L’intégration continue et la virtualisation des tests sur System z

Francois Dumont

Product Manager, IBM Rational

QUESTION :

QUELLE EST LA PART DES COUTS D'UN PROJET LIEE À LA CORRECTION D'ANOMALIES ?

Cost is a significant driver

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

During the
CODING phase

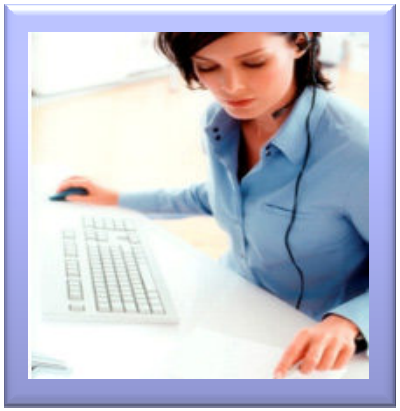
$80/defect

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

*National Institute of Standards & Technology

Source: GBS Industry standard study

Defect cost 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.

QUESTION :

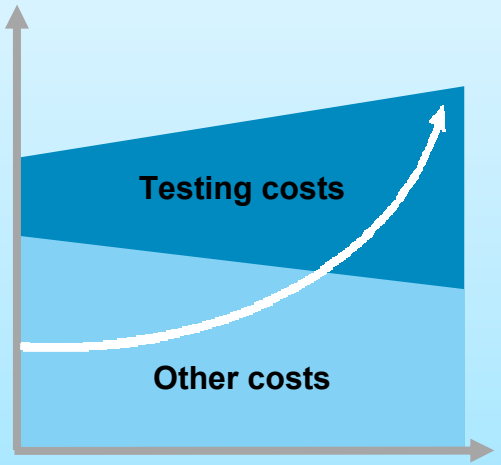
QUELLE EST LA REPARTITION DU TEMPS ENTRE *METTRE EN OEUVRE UN ENVIRONNEMENT DE TEST* ET *TESTER* ?

- 20 % ?
- 40 % ?
- 60 % ?

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

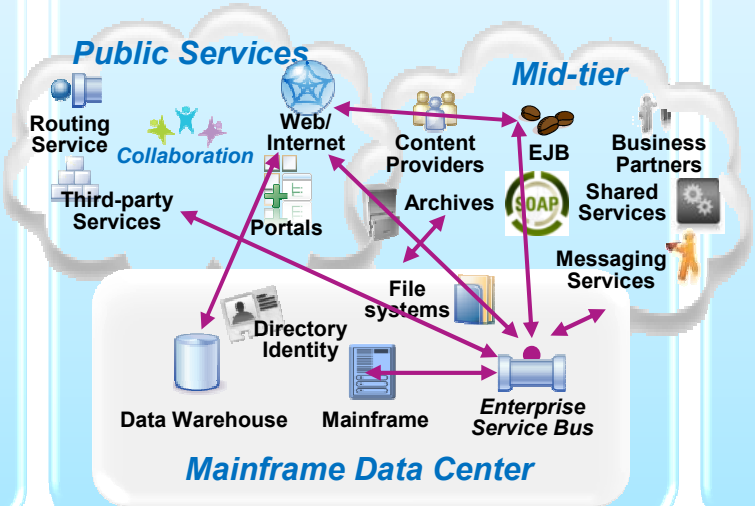


Outsourcing **labor** is no longer a sustainable model as global wages are increasing

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

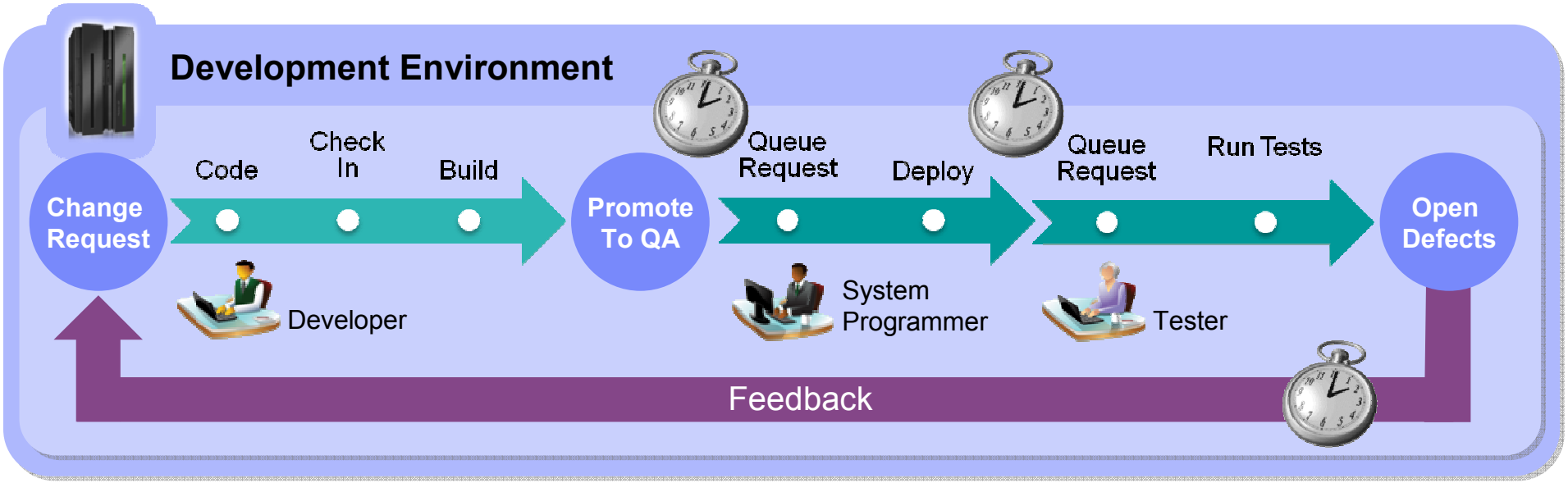
Productivity is inhibited as test teams can no longer keep up with development output

