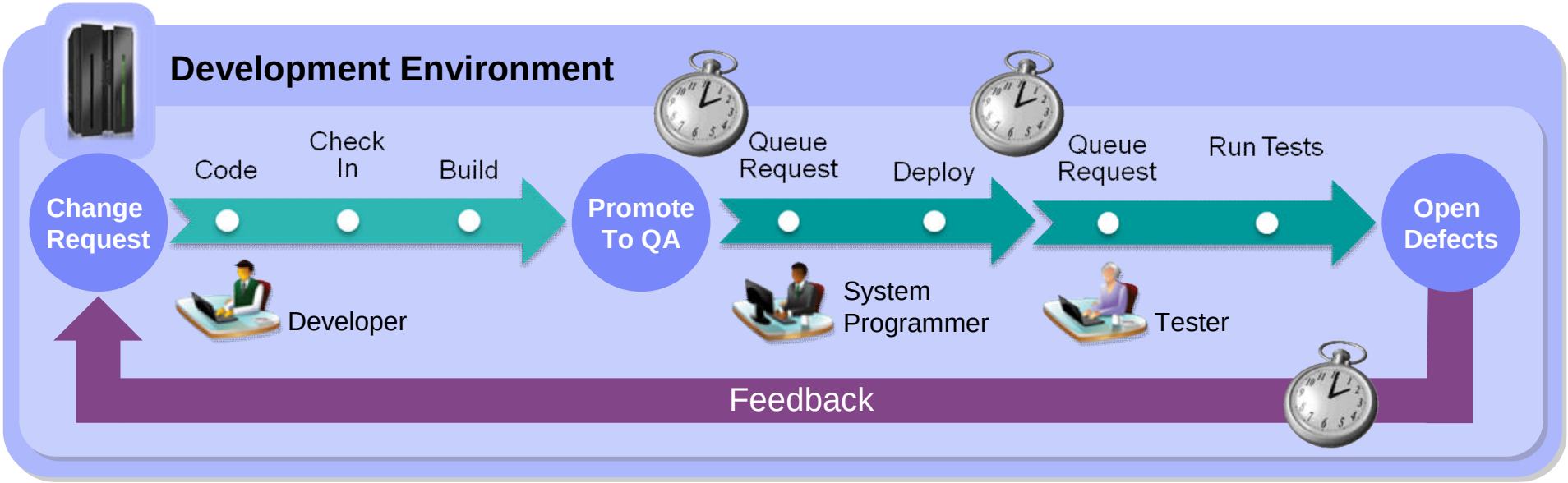
Boost Performance with Smarter Application Testing and Optimization

David Myers

Product Manager, IBM Rational

Enterprises want to… *deliver end-to-end application enhancements quickly to stay competitive, trust that complex enterprise systems can be broadly integrated, and bolster confidence in application quality*



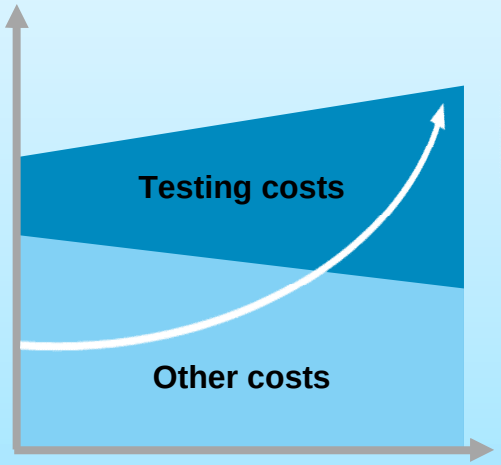
But…

It takes weeks or even months to test and fix changes due to reliance on manual processes and limited access to test resources

Cost, complexity and velocity make today's quality paradigm impractical

An estimated 60 - 80 percent of the cost of software development is in rework*

Increasing Cost of Quality

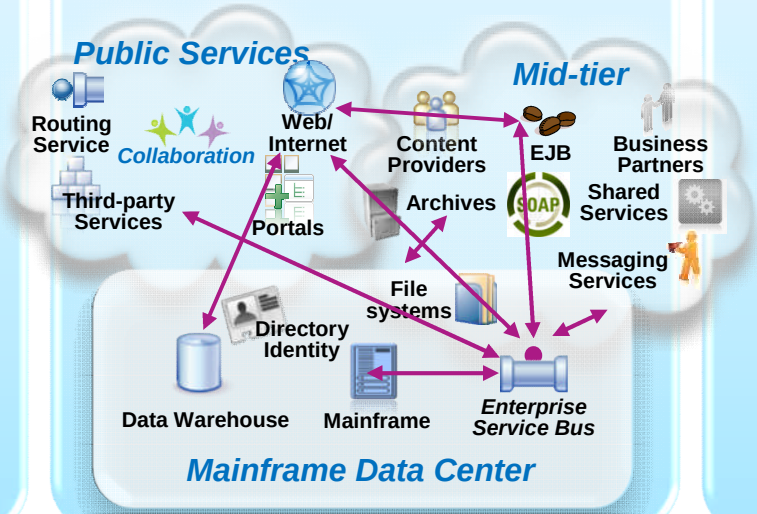


Outsourcing **labor** is no longer a sustainable model as global wages are increasing

13%

The forecasted increase in wages for India IT workforce in 2011[a]

Increasing Development Complexity



Product and application **complexity** and size are increasing

$5-30 million

The typical investment to build a single test lab for a Fortune 500 company. Most have dozens[b]…

Balancing Quality and Speed

Productivity is inhibited as test teams can no longer keep up with development output

30-50%

The average amount of time testing teams spend on setting up test environments, instead of testing[c]

* Source: http://www.sei.cmu.edu/about/message/
Last Updated: 17 January 2012

Business constraints with mainframe development today

Limits the velocity of System z application delivery

"Operations tell me it will take two months to get my test system allocated."

"My development capacity charge-back is consuming my entire budget. I can't afford tools."

"I can only test my batch applications in offline hours. Online apps consume the 9-5 cycles."

"We don't have the capital budget to obtain more mainframe test resources for my developers."

"It is difficult for my developers to learn the mainframe. Operations controls can prevent experimentation by developers.."

"I can't even work on Mondays! Production workload kicks me off."

"I want to try out creating Event Processing and ATOM apps, but my system isn't scheduled for a CICS/IMS update till 2012."

"The Mainframe isn't cool anymore."

Cost is a significant driver

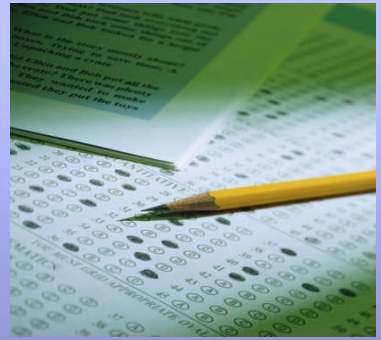
80% of development costs are spent identifying and correcting defects!*

During the **CODING** phase

$80/defect

During the **BUILD** phase

$240/defect



During the **QA/TESTING** phase

$960/defect

Once **released** as a product

$7,600/defect

+

Law suits, loss of customer trust, damage to brand

If admitted or not most development LPARs are managed as if starting here

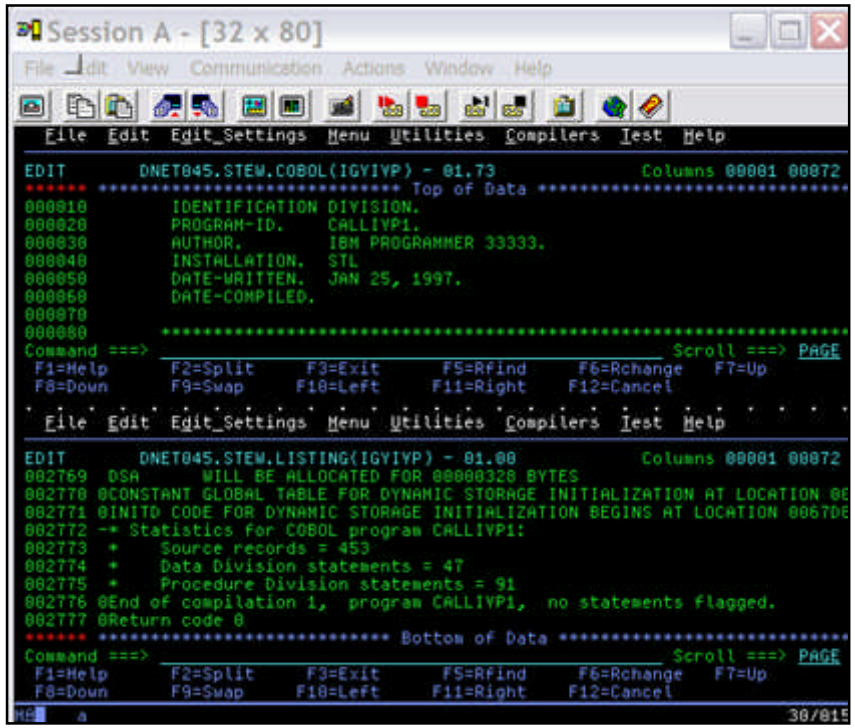
*National Institute of Standards & Technology

Source: GBS Industry standard study

st derived in assuming it takes 8 hrs 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.

