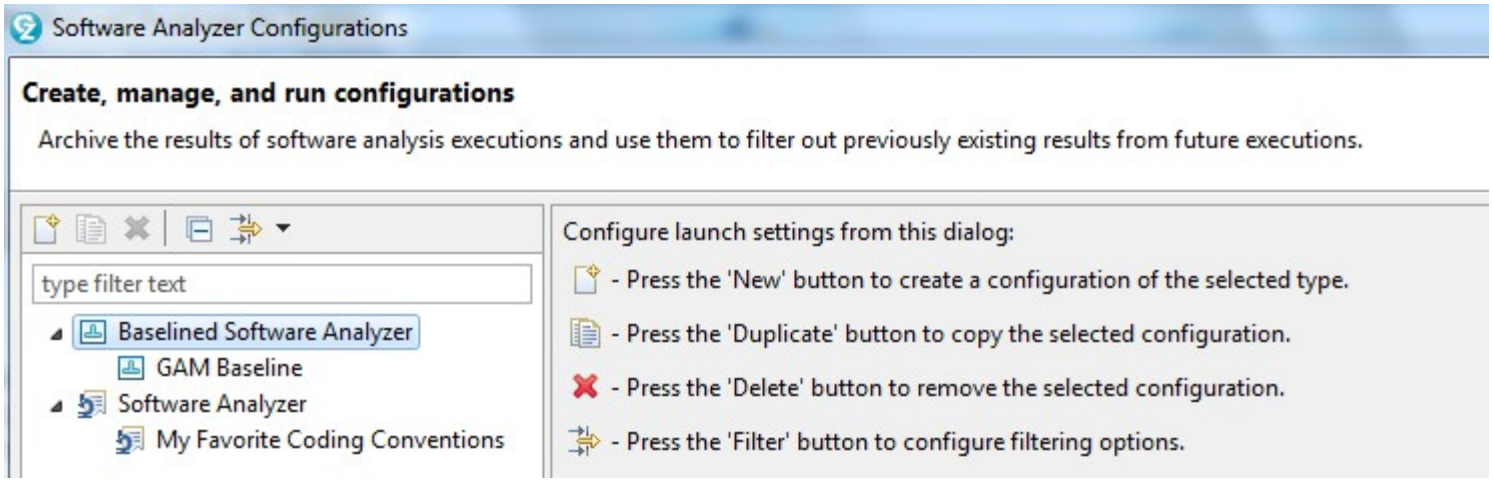
- Most likely you will have coding standards not covered in our list of pre-built and custom templates; therefore, you will need to add your own set of custom rules to the selection lists.
- Process to build your own custom COBOL and/or PLI rule:
  - **Use RDz wizard** (Eclipse PDE new plugin project template) to generate plugin for custom COBOL and/or PLI rules
    - Creates the java plugin project
    - Creates new category(s) to hold all your domain specific rules
    - Adds rules to the categories
    - Creates java class templates for each of your custom rules
  - Rule developer **fills in the template** with java code to implement their custom rule
    - Using RDz published **COBOL and PLIApplication Model API**
  - **Package your plugin** as P2 update site and **install in the RDz** Eclipse environment using Eclipse Software Updater

How to handle “legacy code violations” ?

- Within the RDz Client, you can use baselines to filter out previously existing results using two methods:

  1) Create a new analysis configuration of type “Baselined Software Analyzer“ that will use a baseline archive to filter out results.

  2) From within a code editor, select a previous version of the code you are editing to use as a baseline with your traditional “Software Analysis” configurations.



What is a “Baseline Archive”?

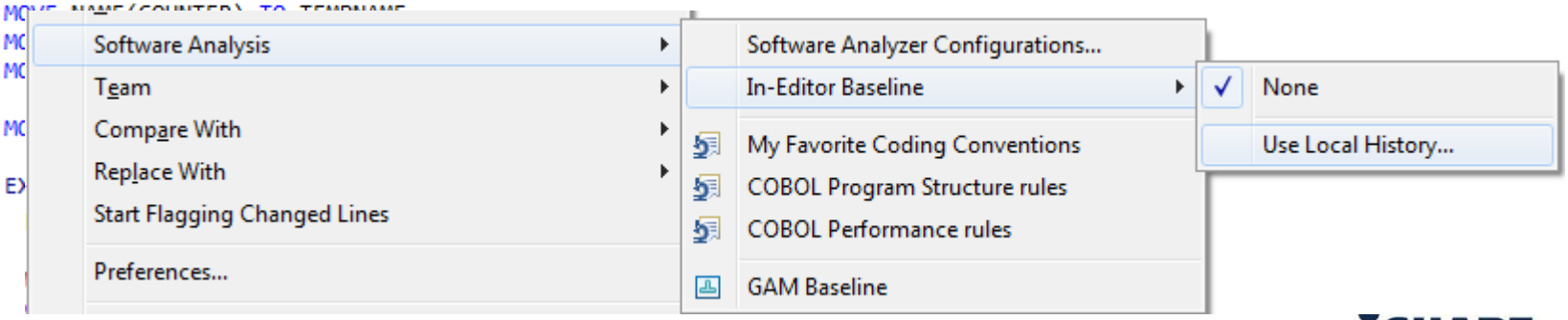
- A ZIP Archive that contains:
  - A list of the software analysis results from an analysis job.
  - Some metadata about the analysis job such as
    - Configuration Name
    - Execution Time
    - Execution Scope
    - Number of Rules in the Configuration
  - Any source file that had an analysis rule violation.