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

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*



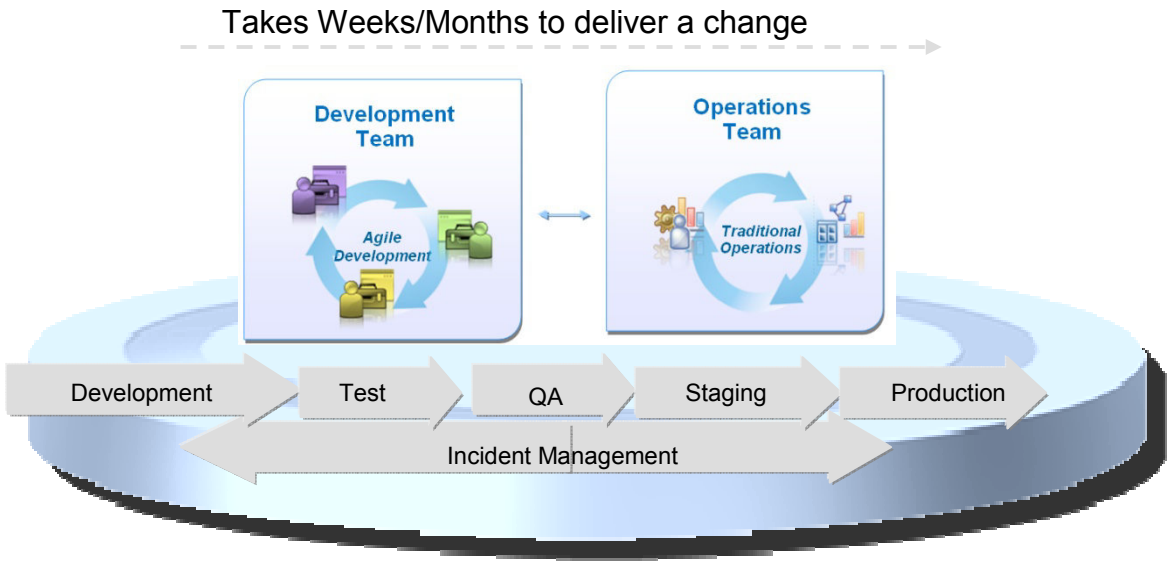
But…

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

Software Delivery Challenges: A Customer Perspective

Needs:

- Reduce cycle time and delays
- Improving software delivery efficiencies with standardization and automation
- Improving Quality of Delivery and reducing roll-backs



Quality Challenges

- Difficulties in reproducing production defects
- Long time to fix defects
- Poor Test coverage
- Lack of automated testing

Release Challenges

- Differences in Dev/Ops environments
- Siloed / Limited automation
- Long set up time

Process and Cultural challenges

- Point-Point, adhoc and Fragile integration of tools
- Poor visibility, stability and extensibility
- Cultural barriers limiting collaboration
- Heavy-handed control of dev environments

Simplified view of a single-release pipeline. In general, there are multiple projects, releases, and technologies at play

INTEGRATION CONTINUE

What is Continuous Integration

Expedite feedback to developers on application quality

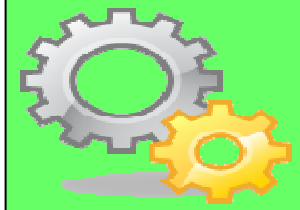
Principles of Continuous Integration

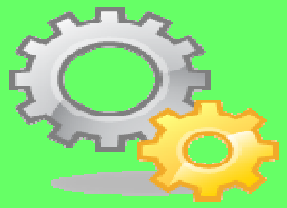
- ☑ **Maintain a code repository**
- ☑ **Automate the build**
- ☒ **Make the build self-testing**
- ☐ **Commit to the baseline every day**
- ☑ **Every commit should be built**
- ☐ Keep the build fast
- ☑ Test in a clone of production
- ☐ Make it easy to get latest deliverables
- ☐ Everyone can see the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

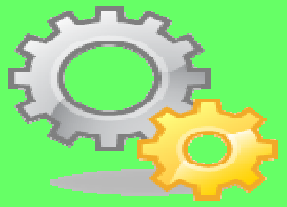
Testing and Delivery – where are customers today?

