A System z Developer's Journey Through the Application Lifecycle

Liam Doherty dohertl@au1.ibm.com

Gary Mazo mazo@us.ibm.com

3 March, 2015
Session: 16517

#SHAREorg

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

Trademarks and Legal Notes

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml: AS/400, DBE, e-business logo, ESCO, eServer, FICON, IBM, IBM Logo, iSeries, MVS, OS/390, pSeries, RS/6000, S/30, VM/ESA, VSE/ESA, Websphere, xSeries, z/OS, zSeries, System z, z/VM

The following are trademarks or registered trademarks of other companies

Lotus, Notes, and Domino are trademarks or registered trademarks of Lotus Development Corporation
Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
LINUX is a registered trademark of Linux Torvalds
UNIX is a registered trademark of The Open Group in the United States and other countries.
Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.
Intel is a registered trademark of Intel Corporation
* All other products may be trademarks or registered trademarks of their respective companies.

NOTES:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon 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 throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

Any proposed use of claims in this presentation outside of the United States must be reviewed by local IBM country counsel prior to such use.

The information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Purpose and Presentation flow

- Purpose … to present a System z developer's use of tools that help manage the software development cycle: "Day in a Life"

- Flow
  - Overview of
    - What is DevOps and its role in current state of Software development
    - Tools for Software development to support the DevOps story
  - Walk through the dev cycle and tools that supports each step in the cycle

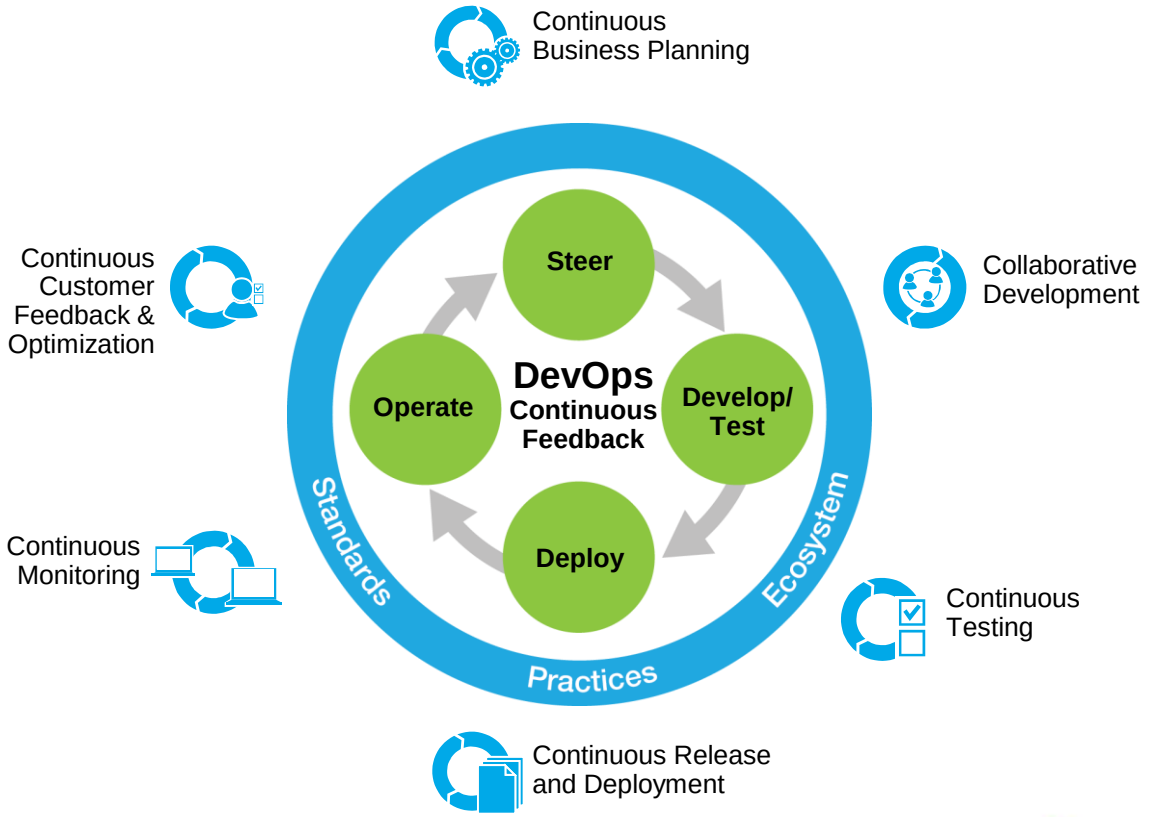
IBM DevOps point of view

Enterprise capability for continuous software delivery that enables organizations 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*



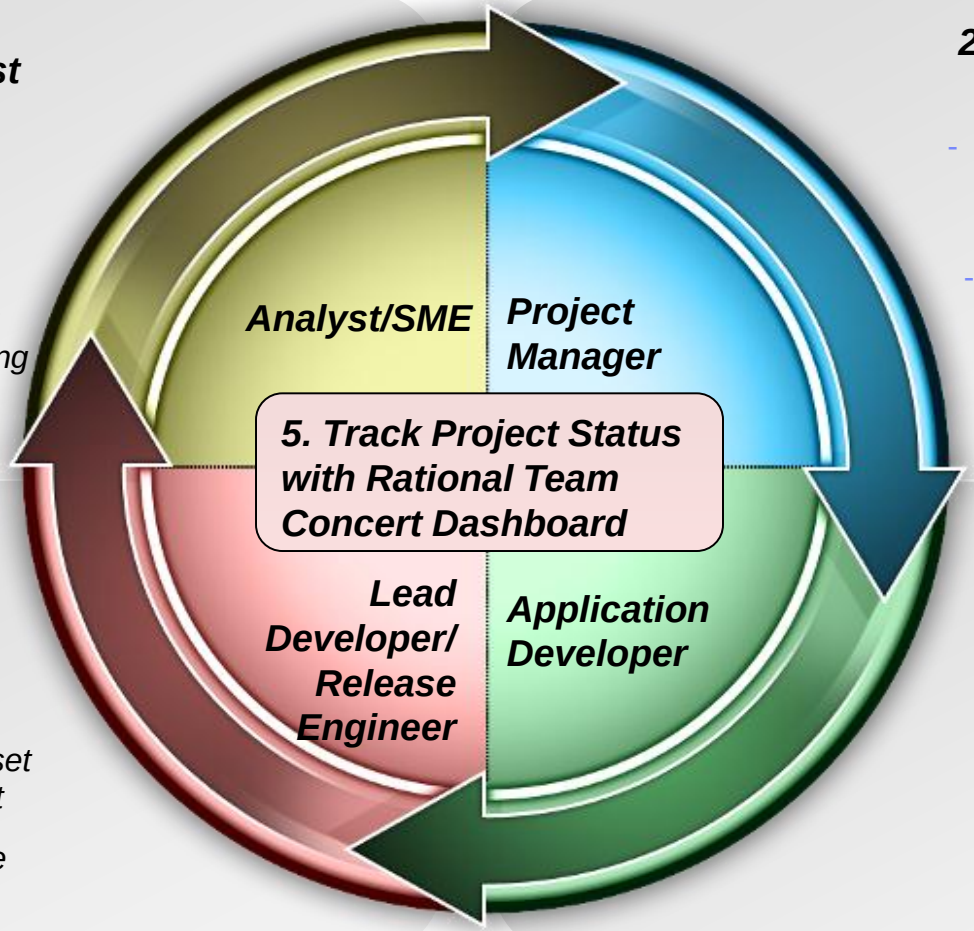
Our DevOps POV is resonating with clients and they are delivering **measurable business outcomes with DevOps**

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*

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*

3: Implement required changes, build and deliver

- *Analyze source to identify modifications*
- *Implement and test modifications*
- *Perform personal build and deliver new features*

Development Life Cycle

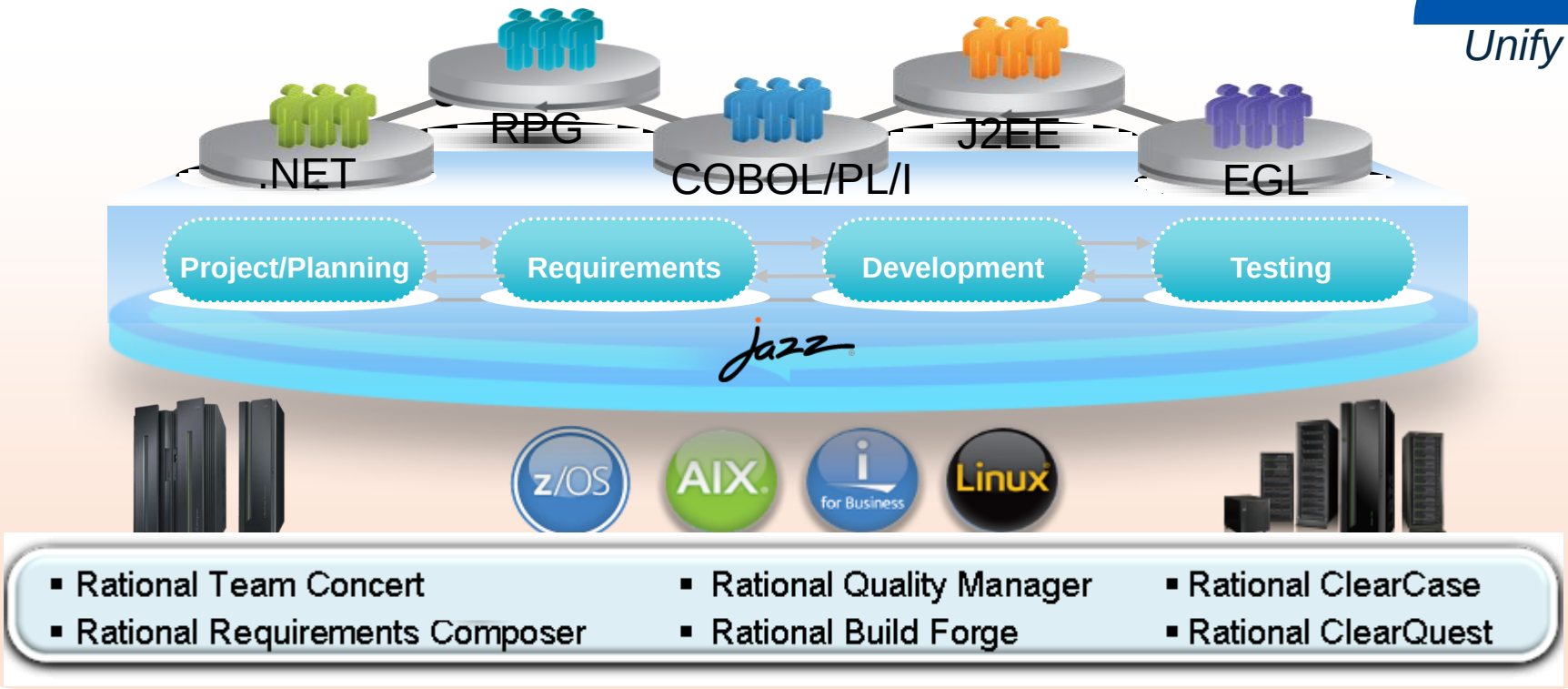
Planning	Source Dev	Governance/Unit test	Build

• 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 RTC	RDz RD&T RTC	RTC RDz

Collaborative application lifecycle management

Deploy new, common team infrastructure for source control, change management and build that empowers your team with integrated collaboration, process automation and reporting



"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."

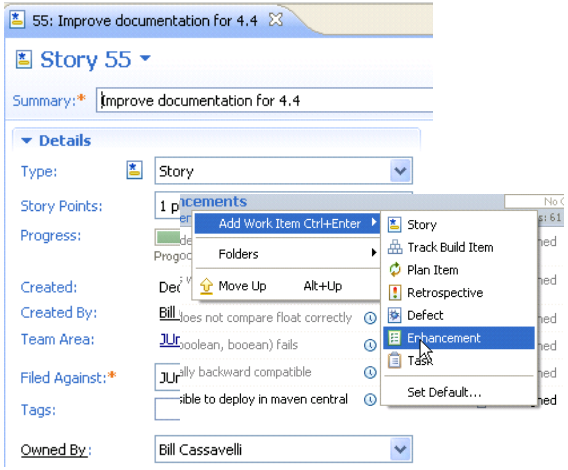
— Hugh Smith, Project Manager, Sodifrance

SODIFRANCE

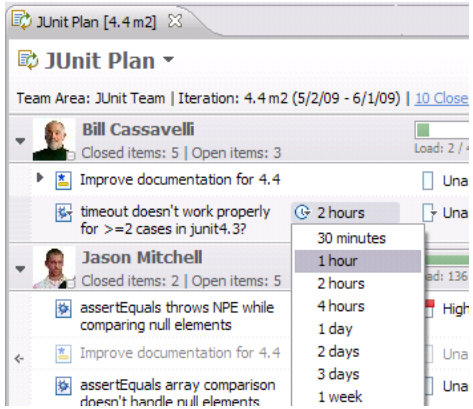
SHARE in Seattle 2015

Rational Team Concert – A single tool, many capabilities

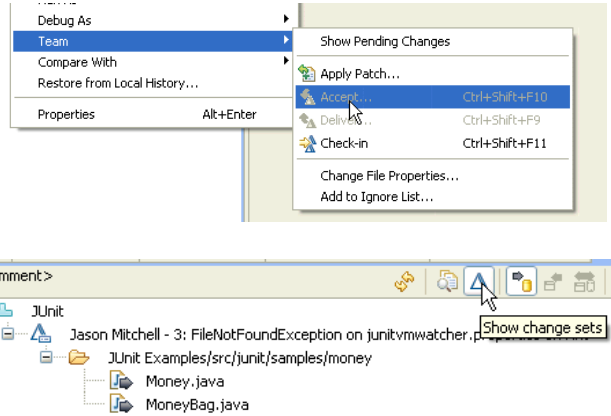
- Work Items



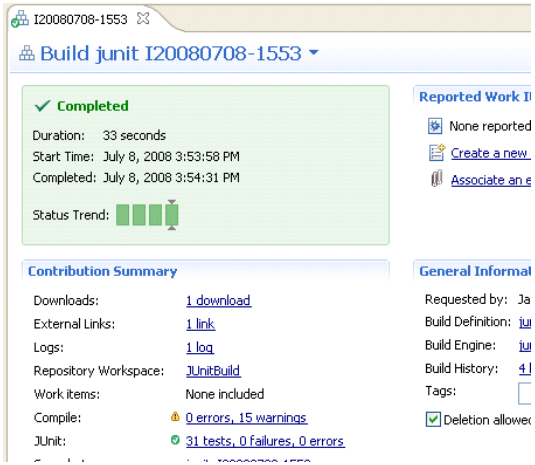
- Planning



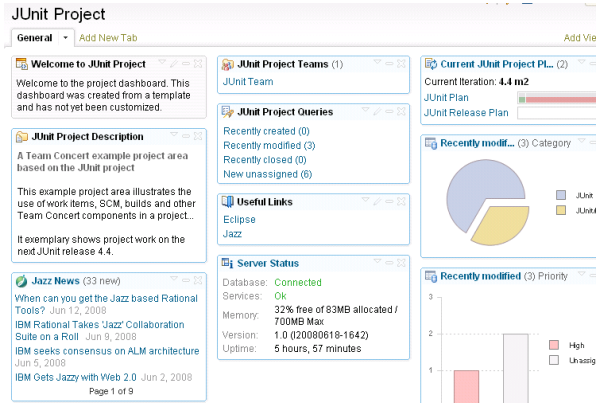
- Source Control



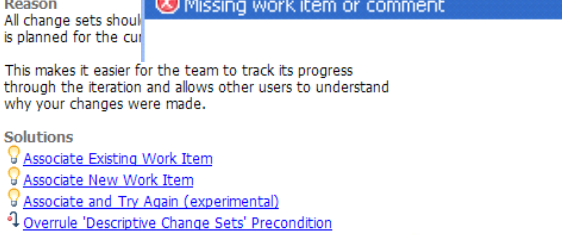
- Builds – Continuous



- Dashboards & Reporting

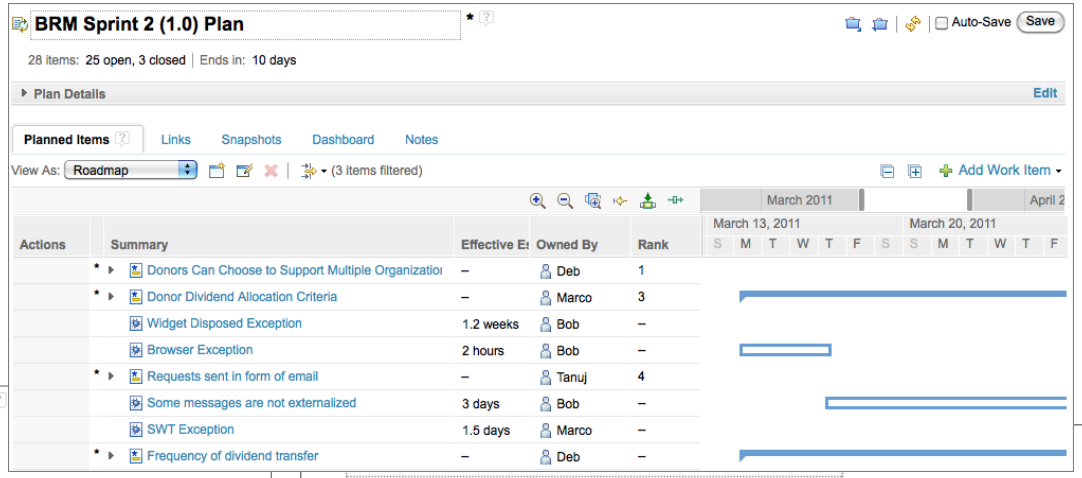


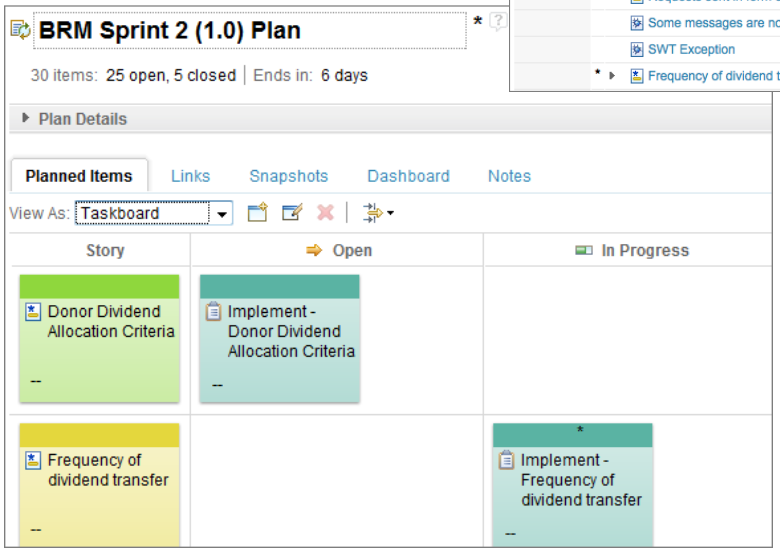
- Method Enforcement and Automation

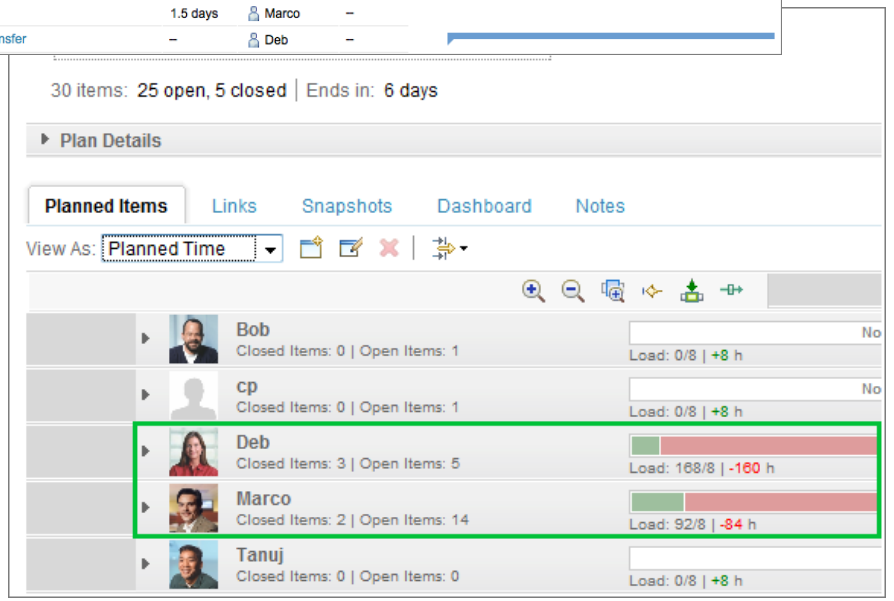


Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Multiple plan views facilitate continuous planning