Existing development environment

- Multiple screens
- Multiple disparate tools
- 20 x 80 characters of content

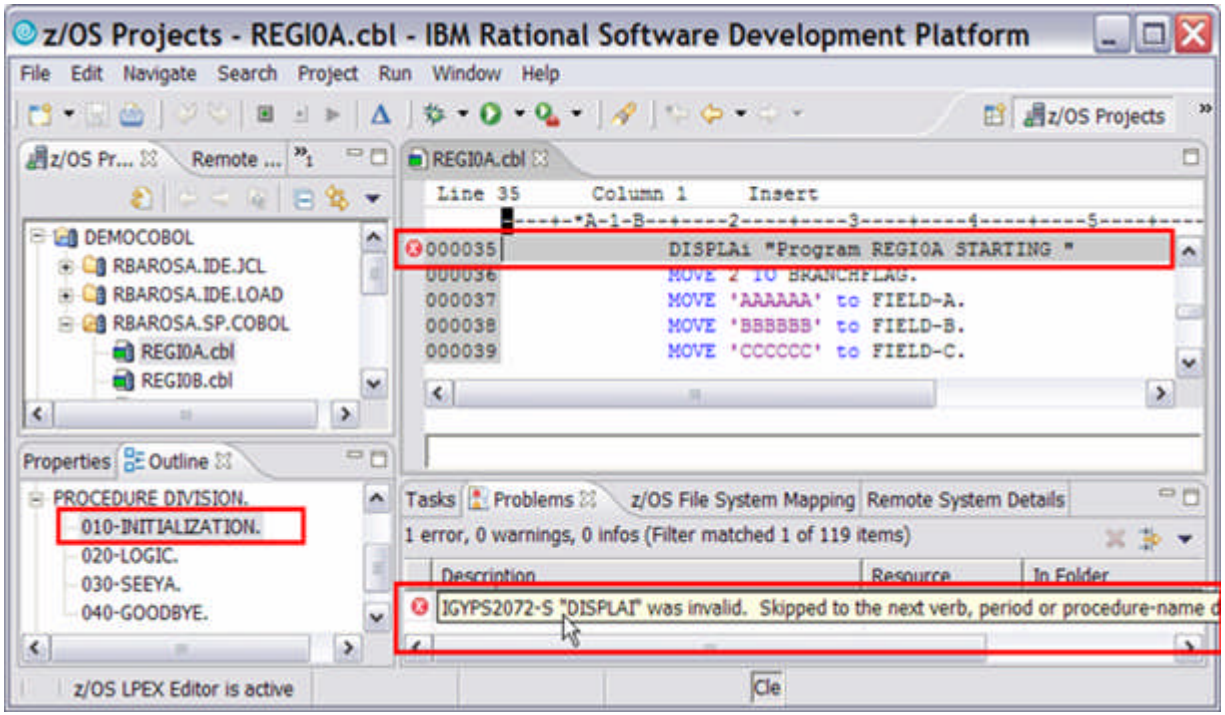


Steps to validate a change

- Edit source
- Find code line
- Change code
- Exit source
- Find JCL
- Edit JCL
- Submit compile job
- Swap to SDSF
- Select job
- Find error message
- Swap to JCL
- Exit JCL
- Find source file
- <repeat>

Tomorrow's development environment

- Information at your fingertips
- Easy navigation
- Automatic feedback

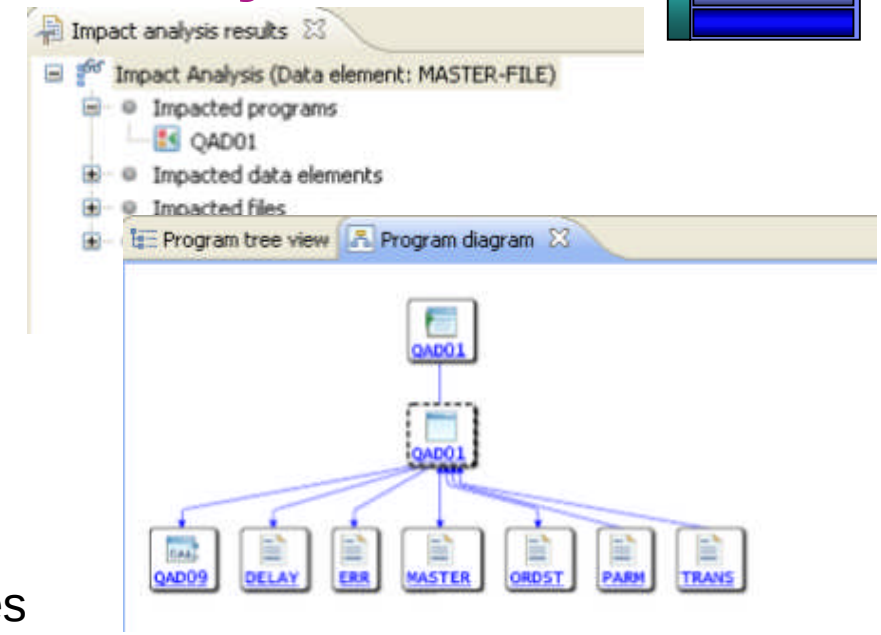


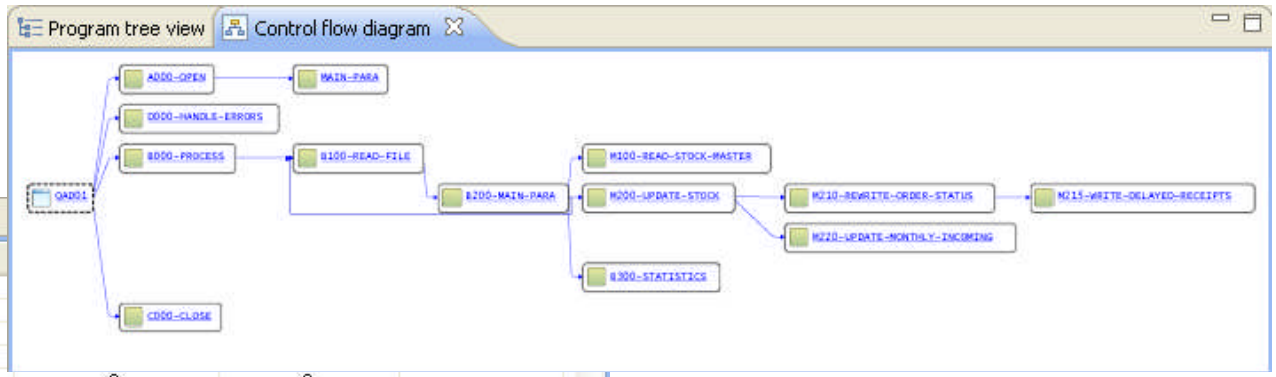
Steps to validate a change

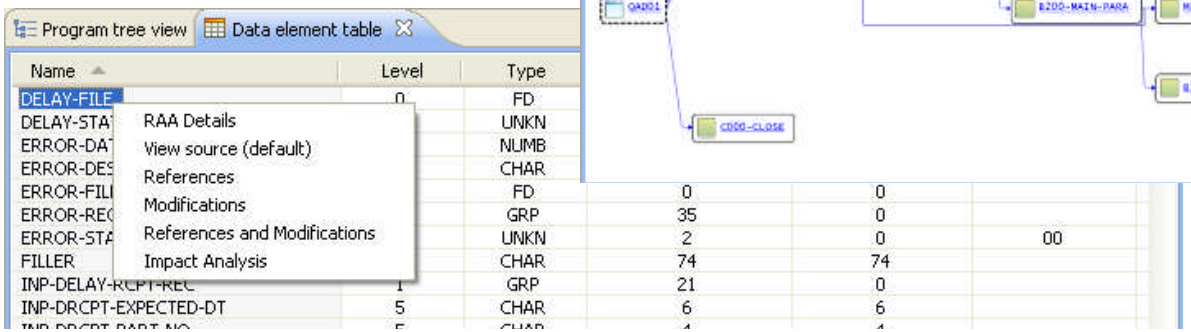
- Edit source
- Find code line
- Change code w/feedback

Analyze applications automatically

- **Integrate application analysis into development**
  - Link code to data and runtime resources
  - Visualize code structure and flow
- **Understand the effect of changes**
  - Run impact analysis on code changes to determine effected production modules
  - Size testing efforts and create workspaces for changes
  - Locate dead code







Create enterprise services…

- Provide standardized access to application assets
- Work across platforms
- Create abstract interfaces for loose coupling
- Vary implementation without affecting consumers
- Ease application reuse
- Improve application testability

IDE Efficiency Benchmarks

- **100 common (daily) ISPF tasks used during maintenance and support assignments**
  - ISPF workflow translated (click-for-click) to RDz development
  - Project participants believed they were trying to find gaps between RDz and ISPF functionality
- **Apples-to-Apples and test scripts**
- **Mix of experienced (veteran) ISPF programmers and new-hire developers**
- **Assumption was that only new-hire developers would be more productive**
  - This turned out to be false

Based on IBM internal productivity study.

All performance data contained in this presentation was obtained in the specific operating environment and under the lab conditions and is presented as an illustration only.

Performance obtained in other operating environments may vary and customers should conduct their own testing.

Analytics	All Participants
Use Case	% Less time to complete tasks with RDz
Source Navigation:	45.00
Program Analysis:	100.90
Primitive Edit Operations:	19.69
COBOL Statement Coding:	35.79
Syntax Check:	57.11
Program Compile/Link Edit:	43.52
DB2 data edit/SQL statement work:	101.30
Total - all use cases:	57.69

Analytics	ISPF Top Guns
Use Case	% Less time to complete tasks with RDz
Source Navigation:	47.21
Program Analysis:	72.60
Primitive Edit Operations:	19.84
COBOL Statement Coding:	36.89
Syntax Check:	68.86
Program Compile/Link Edit:	14.14
DB2 data edit/SQL statement work:	64.68
Total - all use cases:	28.36

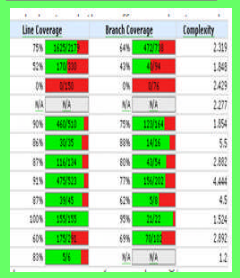
Mainframe Tester Feedback

- **Tester 1 - It takes us 5 to 6 weeks to complete z/OS application testing**
  - Over 1000 test cases to run
  - Manual test effort because there are no z/OS automated test tools
  - No end-to-end testing (one tool that does it all)
- **Tester 2 – We can not respond to regulatory changes**
  - Test cycle takes a minimum of 12 weeks
  - Competition between development teams for testing resources
  - Build and maintain their own test tools. Manual operation.
  - Long batch test runs
- **Tester 3 – Testing applications with multiple teams, components, and runtimes is complex**
  - Have major z/OS application resource constraints that results in long test cycle
  - Off-Shore development and testing requirements
  - Major application failed due to no end-to-end functional and performance testing

Testing and Delivery – where are customers today?