Agile teams goal state

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



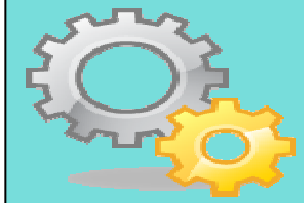


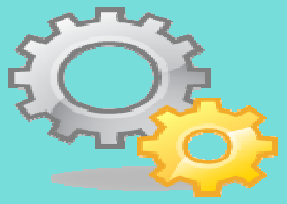


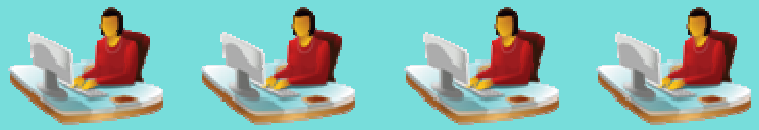
Hours to days

Mainframe teams current state **

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







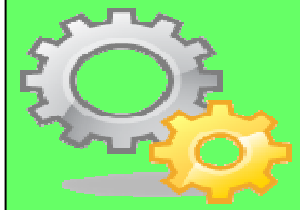
Weeks to Months

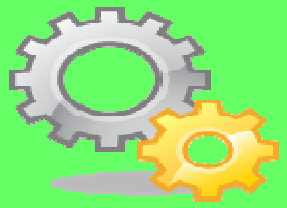
** Feedback from mainframe customers

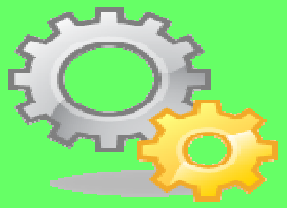
Testing and Delivery – moving one step forward

Java / .Net teams goal state

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



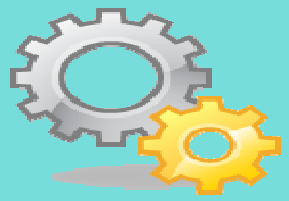


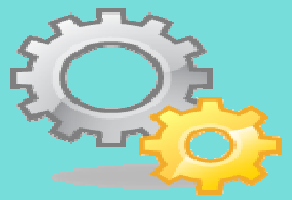


Hours to days

Mainframe teams after CIz

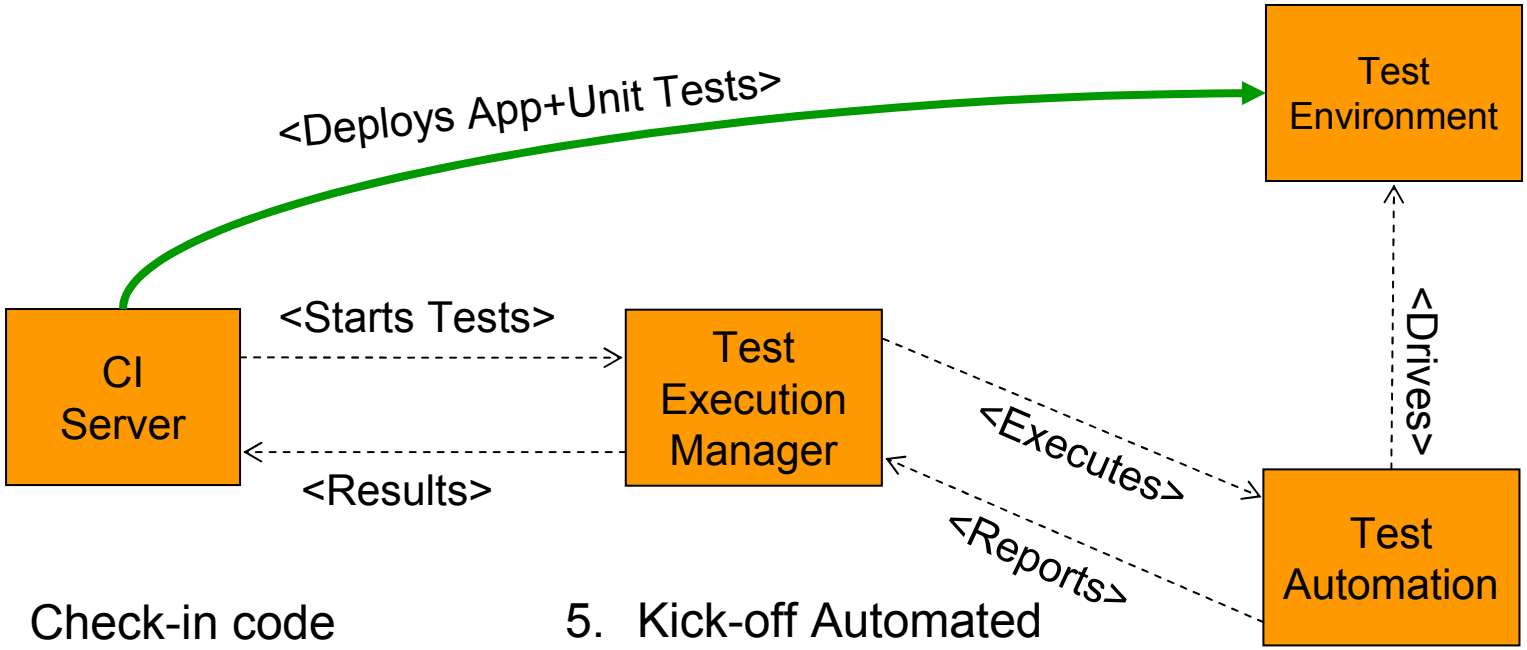
Compile → Promote & Configure → Run Automated Tests → Run Manual Tests → Promote Results