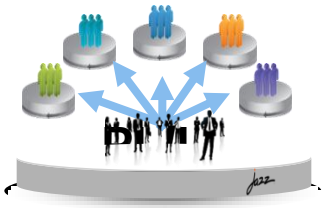






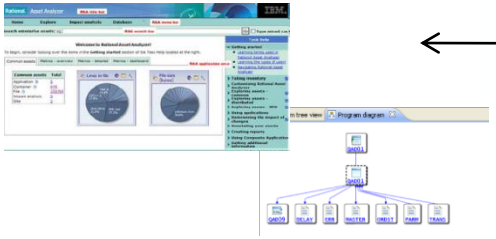
Rational Developer for System z:
An Integrated Development Environment for System z

Integration with Team Concert for Lifecycle and Source Management

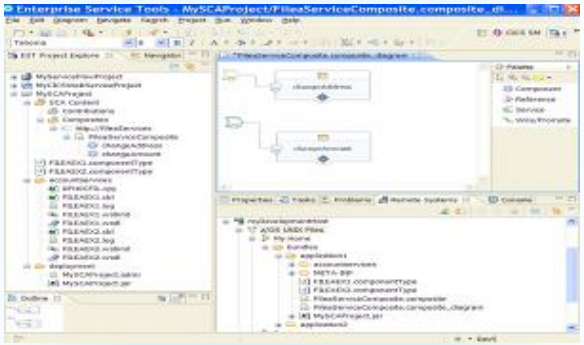


Access to typical System z sub-system functionality in z/OS, CICS, IMS, DB2, WAS

Integration with Asset Analyzer for Application Understanding and Impact Analysis

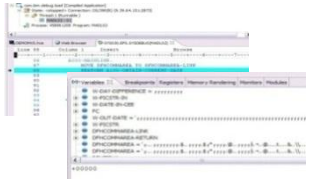


Rational Developer for System z



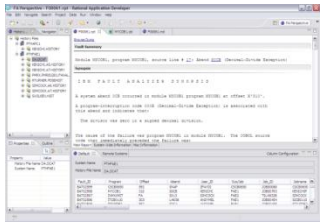
New

Out of the Box debugger and code coverage capabilities



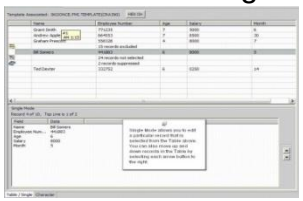
A modern IDE for productive development of cross-platform applications written in COBOL, PL/I, ASM, Java, EGL or C/C++ in System z CICS, IMS, DB2, Batch applications

Integration with Fault Analyzer for Dump Analysis



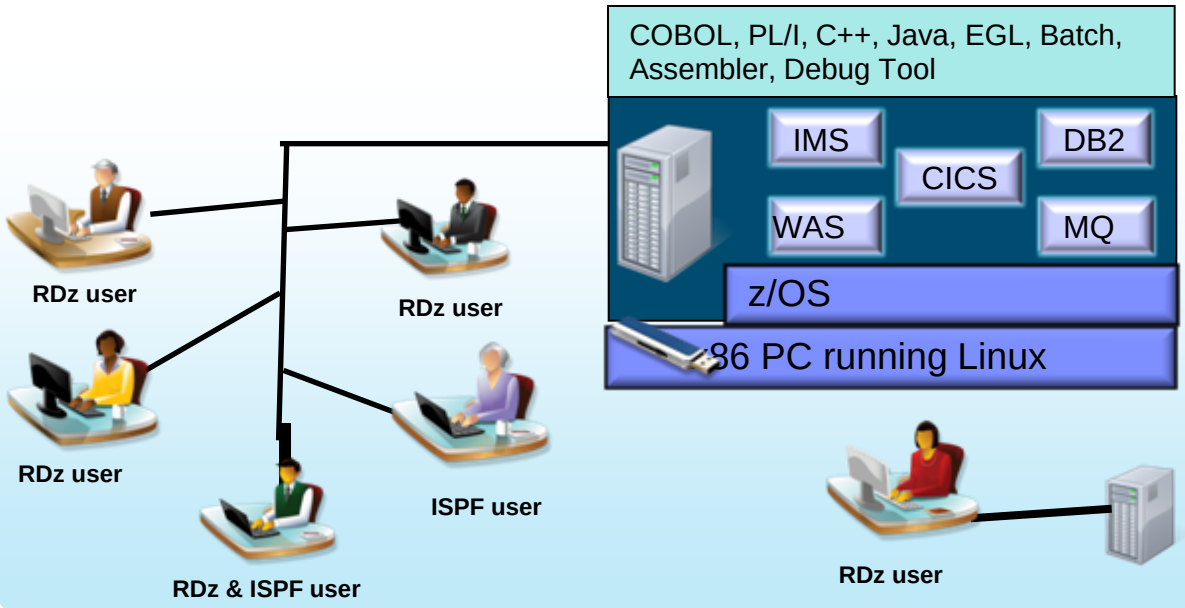
Integration with RD&T for flexible access to System z environment

Integration with File Manager for file and test data handling



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.

Development Life Cycle

Planning	Source Dev	Governance/Unit test	Build

• 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

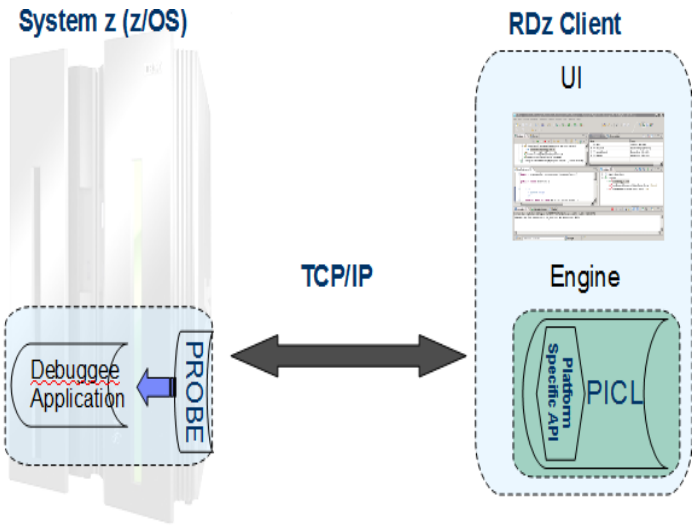
CLM	RDz RTC	RDz RD&T RTC	RTC RDz

Demo: What we will show

- Business Analyst
  - opening a new work item for Developer using RTC
- Software Developer
  - discovering the new work item in his work queue in RTC
  - accessing and loading his code into his work env using RTC
  - working to setup work using integrated RDz/RTC
  - editing the COBOL source file and figuring out how to code the requirement stated in the work item
  - code the requirement, personal build and test
- Build/Release Admin
  - team build, test and deploy
  - mark the work item Complete in RTC
- Business Analyst
  - Verify that requirement is complete and close the work item in RTC

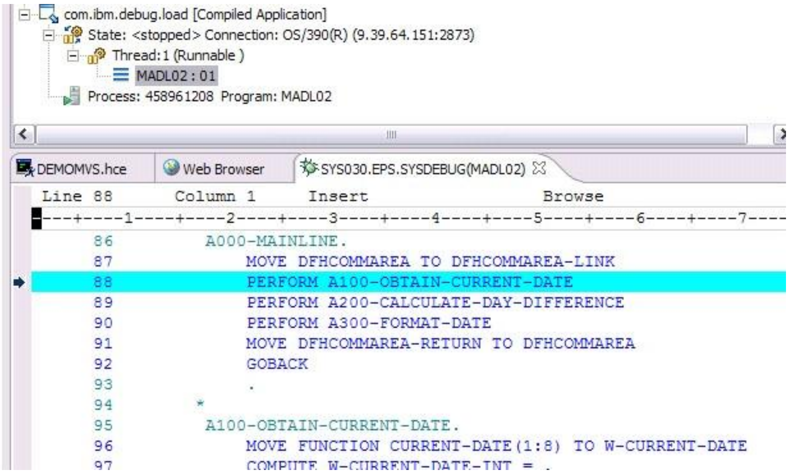
Integrated Debugger

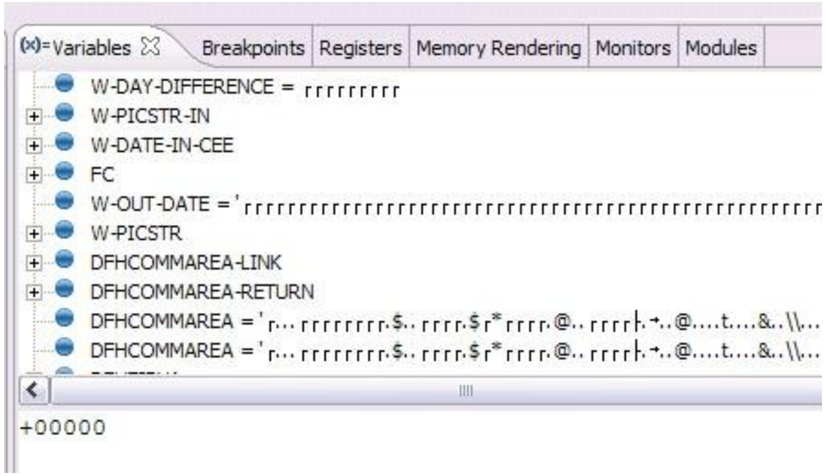
- **Host-offload architecture:**
  - Remote debugger with only a small footprint on the mainframe:
    - Leverages workstation CPUs enabling faster processing of debug information
    - Enables scalability and reliability
  - Debugger client is supported on Windows and Linux
- **Simple and Secure Connections:**
  - Single client can handle multiple debug sessions on multiple hosts or an application the spans multiple systems
  - Client initiated debug – no need to specify client IP address and port (v9.0.1.2)
  - SSL/TLS support



Debug Multiple Runtimes

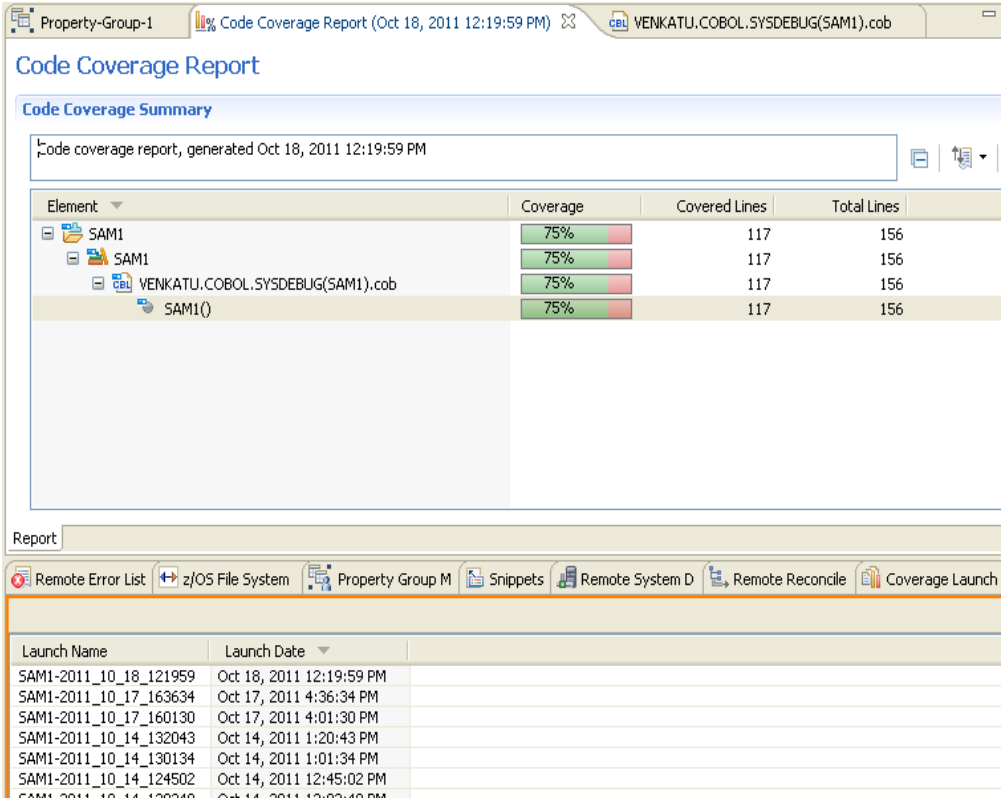
- Use the cross-platform debugger to debug end-to-end systems as they execute in the runtime
  - CICS
  - Batch
  - Java
- From the workstation:
  - View executing source code
  - Step through host code line-by-line
  - Set breakpoints
  - Alter working storage values
  - Alter register values
  - Etc…
- Debug zOS and distributed code in the same interface even stepping between runtimes and platforms!
- Leverage Integration with IBM Debug Tool for other runtimes

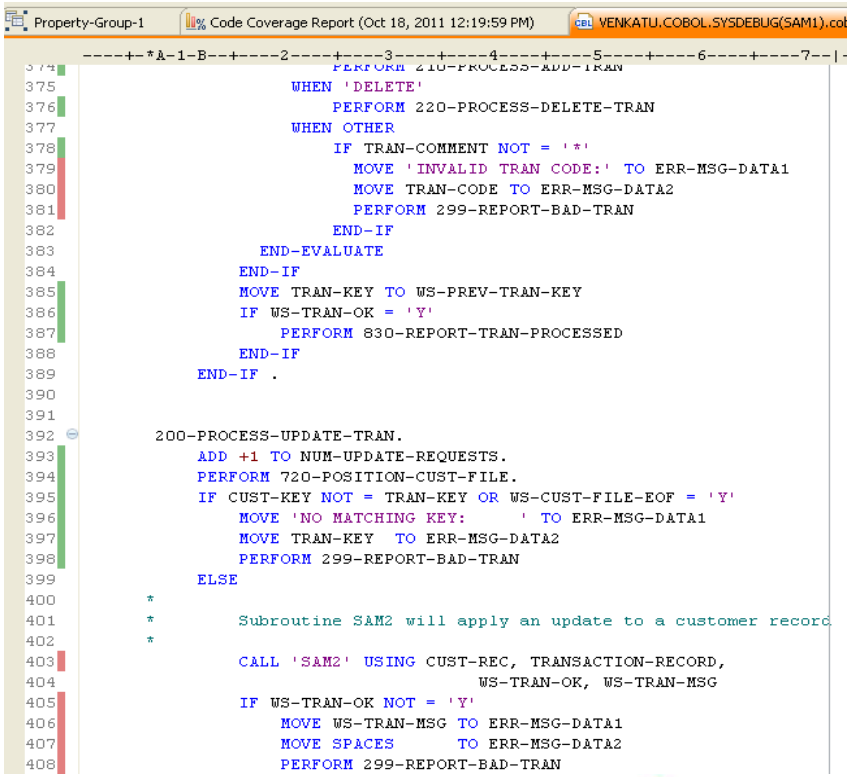




Enhanced Application Quality – Code Coverage

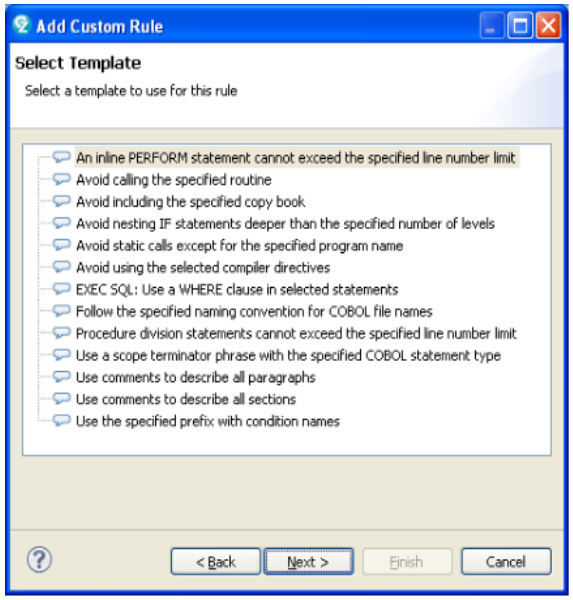
- Line Level Code Coverage - provides tools to measure and report on test coverage of an application
  - Leverages the Integrated Debugger technology
  - Indicating what source code lines were tested and remain to be tested