Java / .Net teams goal state

Compile → Automated Unit Test → Install & Configure → Run Automated Tests → Run Manual Tests → Promote Results



Hours to days

Mainframe teams current state **

Compile → Promote & Configure → Run Manual Tests → Promote Results

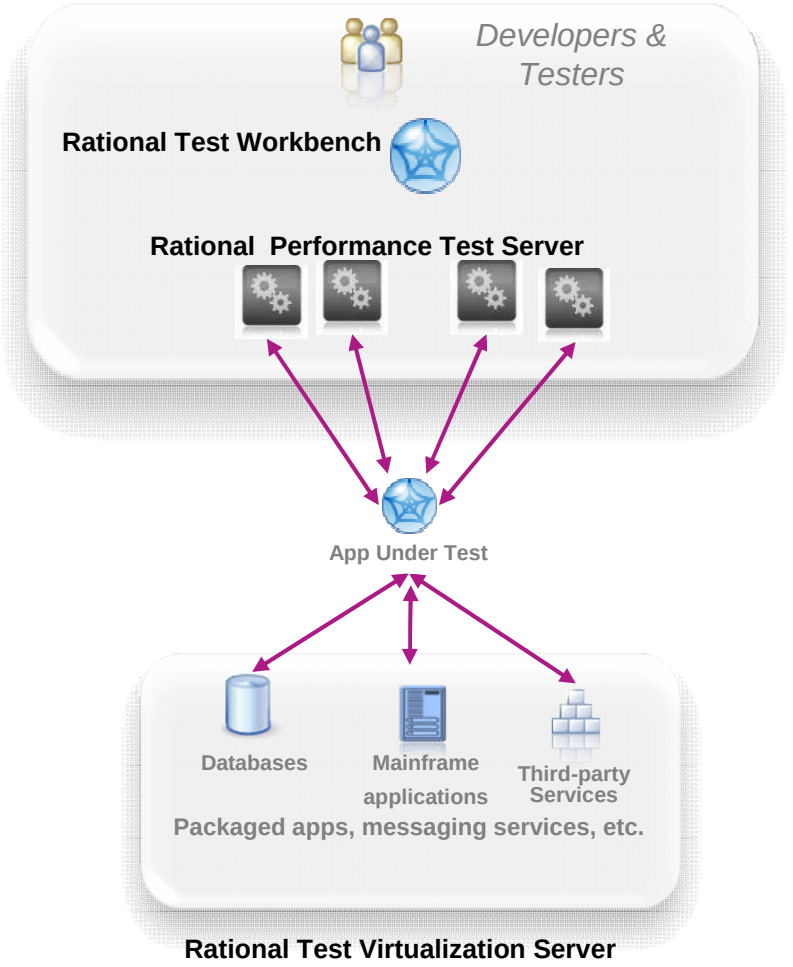
Weeks to Months

** Feedback from mainframe customers

IBM Rational Test Solutions for System z

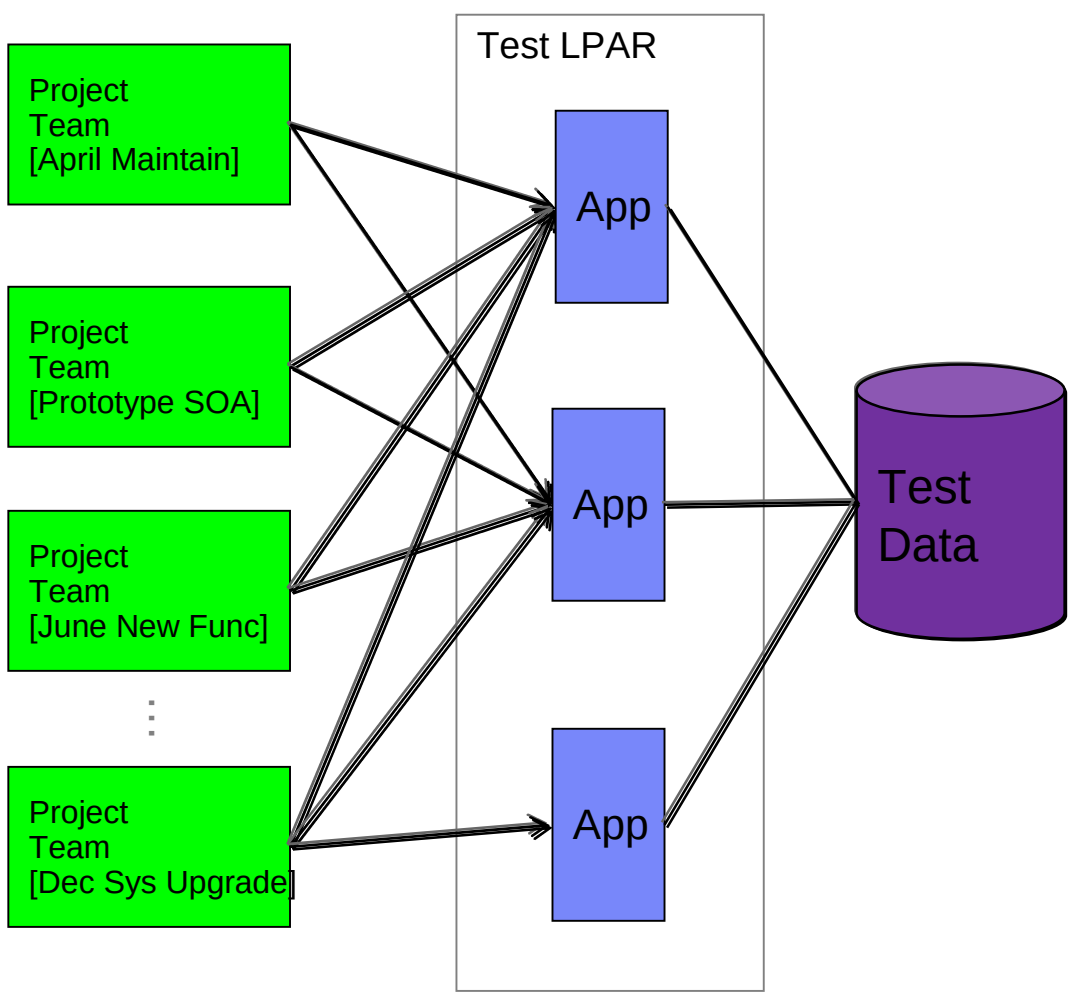
A smarter solution to better quality

- **Rational Test Workbench** is a desktop solution that enables testers/developers to**:**
  - Capture and model virtual services
  - Test services and applications long before their user interfaces becomes available and do integration testing (SOA, BPM)
- **Rational Test Virtualization Server** is a server solution that**:**
  - Provides a central environment to virtualize heterogeneous hardware, software and services to provide 24x7 testing capabilities
  - Reduces infrastructure costs of traditional testing environments
  - Virtual Services can be built from the interface definition of the system for a wide variety of protocols, including HTTP, web services, SOA, JMS, TIBCO, IBM WebSphere MQ, CICS Transaction Gateway, IMS Connect, Oracle, etc.
- **Rational Performance Test Server** enables Rational Test Workbench users to reuse test scripts to drive performance testing
  - Can be used in combination with Virtual Services
  - Probe for identification of system bottlenecks
- **Rational Development and Test Environment for System z** enables provisioning of System z test environments on x86 hardware
  - Enables isolated testing of mainframe-centric applications
  - Provides low cost System z environments for early cycle testing
  - Lowers development MIPS requirements on mainframe hardware



Typical z/OS Testing Architecture

Organized by project team, vertically scaled, sharing resources, limited automation



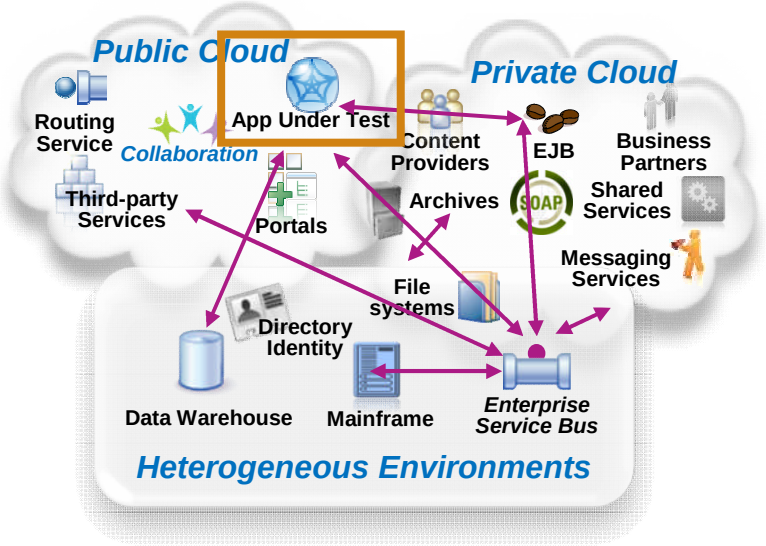
Problems Encountered

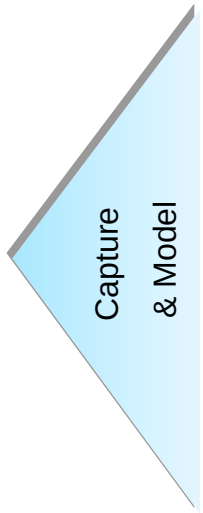
1. Shared resources combined with overlapping schedules can elicit conflicts, impede innovation and slow code delivery

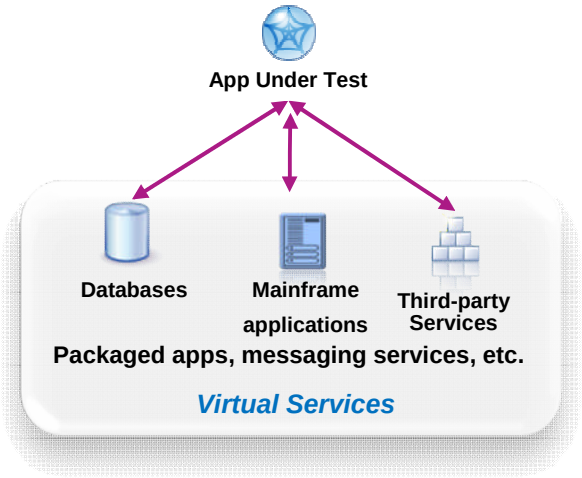
2. Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead

3. Shared test data is difficult to manage and can lead to over testing or incorrect test results

What is Test Virtualization?