Days to Weeks

Detailed Continuous Integration Scenario



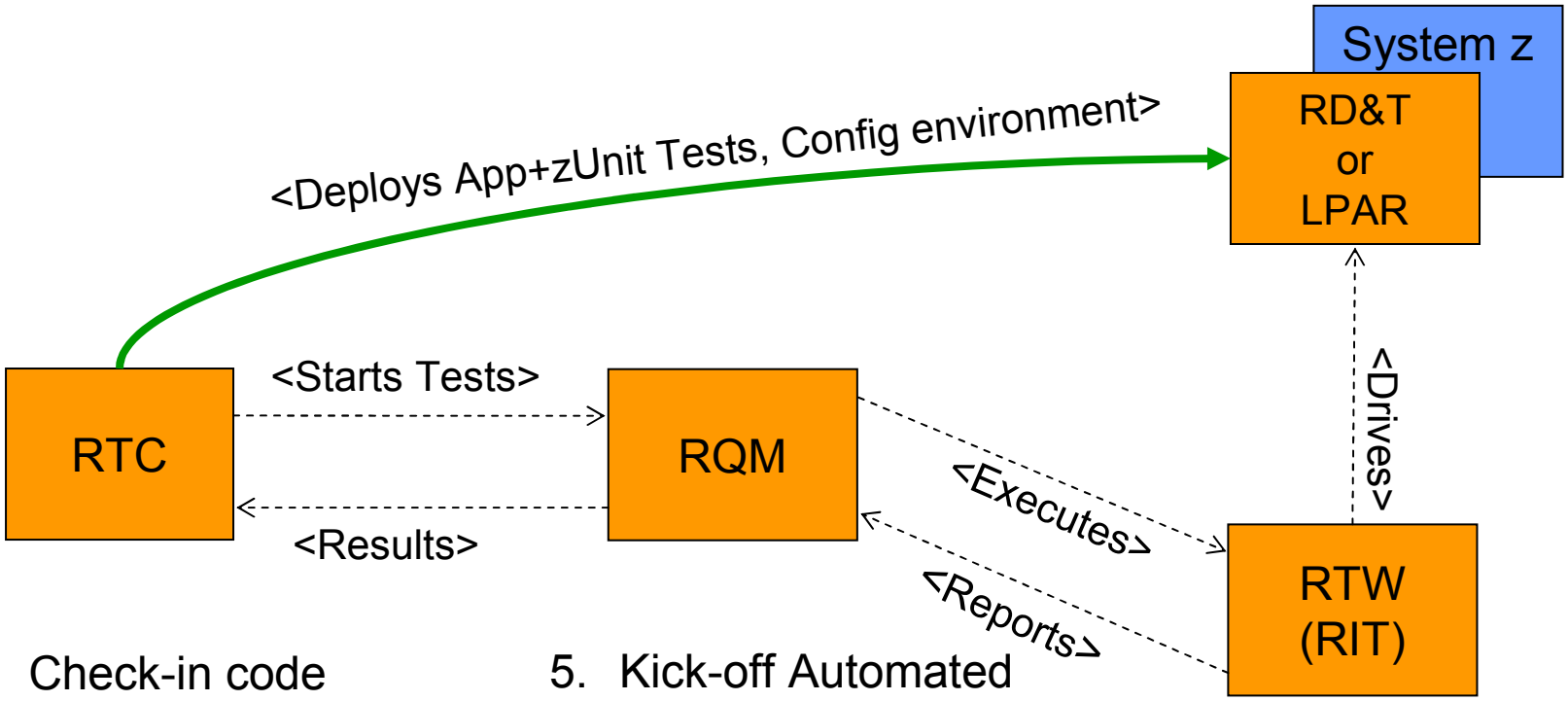
1. Check-in code
2. Build code and Unit tests
3. Deploy build results to Test Environment
4. Execute Unit Tests

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

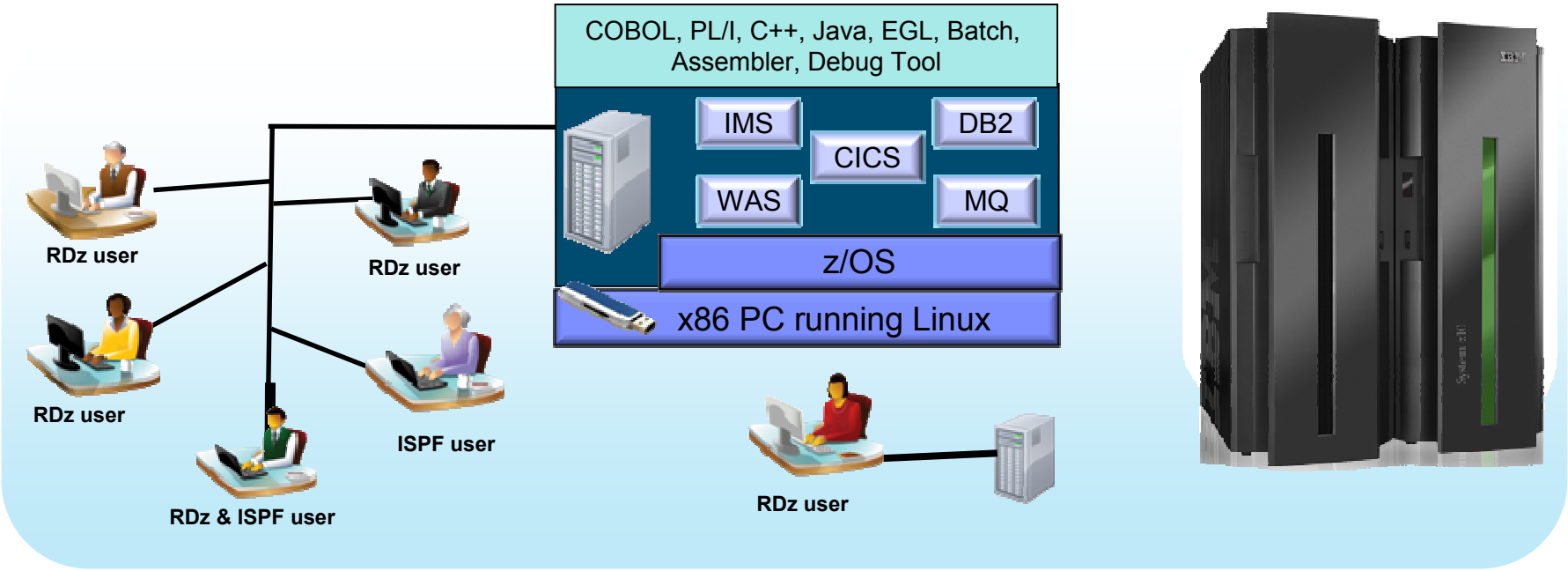
Detailed Continuous Integration for System z Scenario



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.