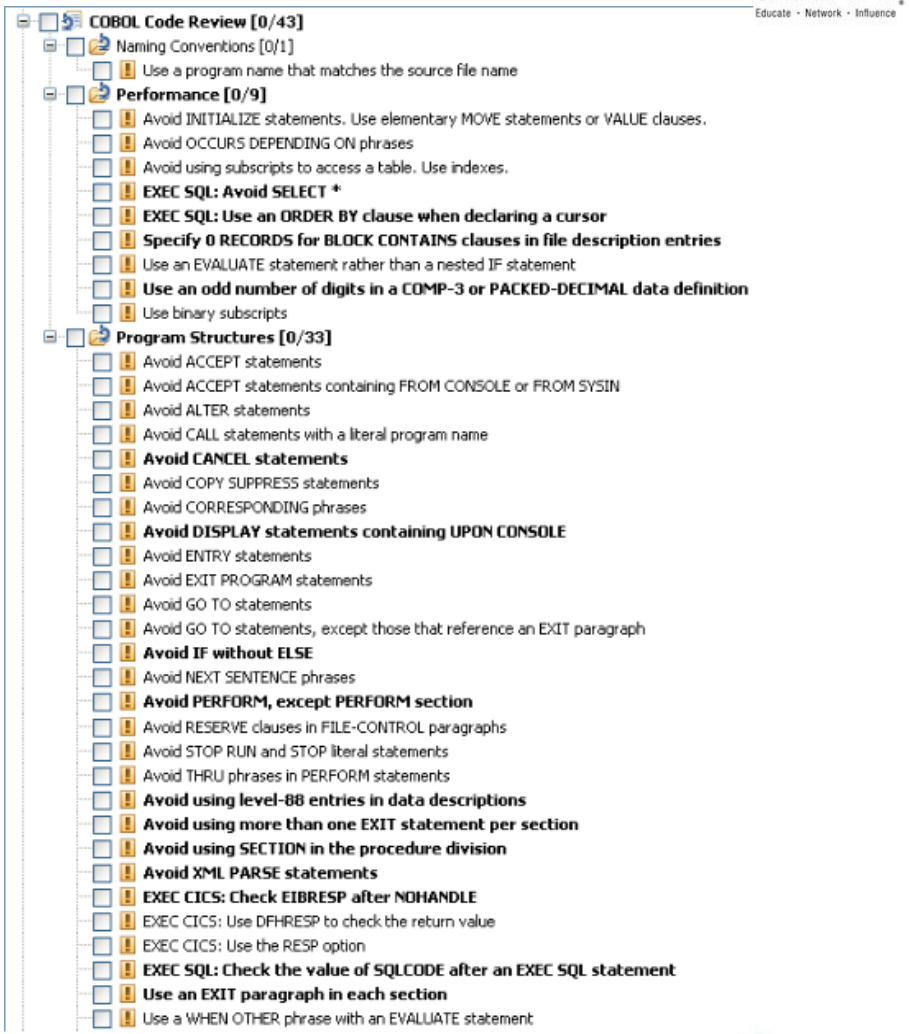




Enhanced Quality & Structure Analysis – Code review

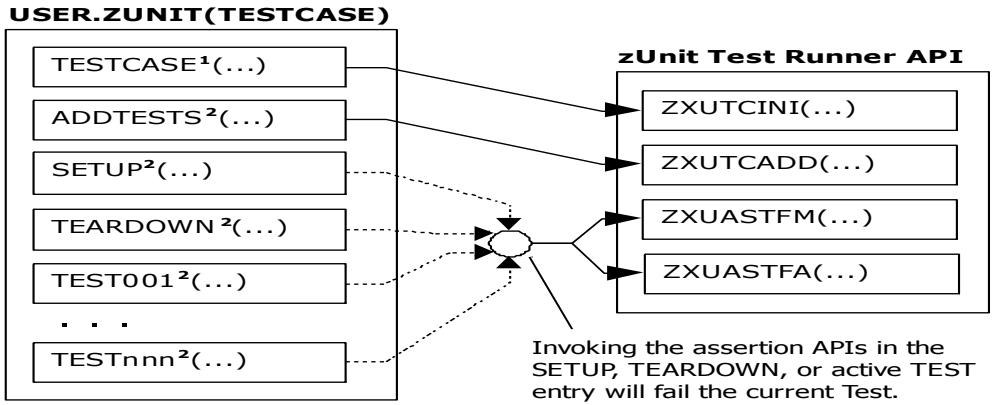
- Code Review/Governance - provides predefined rules and templates for COBOL and PL/I applications
  - Ensure adherence to corporate standards
  - Custom rules for COBOL and PL/I





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 *x*Unit.
- 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



¹Language-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.

²Language-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).

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 Suite referred to in a zUnit configuration file

- zUnit Wizard used to generate Test Cases
  - RDz client feature
  - Eclipse based wizards allow creation of:
    - •Template Test Cases are generated in COBOL or PL/I
      - •Simple pass/fail **assertion** API

    (RDz v9.1) Complete COBOL test cases:

      - **•Identify the interface or set of copy book(s)**
      - **•Generate XML Schema to represent the interface**
      - **•Generate XML files where you would specify test input and expected output**
      - **•Generate a Test Case based on the XML file**
    - **•(Optionally) Generate stubs for called programs**
  - RDz viewers/editors for unit test XML results

What have we learned

- What is Software DevOps and its importance in today's Enterprise
- The tools that help manage every day life of Software DevOps practitioners
- How to use some of the tools and where to find more…

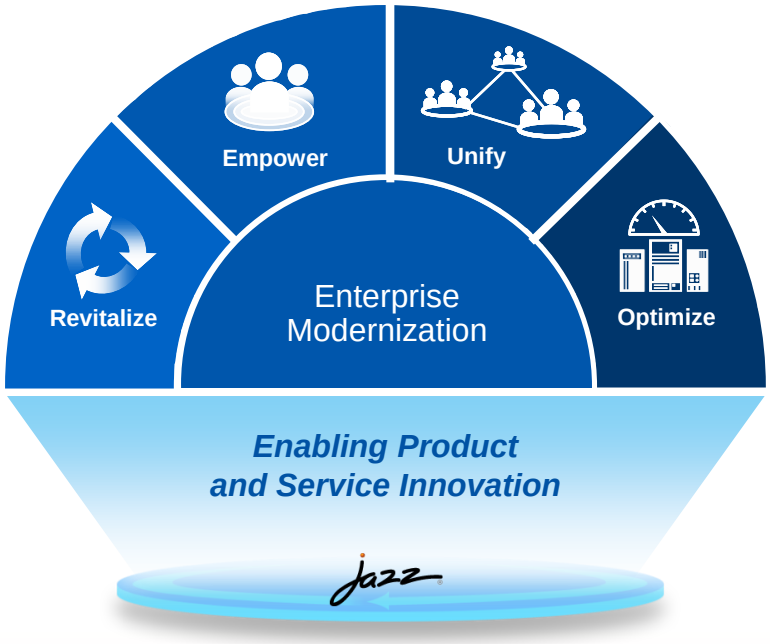
Summary

- 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

www.ibm.com/rational/modernization



- 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

23


QUESTIONS


Thank You

Back up and Reference

Development Life Cycle

Planning	Source Dev	Governance/Unit test	Build

• 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 RTC	RDz RD&T RTC	RTC RDz

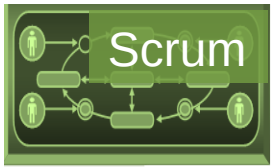
Any process: Executable and repeatable

Use ONE tool to support both agile and non-agile

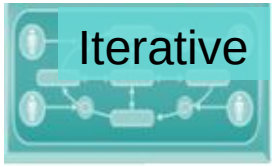
Process Architect

Project Manager

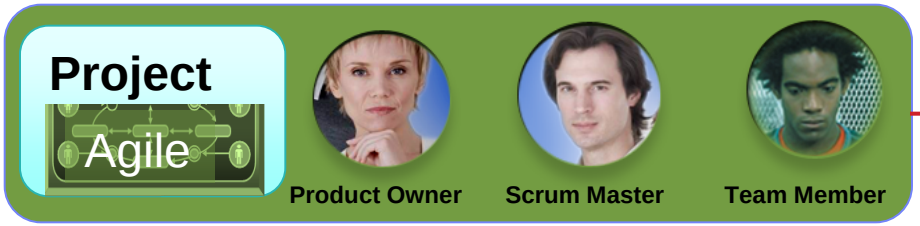
Variant #1

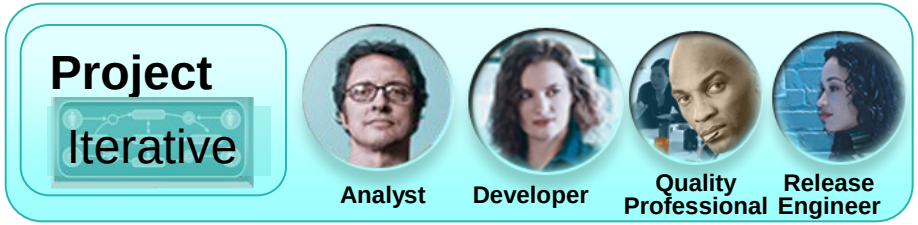


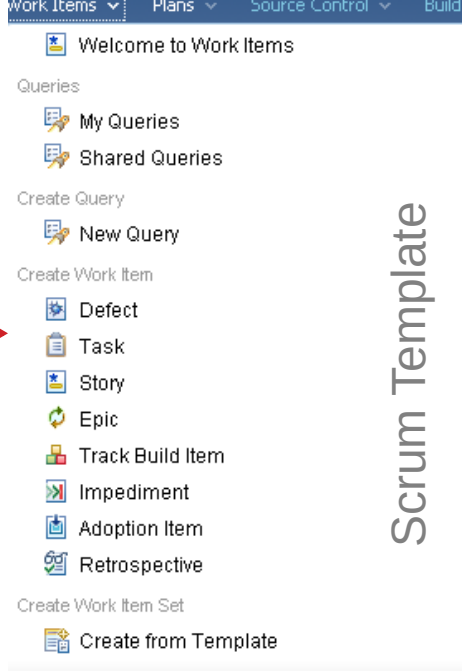
Variant #2

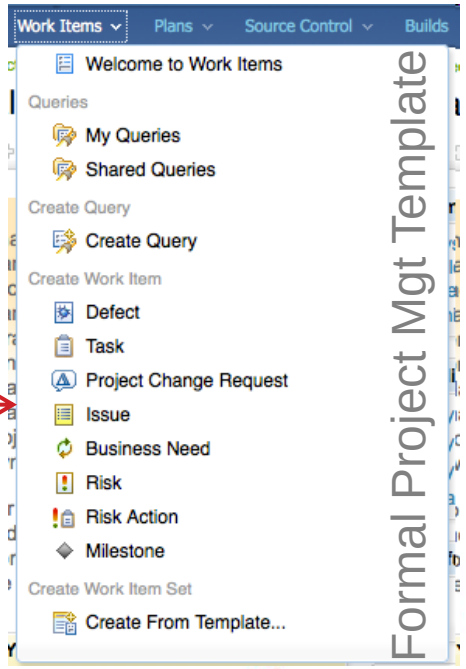


Variant #3





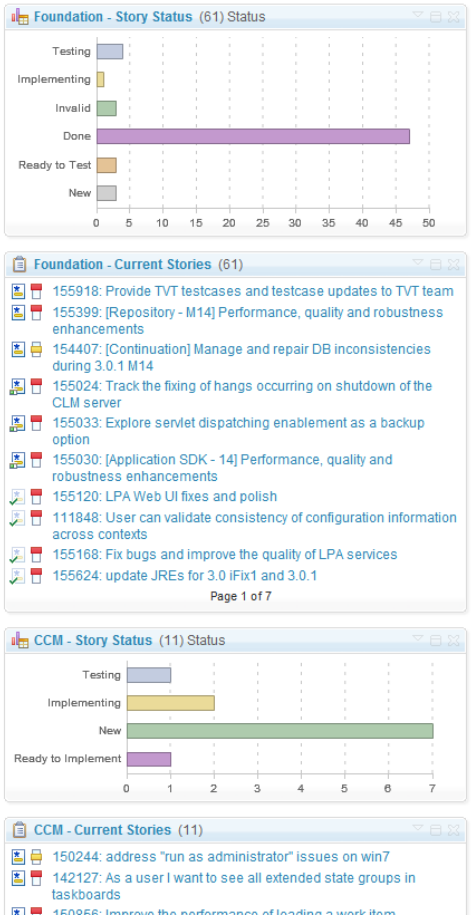




Progress Tracking - Everyone can see live project status

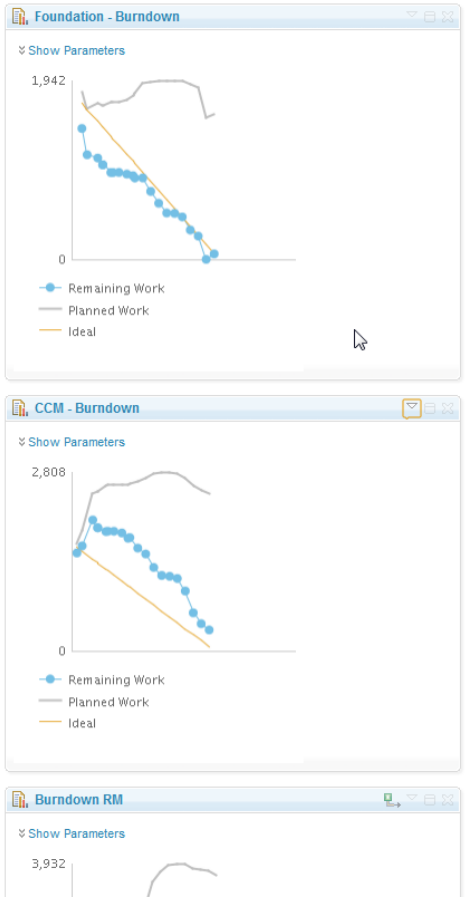
Story Status Current Iteration

Shows the status of all stories planned for the current iteration



Burndown Current Iteration

Shows the remaining amount of estimated work in hours of work items planned for the current iteration.



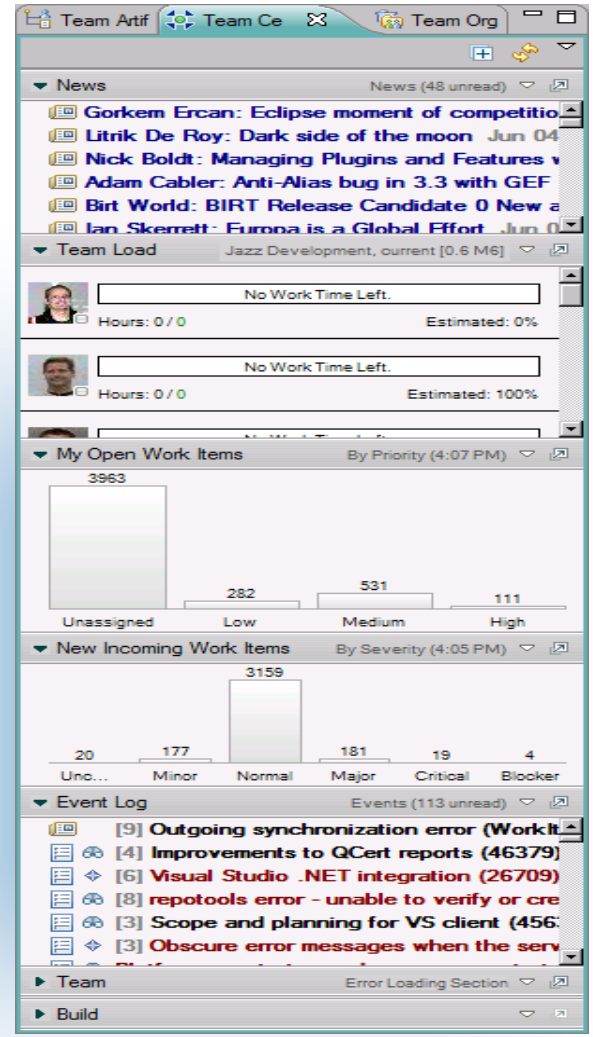
Stories Open/Closed/In Progress

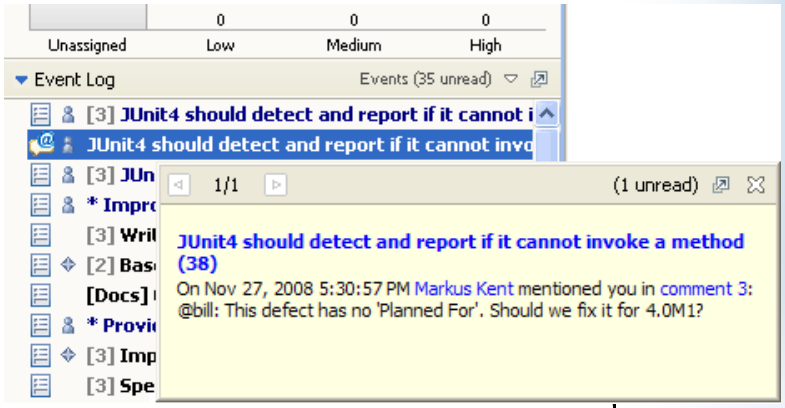
Shows the number of stories which are open, in progress, done during the iteration.

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

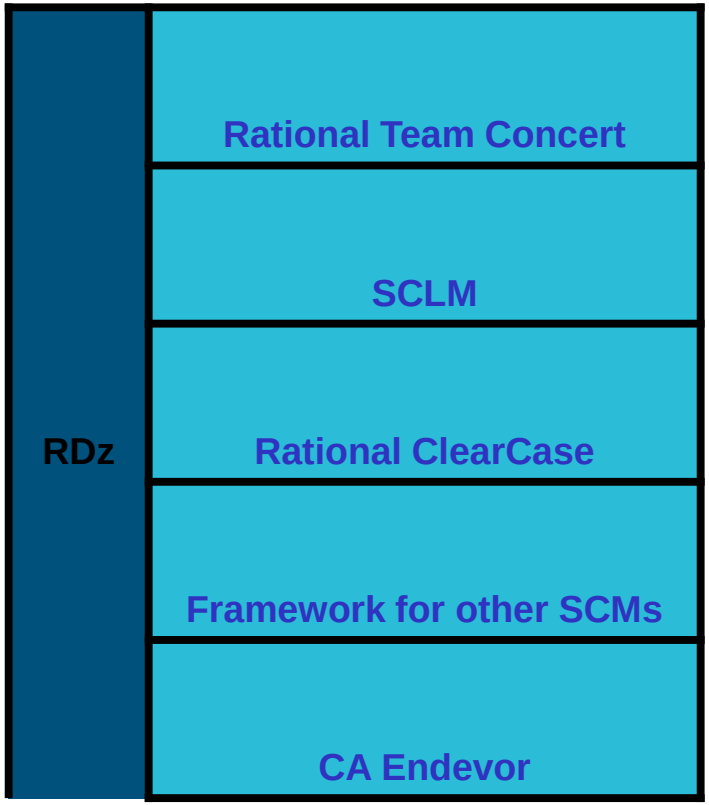
Planning	Source Dev	Governance/Unit test	Build