- Note: Because a baseline archive contain copies of proprietary source code, it should be treated with the same care and protections as the actual source code.

Using File History as a Baseline

- Our second use case supports baselined analysis without the use of a baseline archive and will work with both local and remote files.
- From within a code editor, the context menu allows you to select a version from local history of the edited file to use as a baseline:

```
*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--

 2000-CREATE-DEALERSHIP.

     IF SQLCODE = 0
        MOVE 1 TO COUNTER
        PERFORM UNTIL COUNTER > DEALERLENGTH OR SQLCODE NOT = 0
           MOVE DEALERADDRESS (COUNTER) TO TEMPDEALERADDRESS
           MOVE STATE(COUNTER) TO TEMPSTATE
           MOVE ABBR(COUNTER) TO TEMPABBR
           MOVE CITY(COUNTER) TO TEMPCITY
           MOVE REGION(COUNTER) TO TEMPREGION
           MOVE ZIP(COUNTER) TO TEMPZIP
           MOVE PHONE(COUNTER) TO TEMPPHONE
```



z/OS Batch – Code Review overview

- **Separate install**
  - **Rational Developer for System z Host Utilities**

- Code review functionality targeting COBOL and PL/I source code located in PDS's
- Runs the same analysis code and produces the same results as code review on the RDz workstation client
- Implemented as Eclipse-based application running on z/OS
  - zIIP/zAAP eligible Java workload
- Configure using exported artifacts from RDz workstation client (property groups, code review rule set, etc.)
- JCL/REXX front end drives batch processing of Eclipse plug-ins running on Java VM in z/OS UNIX process

Simple Batch example

```
//*
//* invoke code review
//*
//AKGCREV  EXEC PROC=AKGCR,
//             PRM='',
//             PDS=USER91.BATCH.COBOL(CANCEL)
//RULES    DD PATH='/u/user91/snapshot/snapshot.dat'
//CUSTRULE DD PATH='/u/user91/snapshot/snapshot.custom.ccr'
//PROPERTY DD PATH='/u/user91/snapshot/SNAPSHOT.xml'
//EXTMAP   DD PATH='/u/user91/snapshot/snapshot.extensions.zip
//TMPDIR   DD PATH='/tmp/&SYSUID'
//*
```

A single PDS member specified; artifacts exported from RDz workstation client specified as DD's.

Simple Batch example

```
//*
//* invoke code review
//*
//AKGCREV  EXEC PROC=AKGCR,
//
//
//RULES
//CUSTI
//PROPI
//EXTMI
//TMPDI
//*
```

DDNAME	StepName	ProcStep	DSID	Owner	C	Dest	Rec-Cnt
JESMSGLG	JES2		2	USER91	A	LOCAL	19
JESJCL	JES2		3	USER91	A	LOCAL	388
JESYSMSG	JES2		4	USER91	A	LOCAL	104
STDOUT	CREATE		101	USER91	A	LOCAL	1
SYSTSPRT	AKGCREV	TEST	103	USER91	A	LOCAL	51
SUMMARY	AKGCREV	TEST	104	USER91	A	LOCAL	8
CSV	AKGCREV	TEST	105	USER91	A	LOCAL	4
XML	AKGCREV	TEST	106	USER91	A	LOCAL	24
WORKSPCE	AKGCREV	TEST	107	USER91	A	LOCAL	48
MSGS	AKGCREV	TEST	108	USER91	A	LOCAL	6
ERRMSGS	AKGCREV	TEST	109	USER91	A	LOCAL	40
STDOUT	DELETE		112	USER91	A	LOCAL	1

Output reports and logs written to DD's.

Simple Batch example

```
//*
//* invoke code review
//*
//AKGCREV  EXEC PROC=AKGCR,
//
//
//RULE
//CUSTI
//PROPI
//EXTMI
//TMPDI
//*
```

```
DDNAME   StepName ProcStep DSID Owner    C Dest                Rec-Cnt
JESMSGLG JES2                 2 USER91   A LOCAL                    19
JESJCL   JES2                 3 USER91   A LOCAL                   388
JESYSMSG JES2                 4 USER91   A LOCAL                   104
STDOUT
SYSTSP
SUMMAR
CSV
XML
WORKSP
MSGS
ERRMSG
STDOUT
```

```
<report>
 <provider id="codereview.cobol.analysisProvider" name="COBOL Code Review">
  <category id="codereview.cobol.analysisProvider.SNAPSHOT" name="SNAPSHOT">
   <rule id="codereview.cobol.rules.template.NestedIfLimitRule.1366121813492" na
    <result fileId="1" line="17" />
    <result fileId="1" line="20" />
   </rule>
  </category>
  <category id="codereview.cobol.category.structure" name="Program Structures">
   <rule id="codereview.cobol.rules.CancelRule" name="Avoid CANCEL statements" s
    <result fileId="1" line="27" />
   </rule>
  </category>
 </provider>
 <provider id="codereview.pl1.analysisProvider" name="PL/I Code Review">
  <category id="codereview.pl1.category.structure" name="Program Structures">
   <rule id="codereview.pl1.rules.GoToRule" name="Avoid most GO TO statements" s
```

Reports generated are the same format as reports generated by RDz workstation client with command line invocation (XML report shown).