System *dependencies* are a key challenge in setting up test environments:

- ***Unavailable/inaccessible***: Testing is constrained due to production schedules, security restrictions, contention between teams, or because they are still under development
- ***Costly 3rd party access fees***: Developing or testing against Cloud-based or other shared services can result in costly usage fees
- ***Impractical hardware-based virtualization***: Systems are either too difficult (mainframes) or remote (third-party services) to replicate via traditional hardware-based virtualization approaches

Test Virtualization enables to create "*virtual services*":

- *Virtual Services simulate the behavior of an entire application or system during testing*
- *Virtual Services can run on commodity hardware, private cloud, public cloud*
- *Each developer, tester can easily have their own test environment*
- *Developer and testers continue to use their testing tools (Manual, Web performance, UI test automation)*

Supported Environments and Technologies

Messaging Protocols

- ActiveMQ
- CICS Transaction Gateway
- Email (SMTP, IMAP)
- Files
- FTP/S
- HTTP/S
  JMS (JBOSS et al)
  IBM WebSphere MQ
- IMS Connect
- JBoss MQ
- SAP IDoc, BAPI, RFC & XI/PI
- Software AG's IB & IS
- Solace
- Sonic MQ
- TCP
  TIBCO Rendezvous, Smart Sockets & EMS
- Custom

SOA, ESB, Others

- CentraSite
- Oracle Fusion
- SCA Domain
- Software AG IS, BPMS
- Sonic ESB
- TIBCO ActiveMatrix
- UDDI
- Web Services
- WebSphere RR
- WSDL

- BPM
- Databases
- Log Files

Message Formats

- .Net Objects
- Bytes
- COBOL Copybook
- ebXML
- EDI
- Fixed Width
- HL7
- IATA
- Java Objects
- MIME
- OAG
- SOAP
- Software AG Broker Docs
- SWIFT
- TIBCO ActiveEnterprise
- XML (DTD, XSD, WSDL)
- Custom

Note : Custom protocol support can be developed

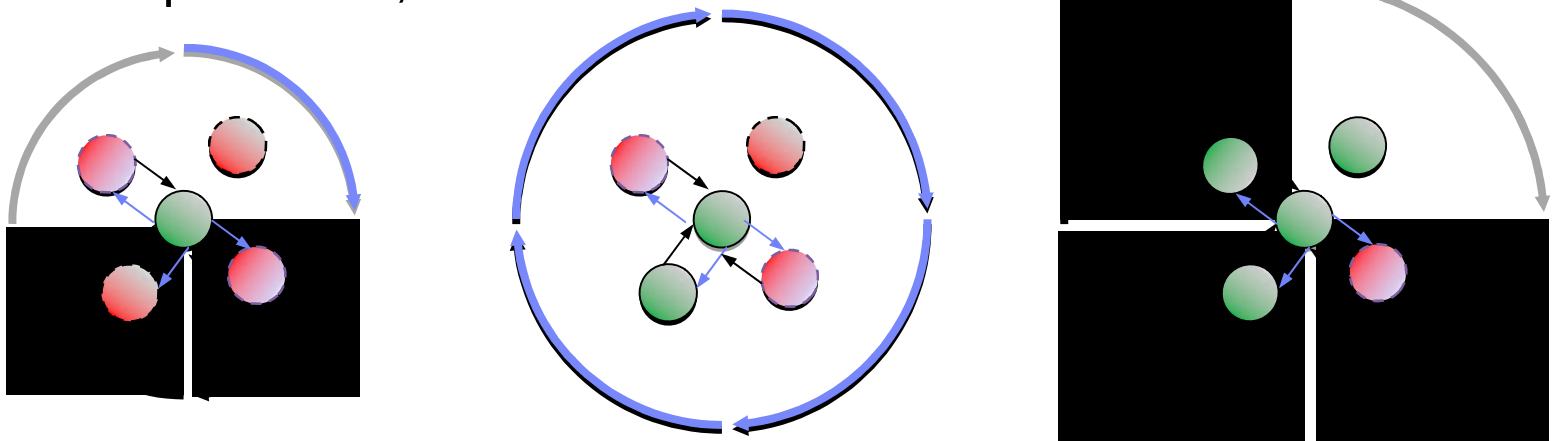
IBM Rational Test Virtualization Solution is a key enabler for Continuous Integration Testing

✓Test Virtualization is an enabler for continuous Integration Testing

Actual Service/App

Virtual Service/App

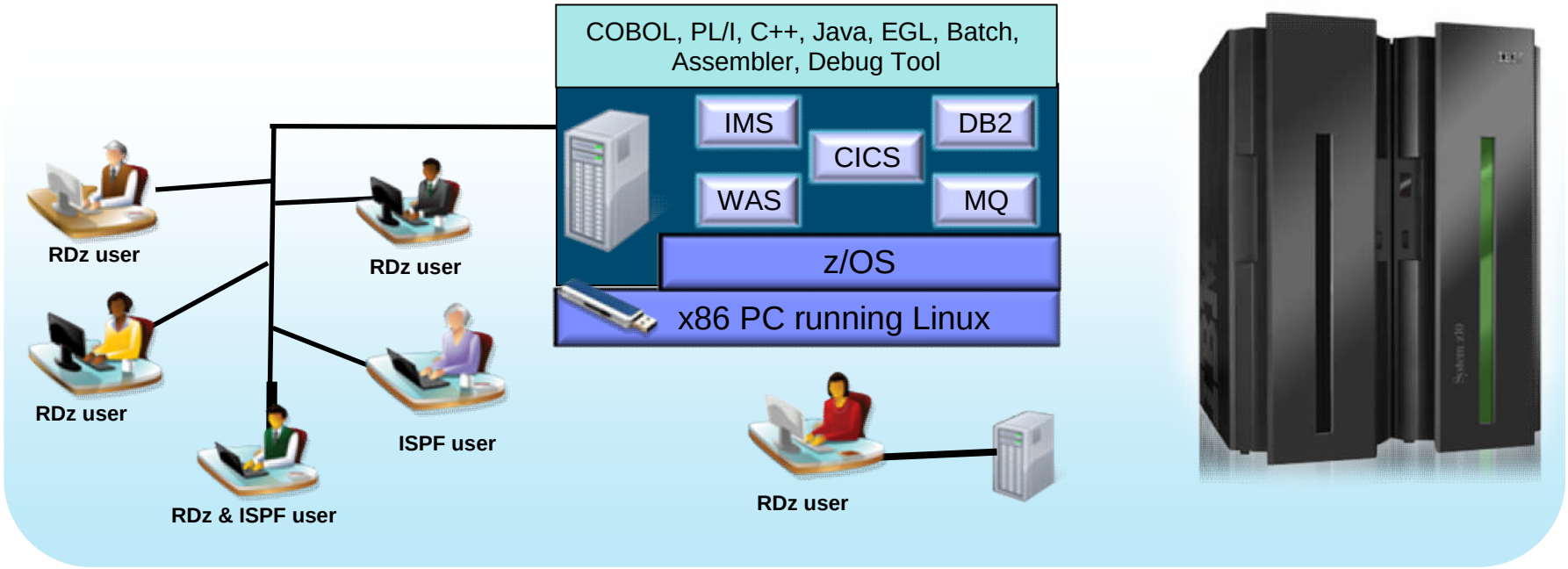
✓Services, applications, systems are introduced into the continuous integration cycle in a prioritized, controlled fashion.