• 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

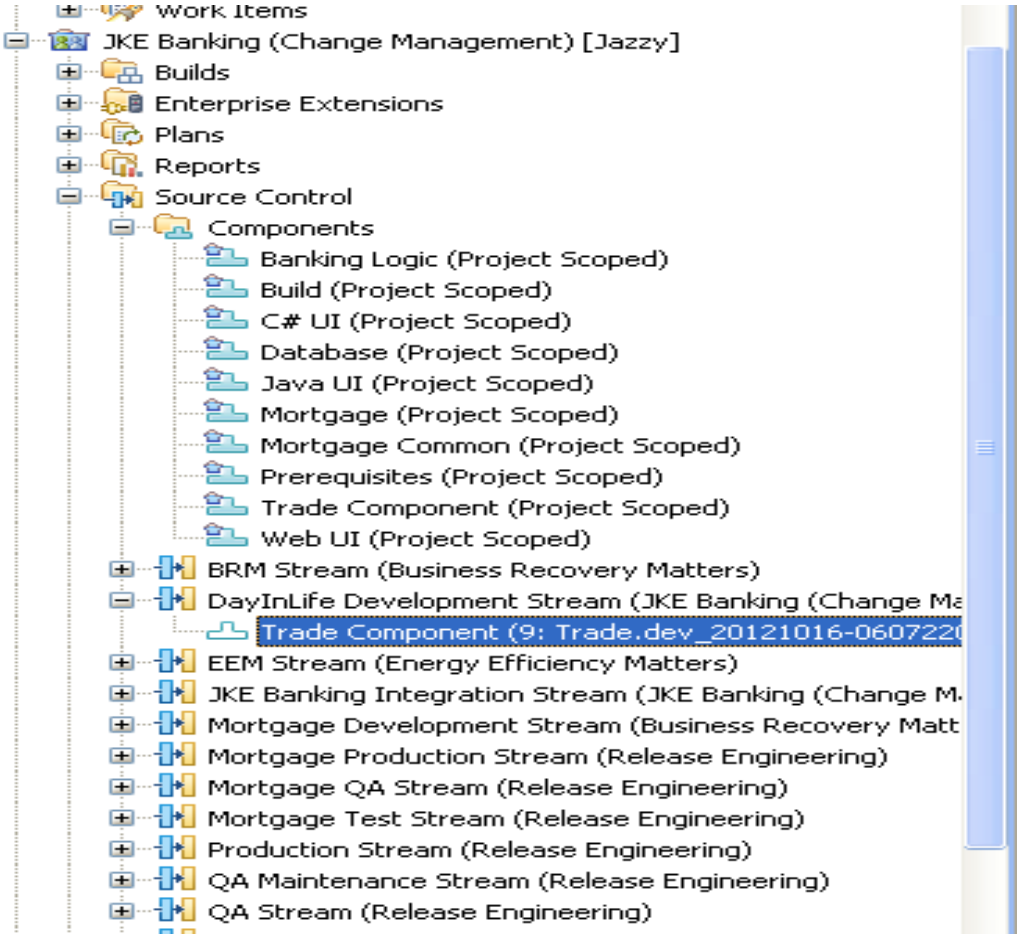
CLM	**RDz** **RTC**	RDz RD&T RTC	RTC RDz

RDz Source Code Integration

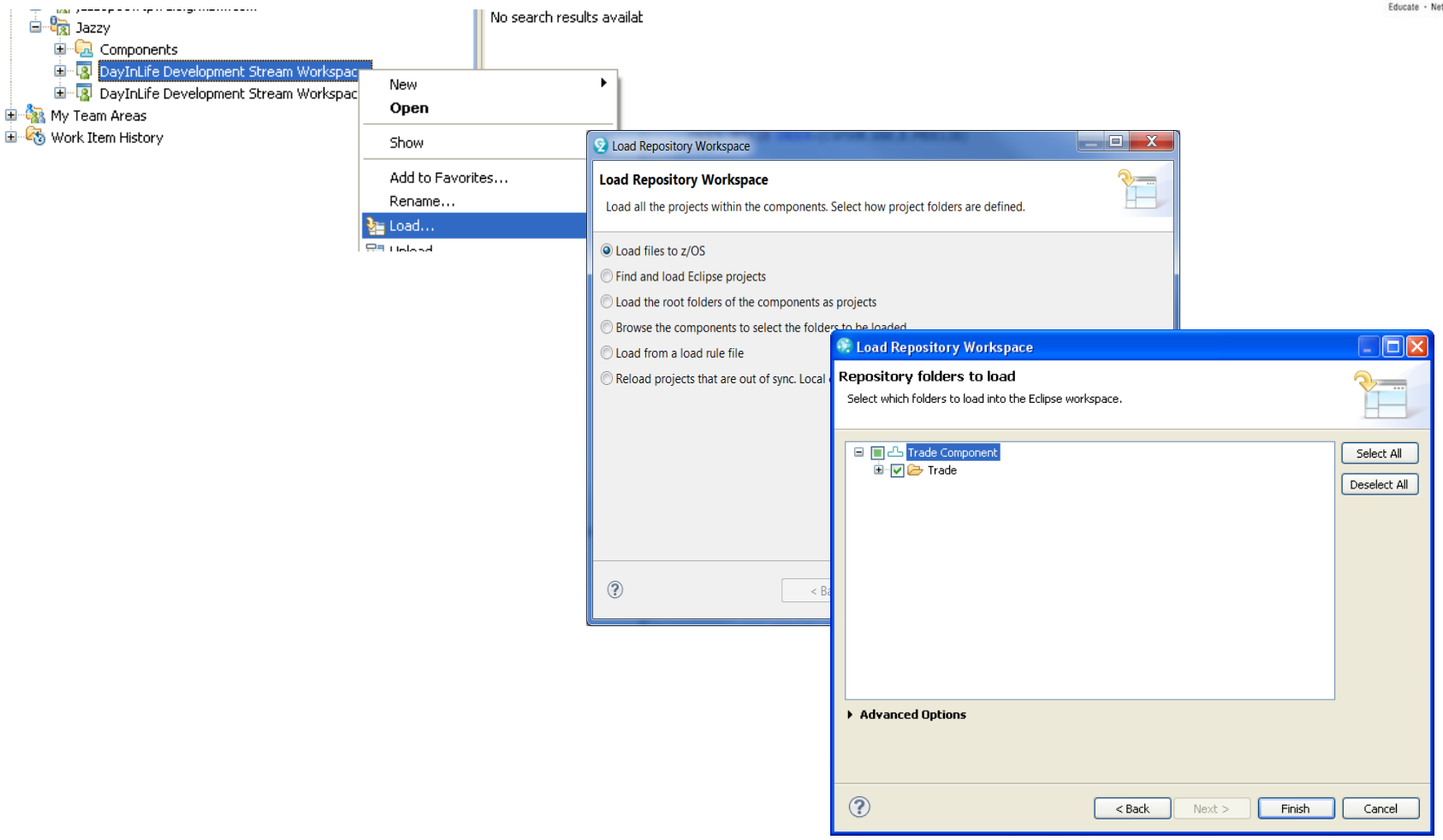
- Rational's Strategic Source Code tooling is RTC and RDz provides tight integration
- RDz offers integration into a variety of other Source Code Management (SCM) tools as well as a framework for creating SCM integration on your own (CARMA)
- Variety of vendors supply plug-ins to RDz to provide easy access to processes and source code controlled by their products



Source Control Management

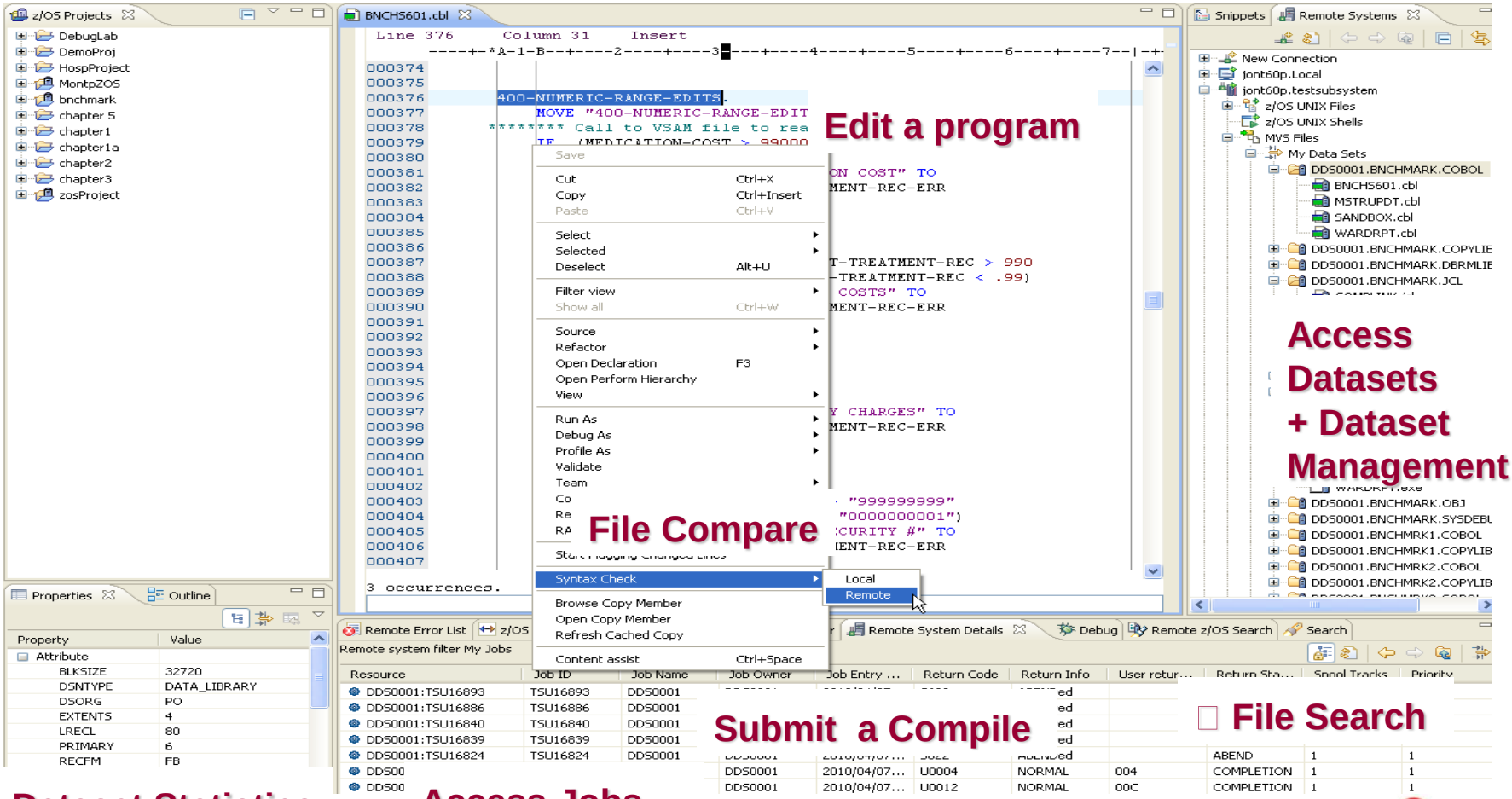


Load the source artifacts



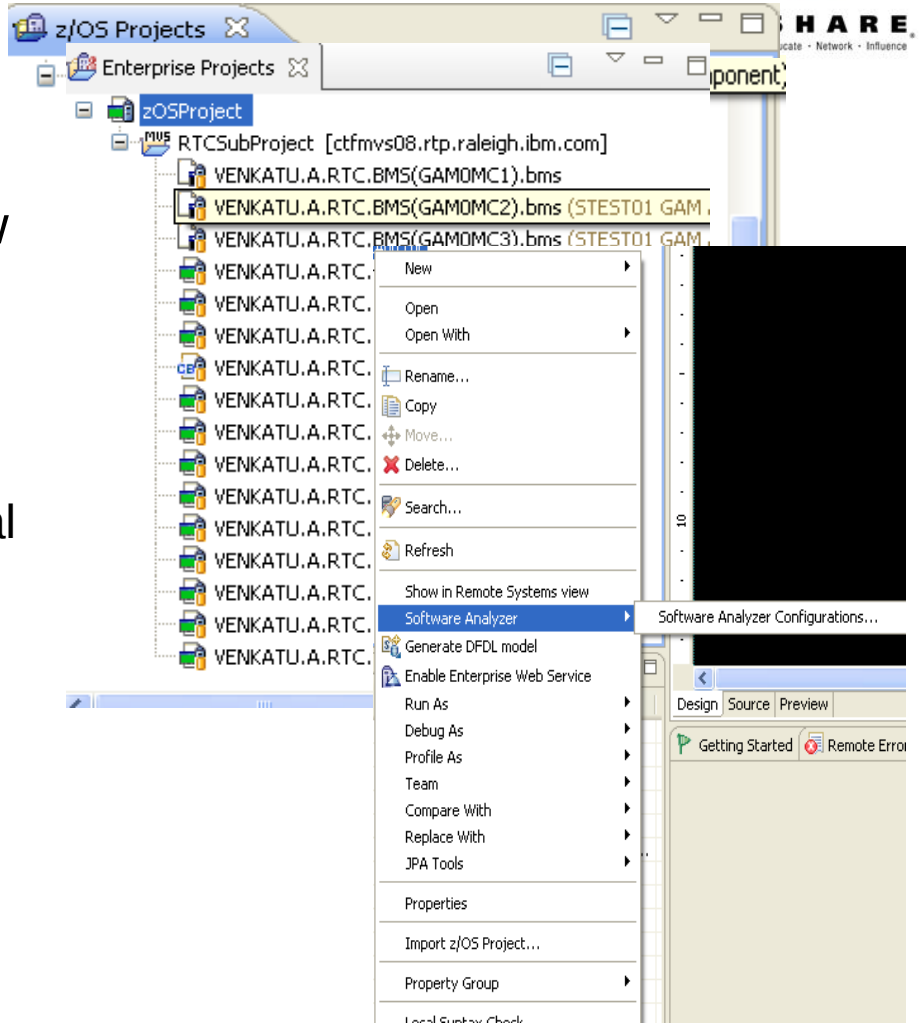
The Benefits of RDz

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



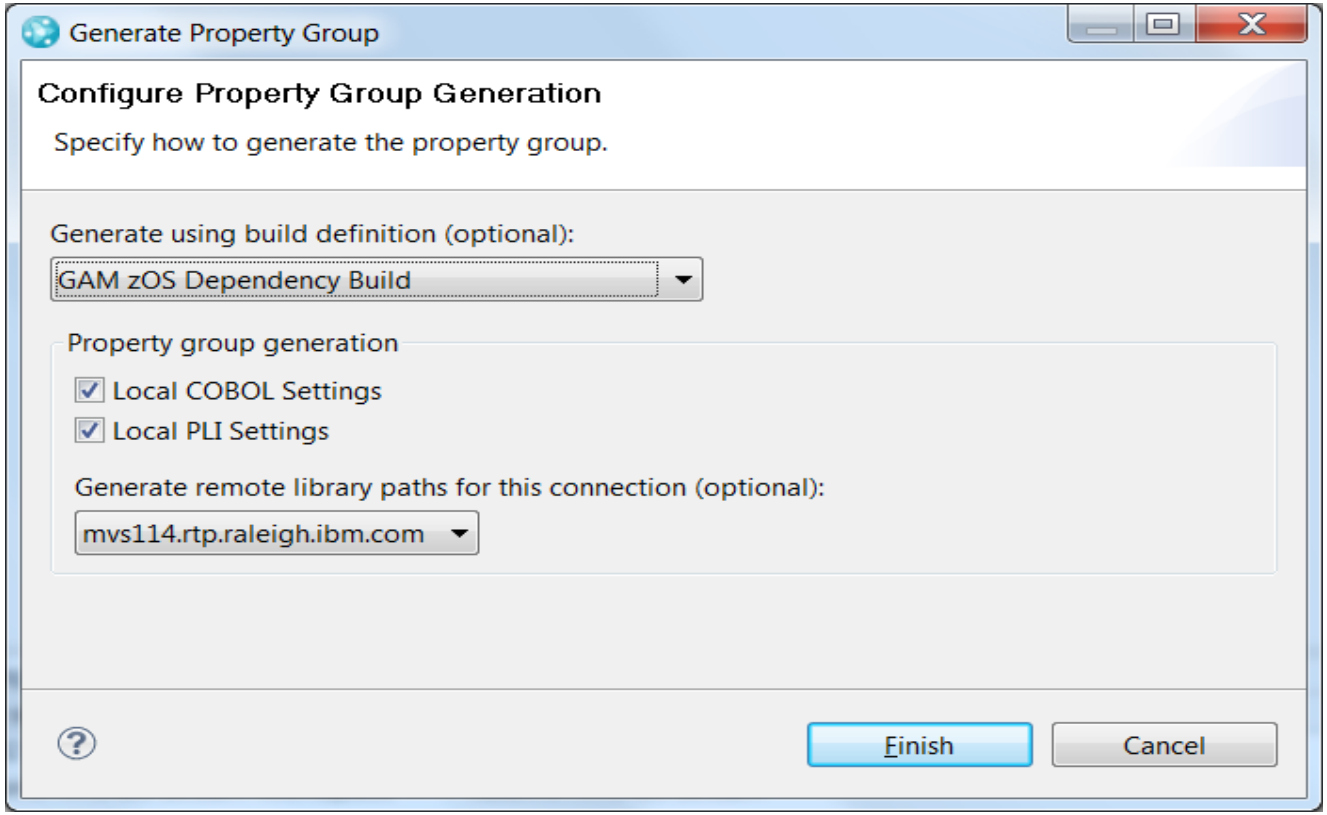
RDz and RTC together

- 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.)



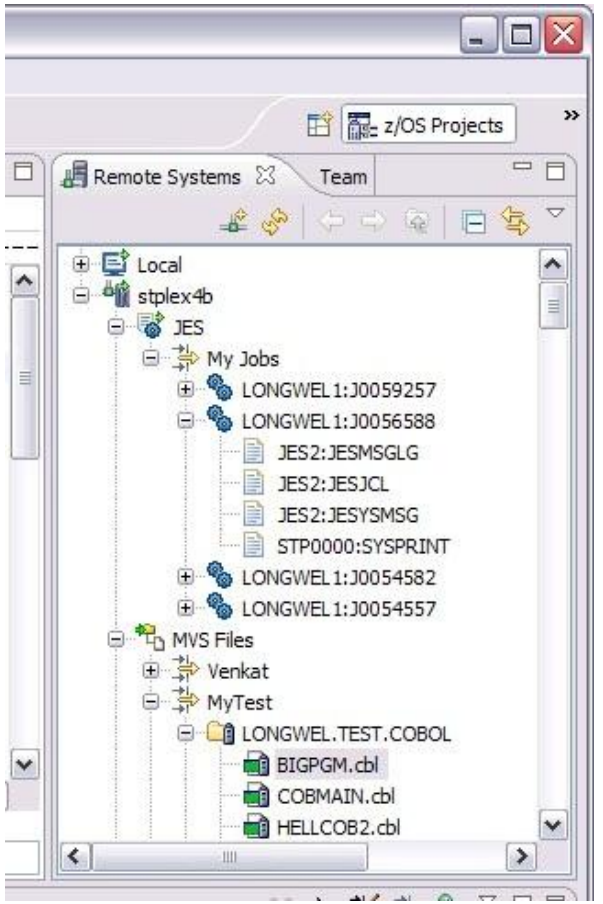
Create a Property Group

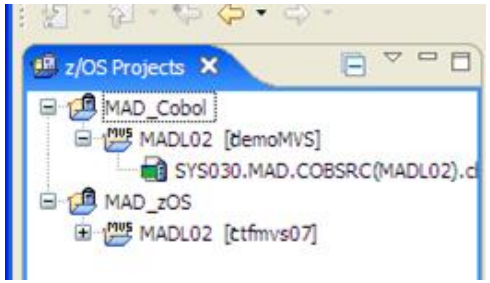
- Generate property groups for your project based on RTC build definition
- Allows RDz to resolve the dependencies and thus offer all the tooling