Purpose and Presentation flow

- Purpose … to present the features in Rational Developer for System z that help organizations enhance code quality and streamline the delivery of new functions into existing code.

- Flow
  - Code quality and governance
    - Importance of code quality and governance
    - Tools available
  - Unit testing
    - Why Unit test ?
    - Tools available

The value of early and extensive testing

"80% of development costs are spent identifying and correcting defects" **

During the
Coding or
Unit Testing
phases

$80/defect

During the
BUILD phase

$240/defect

During
Quality Assurance
or the System Test
phases

$960/defect

Once released
into production

$7,600/defect
+
Law suits, loss
of customer trust,
damage to brand

**National Institute of Standards & Technology

Source: GBS Industry standard study

Defect cost derived in assuming it takes 8 hours to find, fix and repair a defect when found in code and unit test.
Defect FFR cost for other phases calculated by using the multiplier on a blended rate of $80/hr.

Unit Testing Focusses on...

Creation, Automation of testing during the CODING phase

What is Unit Testing?

Unit Testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. *Wikipedia*

An *individual unit of software* is a single test-able logic construct or routine within a call-able program:

- Date validation
- Credit Card number look-up
- Tax computation
- Co-pay calculation

This method of testing is sometimes called "white Box testing"

(See Slide Notes for more on White Box Testing)

SAM2.cbl

```
      ---+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--
000084
000085       100-VALIDATE-TRAN.
000086           EVALUATE TRAN-CODE
000087               WHEN  'UPDATE '
000088                   CONTINUE
000089               WHEN OTHER
000090                   MOVE 'N' TO TRAN-OK
000091                   MOVE 'REQUEST TYPE IS INVALID' TO TRAN-MSG
000092           END-EVALUATE.
000093           EVALUATE TRAN-FIELD-NAME
000094               WHEN 'BALANCE '
000095               WHEN 'ORDERS  '
000096                   IF TRAN-UPDATE-NUM-HH  NOT NUMERIC
000097                       MOVE 0 TO TRAN-UPDATE-NUM-HH
000098                   END-IF
000099                   MOVE 'N' TO WS-UPDATE-NUM-NEG
000100                   IF TRAN-UPDATE-NUMX1 = '-'
000101                       MOVE 'Y' TO WS-UPDATE-NUM-NEG
000102                       MOVE '0' TO TRAN-UPDATE-NUMX1
000103                   END-IF
000104                   IF TRAN-UPDATE-NUMX1 = '+'
000105                       MOVE '0' TO TRAN-UPDATE-NUMX1
000106                   END-IF
000107                   IF TRAN-UPDATE-NUM NOT NUMERIC
000108                       MOVE 'N' TO TRAN-OK
000109                       MOVE 'DATA IS NOT NUMERIC' TO TRAN-MSG
000110                   ELSE
000111                       MOVE TRAN-UPDATE-NUM TO WS-UPDATE-NUM
000112                       IF WS-UPDATE-NUM-NEG = 'Y'
000113                           COMPUTE WS-UPDATE-NUM = WS-UPDATE-NUM * -1
000114                       END-IF
000115                   END-IF
000116           END-EVALUATE .
000117           EVALUATE TRAN-ACTION
000118               WHEN 'REPLACE '
000119               WHEN 'ADD     '
000120                   CONTINUE
000121               WHEN OTHER
000122                   MOVE 'N' TO TRAN-OK
000123                   MOVE 'INVALID ACTION CODE   ' TO TRAN-MSG
```

Why bother to Unit Test?

By testing individual logic routines in your programs:

- You can move through the lifecycle more quickly, because you have precise feedback about separate logic routines
  - So you can better understand cause & effect
  - And you know **that** your code works and you know **how** your code works, which gives you confidence to make enhancements and modifications
- Because you execute zUnit Tests through JCL:
  - The testing can be **automated**
  - The end-to-end process takes less time than interactive debugging.
  - And it can be more **systematic**