Incremental Integration Testing

Rational Development and Test Environment for System z

The ultimate in modern application development for System z

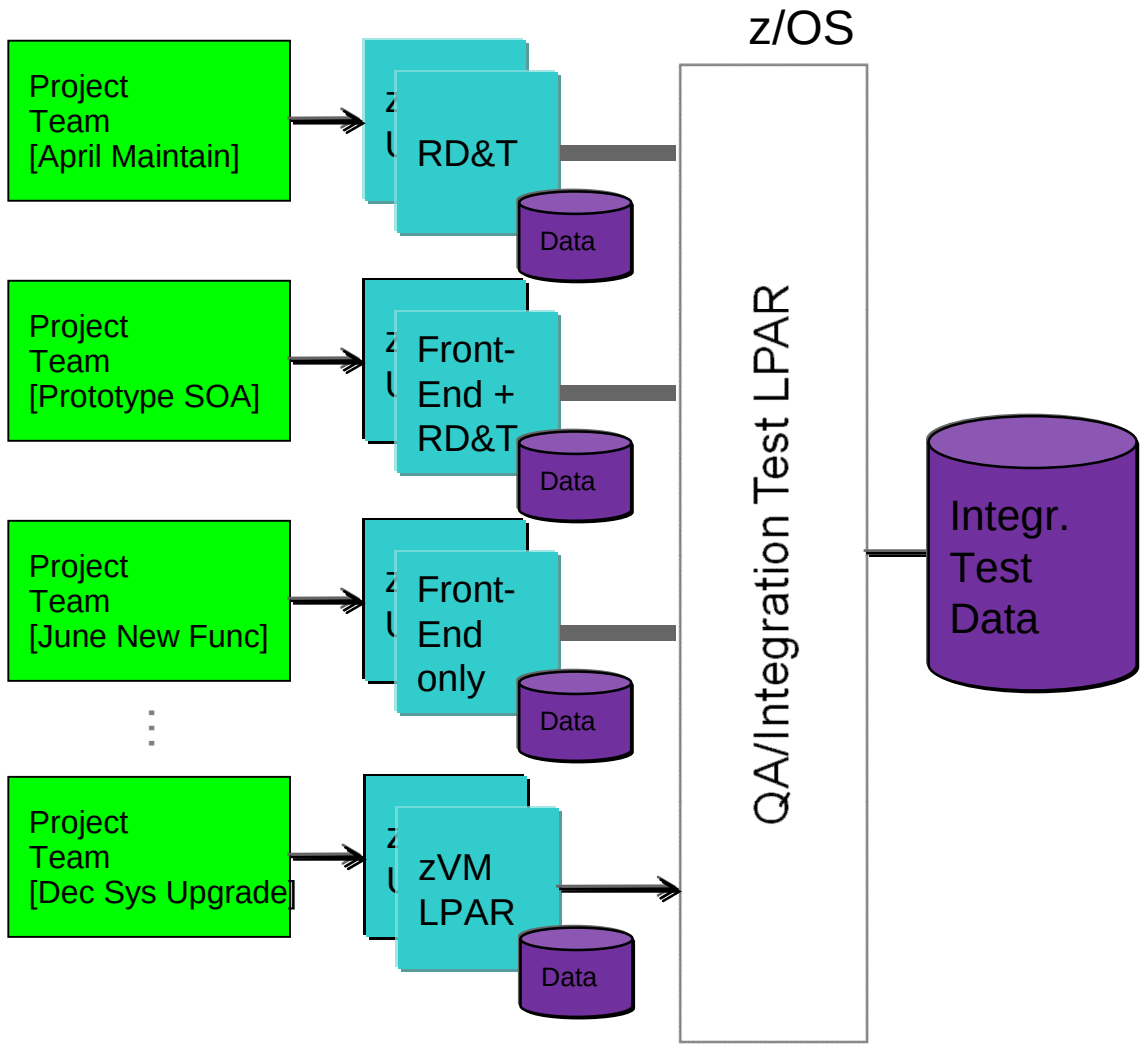


- Liberate developers to rapidly prototype new applications
- Develop and test System z applications anywhere, anytime!
- Free up mainframe development MIPS for production capacity
- Eliminate costly delays by reducing dependencies on operations staff

Note: This Program is licensed only for development , test, and internal training 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.

Testing Organized for Flexibility and Quick Delivery

Organized by application team, horizontally sliced, dedicated resources, highly automated



Problems Encountered

1. Shared resources combi… ov…lapp…g sched…es ca…elicit co…nts impe…n and slow code delivery
2. Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead
3. Share… is dif…cult to …nage a…d can l…d to ov…ting or incorrect test results
4. Provisioning, managing, and synchronizing project test environments including data

IBM Rational Test Solutions for System z
A Smarter Solution for Better Quality

Significantly Lesser Test Lab costs

- Test lab infrastructure **costs can be reduced by up to 90%**
- **Labor involved** in setting up test environments can be **reduced by 80%+**
- **Reduced or eliminated the cost of invoking 3rd party systems** for non-production use, fee-based web services

Reduced Cycle Time

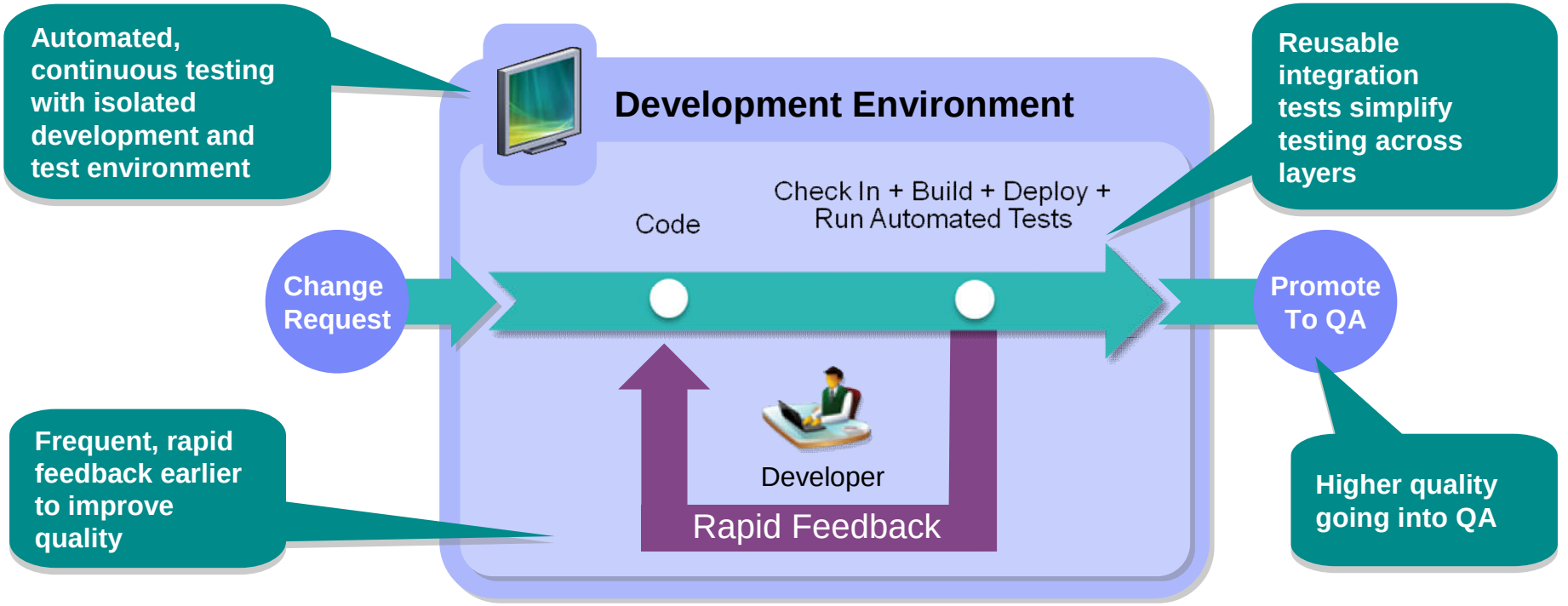
- Test environments can be **configured in minutes vs weeks**
- More testers can be focused on testing, rather than configuring test environments
- **More regression testing can be done** independently from the User Interface, during development

Lower Risk

- Developers have the means to **test software earlier** at the Service/API level
- Large teams working on different parts of an application or system can effectively **do parallel development by virtualizing** different parts of the system

Continuous Integration

Reduced delivery time, end-to-end visibility of test activities, safer and faster upgrades (V2V)



- Fast, dependable, automatic feedback speeds time to market
- Lower cost of application testing using off-mainframe z/OS test environment
- Enables confidence by automatically tracking and promoting code health

What is Continuous Integration

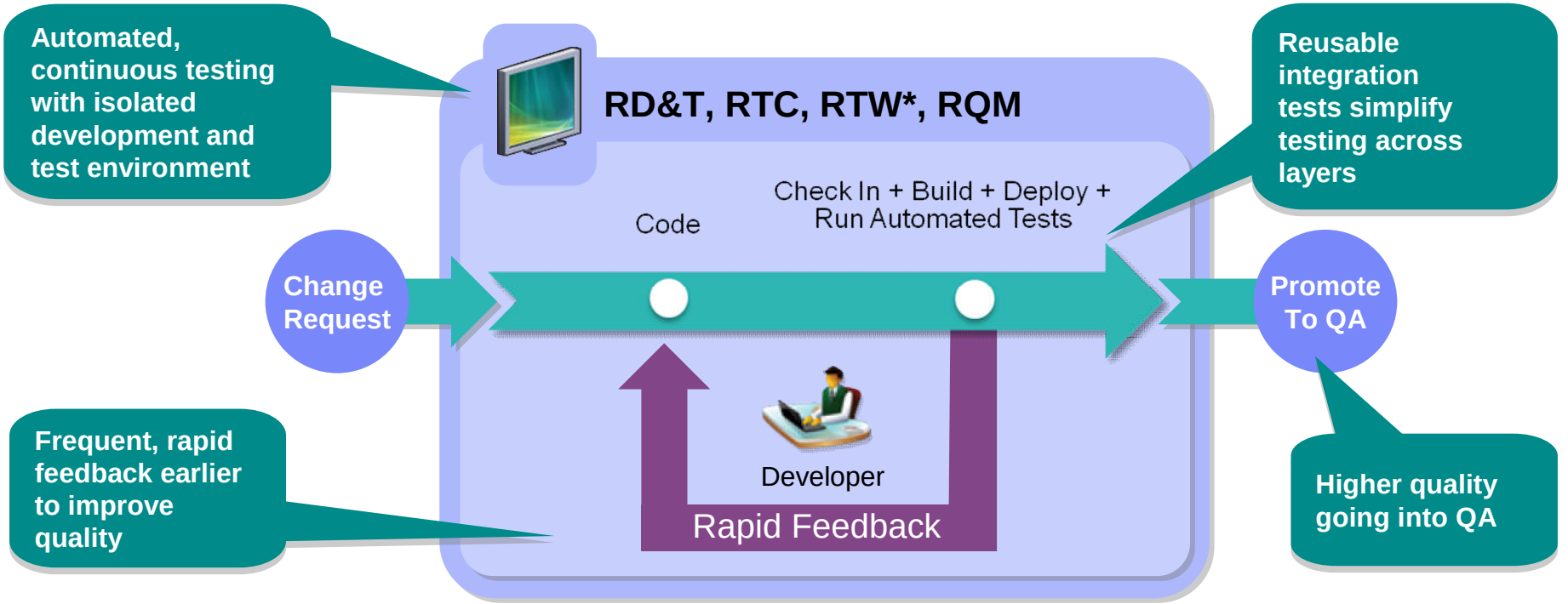
Expedite feedback to developers on application quality

