Modern environment for z/OS development

Rosalind Radcliffe
Venkat Balabhadrapatruni

Feb 4, 2013
Session: 12503
Four key barriers preventing optimal return on IT investments

Decades of application investments

“We don’t understand the effort, risk and impact of modernizing our legacy applications.”

Islands of skills, languages and platforms

“Our skills gap keeps growing. How do we stay current with all the language and technology changes?”

Poorly integrated teams

“We need to enable our teams to collaborate across platforms, languages, and environments.”

Infrastructure inefficiency

“We need a cost effective way to improve our infrastructure efficiency and free up capacity to handle more workload.”
Overview of Supported Production Scenario

Project Manager or Support Team has submitted Project Change Request ...

1: Review Change Request
- Analyze application to be changed
- Size/scope effort and risk of change
- Submit to Project Manager for review, approval and scheduling

2: Review and Approve Change Request
- Review analysis for change request and approve for scheduling
- Create development work item(s) for implementation
- Add work to project plan

3: Implement required changes, build and deliver
- Analyze source to identify modifications
- Implement and test modifications
- Perform personal build and deliver new features

4: Promote and deploy enhancement
- Promote changes from development to test
- Create update package with set of changes from development
- Deploy update package to the test environment

5: Track Project Status with Rational Team Concert Dashboard
## Development Life Cycle

<table>
<thead>
<tr>
<th>Planning</th>
<th>Source Dev</th>
<th>Governance/Unit test</th>
<th>Build</th>
</tr>
</thead>
</table>
| • Define the tasks  
• Create a plan  
• Create a work item  
• Assign the work item to a developer | • Load the project/source artifacts from SCM  
• Navigate, Analyze, Edit, Syntax check source code | • Compile  
• Quality assurance  
  • Debug  
  • Code Coverage  
  • Code review  
  • Unit Testing | • Check-in/Deliver the source code  
• Build |

| CLM | RDz | RDz | RTC  
| RDz | RTC | RD& T  
| RTC | RDz |

Complete your sessions evaluation online at SHARE.org/SFEval
Building an agile development team requires a multiplatform approach, and Sodifrance uses Rational Developer for System z and Rational Team Concert for System z to help application teams synchronize their efforts and improve collaboration. Rational on System z offers a powerful and valuable combination for any company that wants to boost its development team’s productivity.

— Hugh Smith, Project Manager, Sodifrance
Rational Team Concert – A single tool, many capabilities

- **Work Items**
- **Planning**
- **Builds – Continuous**
- **Dashboards & Reporting**
- **Source Control**
- **Method Enforcement and Automation**

### Work Items

- **555: Improve documentation for 4.4**

  **Story 55**

  **Summary:** Improve documentation for 4.4

  **Details:**

  - **Type:** Story
  - **Created:**
  - **Created By:**
  - **Team Area:** Joint Team
  - **Filed Against:**
  - **Tags:**
  - **Owner:** Bill Cassaveli

### Planning

- **JUnit Plan**

  **Team Area:** Joint Team

  **Iteration:** 4.4 m2 (5/209 - 6/109)

  **Task:**

  - **Bill Cassaveli**
  - **Closed Items:** 5, **Open Items:** 3
  - **Due:** 1 week

### Builds – Continuous

- **Build junit 120080708-1553**

  **Completed:**

  - **Duration:** 33 seconds
  - **Start Time:** July 8, 2008 3:59:59 AM
  - **Completed:** July 8, 2008 3:59:59 AM

### Dashboards & Reporting

- **JUnit Project**

  **JUnit Project Details**

  - **JUnit:**
  - **Class:** org.junit
  - **Method:** assert

### Source Control

- **No Changes to Track**

### Method Enforcement and Automation

- **Problem**

  - A work item must be associated with the change set or a comment must be set.

- **Reason**

  - All change sets should be associated with a work item which is planned for delivery.