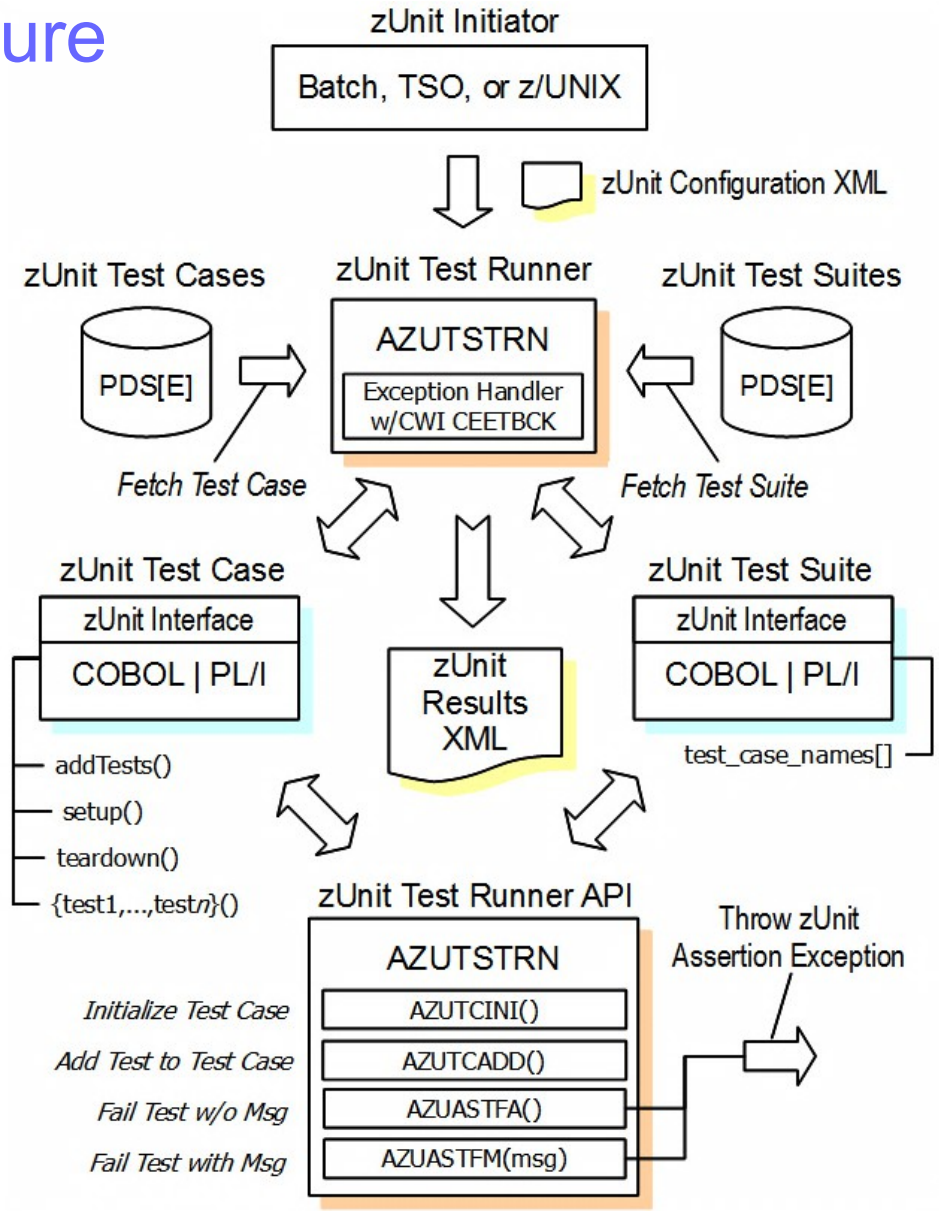
- By isolating and verifying code fragments Unit Testing allows you to understand “cause & effect” in your program logic

How do we unit test ?

The zUnit feature of Rational Developer for System z provides a code-driven unit testing framework for Enterprise COBOL and PL/I. zUnit provides an automated solution for executing and verifying Enterprise COBOL and PL/I unit test cases that are written using the zUnit framework.

- An xUnit instance for Enterprise COBOL and PL/I on System z.
- Test cases can be written in either COBOL or PL/I.
- Provides generation of COBOL or PL/I test case templates.
- Can run a sequence of test cases, mixing COBOL and PL/I is OK.
- Test cases must be LE-enabled batch applications and built into PDSEs.
- Provides a simple fail-type assertion API for COBOL and PL/I.
- Simple test runner configuration XML specifies which test cases to run.
- Comprehensive test runner results XML provides detailed test results.
- Eclipse viewers/editors for the configuration and results XML formats.