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

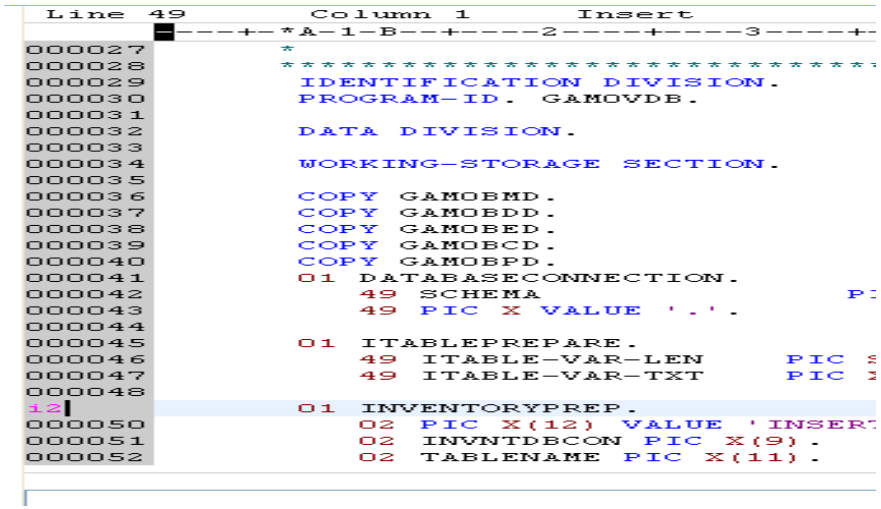




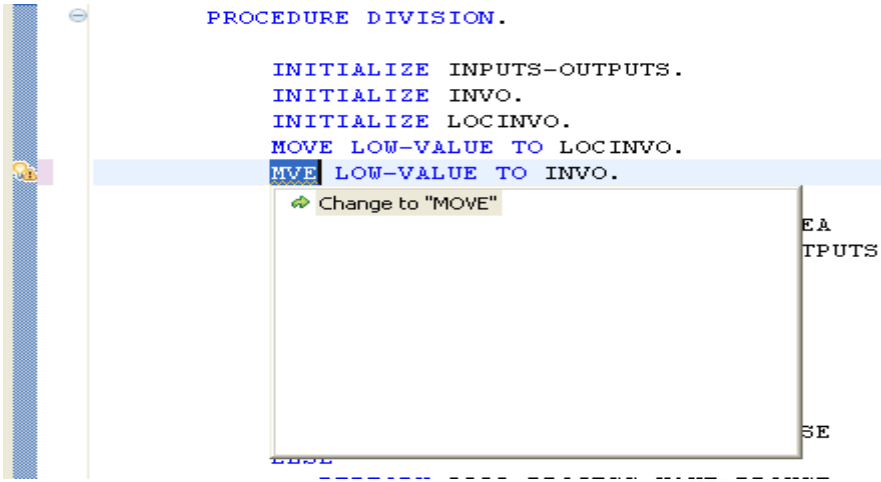
Edit capabilities in RDz

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



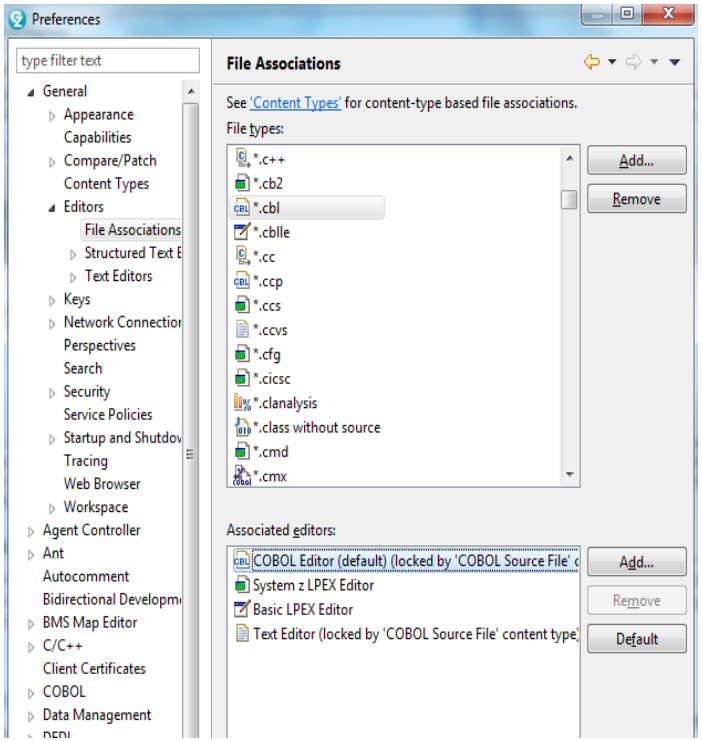
- COBOL, PLI, and JCL advanced editors
  - Based on the Eclipse editor infrastructure, provide more advanced edit capabilities like quick fixes, hyper-linking, hover, easy navigation between various edit sessions or within the same edit session.
  - Supports real time syntax checking, content assist, key word highlighting etc.

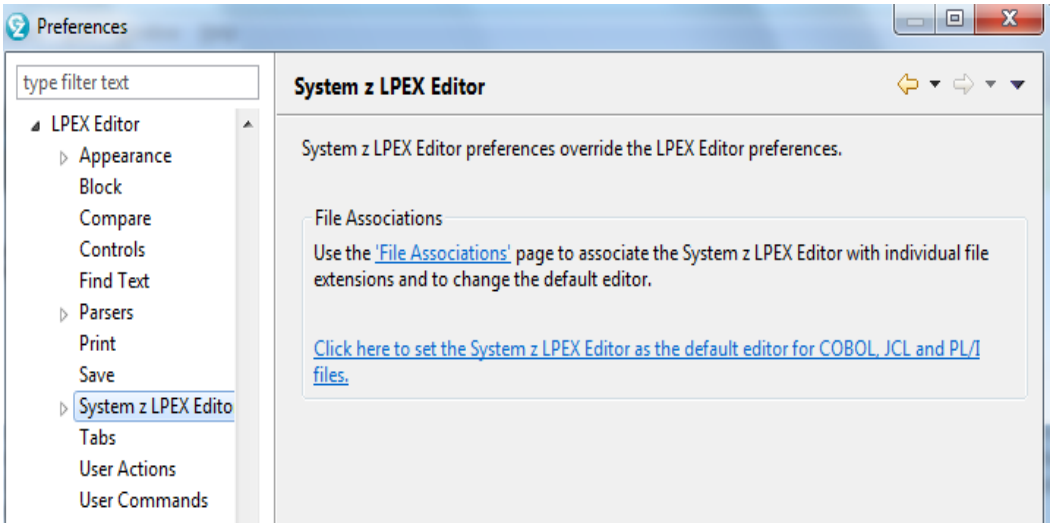


COBOL, PL/I and JCL Editor – the new default

- New workspaces created in RDz 9.x
  - New editors are the default
  - Single click switch to LPEX is available

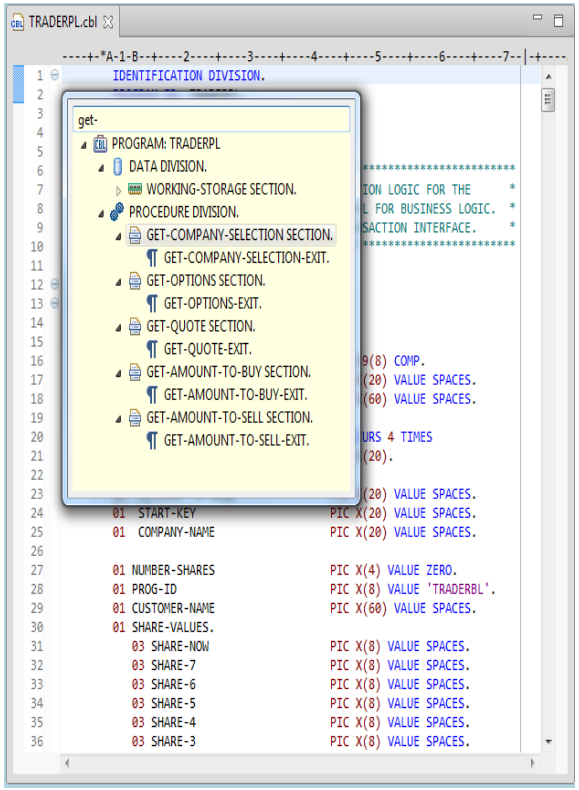
Applicable to source and include/copybooks

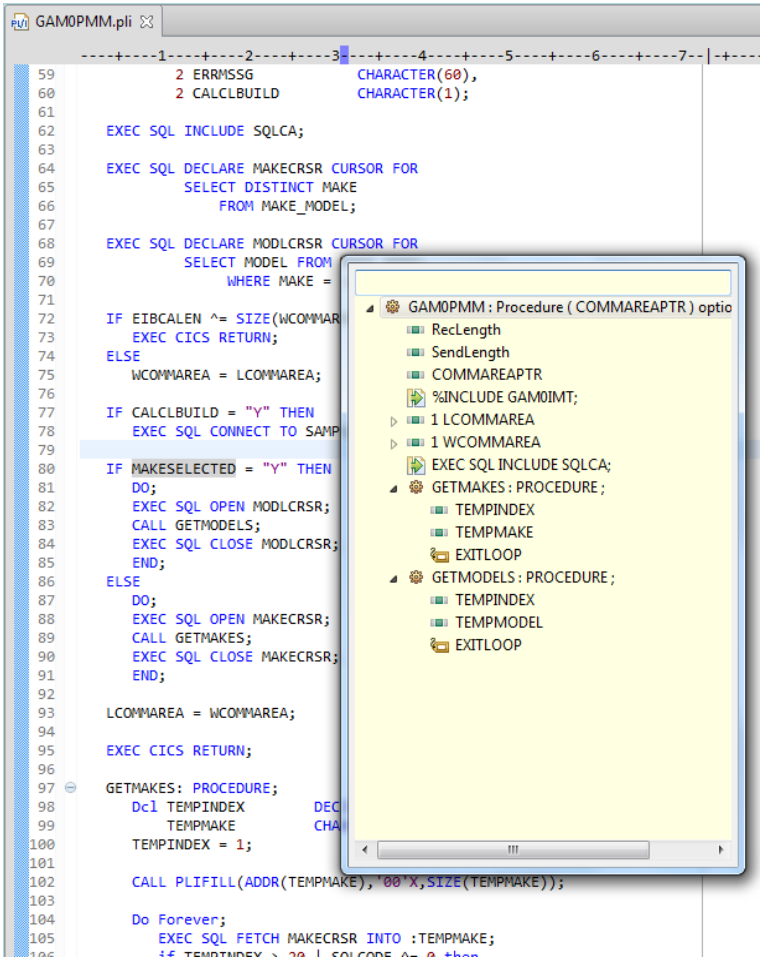




COBOL, PL/I and JCL Editor improvements

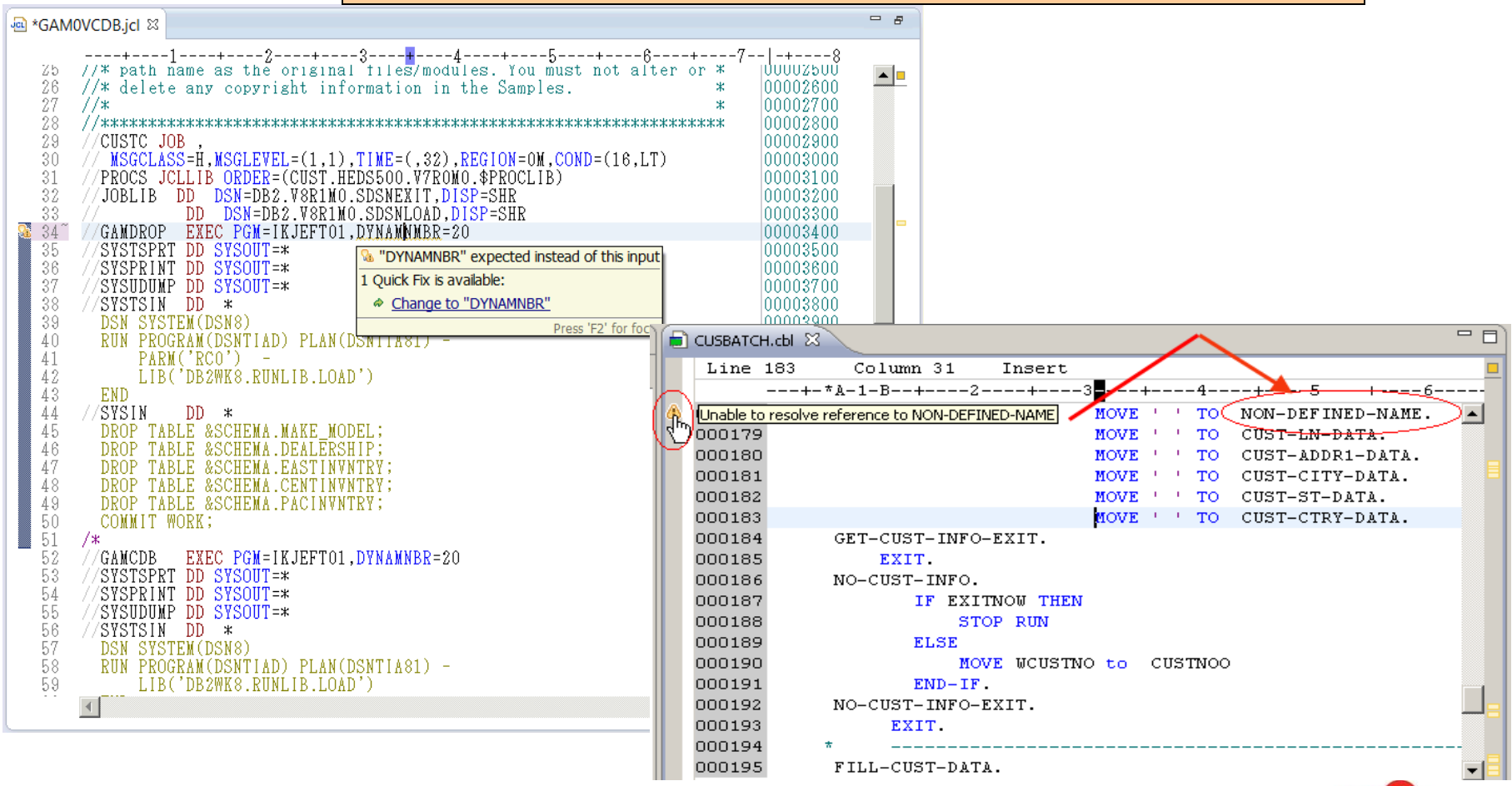
- **Quick Outline**
  - Press Ctrl+O to activate





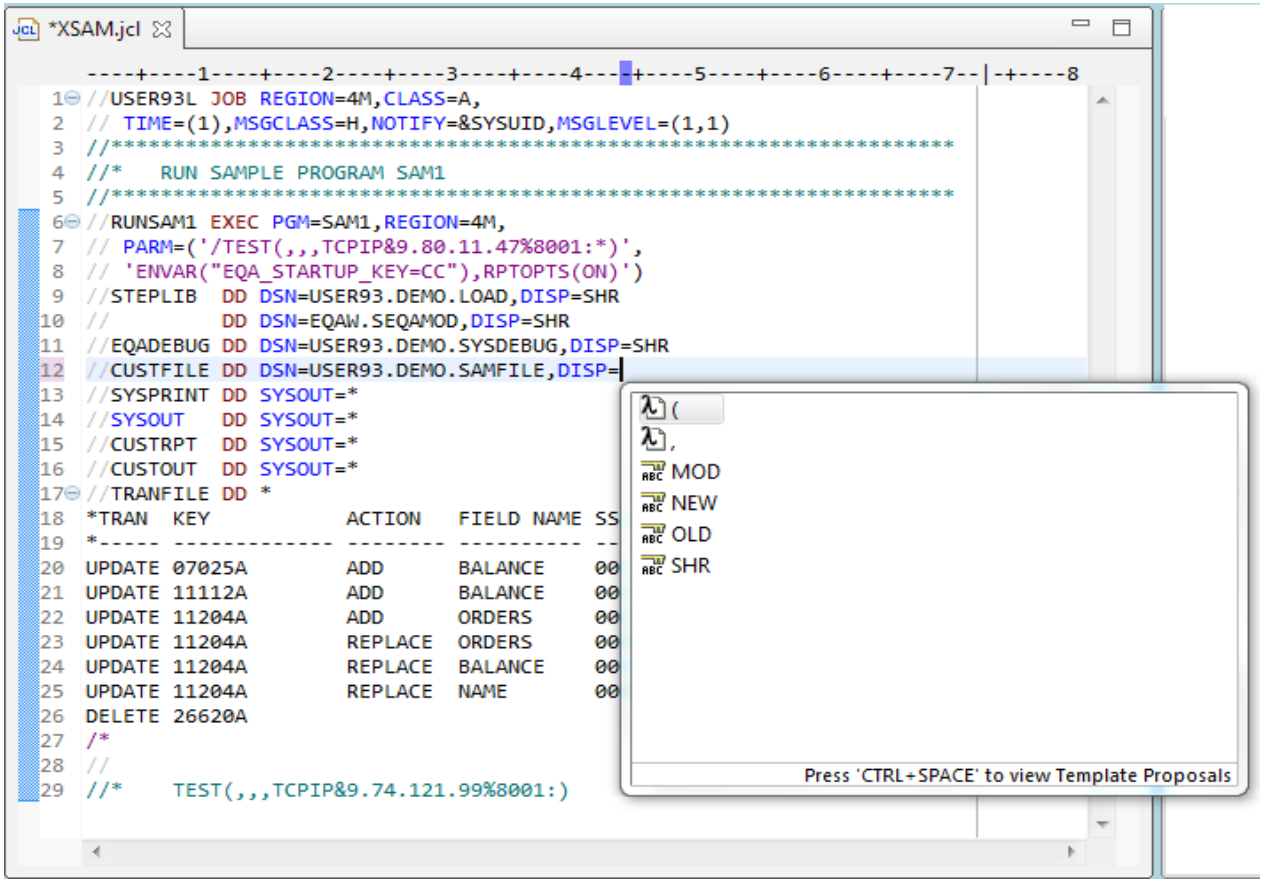
Editor Productivity Features – real time syntax checking

Real-time syntax check without requiring code compile or save



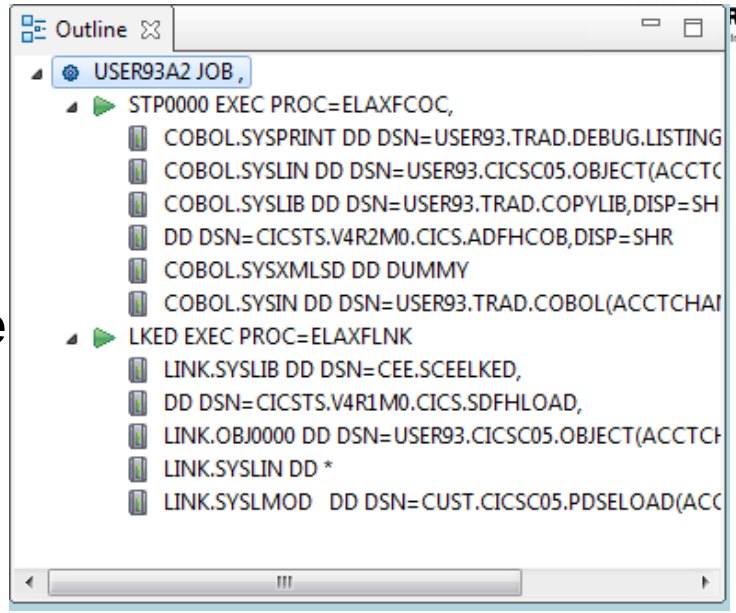
JCL Editor Content Assist

- **Keyword syntax proposals**
  - Press Ctrl+SPACE to activate

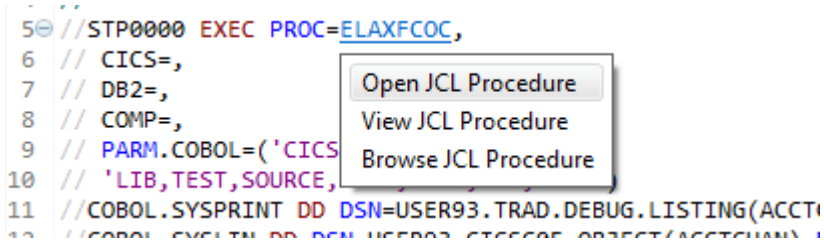


JCL Editor improvements

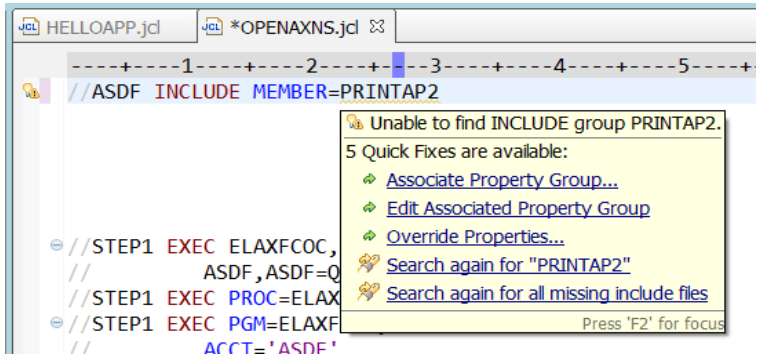
- **JCL Outline shows all DDs**
  - No longer only instream