- **Deliver (failed)**

  - Missing work item or comment

  - **Missing work item or comment**

  - **Solutions**

    - Associate Existing Work Item
    - Associate New Work Item
    - Associate and Try Again (experimental)
    - Overrule ‘Descriptive Change Sets’ Precondition
Rational Team Concert: Built on an open, Web 2.0 platform
Supporting a broad range of desktop clients, IDE’s and languages
Rational Developer for System z:
An Integrated Development Environment for System z

- A modern IDE for productive 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
- Access to typical System z sub-system functionality in z/OS, CICS, IMS, DB2, WAS
- Integration with Debug Tool for Development and Test
- Integration with File Manager for file and test data handling
- Integration with Asset Analyzer for Application Understanding and Impact Analysis
- Integration with Fault Analyzer for Dump Analysis
- Integration with Team Concert for Lifecycle and Source Management
The Benefits of a RDz’s IDE

Instead of maneuvering to access panels and working sequentially, in RDz the functionality you need is always in-focus – you work concurrently.

- Access Datasets
- + Dataset Management
- Access Jobs (Outlist facility)
- Dataset Statistics
- Edit a program
- File Compare
- Submit a Compile
- File Search

Complete your sessions evaluation online at SHARE.org/SFEval
Traditional development and Enterprise web services

Supports traditional development/maintenance
• Cobol, PL/I, Assembler, JCL

Supports modern architecture development

Enterprise Service Tools

Single service projects
Top down, bottom up, and meet in the middle web service enablement for CICS, IMS, and Batch/TSO environments.

Enable Enterprise Applications for Mobile and Web

Service flow projects
Graphical composition of CICS applications chained together to form a new business service.
Rational Development and Test Environment for System z

The ultimate in modern application development for System z

- Increase availability of z/OS testing environment and resources
  - Liberate developers to rapidly prototype new applications
  - Develop and test System z applications anywhere, anytime!
  - Eliminate costly delays by reducing dependencies on operations staff

- Improve quality and lower risk via automation, measurement, and collaboration
- Focus on what is required for the change at hand, then scale

Note: This Program is licensed only for development and test of applications that run on IBM z/OS. The Program may not be used to run production workloads of any kind, nor more robust development workloads including without limitation production module builds, pre-production testing, stress testing, or performance testing.
Centralized administration

Push to Client - Configuration Files management and product updates

Desktop configuration setup

- upon connecting to the host, a user is notified if they have incoming changes to their configuration files.

Product updates

- triggered from central place to all the users connecting to the host

An Export Wizard – to help an Admin configure various settings on RDz client and upload it to a central location.

Settings are grouped by function into configuration files such as remote connections or system mappings.
# Development Life Cycle

<table>
<thead>
<tr>
<th>Planning</th>
<th>Source Dev</th>
<th>Governance/Unit test</th>
<th>Build</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the tasks</td>
<td>Load the project/source artifacts from SCM</td>
<td>Compile</td>
<td>Check-in/Deliver the source code</td>
</tr>
<tr>
<td>Create a plan</td>
<td>Navigate, Analyze, Edit, Syntax check source code</td>
<td>Quality assurance</td>
<td>Build</td>
</tr>
<tr>
<td>Create a work item</td>
<td></td>
<td>• Debug</td>
<td></td>
</tr>
<tr>
<td>Assign the work item to a developer</td>
<td></td>
<td>• Code Coverage</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Code review</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Unit Testing</td>
<td></td>
</tr>
</tbody>
</table>

- **CLM**
- **RDz**
- **RTC**
- **RD&TRTC**
- **RTC RDz**
Any process: Executable and repeatable

Use ONE tool to support both agile and non-agile
Multiple plan views facilitate continuous planning
Progress Tracking - Everyone can see live project status

Story Status Current Iteration
Shows the status of all stories planned for the current iteration