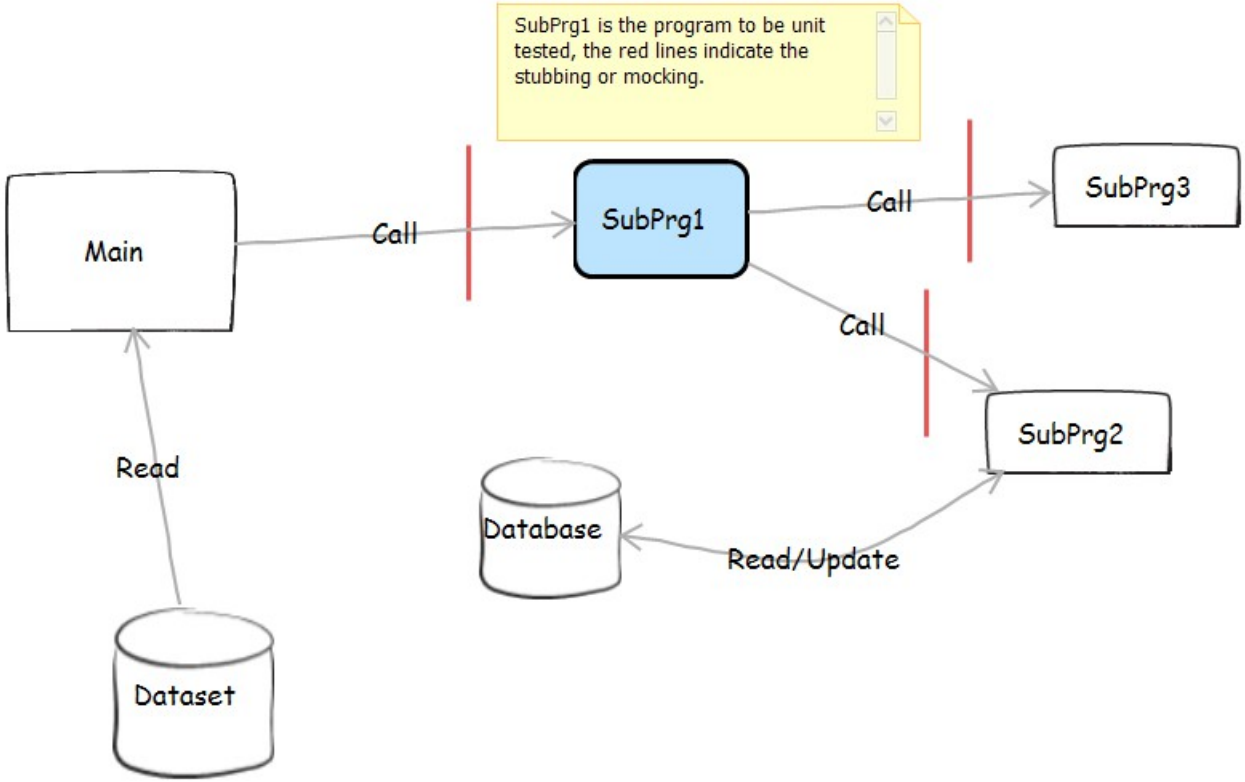
zUnit Architecture



zUnit capabilities

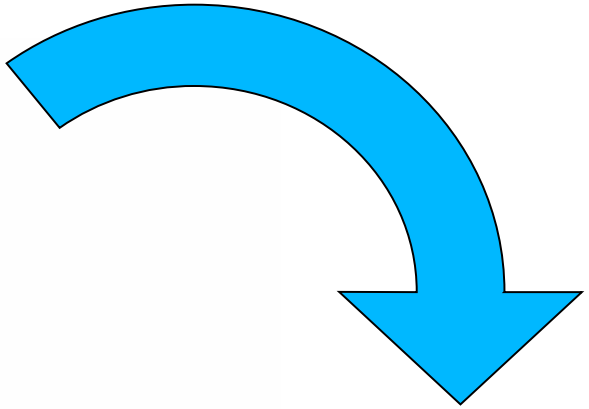
- zUnit Test Runner
  - Runs on z/OS
    - Installed and configured on z/OS as part of RDz Host install and customization
  - Fetches and runs the test cases referred to in the configuration file that is the input to test runner
- zUnit Wizards to generate test cases
  - Client feature
  - Eclipse based wizards that allow creation of
    - Template COBOL or PLI test cases
    - Complete COBOL test cases NEW in RDz v9.1!
      - Identify the interface or set of copy book(s)
      - Generate XML Schema to represent the interface
      - Generate XML files to specify the test input and expected output
      - Generate a test case based on the XML file
      - (Optionally) Generate stubs for called programs