- **Open actions allow Open, View, Browse**

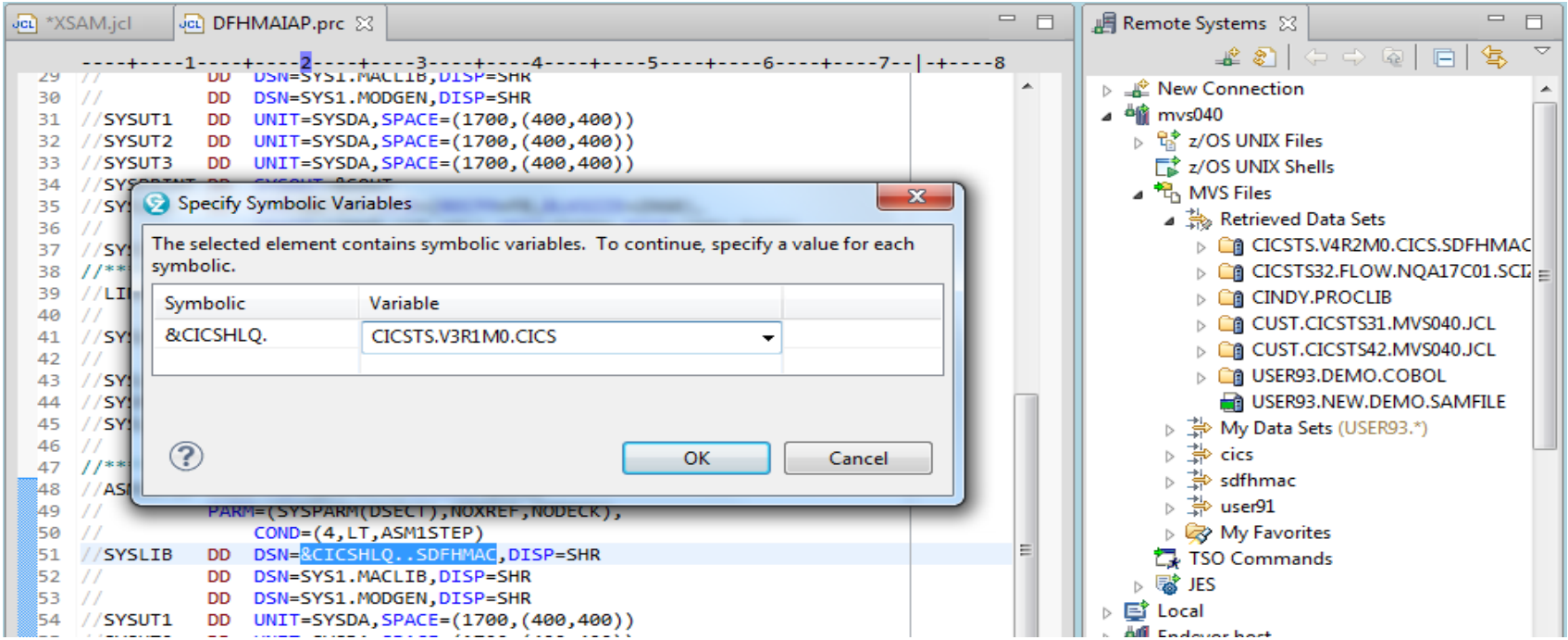


- **Improvements for INCLUDE members**
  - Hover to see contents
  - Open actions support
  - Syntax check with Quick fixes

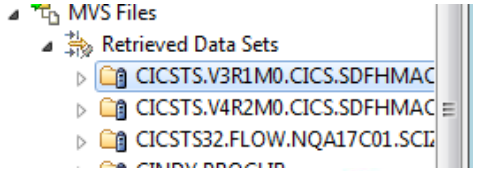


JCL Editor Symbolic resolution

- **Open Declaration (F3) on data set with Symbolic variables**

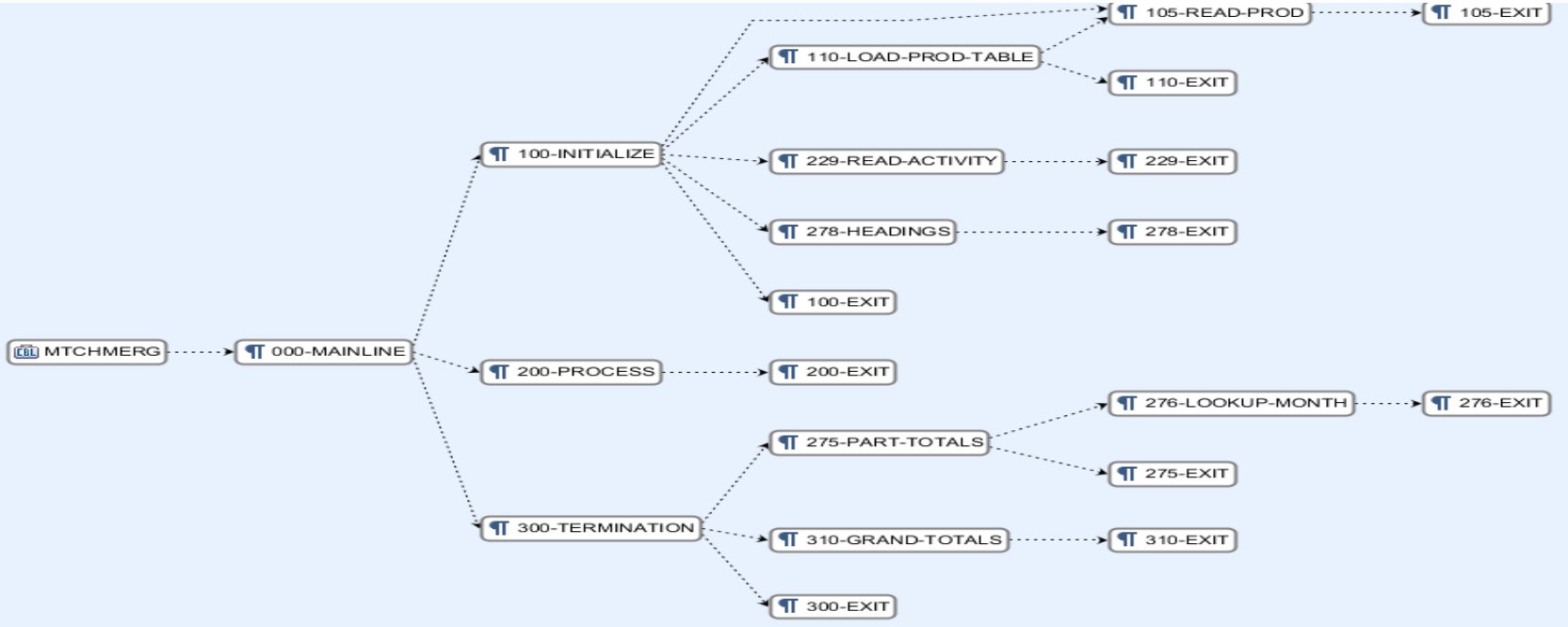


- **Data set is Retrieved using specified value**



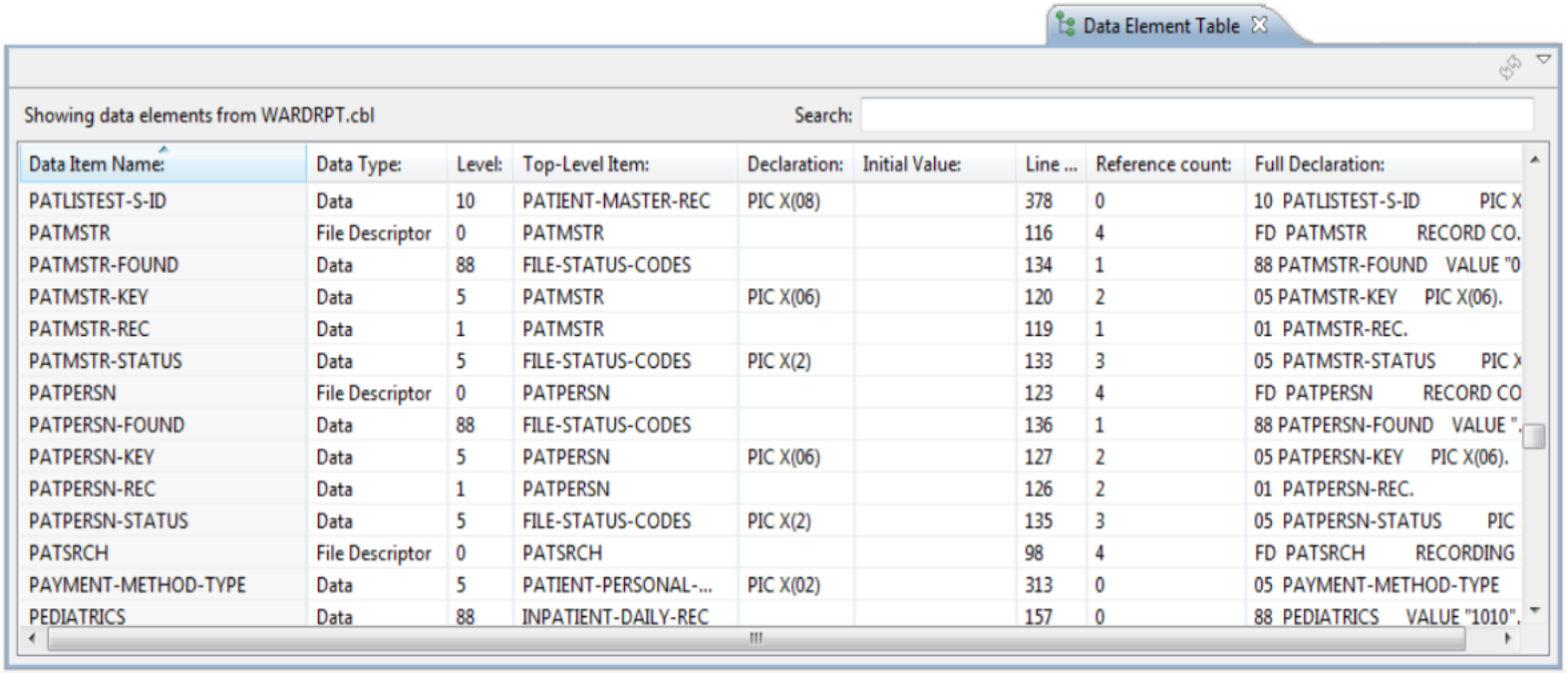
Enhanced Application Quality & Structure Analysis

- Application Analysis
  - Control flow diagrams for COBOL and **PLI** programs,
    - ***Graphical representation of the program flow with links to the source***
  - Helps identify and highlight potential unreachable code



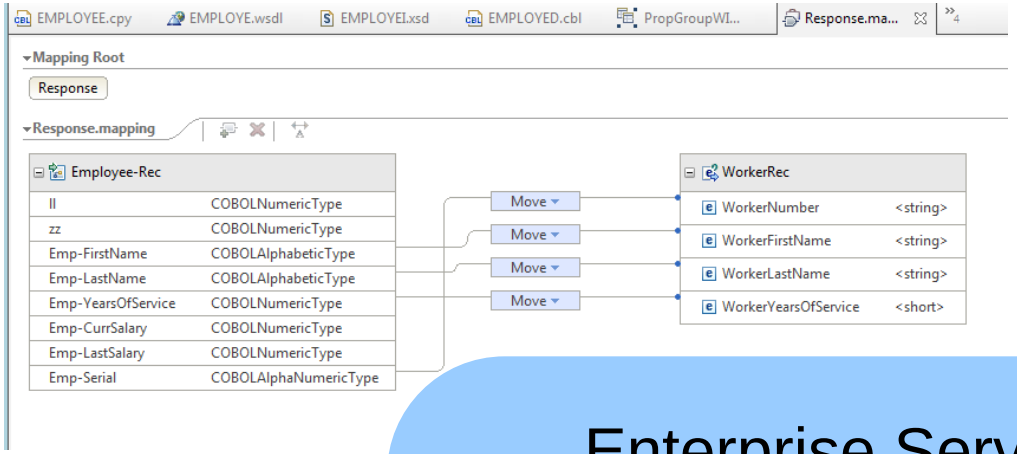
Enhanced Structure Analysis – Data Element Table

- 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 Item Name:	Data Type:	Level:	Top-Level Item:	Declaration:	Initial Value:	Line ...	Reference count:	Full Declaration:
PATLISTEST-S-ID	Data	10	PATIENT-MASTER-REC	PIC X(08)		378	0	10 PATLISTEST-S-ID PIC X
PATMSTR	File Descriptor	0	PATMSTR			116	4	FD PATMSTR RECORD CO.
PATMSTR-FOUND	Data	88	FILE-STATUS-CODES			134	1	88 PATMSTR-FOUND VALUE "0
PATMSTR-KEY	Data	5	PATMSTR	PIC X(06)		120	2	05 PATMSTR-KEY PIC X(06).
PATMSTR-REC	Data	1	PATMSTR			119	1	01 PATMSTR-REC.
PATMSTR-STATUS	Data	5	FILE-STATUS-CODES	PIC X(2)		133	3	05 PATMSTR-STATUS PIC X
PATPERSN	File Descriptor	0	PATPERSN			123	4	FD PATPERSN RECORD CO
PATPERSN-FOUND	Data	88	FILE-STATUS-CODES			136	1	88 PATPERSN-FOUND VALUE "
PATPERSN-KEY	Data	5	PATPERSN	PIC X(06)		127	2	05 PATPERSN-KEY PIC X(06).
PATPERSN-REC	Data	1	PATPERSN			126	2	01 PATPERSN-REC.
PATPERSN-STATUS	Data	5	FILE-STATUS-CODES	PIC X(2)		135	3	05 PATPERSN-STATUS PIC
PATSRCH	File Descriptor	0	PATSRCH			98	4	FD PATSRCH RECORDING
PAYMENT-METHOD-TYPE	Data	5	PATIENT-PERSONAL-...	PIC X(02)		313	0	05 PAYMENT-METHOD-TYPE
PEDIATRICS	Data	88	INPATIENT-DAILY-REC			157	0	88 PEDIATRICS VALUE "1010".

Traditional development and Enterprise web services



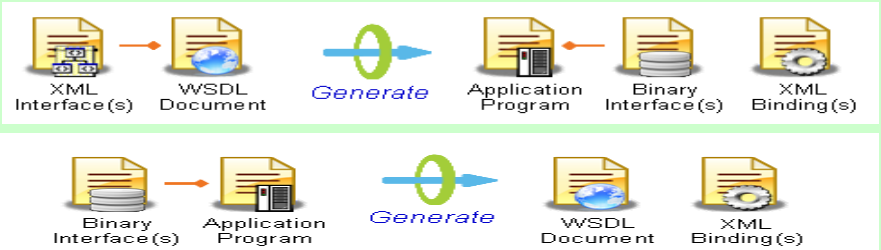
Supports traditional development/maintenance

- Cobol, PL/I, Assembler, JCL

Supports modern architecture development

Enterprise Service Tools

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

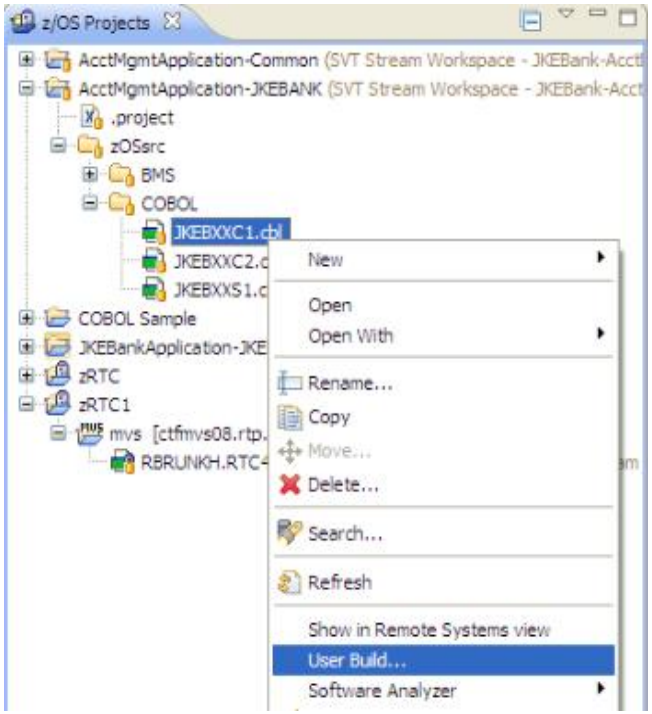


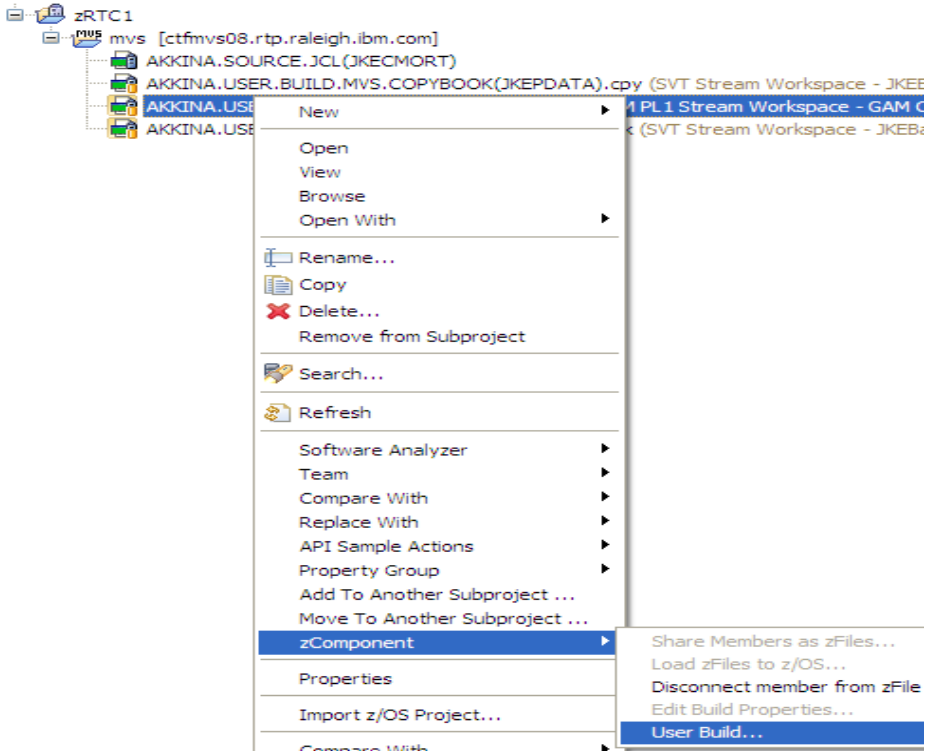
Enable Enterprise Applications for Mobile and Web