Rational Development and Test Environment for System z

The ultimate in modern application development for System z

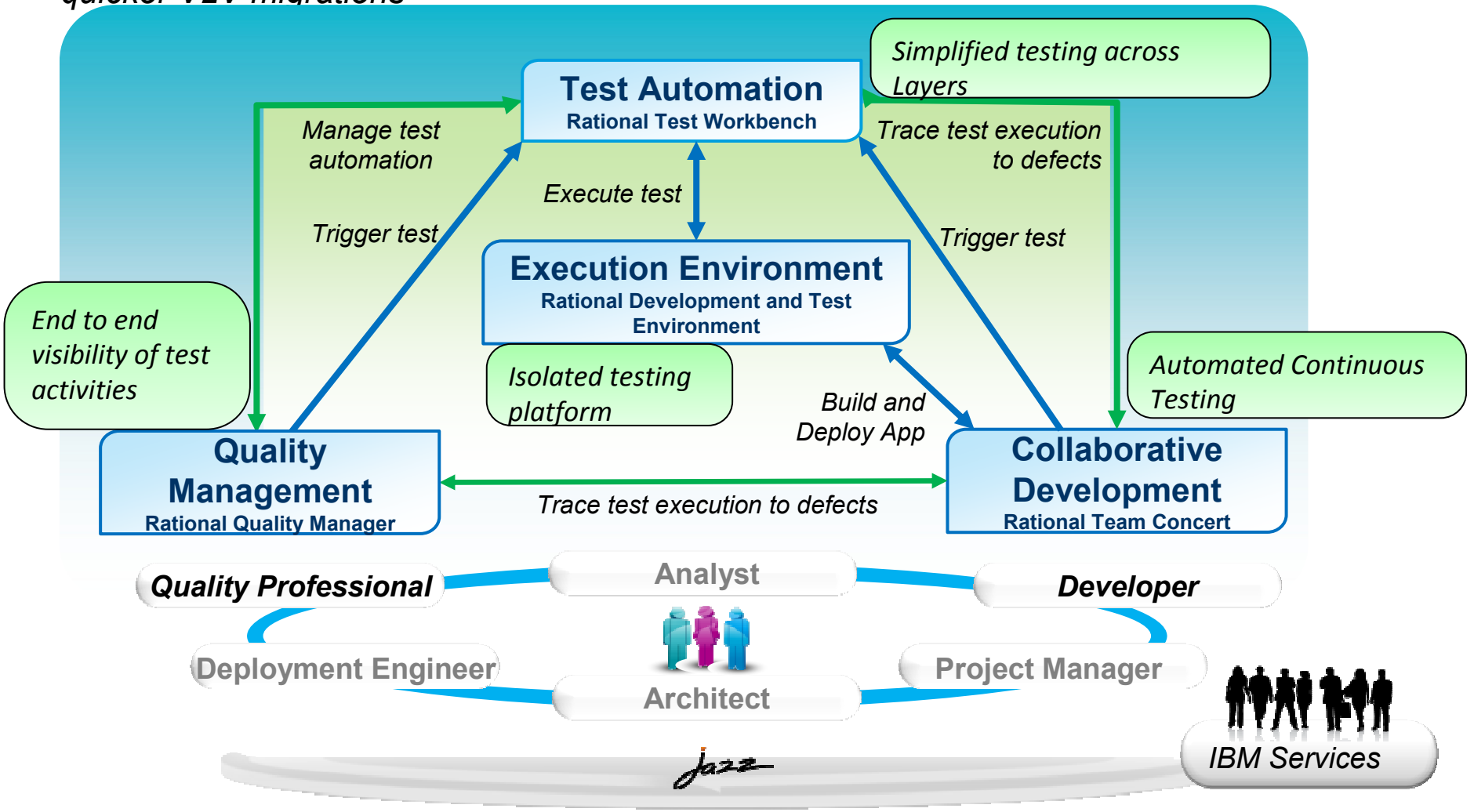


- Liberate developers to rapidly prototype new applications
- Develop and test System z applications anywhere, anytime!
- Free up mainframe development MIPS for production capacity
- Eliminate costly delays by reducing dependencies on operations staff

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.