Principles of Continuous Integration

- ☑ **Maintain a code repository**
- ☑ **Automate the build**
- ☒ **Make the build self-testing**
- ? **Commit to the baseline every day**
- ? **Every commit should be built**
- ? Keep the build fast
- ☑ Test in a clone of production
- ? Make it easy to get latest deliverables
- ? Everyone can see the latest build results
- ☑ Automate deployment

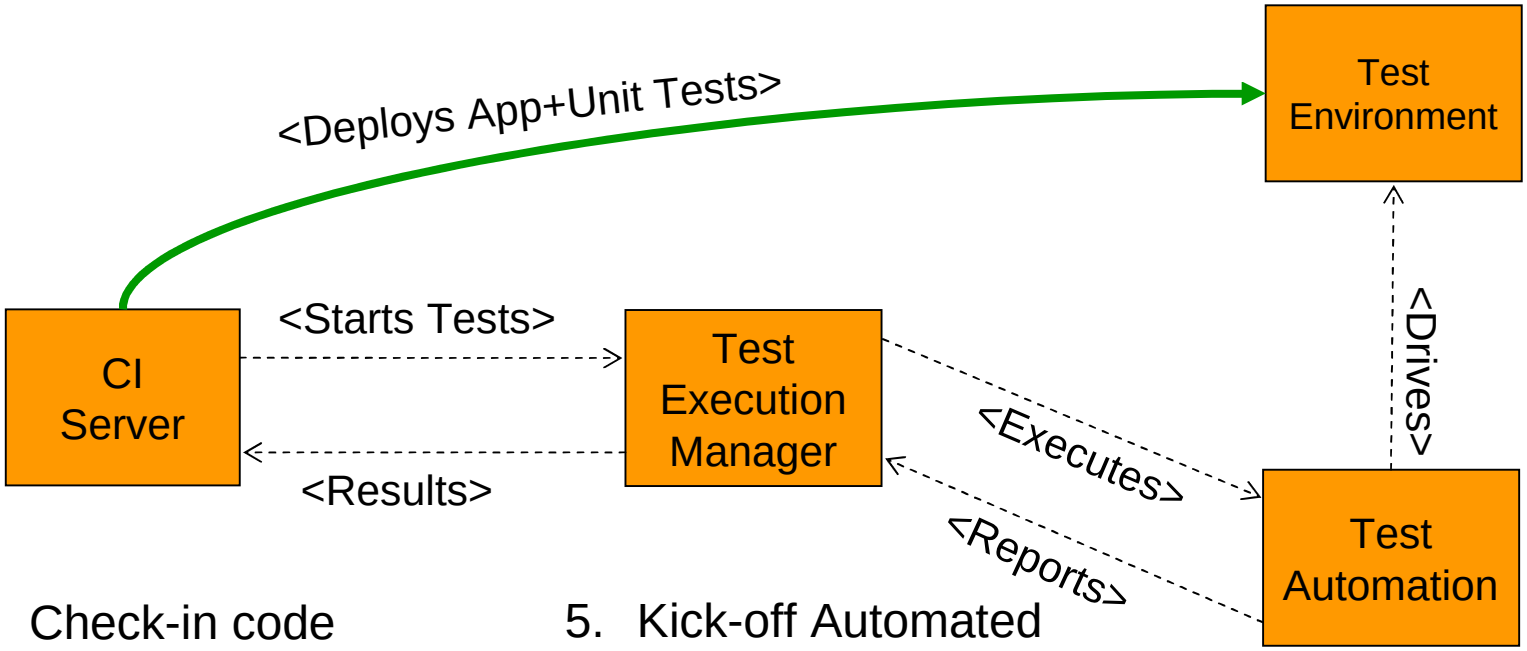
IBM Continuous Integration Solution for System z

Reduced delivery time, end-to-end visibility of test activities, safer and faster V2V migrations



- Rational Team Concert 4.0
- Rational Quality Manager 4.0
- Rational Development and Test Environment for System z 8.5 NEW!
- Rational Testing Workbench powered by Green Hat Technology

Detailed Continuous Integration Scenario



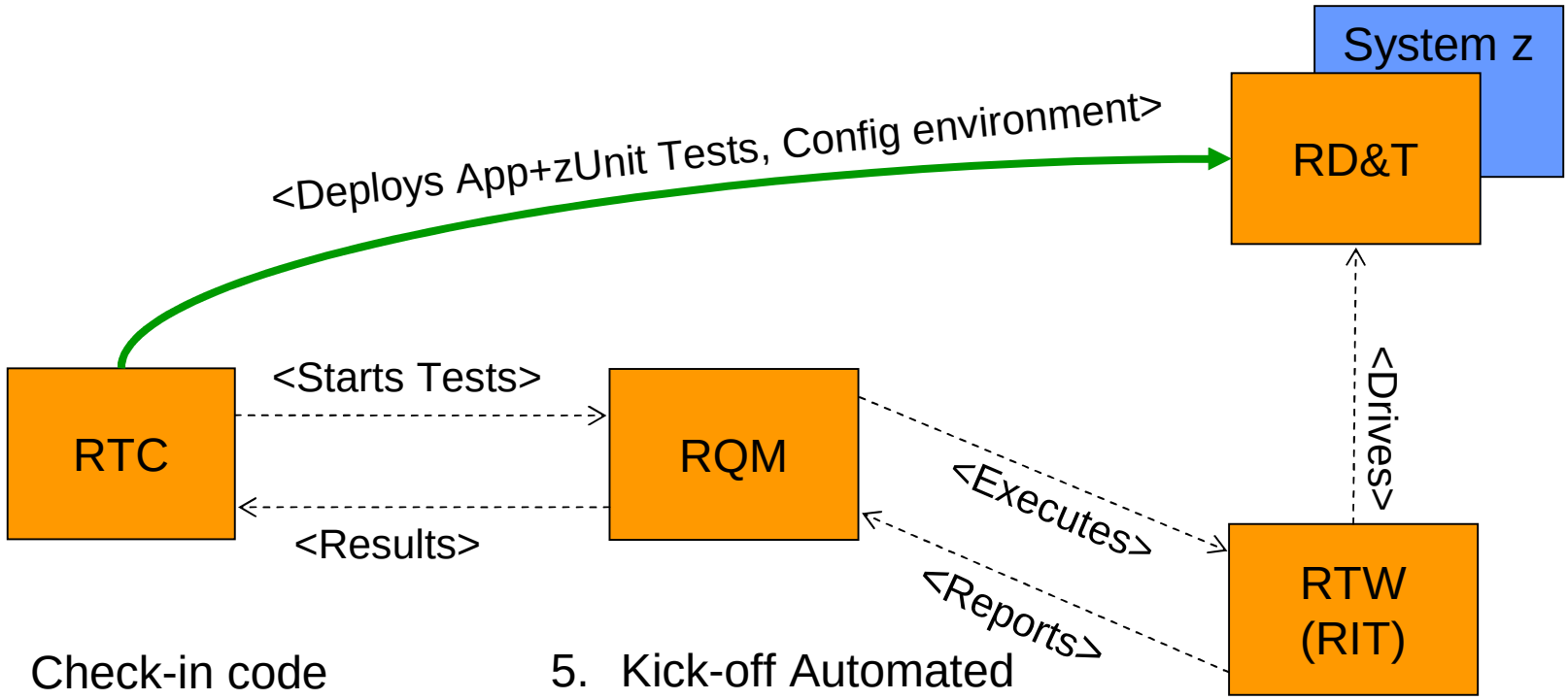
1. Check-in code
2. Build code and Unit tests
3. Deploy build results to Test Environment
4. Execute Unit Tests

5. Kick-off Automated Test Plan

8. Report test results in dashboard/build results/defect records in CI server.

6. Run automated interface tests against Test Environment
7. Mark execution records Pass/Fail in Test Execution Manager

Detailed Continuous Integration for System z Scenario

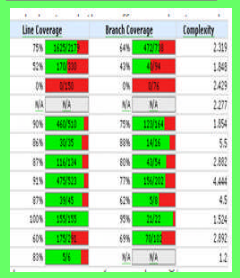


1. Check-in code
2. Build code *and zUnit tests*
3. Deploy build results *and test data* to RD&T
4. *Execute zUnit Tests*
5. Kick-off Automated Test Plan
6. Run automated interface tests against RD&T or System z
7. Mark RQM execution records Pass/Fail
8. Report test results in dashboard/build results/defect records in RTC.