User Build from RDz

"User build", is supported both in zComponent projects and RDz remote z/OS projects

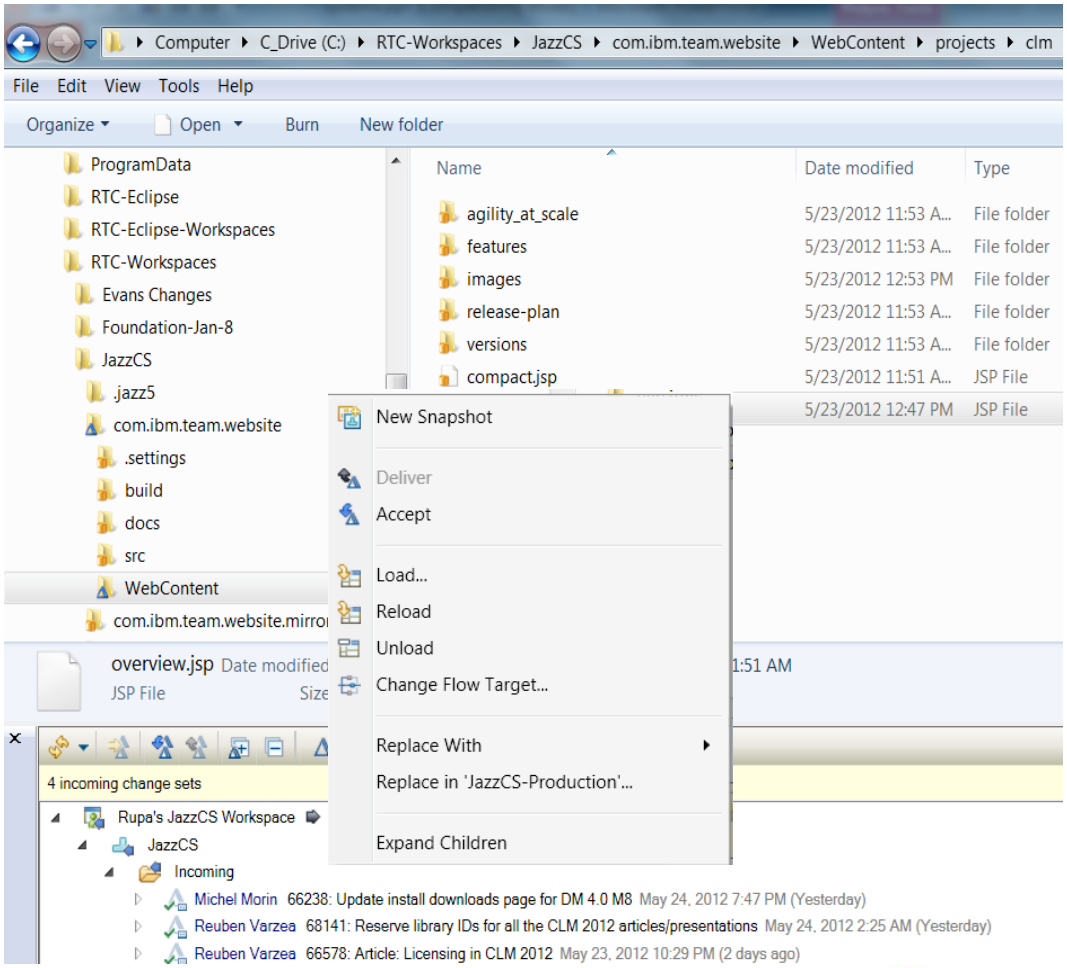
- Builds just one the single file selected, supports Error feedback
- Generates JCL based on the associated RTC Language definitions and Translators





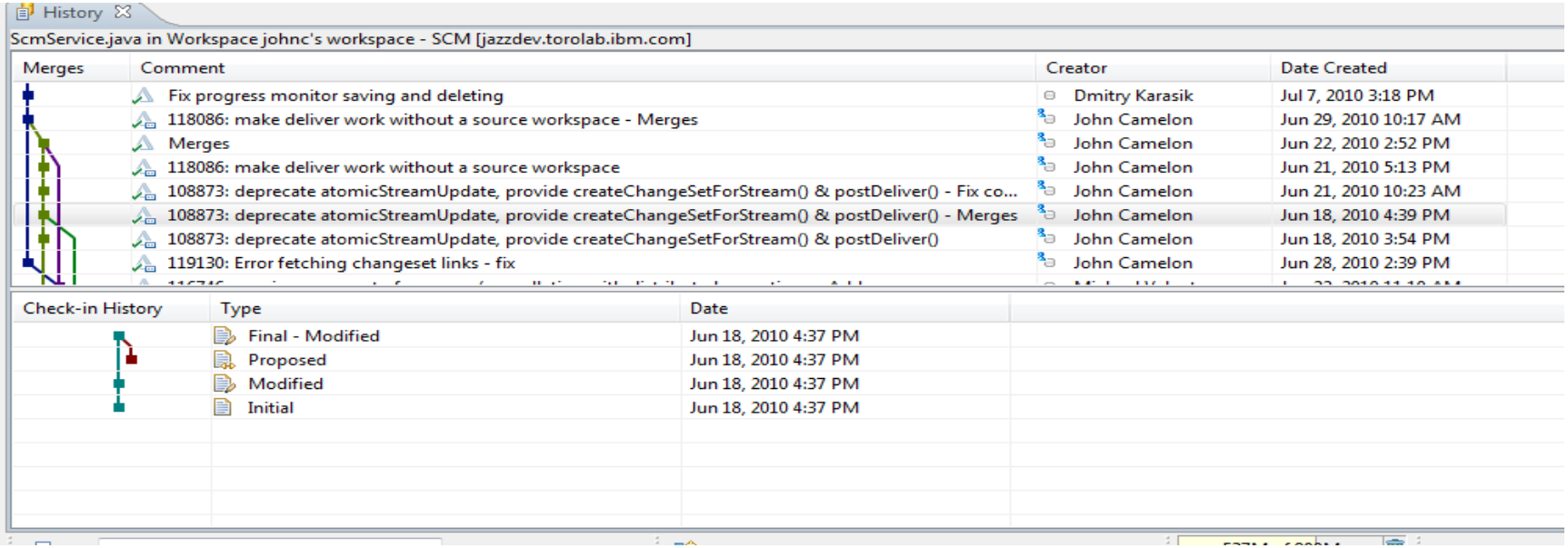
Pending Changes

- 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	Source Dev	Governance/Unit test	Build

• 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

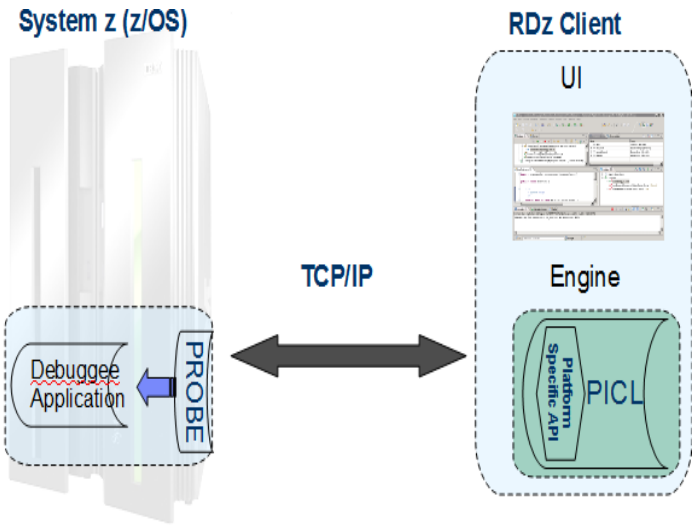
CLM	RDz RTC	**RDz** **RD&T** **RTC**	RTC RDz

Integrated Debugger

- **RDz introduced a new feature called Integrated Debugger**
  - ✓ **A GUI-based multi-platform, multi-language debugger**
    - ✓ Full asynchronous mode
      - ✓ Thread-level control of multi-threaded applications
    - ✓ Automonitor support
  - ✓ RDz v9.0.1 Supported**:**
    - ✓ COBOL V5.1, V4, V3.4
    - ✓ Batch, Batch IMS, Batch DB2, CICS 5.1, 4.2, 4.1
    - ✓ Interactive Code coverage – Out of the box
  - ✓ **RDz 9.1 added support for**:
    - ✓ PLI v4.x, v3.9
    - ✓ C/C++ V1R13, V2R1
    - ✓ IMS TM
    - ✓ DB2 Stored procedures

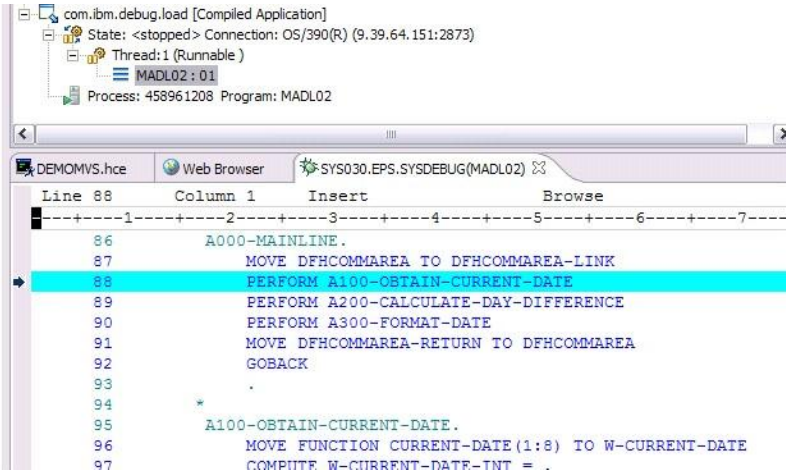
Integrated Debugger

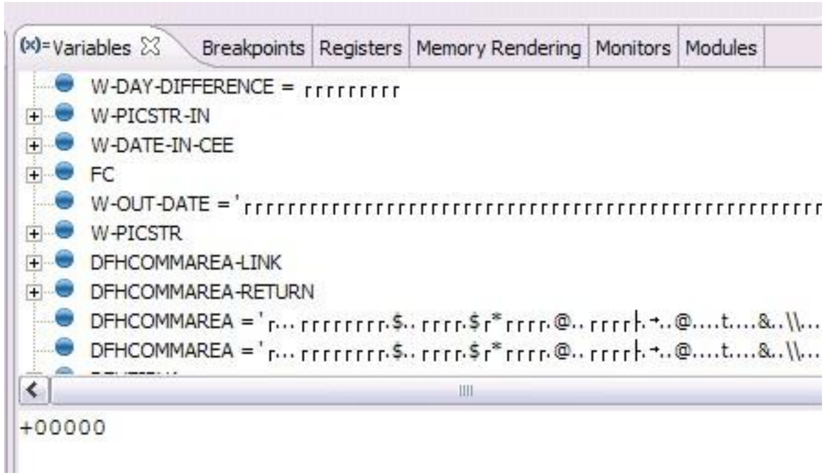
- **Host-offload architecture:**
  - Remote debugger with only a small footprint on the mainframe:
    - Leverages workstation CPUs enabling faster processing of debug information
    - Enables scalability and reliability
  - Debugger client is supported on Windows and Linux
- **Simple and Secure Connections:**
  - Single client can handle multiple debug sessions on multiple hosts or an application the spans multiple systems
  - Client initiated debug – no need to specify client IP address and port (v9.0.1.2)
  - SSL/TLS support



Debug Multiple Runtimes

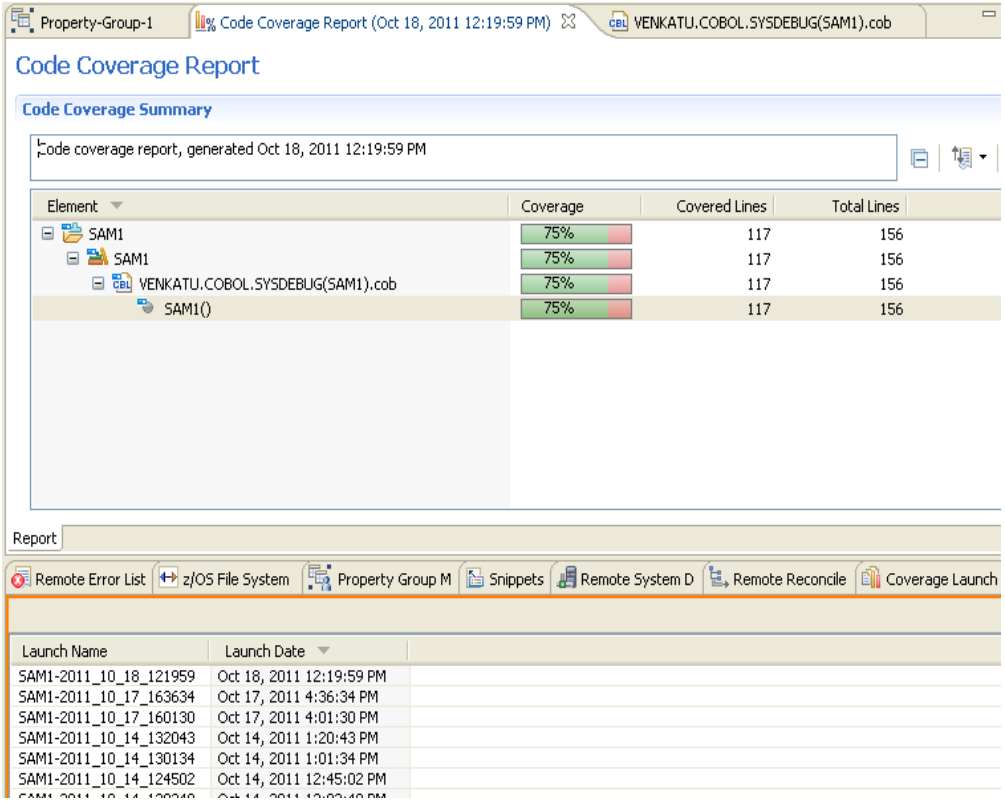
- Use the cross-platform debugger to debug end-to-end systems as they execute in the runtime
  - CICS
  - Batch
  - Java
- From the workstation:
  - View executing source code
  - Step through host code line-by-line
  - Set breakpoints
  - Alter working storage values
  - Alter register values
  - Etc…
- Debug zOS and distributed code in the same interface even stepping between runtimes and platforms!
- Leverage Integration with IBM Debug Tool for other runtimes

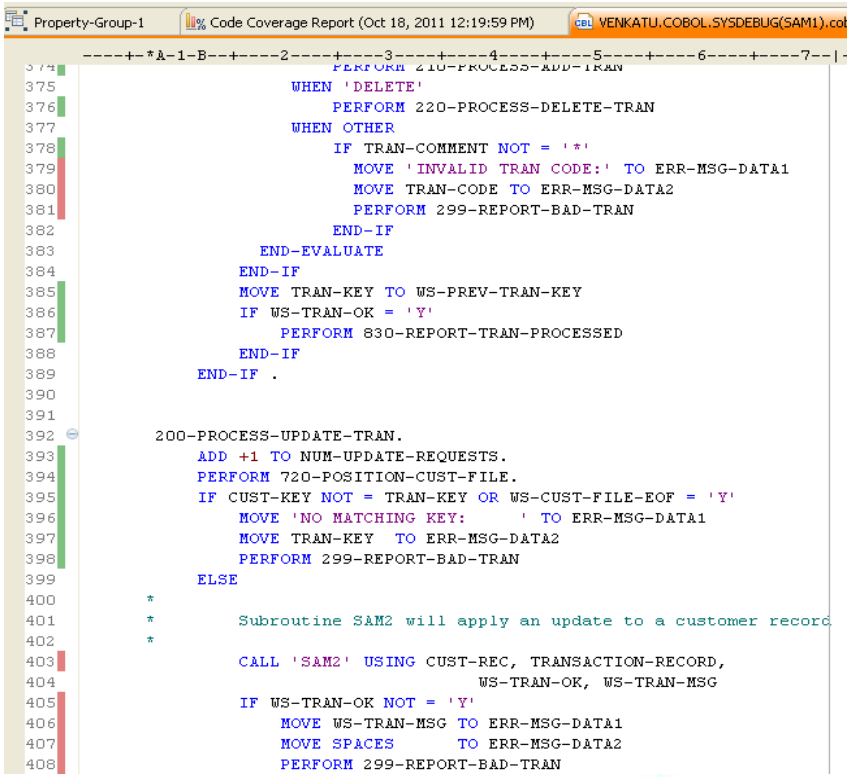




Enhanced Application Quality – Code Coverage

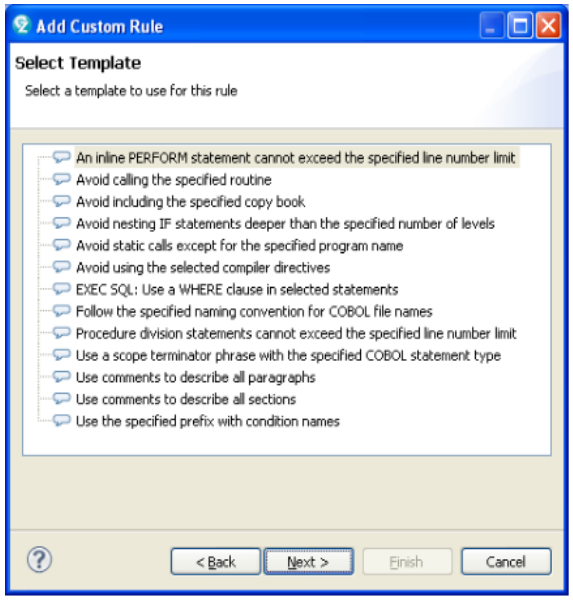
- Line Level Code Coverage - provides tools to measure and report on test coverage of an application
  - Leverages the Integrated Debugger technology
  - Indicating what source code lines were tested and remain to be tested