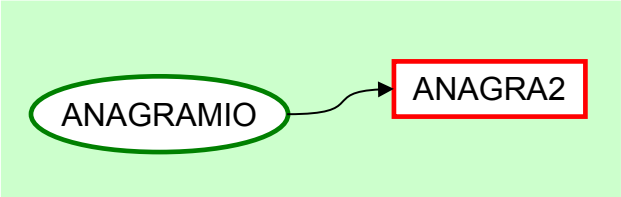
Module level testing

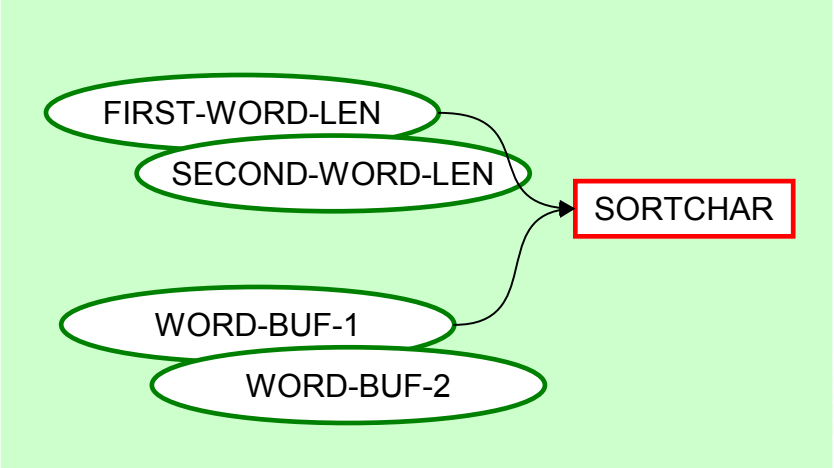


Source Program to be tested

```
IDENTIFICATION DIVISION.
 PROGRAM-ID. 'ANAGRA2'
DATA DIVISION.

 ........................................

COPY ANAGRAM.
PROCEDURE DIVISION USING ANAGRMIO.
MAINLINE SECTION.
    DISPLAY 'ANAGRAM STARTED...'
    PERFORM VALIDATE-INPUT
    PERFORM ANAGRAM-TEST
    DISPLAY 'ANAGRAM SUCCESSFUL'

  ........................................

ANAGRAM-TEST.
    SET NOT-ANAGRAM TO TRUE
    INITIALIZE WORD-BUF-1
    DISPLAY FUNCTION UPPER-CASE(FIRST-WORD(1:FIRST-WORD-LEN))
    MOVE FUNCTION UPPER-CASE(FIRST-WORD(1:FIRST-WORD-LEN))
      TO WORD-BUF-1(1:FIRST-WORD-LEN)
    DISPLAY
       'o FIRST-WORD:' FIRST-WORD(1:FIRST-WORD-LEN) '.'

    CALL 'SORTCHAR' USING FIRST-WORD-LEN WORD-BUF-1
    DISPLAY
       'o WORD-BUF-1:' WORD-BUF-1(1:FIRST-WORD-LEN) '.'
    INITIALIZE WORD-BUF-2
    MOVE FUNCTION UPPER-CASE(SECOND-WORD(1:SECOND-WORD-LEN))
      TO WORD-BUF-2(1:FIRST-WORD-LEN)
    DISPLAY
       'o SECOND-WORD:' SECOND-WORD(1:SECOND-WORD-LEN) '.'
    CALL 'SORTCHAR' USING SECOND-WORD-LEN WORD-BUF-2
```







Test Data Schema wizard

New

Select a wizard

Wizards:

type filter text

- z/OS Automated Unit Testing Framework (zUn
  - Enterprise COBOL zUnit Test Case
  - Enterprise COBOL zUnit Test Data Schema
  - Enterprise PL/I zUnit Test Case
  - zUnit Runner Configuration
- z/OS UNIX
- zComponent

Show All Wizards.

< Back | Next >

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Data Schema

New COBOL zUnit Test Data Schema

Create COBOL zUnit test data schema

Source files to import

ctfmvs08.rtp.raleigh.ibm.com\VENKATU\VENKATU.ZUNIT.COBCOPY\ANA
ctfmvs08.rtp.raleigh.ibm.com\VENKATU\VENKATU.ZUNIT.SOURCE.COBO

Entry Point

ctfmvs08.rtp.raleigh.ibm.com\VENKATU\VENKATU.ZUNIT.SOURCE.COBO

Change COBOL preferences

< Back | Next > | Finish

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Data Schema

New COBOL zUnit Test Data Schema

Select a location for the test data schema

Target for the test data schema

Data set name: VENKATU.ZUNIT.SCHEMA.XSD

Remote...

Workspace...

Overwrite member

< Back | Next > | Finish | Cancel

Generating XML files for test Data

z/OS Projects
- COBOLSample
- Cobsampl
- zOSProject
- zUnitProject
  - zUnitDemo [ctfmvs08.rtp.raleigh.ibm.com]
    - VENKATU.ZUNIT.AZUCFG
    - VENKATU.ZUNIT.COBCOPY
    - VENKATU.ZUNIT.SCHEMA.XSD
      - ANAGRMI
      - FIRSTWO
      - SECOND
      - WORDBU
      - WORDBU
    - VENKATU.ZU
      - ANAGRA
      - ANAGRA
    - VENKATU.ZU

GAD Demo | ANAGRA2.cbl

```
----+-*A-1-B--+----2----
            END-IF
            .
        ANAGRAM-TEST.
            SET NOT-ANAGRA
            INITIALIZE WOR
            MOVE FUNCTION
              TO WORD-BUF-
            DISPLAY
                'o FIRST-WO
            CALL 'SORTCHAR
            DISPLAY
                'o WORD-BUF
            INITIALIZE WOR
            MOVE FUNCTION
              TO WORD-BUF-
          * The above phrase
          *     TO WORD-BUF
            DISPLAY
                'o SECOND-W
            CALL 'SORTCHAR
```

- New
- Open
- View
- Browse
- Open With
- Rename...
- Copy
- Delete...
- Remove...
- Generate
  - XML Files...
- Search...
- Refresh
- Team
- Enterprise Extensions
- Property Group
- Add To Another Subproject ...
- Move To Another Subproject ...
- zComponent
- Properties
- Import z/OS Project...
- Compare With

Properties | Out | g Started | Remote Error

- PROGRAM: ANAGI
  - IDENTIFICATIO
  - DATA DIVISION
  - PROCEDURE DI
    - MAINLINE S
      - VALIDATE
      - ANAGRA

m JES

e	Parent filt...	Par
ved Jobs	CN-ctfmv...	Not
S Jobs	CN-ctfmv...	Not

Test case creation wizard

New

Select a wizard

Wizards:

type filter text

- z/OS
  - Application Deployment Manager Mani
  - Database Application Project
  - Host Connection File
  - MVS Subproject
  - z/OS Project
  - BMS
  - MFS
  - z/OS Automated Unit Testing Framewor
    - Enterprise COBOL zUnit Test Case
    - Enterprise COBOL zUnit Test Data Sc
    - Enterprise PL/I zUnit Test Case
    - zUnit Runner Configuration
- z/OS UNIX
- zComponent
- Other
- Examples

Show All Wizards.