Testing and Delivery – moving one step forward

Java / .Net teams goal state

Compile → Automated Unit Test → Install & Configure → Run Automated Tests → Run Manual Tests → Promote Results



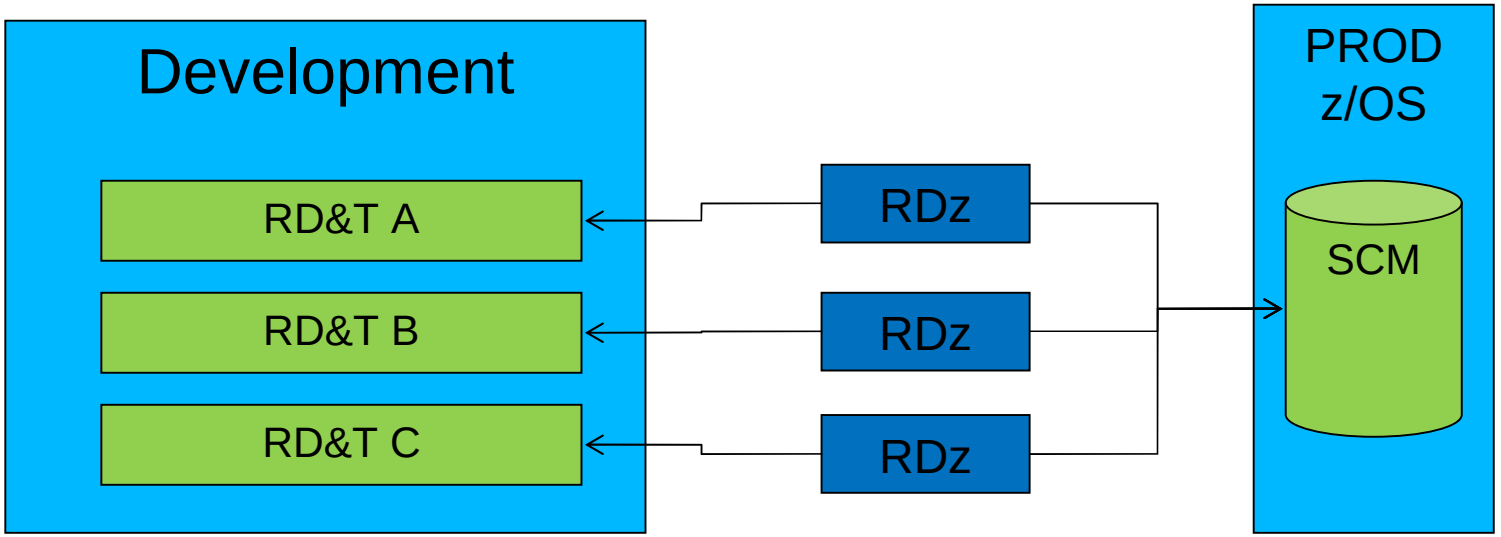
Hours to days

Mainframe teams after CIz

Compile → Promote & Configure → Run Automated Tests → Run Manual Tests → Promote Results

Days to Weeks

Customer example

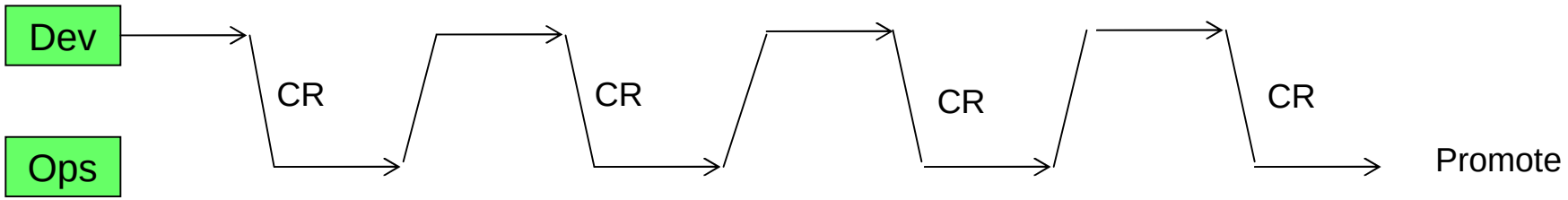


Large US Financial Customer - Implementation

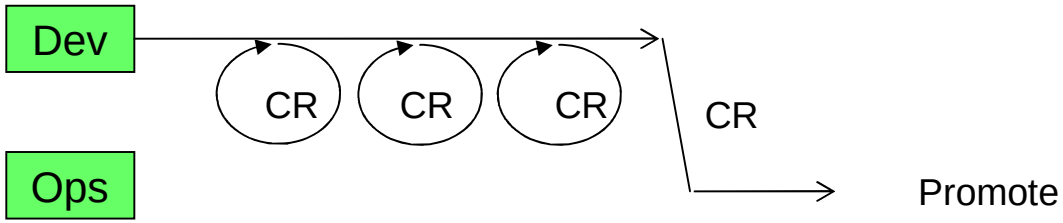
- Requirement
  - Provide more responsive System z access for application developers
  - Reduce application software delivery through faster test cycle
  - Provide each application group their own unit test facility
- RD&T Solution
  - Application developers have use of a uniquely defined RD&T feature for System z access
  - RD&T provides a unique test environment for different application development groups
  - Supports multiple RD&T development environments on Intel blade to reduce server hardware
  - Has reduced application development test time

Customer story

Mainframe process today (2 months)



CIz process using RD&T (5 days)



Project cycle time reduced from 2 months to 1 week

"Normally it can take up to 5 days for the mainframe staff to process an request to make a change to CICS. If a project is trying to get something to work, it may take many change requests and several weeks to resolve a problem. However with CICS on RD&T, the project architects or developers can try the changes themselves in real-time until they get the configuration correct. Then an change request can be submitted with correct configuration parameters to the systems people to implement on the mainframe. This saved the development team weeks of delivery time!"

Compilers for zEnterprise

Improve application performance, productivity, and return on investment

IBM zEnterprise 196

- z/OS XL C/C++
- Enterprise PL/I for z/OS
- Enterprise COBOL for z/OS

zEnterprise BladeCenter Extension (zBX)

- XL C/C++ for AIX and Linux
- XL Fortran for AIX and Linux
- COBOL for AIX
- PL/I for AIX

Delivering…

- Performance improvements for applications with enhanced compiler optimization technology
- Exploitation of new zEnterprise 196 and POWER7 hardware
- Productivity improvements with enhanced reporting, problem determination, performance tuning and code portability support

Compiler Business Values

- Increase return on investment
  - Maximize application performance on System z
    - *Exploit z/Architecture and middleware*
    - *Leverage advanced optimization technology*
    - *Reduce total cost of ownership*
- Improve programmer productivity
  - Simplify programming
  - Improve usability
  - Reduce risk, cost, and development time
- Protect investment in business critical applications
  - Modernize business critical applications
    - *Reduce risk, reduce cost*
  - Maintain release-to-release compatibility
  - Support Industry programming language standards and extensions

IBM z/OS XL C/C++

- **Optionally priced feature of z/OS**
  - – Enables development of high performing business applications, system programs and low level C applications
- **IBM has been delivering leading edge C/C++ compilers on z/OS for over 20 years**
  - – Every release sets new standard for performance
  - – Includes advanced optimization technology originally designed for HPC applications, and innovations to improve programmer productivity
- **Provides system programming capabilities with Metal C option**
  - – Allows developers to use C syntax to develop system programs and low level free standing applications on z/OS without coding in HLASM
    - Significantly shortens the learning curve
    - Leverage advanced optimization technology to generate high performance optimized code

What's in z/OS v1.13 XL C/C++?

- **Ships with z/OS v1.13**
- **Improved exploitation of zEnterprise 196 processor**
  - – Improve support for new instructions with new New ARCH(9) functions
- **Improved application performance[1]**
  - – 4% over v1.12 for a compute intensive integer benchmark suite
  - – 7% over v1.12 for a compute intensive floating point benchmark suite
- **Added language support to enable straightforward porting of C/C++ applications to z/OS**
- **"Metal C" functional and performance enhancements**
  - – Enabled advanced optimization with IPA and HOT options
- **Improved debugging and programmability support**

[1]Results are based on a compute-intensive integer and floating point benchmark suites compiled with z/OS C/C++ V1R13 executing on a System zEnterprise 196 server. Performance gains from other applications may vary

z/OS XL C/C++ v1.13 support for zEC12

- *Next release of z/OS XL C/C++ will be aligned with z/OS schedule*
  - Follow a new two-year release cycle announced in the April 2012 z/OS SOD

- *z/OS XL C/C++ V1.13 will provide initial support for zEC12 in the Sept. 2012 PTF*

C/C++ compiler	UK80670 and UK80671
C/C++ readme	UK80039
C++RT builtins.h	UK79899

  - New “Arch(10)” and “Tune(10)” options
  - Enable developers to exploit new Transactional Execution Facility via built-in functions
  - PTF Web site: http://www-01.ibm.com/support/docview.wss?uid=swg21108506

- *Performance Improvements of up 23% for CPU intensive applications on zEC12*
  - From new hardware, No recompilation required

Rocket Software, Inc. increases development efficiency on the IBM System z platform

The Metal C feature of the IBM z/OS XL C/C++ compiler makes it easier to leverage its C programming skills

The need:

Rocket Software, Inc. wanted to increase efficiency and improve time to market for its IBM® System z® operating system-based software products.

The solution:

Rocket Software used the Metal C feature of the IBM z/OS® XL C/C++ compiler to develop high performance system level programs. With the Metal C feature, programmers can write code in the C syntax while taking advantage of advanced optimization technology in the z/OS XL C/C++ compiler, resulting in high-performance code that works seamlessly with code written in IBM High-Level Assembler language (HLASM).

The benefits:

- Significantly increased development efficiency
- Reduced development time by half
- Enabled the company to leverage C programming skills

"Metal C in z/OS XL C/C++ is yet another powerful tool helping turn the economics of System z software development into a complete equation."