<table>
<thead>
<tr>
<th>Foundation - Story Status (61) Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Foundation - Current Stories (61)</th>
</tr>
</thead>
<tbody>
<tr>
<td>155918: Provide TVT testcases and testcase updates to TVT team</td>
</tr>
<tr>
<td>155399: [Repository - M14] Performance, quality and robustness enhancements</td>
</tr>
<tr>
<td>15407: [Login] Manage and repair DB inconsistencies during 3.0.1 M14</td>
</tr>
<tr>
<td>155924: Track the fixing of hangs occurring on shutdown of the GLM server</td>
</tr>
<tr>
<td>155923: [Application SDK - 14] Performance, quality and robustness enhancements</td>
</tr>
<tr>
<td>155092: LPA Web UI Tests and polish</td>
</tr>
<tr>
<td>15518: User can validate consistency of configuration information across contexts</td>
</tr>
<tr>
<td>155181: Fix bugs and improve the quality of LPA services</td>
</tr>
<tr>
<td>155924: Update JREs for 3.0.1 M14 and 3.0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCM - Story Status (11) Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCM - Current Stories (11)</th>
</tr>
</thead>
<tbody>
<tr>
<td>150244: address &quot;run as administrator&quot; issues on win7</td>
</tr>
<tr>
<td>142127: As a user I want to see all extended state groups in taskboards</td>
</tr>
<tr>
<td>150169: Increase the performance of loading a workbook</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Burndown Current Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows the remaining amount of estimated work in hours of work items planned for the current iteration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Foundation - Burndown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining Work</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCM - Burndown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining Work</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RM - Stories Open, Closed, In Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Parameters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RM - Stories Open, Closed, In Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Parameters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stories Open/Closed/In Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shows the number of stories which are open, in progress, done during the iteration.</td>
</tr>
</tbody>
</table>

Complete your sessions evaluation online at SHARE.org/SFEval
In-context Collaboration – Team View

Team Central

- Shows what is happening on project:
  - News & events
  - What’s being worked on
  - Changes

- Configurable (RSS feeds) - New kinds of information easily added

- Personalized, Persistent - Each team member can tailor to their needs
# Development Life Cycle

<table>
<thead>
<tr>
<th>Planning</th>
<th>Source Dev</th>
<th>Governance/Unit test</th>
<th>Build</th>
</tr>
</thead>
</table>
| - Define the tasks  
- Create a plan  
- Create a work item  
- Assign the work item to a developer | - Load the project/source artifacts from SCM  
- Navigate, Analyze, Edit, Syntax check source code | - Compile  
- Quality assurance  
  - Debug  
  - Code Coverage  
  - Code review  
  - Unit Testing | - Check-in/Deliver the source code  
- Build |

<table>
<thead>
<tr>
<th>CLM</th>
<th>RDz</th>
<th>RDz</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
<td>RTC</td>
<td>RTC</td>
<td>RDz</td>
</tr>
</tbody>
</table>
RDz Source Code Integration

- Rational’s Strategic Source Code tooling is RTC and RDz provides tight integration

- RDz offers integration into a variety of other Source Code Management (SCM) tools as well as a framework for creating SCM integration on your own (CARMA)

- Variety of vendors supply plug-ins to RDz to provide easy access to processes and source code controlled by their products
Source Control Management
Load the source artifacts
RDz and RTC together

- Once the project is loaded, it will appear in the RDz z/OS projects view

- RDz augments the development productivity & experience
  - Appropriate editors (COBOL, maps, etc.) and functions (content assist, real-time syntax check, etc.)
  - High value functions (Enterprise web services, SFM, Code review, Unit testing, program analysis/control flow etc.)
Create a Property Group

- Generate property groups for your project based on RTC build definition
- Allows RDz to resolve the dependencies and thus offer all the tooling

Configure Property Group Generation

Specify how to generate the property group.

Generate using build definition (optional):
- GAM zOS Dependency Build

Property group generation
- Local COBOL Settings
- Local PLI Settings

Generate remote library paths for this connection (optional):
- mvs114.rtp.raleigh.ibm.com

Finish  Cancel
Navigate datasets and jobs live on zOS

- Connect to multiple hosts concurrently
- Respects existing security configurations and user IDs
- Search, filter, browse, edit, compare, migrate, and allocate new MVS datasets and USS files
- Copy source code, members, or datasets between systems with a few mouse clicks.
- Access JES queues submit jobs, view job state, and open output spools
- Submit TSO or USS commands
- Add datasets and members into projects to group applications and work items together logically
- Open an emulator in the IDE to configured hosts
Edit capabilities in RDz

- RDz at a high level has different types of editors
  - Lpex Editor
    - Supports editing of COBOL, PLI, HLASM, JCL, C/C++, Rexx etc.
    - Provides ISPF like edit experience including prefix commands, command line and even look and feel
    - Supports advanced edit functions for COBOL, PLI and HLASM like real time syntax checking, content assist
  - COBOL and PLI advanced editors
    - Based on the Eclipse editor infrastructure, provide more advanced edit capabilities like quick fixes, hyperlinking, hover, easy navigation between various edit sessions or within the same edit session.
    - Supports real time syntax checking, content assist, key word highlighting etc.
Edit capabilities in RDz