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Case

File, Data Set, or Member Selection

Create a COBOL zUnit test cas

Target for test case program

Data set name: VENKATU.

Member name: DEMO01

Overwrit

Identification properties for t

Test case ID:

Test case name: DEMO01

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Case

Scenario Selection

Select a scenario for generating a zUnit t

Generate zUnit test cases:

From a template

From target source and test data (You

Source files to import

ctfmvs08.rtp.raleigh.ibm.com\VENKATU

ctfmvs08.rtp.raleigh.ibm.com\VENKATU

Entry Point

ctfmvs08.rtp.raleigh.ibm.com\VENKATU

Test data files to import

C:\Users\IBM_ADMIN\IBM\rationalsdp\

C:\Users\IBM_ADMIN\IBM\rationalsdp\

C:\Users\IBM_ADMIN\IBM\rationalsdp\

C:\Users\IBM_ADMIN\IBM\rationalsdp\

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Case

Stubs Selection

Select subprograms to generate stub programs

- [x] PROCEDURE DIVISION
- [x] 'SORTCHAR'

< Back | Next > | Finish | Cancel

Test case creation wizard

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Case

Test Data Setting

Specify test data for parameter

PROCEDURE DIVISION | 'SORTCHAR'

Specify test data for each parameter of the test entry

Entry Name	Test Name
TEST0001	TEST0001

Selected test entry TEST0001

Parameter	Input Data	Expected Output Data
ANAGRMIO		

Select expected output test data file

Test data for expected output

tc1_ANAGRAMIO_expected.xml
tc1_ANAGRAMIO_input.xml
tc1_WORD-BUF1_expected.xml
tc1_WORD-BUF1_input.xml

Go back to the Scenario Selection page to add a test data file.

OK | Cancel

New COBOL z/OS Automated Unit Testing Framework (zUnit) Test Case

Test Data Setting

Specify test data for parameter

PROCEDURE DIVISION | 'SORTCHAR'

Specify test data for each parameter of the test entry

Entry Name	Test Name
TEST0001	TEST0001

Add... | Edit... | Remove | Move Up | Move Down

Selected test entry TEST0001

Parameter	Input Data	Expected Output Data
FIRST-WORD-LEN		
WORD-BUF-1	tc1_WORD-BUF1_input.xml	tc1_WORD-BUF1_expected.xml

Select data file: Select... | Remove

< Back | Next > | Finish | Cancel

Generated Test case & Stub

```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
      *|     Test Case. Upon return from this program, the            |
      *|     Test Runner will attempt to call the ADDTESTS            |
      *|     program.                                                 |
      *|                                                              |
      *| @param TEST-CASE-PTR (input),                                |
      *|     A pointer-by-value to an area maintained by the          |
      *|     zUnit Test Runner that identifies the Test Case          |
      *|     and associated resources.                                |
      *|
      *| Note: this program does not require editing.
      *+-----------------------------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. 'AZUTC003'.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       1 CBLTESTC-ID PIC X(36)
           VALUE 'd9f4aba9-e187-4b84-963b-ecfdbbaadb8
       1 CBLTESTC-NAME PIC X(8)
           VALUE 'AZUTC003'.
       1 CBLTESTC-ID-LEN PIC S9(9) COMP-5.
       1 CBLTESTC-NAME-LEN PIC S9(9) COMP-5.
```

```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
       PROCESS DLL,EXPORTALL,TEST(NOSEP)
      *+-------------------------------------------------------------+
      *| SORTCHAR                                                    |
      *|     This is STUB program generated by the IBM z/OS Automated |
      *|     Unit Testing Framework (zUnit)                          |
      *| Date Generated: 03/09/2014 03:43 AM                         |
      *+-------------------------------------------------------------+
       IDENTIFICATION DIVISION.
       PROGRAM-ID. SORTCHAR.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       1 FAIL-MESSAGE-TXT PIC X(254).
       1 FAIL-MESSAGE-LEN PIC S9(9) COMP-5 VALUE 19.
       1 AZ-TEST-NAME-FOR-SUB   PIC X(254) EXTERNAL.
       1 AZ-TEST-NAME-LEN  PIC S9(9) COMP-5 EXTERNAL.
       1 AZ-TEST-CASE-PTR    POINTER EXTERNAL.
       1 AZ-RECORD-COUNT PIC S9(9) COMP-5.
       1 AZ-LAST-TEST-NAME   PIC X(254) VALUE SPACES.
       1 AZ-LAST-TEST-NAME-LEN  PIC S9(9) COMP-5 VALUE 0.
       1 AZ-TEST-EXPECTED-DATA-VALUE
```

Code Coverage in the IDE

z/OS Projects - Code Coverage Report (Feb 25, 2014 8:42:06 AM) - IBM Rational Developer for System z

File Edit Navigate Search Project Run Window Help

Quick Access | z/OS Projects | Debug

z/OS Projects

*R42CIBZP.jcl | Code Coverage Report (ATMCIB00_2014_02_25_084206_0333)

Code Coverage Report

Code Coverage Summary

Code coverage report (analyzed at Feb 25, 2014 8:42:06 AM, generated at Feb 25, 2014 8:42:06 AM)