IBM Continuous Integration Solution for System z

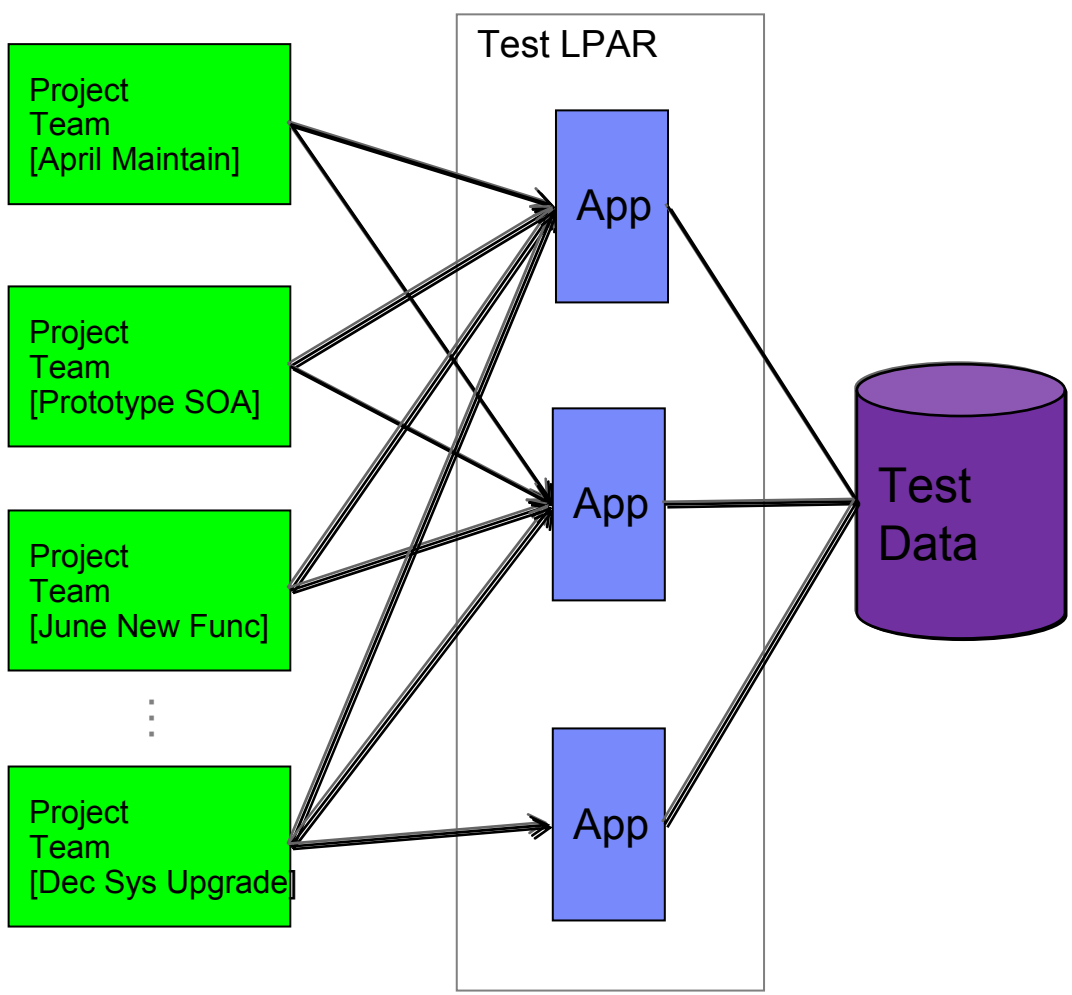
Reduced delivery time, improved end-to-end visibility of test activities, reduced risk and quicker V2V migrations



VIRTUALISATION DES TESTS

Typical z/OS Testing Architecture

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



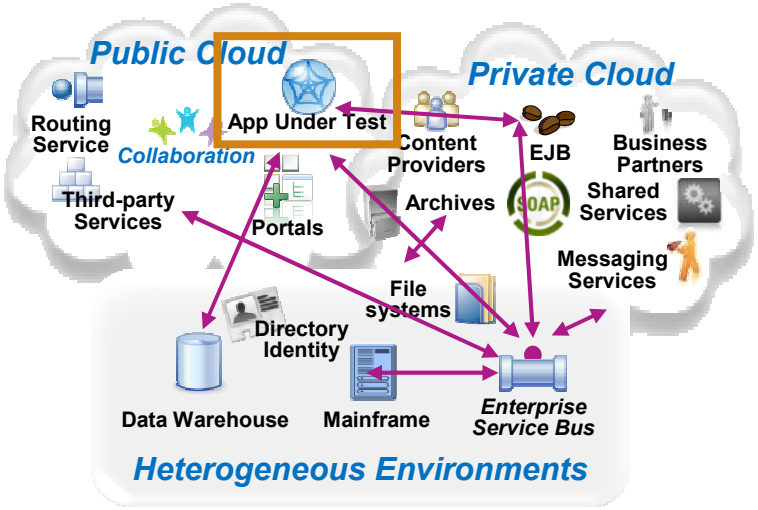
Problems Encountered

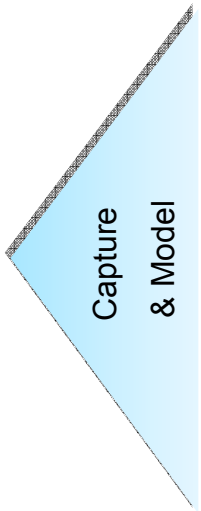
1. Shared resources combined with overlapping schedules can elicit conflicts, impede innovation and slow code delivery

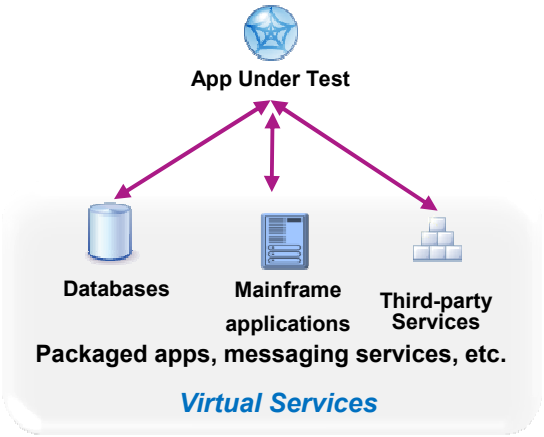
2. Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead

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

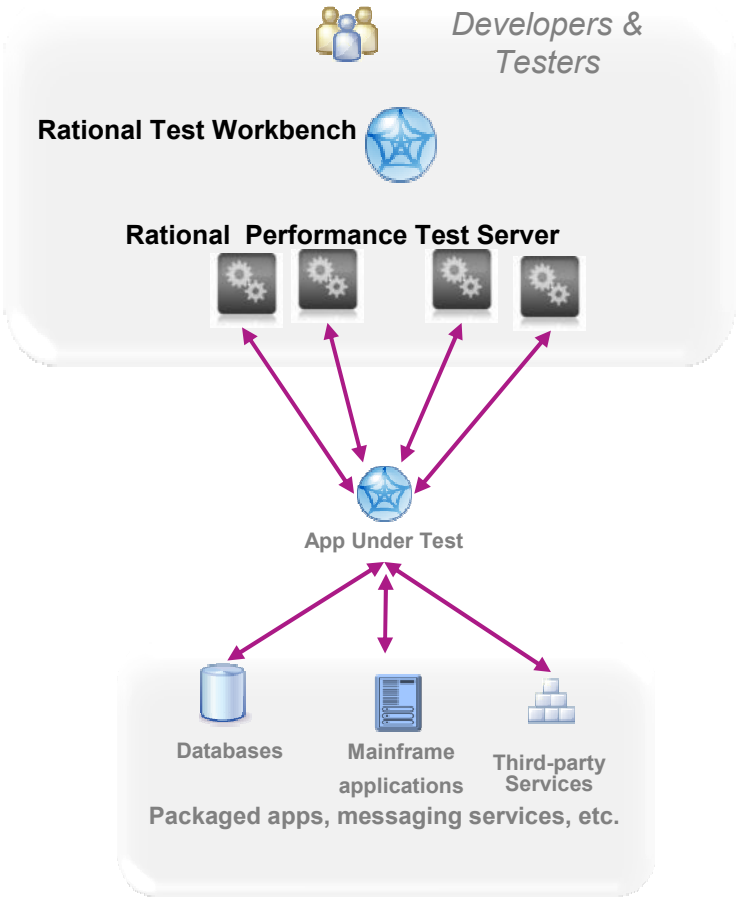
Test Virtualization enables to create "*virtual services*":

- *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)*

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



Rational Test Virtualization Server

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

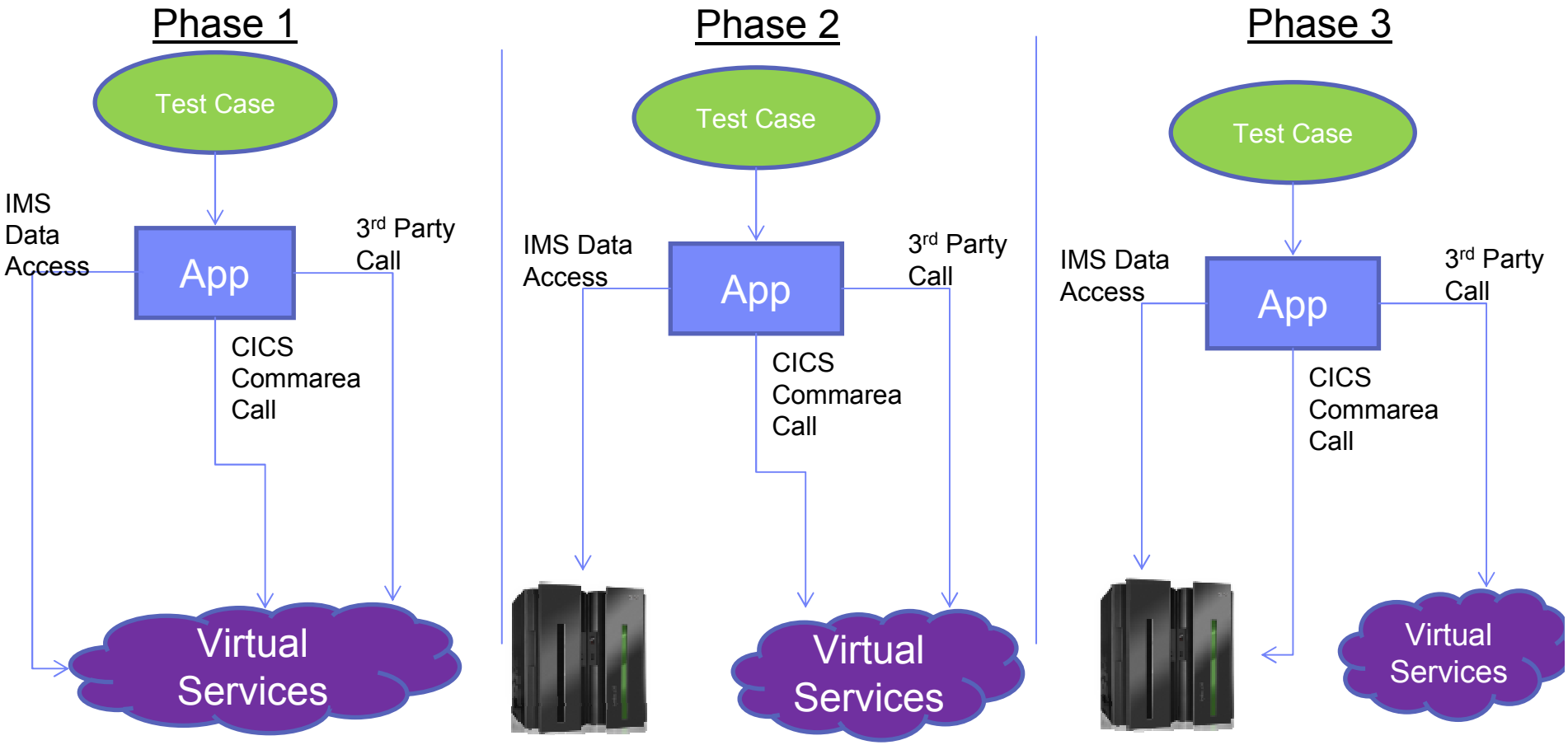
- .Net Objects
- Bytes
- COBOL Copybook
- ebXML
- EDI
- Fixed Width
- HL7
- IATA
- Java Objects
- MIME
- OAG
- SOAP
- Software AG Broker Docs
- SWIFT
- TIBCO ActiveEnterprise
- XML (DTD, XSD, WSDL)
- Custom