- RDz at a high level has different types of editors
  - WYSIWYG editors
    - Creation, edit of BMS and MFS maps
    - Has the source and design view – allows drag and drop of fields in the design view which generates the appropriate source
  - Data Editors
    - Edit QSAM data
Editor Productivity features - Develop code more efficiently

Find all statements and variables (including from COPYBOOKS or INCLUDE)

```
000066 * EXIT.
000067 0150-SECOND-PART.
000068  MOVE 2 TO BRANCHFLAG.
000070  exec cics s
000071  MOVE
000072  MOVE
000073  MOVE
000074  MOVE
000075  0200-LOGIC.
000076  IF WHICH
000077  If is LAB2
000078  CALL
000079
```

```
000067 * 0150-SECOND-PART.
000068  MOVE 2 TO BRANCHFLAG.
000070
000071  CUST-ADDR1
000072  CUST-CITY
000073  CUST-TRY
000074  CUST-FN
000075  CUST-LN
000076  CUST-NO
000077  POTVSAM-RECORD-REC
000078  POTVSAM-RECORD-REC
000079  POTVSAM-RECORD-REC
```
Editor Productivity Features – real time syntax checking

Real-time syntax check without requiring code compile or save
Editor Productivity Features

- Provide "Open Called Program" action

- Hyper linking support for Open/ Browse/ View copybooks/include files

- Show In > Outline action to COBOL and PL/I Editor
Editor Productivity Features

- Mark “Write occurrences” capability to the supported EXEC statements

- Occurrences within EXEC statements known to be “writes” are highlighted with a BROWN background

- All “read” statements will continue to be highlighted with a GREY background
Search for Occurrences Action

- Once a variable is selected the user triggers the “Find Occurrences” action using the Menu under search or keyboard shortcut “Ctrl+Shift+U”
- The occurrences are shown in the “Search results” page
Copy book and Include file resolution

- Hover over a COPY book name or a INCLUDE file to see the contents
- Pressing F2 when hovering will “pin” the hover as shown
- The window can then be dragged to expand, the Pencil icon shown below can be used to edit the copy book
JCL Template Support

- Templates are provided for standard JCL statements, and users can create their own Templates.
- When editing .jcl file using “Ctrl+Space” in the editor will trigger a pop list allowing the user to select the template to insert into the editor contents.
Enhanced Application Quality & Structure Analysis

- **Application Analysis**
  - Control flow diagrams for COBOL programs,
    - *Graphical representation of the program flow with links to the source*
  - Helps identify and highlight potential unreachable code
Enhanced Structure Analysis – Data Element Table

- A table representation of the user-defined data items and symbols in a program
  - Hyperlinks in the table are integrated with the editor allowing easy access to the declaration of the data items.
- Generated by showing the “symbol table” generated when RDz real-time syntax check parses the program

![Data Element Table](image)
Pending Changes

- If you want finer grained control on your SCM operations, then the Pending Changes view is for you
  - Check in, deliver, accept changes
  - Suspend, resume, discard changes
  - Replace, reload out-of-sync
  - Resolve conflicts
  - Open change sets and work items via the web client
Traceability: Check-in History

- Someone made a costly mistake merging and you want to understand exactly where the mistake was made
  - Problem: Traditional history commands & UI only show before/after & merge states for a change set … it does not show intermediates
  - Solution: Use Check-in history in Eclipse, CLI or .NET clients
# Development Life Cycle

## Planning
- Define the tasks
- Create a plan
- Create a work item
- Assign the work item to a developer

## Source Dev
- Load the project/source artifacts from SCM
- Navigate, Analyze, Edit, Syntax check source code

## Governance/Unit test
- Compile
- Quality assurance
  - Debug
  - Code Coverage
  - Code review
  - Unit Testing

## Build
- Check-in/Deliver the source code
- Build

<table>
<thead>
<tr>
<th>CLM</th>
<th>RDz</th>
<th>RDz</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDTC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Debug Multiple Runtimes – using IBM Debug tool

- Use the cross-platform debugger to debug end-to-end systems as they execute in the runtime
  - IMS
  - CICS
  - DB2
  - Batch
  - WAS
  - Java