Element	Coverage	Covered Lines	Total Lir
ATMVITF0	47%	74	
ATMURND0	81%	115	
ATMSICU0	52%	174	
ATMSICT0	52%	75	
ATMSIBR0	65%	124	
ATMSIAC0	59%	147	
ATMRIBCL	71%	65	
ATMRIBBL	64%	47	
ATMRIBAL	63%	55	
ATMRCLG0	42%	17	
ATMRBLG0	42%	17	
ATMRALG0	42%	17	

Report

Remote Systems

I42CIB00.jcl
I51CIB00.jcl
R42CIB00.jcl
R42CIBDM.jcl
R42CIBDT.jcl
R42CIBND.jcl
R42CIBZP.jcl
R42G0000.jcl
R42ROFFL.jcl
R51CIB00.jcl
R51CIBDM.jcl
R51CIBDT.jcl
R51CIBZP.jcl
R51G0000.jcl
R51ROFFL.jcl
V4206.jcl
V42CIB00.jcl
V42M0000.jcl
V42RALG0.jcl
V42RBLG0.jcl
V42RCLG0.jcl

Properties | Outline

Property	Value
Connection-privat	Yes
Name	Retrieved Jobs
Number of childre	0
Number of filter st	1
Parent filter	Not applicable
Parent filter pool	CN-torolabw-com.ibm.zos.jes
Type	Remote system filter

Getting Started | Remote Error List | z/OS File System Map... | Property Group Mana... | Snippets | Remote System Details

Subsystem JES

Resource	Parent filter pool	Parent filter	Number of filter strings	Connection-private
Retrieved Jobs	CN-torolabw-com.ibm.zos.jes	Not applicable	1	Yes
My Jobs	boxall:com.ibm.zos.jes	Not applicable	1	No

ATMSICU0 - Local Workspace

Code Coverage in the IDE

z/OS Projects - RemoteSystemsTempFiles/DebugViewFiles/3/ATMSIAC0.list.cob - IBM Rational Developer for System z

File Edit Navigate Search Project Run Window Help

Quick Access | z/OS Projects | Debug

Code Coverage Report (ATMCIB00_2014_02_25_110758_0871) | ATMSIAC0.list.cob

```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
                                        WS-AIB-RSN-CD
          MOVE LS-FUNCTION            TO WS-FUNCTION
     *    MOVE LS-ACCT-POSITION-F     TO WS-ACCT-POSITION-F
          MOVE ATM-ACCT-BRANCH TO WS-ACCT-BRANCH
          MOVE ATM-ACCT-NO     TO WS-ACCT-NO
          IF WS-FUNCTION-INIT       OR
             WS-FUNCTION-GET        OR WS-FUNCTION-ISRT OR
             WS-FUNCTION-UPDT       OR WS-FUNCTION-SIZE OR
             WS-FUNCTION-GET-FIRST OR WS-FUNCTION-GET-NEXT
                                    OR WS-FUNCTION-LOG-TXN
     *       WS-FUNCTION-INIT-TBL   OR WS-FUNCTION-SORT OR
             CONTINUE
          ELSE
             PERFORM I100-INPUT-PARM-ERROR
             DISPLAY "ATMSIAC0: INVALID WS-FUNCTION IN INIT"
                     " WS-FUNCTION = " WS-FUNCTION
          END-IF
          IF WS-ATMSIAC0-RETURN-CODE-OK
             IF WS-FUNCTION-GET OR WS-FUNCTION-ISRT OR
                WS-FUNCTION-UPDT
                IF WS-ACCT-BRANCH IS EQUAL ZEROES OR
                   WS-ACCT-NO     IS EQUAL ZEROES
                   PERFORM I100-INPUT-PARM-ERROR
                   DISPLAY "ATMSIAC0: ACCT KEY DATA IS ZERO"
```

Remote Systems

R42CIB00.jcl
R42CIBDH.jcl
R42CIBDM.jcl
R42CIBDT.jcl
R42CIBND.jcl
R42CIBZP.jcl
R42G0000.jcl
R42ROFFL.jcl
R51CIB00.jcl
R51CIBDM.jcl
R51CIBDT.jcl
R51CIBZP.jcl
R51G0000.jcl
R51ROFFL.jcl
V4206.jcl
V42CIB00.jcl
V42M0000.jcl
V42RALG0.jcl
V42RBLG0.jcl

Property	Value
Attribute	
BLKSIZE	32720
DSNTYPE	DATA
DSORG	PO
EXTENTS	1
LRECL	80
PRIMARY	201
RECFM	FB
SECONDA	100
SPACE UN	BLOC
VOLUME	LW00

Getting... | Remot... | z/OS Fi... | Proper... | Snippets | Remot... | Remot... | Compil...

Subsystem JES

Resource	Parent filter pool	Parent filter	Number of filter strings	Connection-private
Retrieved Jobs	CN-torolabw-com.ibm...	Not applicable	1	Yes
My Jobs	boxall:com.ibm.zos.jes	Not applicable	1	No

System z LPEX Editor | Browse | Insert | 476 : 39

Summary

- z/OS application development and maintenance can greatly be enhanced by utilizing the Code quality tooling to achieve:
  - Automation
  - Maintainable code
  - Structured code
- More productive development enabling faster turn-around on fixes, changes, and enhancements

Questions ?


Thank You