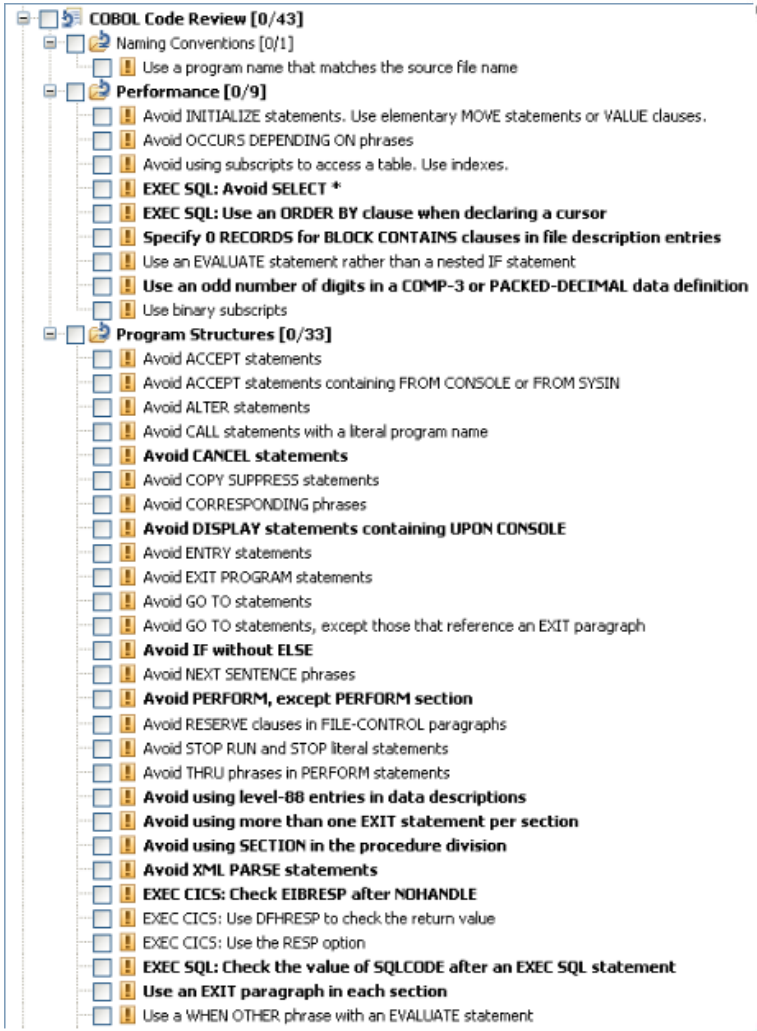




Enhanced Quality & Structure Analysis – Code review

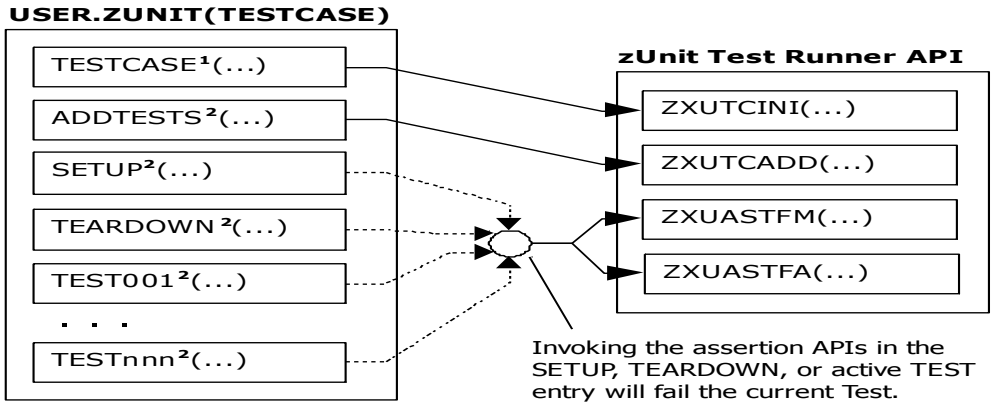
- Code Review/Governance - provides predefined rules and templates for COBOL and PL/I applications
  - Ensure adherence to corporate standards
  - Custom rules for COBOL and PL/I





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 *x*Unit.
- 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



¹Language-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.

²Language-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).

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 Suite referred to in a zUnit configuration file

- zUnit Wizard used to generate Test Cases
  - RDz client feature
  - Eclipse based wizards allow creation of:
    - •Template Test Cases are generated in COBOL or PL/I
      - •Simple pass/fail **assertion** API

    (RDz v9.1) Complete COBOL test cases:

      - **•Identify the interface or set of copy book(s)**
      - **•Generate XML Schema to represent the interface**
      - **•Generate XML files where you would specify test input and expected output**
      - **•Generate a Test Case based on the XML file**
    - **•(Optionally) Generate stubs for called programs**
  - RDz viewers/editors for unit test XML results

Development Life Cycle

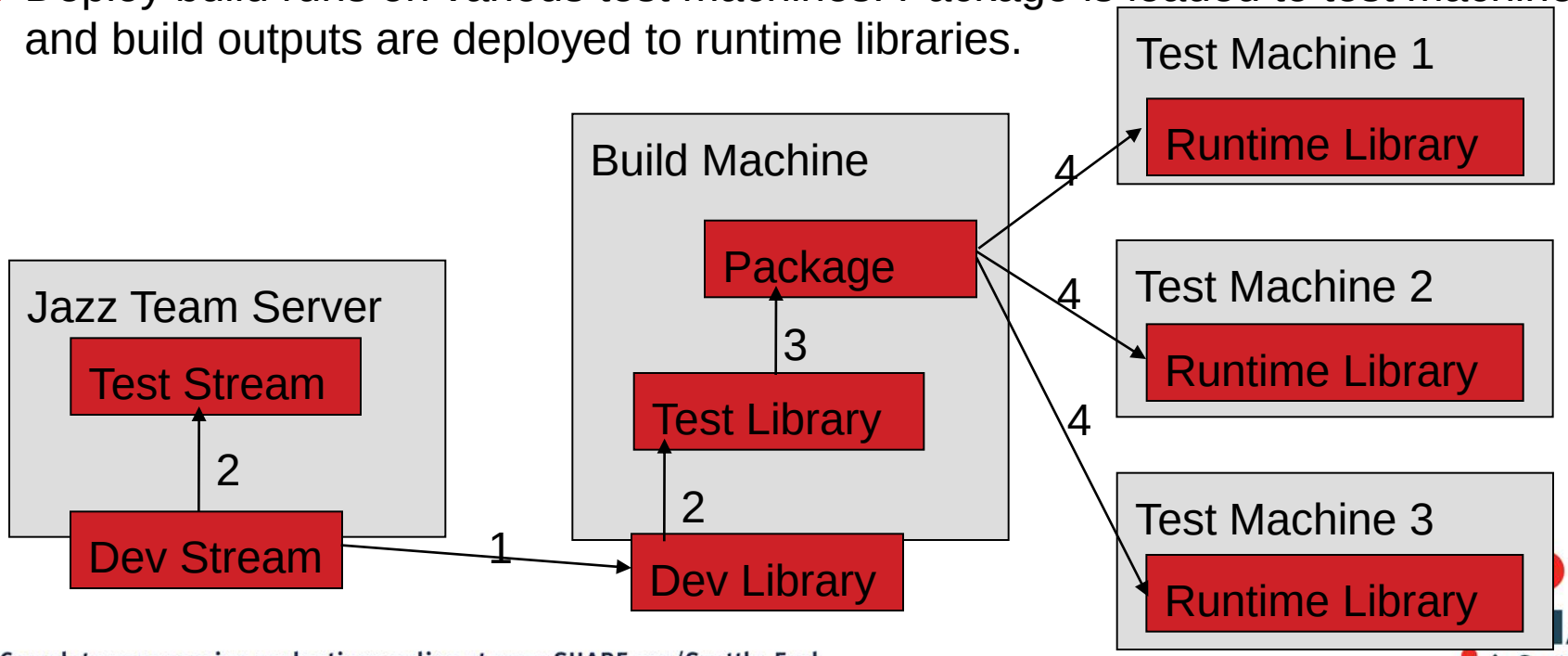
Planning	Source Dev	Governance/Unit test	Build

• 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**

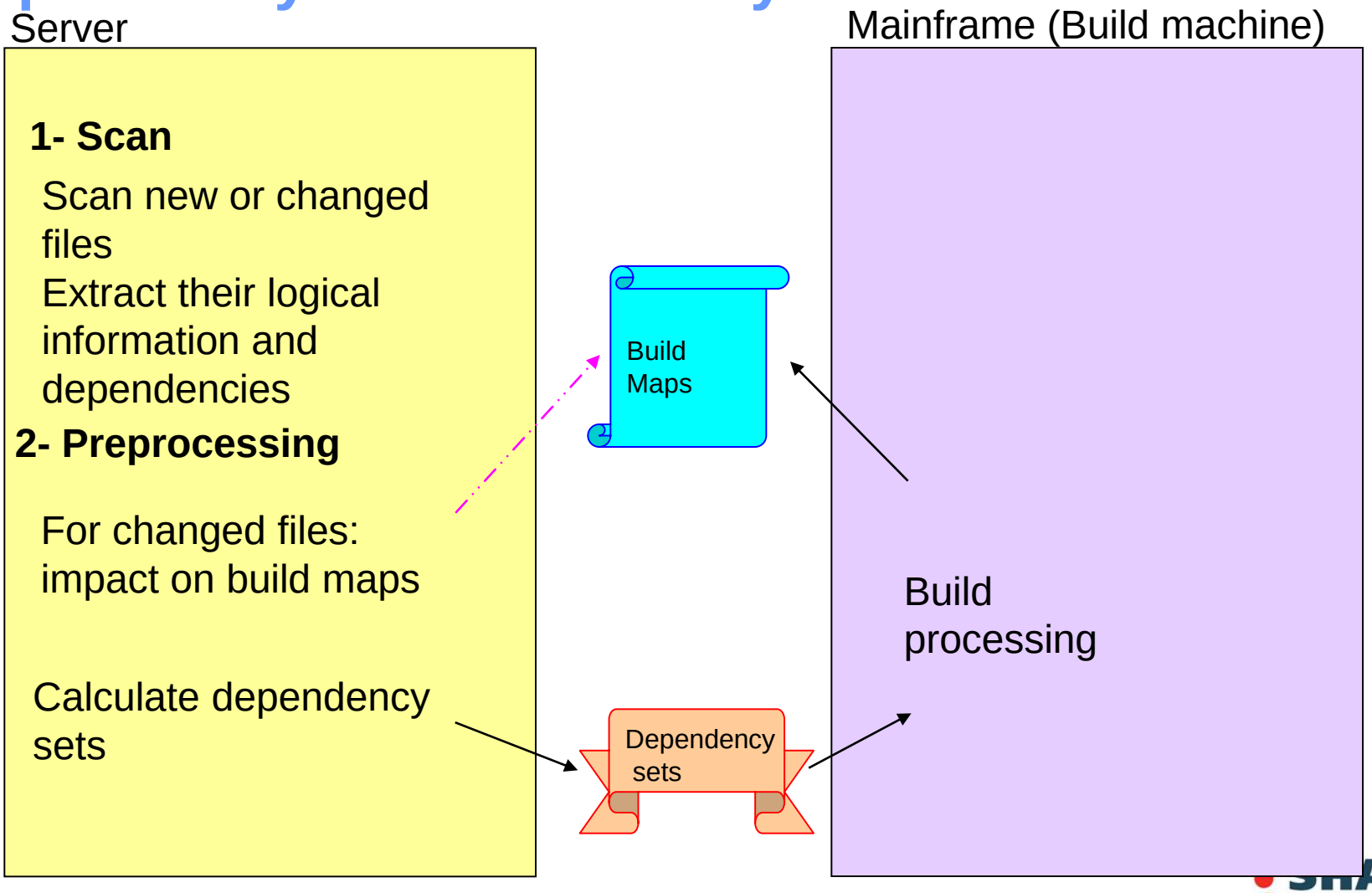
CLM	RDz RTC	RDz RD&T RTC	RTC RDz

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.

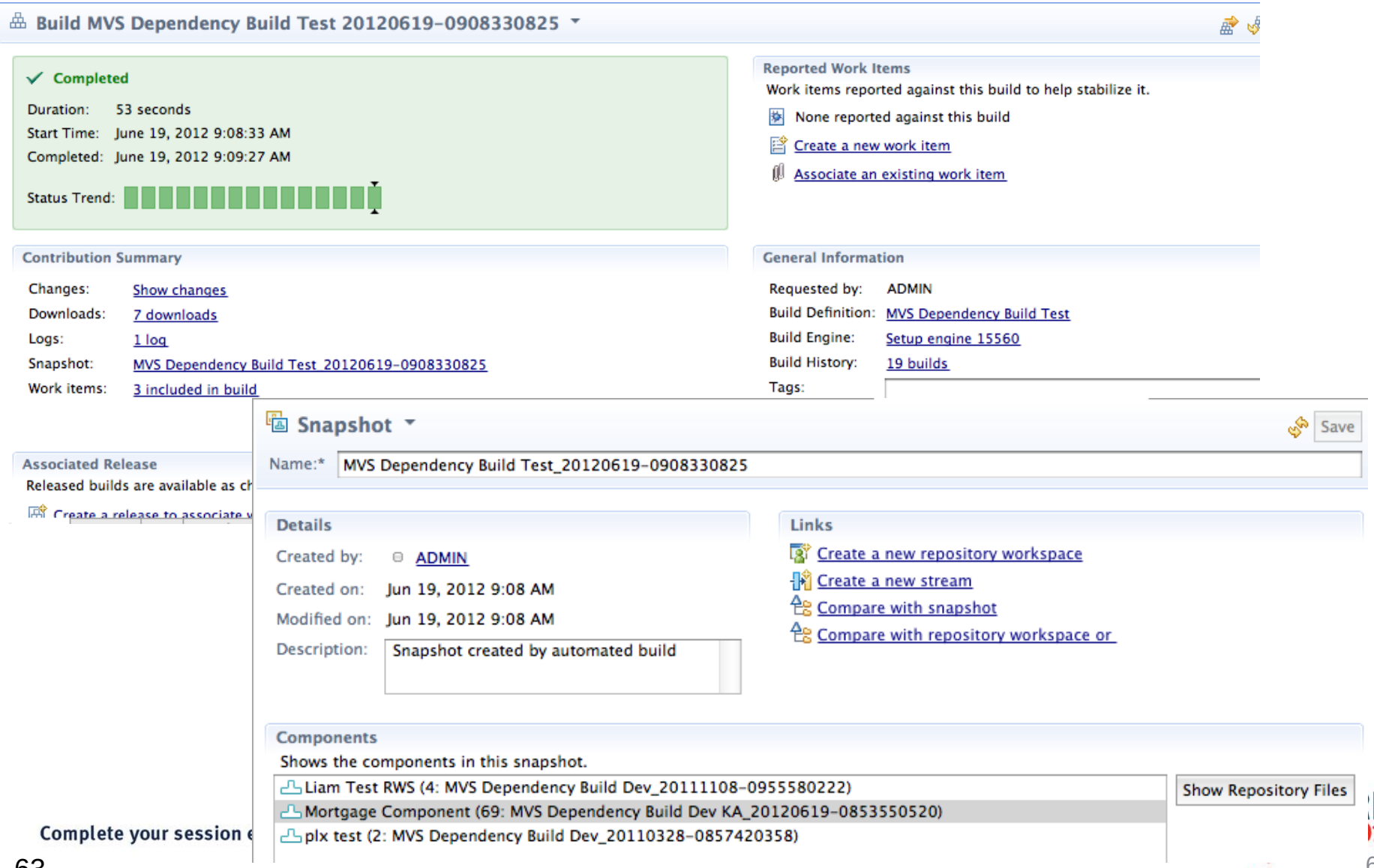


Dependency Build Summary



62

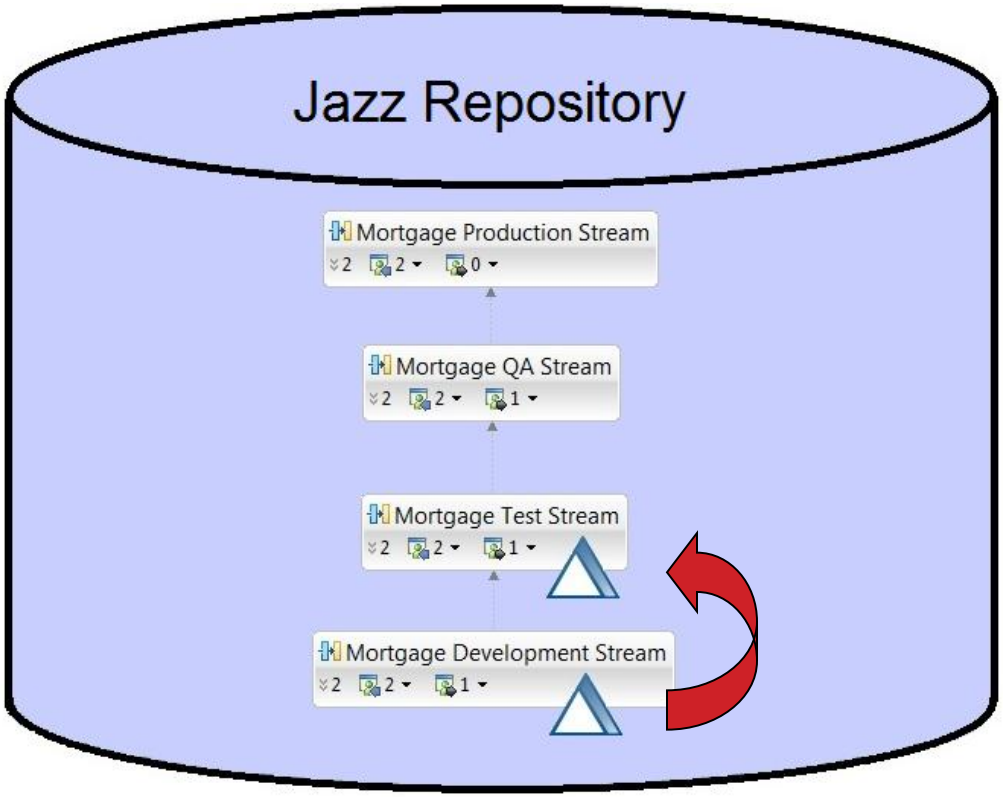
Snapshots for every build



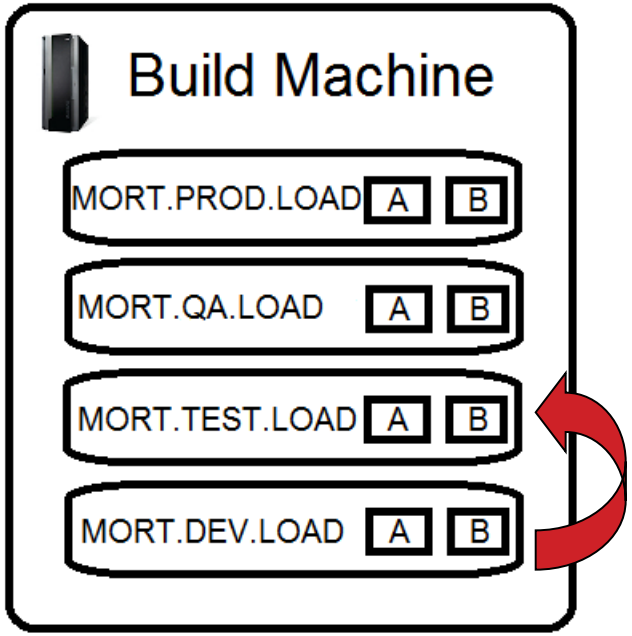
63

Promotion

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



Source



Outputs