—Joseph Devlin, managing director, R&D, Rocket Software

Solution components:

- IBM® z/OS® XL C/C++
- IBM® System z®

Rocket

Enterprise PL/I

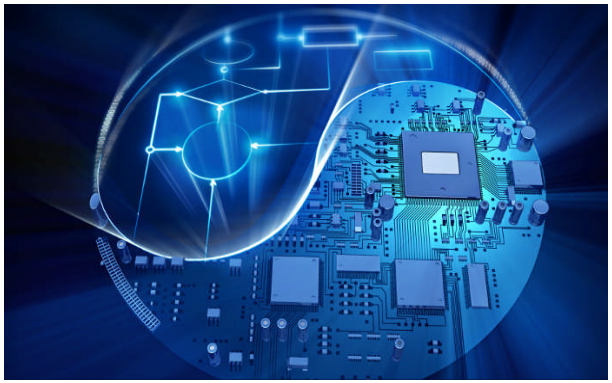
- **Strategic Programming Language**
  - Significant use in business applications but also in some scientific and engineering applications
  - Introduced new version (v4) in 2010
- **Advanced optimization technology**
  - Shares optimizing back-end technology with z/OS XL C/C++
    - Enables timely delivery of leading edge optimization and hardware exploitation to PL/I customers
- **Time proven**
  - First Enterprise PL/I product released in 2001 (Enterprise PL/I for z/OS and OS/390 v3.1)
  - Latest release of Enterprise PL/I for z/OS (v4.3) is based on same architecture
    - Provides easy migration
- **Shipped new release every year since 1999**
  - Improved optimization technology, z/Architecture exploitation, usability, middleware support, and application modernization features.
  - Addressed customer requirements

What's new in Enterprise PL/I for z/OS v4.3?

- **Improved performance**
  - – zEC12 exploitation with new ARCH(10) option
    - Exploits new Decimal-Floating-Point Zoned-Conversion Facility
      - – Up to 40% faster for PICTURE to FIXED BIN conversions
      - – Up to 4X faster for PICTURE to FLOAT DEC conversions
    - CPU-Intensive PL/I benchmarks running on zEC12 see an improvement of up to 31% over zE196

- **Enhanced middleware support**
  - – Support latest middleware CICS, DB2 and IMS
  - – SQL improvements
    - Support for ONEPASS option
    - Improved display of EXEC SQL statements in listings
    - Usability improvements

- **Increased productivity (Requirements from users)**
  - – New built-in functions for better UTF-8 support
  - – New Assert statement
  - – New options for compiler messages and enforcing coding rules.



COBOL Overview

- **COBOL celebrated its 50th birthday in September 2009.**
- **COBOL is still a dominant programming language for processing critical business transactions around the world.**
- **COBOL programs are simple, readable and very maintainable.**
- **Today, most business transactions are still processed with COBOL on IBM System z servers.**
  - There are 200 times more COBOL transactions per day than Google Searches worldwide[1]
- **COBOL is strategic**
  - IBM is investing in improving the underlying technology to bring more value to customers.

[1] eWeek.com: 20 Things You Might Not Know About COBOL

Enterprise COBOL for z/OS v4.2

GA Sept. 2009

- **Validated on zEnterprise 196 and zEnterprise EC12 servers with IBM's latest middleware**
  - – Support latest CICS, IMS, and DB2
- **Provides significant improvements to UNICODE performance**
- **Enables the integration of existing applications with web applications**
  - – Supports Java interoperability by object-oriented COBOL syntax
  - – Supports access to enterprise beans that run on WebSphere Application Server or J2EE-compliant EJB server
  - – Supports Java 5, Java 6 SDK
- **Built-in language support for high speed parsing and validating of XML documents:**
  - – Offloading of XML parsing to zAAP specialty processors
  - – Encoding in UTF-8, UTF-16, and various EBCDIC codepages
- **Improved Debug Tool support for dynamically debugging optimized production programs**

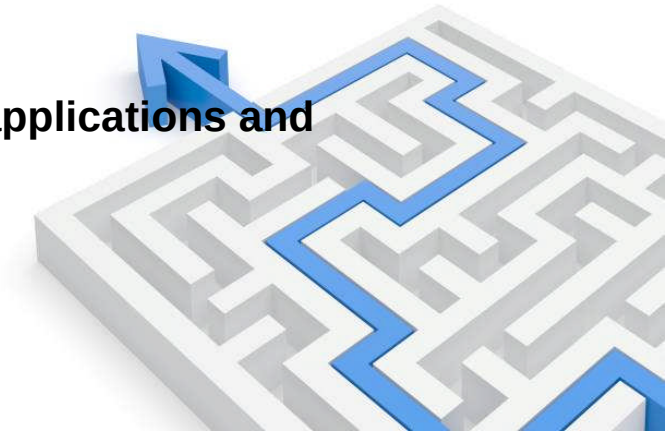
COBOL VNext

Intentions for this release:

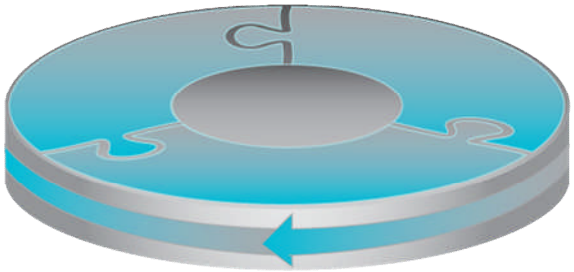
- **Incorporate leading-edge optimization and code-generation technology**
  - Improve delivery of z/Architecture exploitation
  - Maximize hardware utilization
  - Improve application performance

- **Establish solid foundation to support future z processors**
  - Advanced hardware features
  - Exploit 64 bits architecture

- **Improve capabilities for modernizing business critical applications and creating new applications**
  - XML, Java interoperability, UTF-8 Unicode,
  - Usability and problem determination



Compatibility



Our intent:

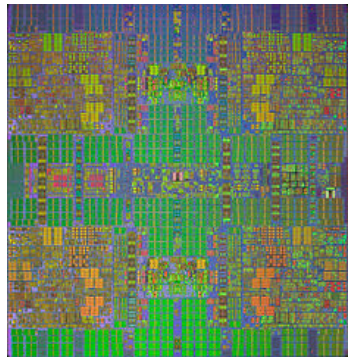
- Provide Source and binary compatibility
  - No need to recompile entire application
    - "Old" and "new" code can be mixed within an application, and communicate with static or dynamic call
  - Most correct COBOL programs will compile without changes
    - Correct programs will run to produce the same results
- Plan to remove some old/obsolete language extensions and options

IBM reserves the right to change strategy and plans at any time.

Capabilities

Our intent:

- New capability to select optimization and processor architecture levels
- Improve capability to process large data items
- Improve capability for modernizing business critical applications
  - *XML enhancements, web-services, Java interoperability*
- Provide support for latest Middleware
  - *CICS, DB2, IMS*
- New debugging interface



IBM reserves the right to change strategy and plans at any time.

COBOL VNext Managed Beta Program

- Objectives
  - Provide early access to ISVs to enable their tools to support the new COBOL compiler
  - Provide early access to *Enterprise COBOL v3 and v4* customers to enable them to preview the new product
  - Validate product acceptability
  - Collect feedback/suggestions on areas of improvement

- Requires active participation
  - COBOL development team will be directly involved
  - Regular calls will be held to discuss progress and exchange technical infor

- 4 Stages (i.e. code drops) planned:
  1. Tools Interface (Started 04/12)
  2. Compatibility (Started 07/12)
  3. New features + Performance
  4. Performance + Quality

ISVs participate in all 4 phases

Customers participate in phases 2 - 4

IBM reserves the right to change strategy and plans at any time.

COBOL VNext Managed Beta Program – more Information

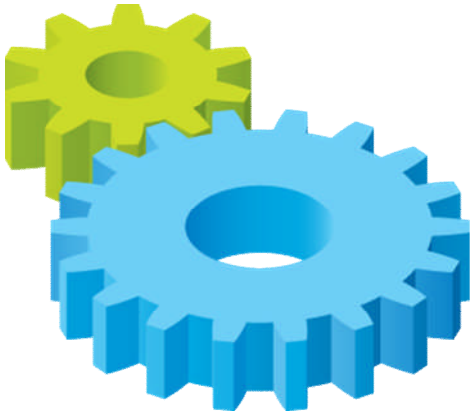
- Submit Nomination Form
  - Self nomination
    https://www-304.ibm.com/software/support/trial/cst/forms/nomination.wss?id=3869
  - Contact IBM rep.

- Program contacts:
  Marie Bradford (mabrad@us.ibm.com)
  Roland Koo (rkoo@ca.ibm.com)

Compilers and middleware

New releases of COBOL, PL/I and C/C++ provide improved support for middleware

- **Integrated CICS and SQL translators**
  - COBOL, PL/I and C/C++
  - Enterprise PL/I v4.2 improved performance of processing SQL source by up to 40%
- **Programming support for new middleware features**
  - CICS co-processor options, DB2 features (e.g. multiple-row insert, multiple-row fetch…)
  - Support for new SQL new data types and SQL syntax first introduced in DB2 v9
- **Problem determination support with program listings and Debug Tool**
  - Display SQL and CICS options in effect in COBOL and PL/I listing
  - Debug COBOL, C/C++, and PL/I applications with CICS, DB2, and IMS
  - Debug optimized COBOL applications in production
- **Java Interoperability**
  - Support Java 5 and Java 6 runtimes
  - Execute COBOL programs in IMS Java region
- **XML Support**
  - COBOL and PL/I programs can send, receive and process XML documents from middleware



Best practices

- **Upgrade compilers when you upgrade System z hardware, or Middleware (CICS, DB2, IMS)**
  - Minimize quality assurance effort
  - Maximize performance
  - Leverage compiler support for new middleware features
  - Improve debugging and programmability
- **Recompile only modified parts of the application**
- **Leverage new compiler features to modernize existing business critical applications**
  - “Rip and Replace is expensive and risky
  - Modernization promotes reuse and delivery of new solutions faster, at lower cost and lower risk,
- **Use Rational development tools to improve programmer productivity, and help attract new talent**
  - Rational Developer for z, Rational Development and Test Environment for System z, Rational Team Concert

www.ibm.com/software/rational