- From the workstation:
  - View executing source code
  - Step through host code line-by-line
  - Set breakpoints
  - Alter working storage values
  - Alter register values
  - Etc…

- Debug zOS and distributed code in the same interface even stepping between runtimes and platforms!
Enhanced Application Quality – Code Coverage

- Line Level Code Coverage - provides tools to measure and report on test coverage of an application
- Indicating what source code lines were tested and remain to be tested
Innovative Debugging using Collaboration

- Collaborative debug with RDz and IBM Debug Tool via the Rational Team Concert Server!
  - Share breakpoints and monitors with other team members
  - Transfer debug session control to other users
  - Save debugging sessions for later retrieval in the team environment
  - Works for WAS (JEE) and System z applications
Enhanced Quality & Structure Analysis – Code review

• Code Review/Governance - provides predefined rules and templates for COBOL and PL/I applications
  • Ensure adherence to corporate standards
zUnit – Unit testing framework for z/OS

- Frameworks that assist developers in writing code to perform repeatable, self-checking unit tests are collectively known as xUnit.

- **xUnit** defines a set of concepts that together provide a light-weight architecture for implementing unit testing frameworks.
  - JUnit, for example, is a very popular instance of the xUnit architecture.

- **zUnit** is a xUnit instance for System z

- Goal is to encourage the **continuous integration and continuous testing** methodology for System z Application development and maintenance

---

1. **Language-specific details:**
   - In COBOL, this is the first program appearing in the Test Case source file and it will be invoked by the Test Runner for Test Case initialization.
   - In PL/I, the is the procedure declared with option(fetchable) in the Test Case source file and it will be invoked by the Test Runner for Test Case initialization.

2. **Language-specific details:**
   - In COBOL, these are expected to be subprograms (non-nested and therefore compatible with FUNCTION-POINTER).
   - In PL/I, these are expected to be internal procedures that are declared at the package level (non-nested).
# Development Life Cycle

## Planning
- Define the tasks
- Create a plan
- Create a work item
- Assign the work item to a developer

## Source Dev
- Load the project/source artifacts from SCM
- Navigate, Analyze, Edit, Syntax check source code

## Governance/Unit test
- Compile
- Quality assurance
  - Debug
  - Code Coverage
  - Code review
  - Unit Testing

## Build
- Check-in/Deliver the source code
- Build

<table>
<thead>
<tr>
<th>CLM</th>
<th>RDz</th>
<th>RDz</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RTC</td>
<td>RD&amp;T</td>
<td>RDz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC</td>
<td></td>
</tr>
</tbody>
</table>
Enterprise Extensions Specific Functions

• Dependency build
  • “Smart build” of z/OS and IBM i applications, based on what has changed

• Promotion
  • Flow of source code changes and build outputs through development hierarchy

• Packaging and deployment
  • Package build outputs and deploy to another system (e.g. test environment, QA, production, etc)
The big picture

1. Dependency build runs on build machine. Source is loaded from Dev Stream and outputs are built in Dev Library.
2. Promotion build runs on build machine. Source is promoted from Dev Stream to Test Stream and build outputs are copied from Dev Library to Test Library.
3. Package build runs on build machine. Test Library build outputs are archived in a package.
4. Deploy build runs on various test machines. Package is loaded to test machine and build outputs are deployed to runtime libraries.
Dependency Build Summary

1- Scan
Scan new or changed files
Extract their logical information and dependencies

2- Preprocessing
For changed files: impact on build maps
Calculate dependency sets

Server

Mainframe (Build machine)

Build Maps

Build processing

Dependency sets
Snapshots for every build
Promotion

- Flow source code changes and build outputs through the development hierarchy
Summary

• Many companies spend more than 70% on keeping lights on, and that amount is increasing
• IT organizations have problems modifying applications at speed of business
• IBM provides a structured approach to incrementally modernize your portfolio based on business priorities
• Change without a plan is chaos
• A Plan without change is stagnation
• Business goals change
  • applications need to change to address them
• Continual renewal is required
  • tools help to guide, govern, drive, and accomplish this change
Getting started
Next steps to modernize your enterprise applications

- Try latest System z software for free
- Sign up for free web-based training
- Join IBM Rational Cafe Communities
- Get prescriptive service solutions
- Success stories
- Latest news on System z twitter
- Latest customer videos
- Latest skills: System z job board
Thank You