Note : Custom protocol support can be developed

Using test virtualization

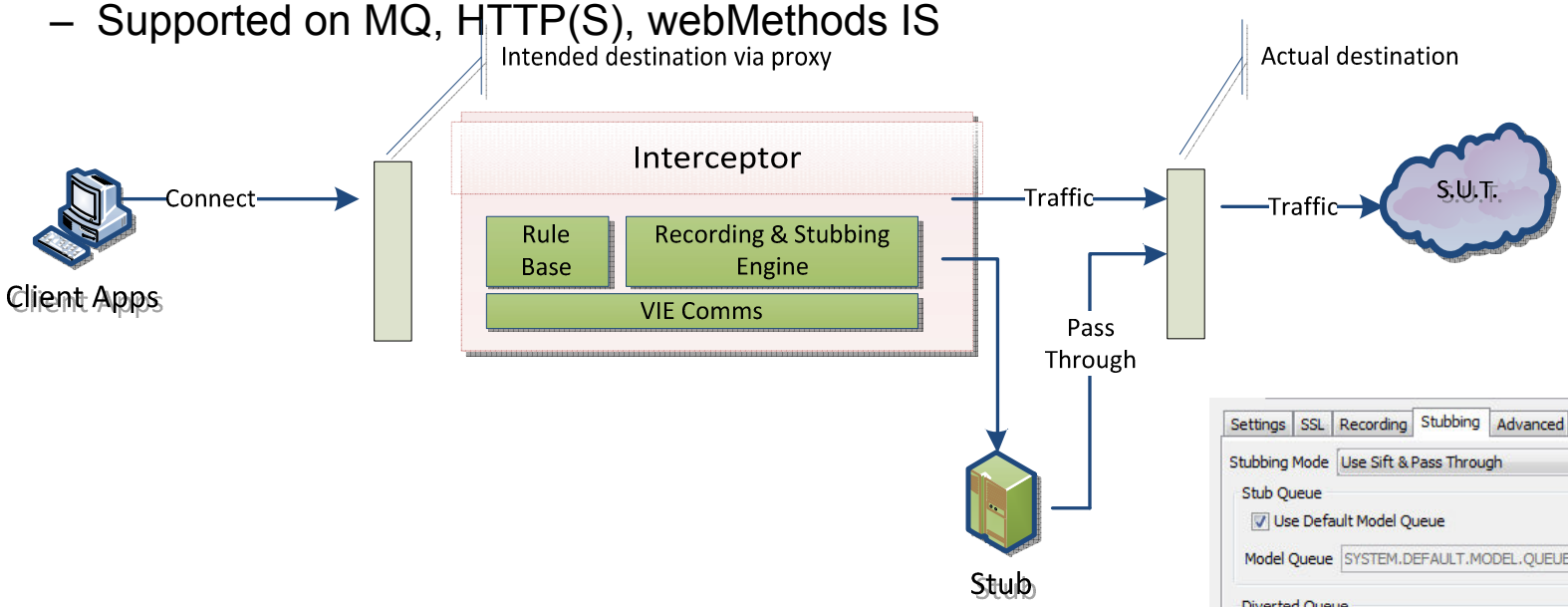
Controlled large system testing by isolating components under test

- Easier problem determination
- Lower test environment capacity requirements
- Improved component quality



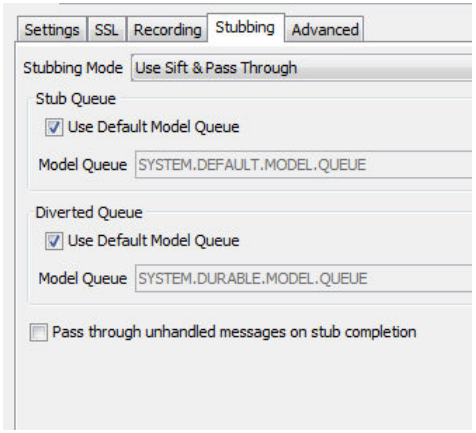
Virtualization Support – Sift & Pass Through

- Real and Virtual blending
- Reduce stub complexity
- Easy error simulation or delays
- Supported on MQ, HTTP(S), webMethods IS



Pass Through

Event	Configuration
CreditCheck	Discard -
AddressLookupByZipCode	Pass Through - delay of 0 ms



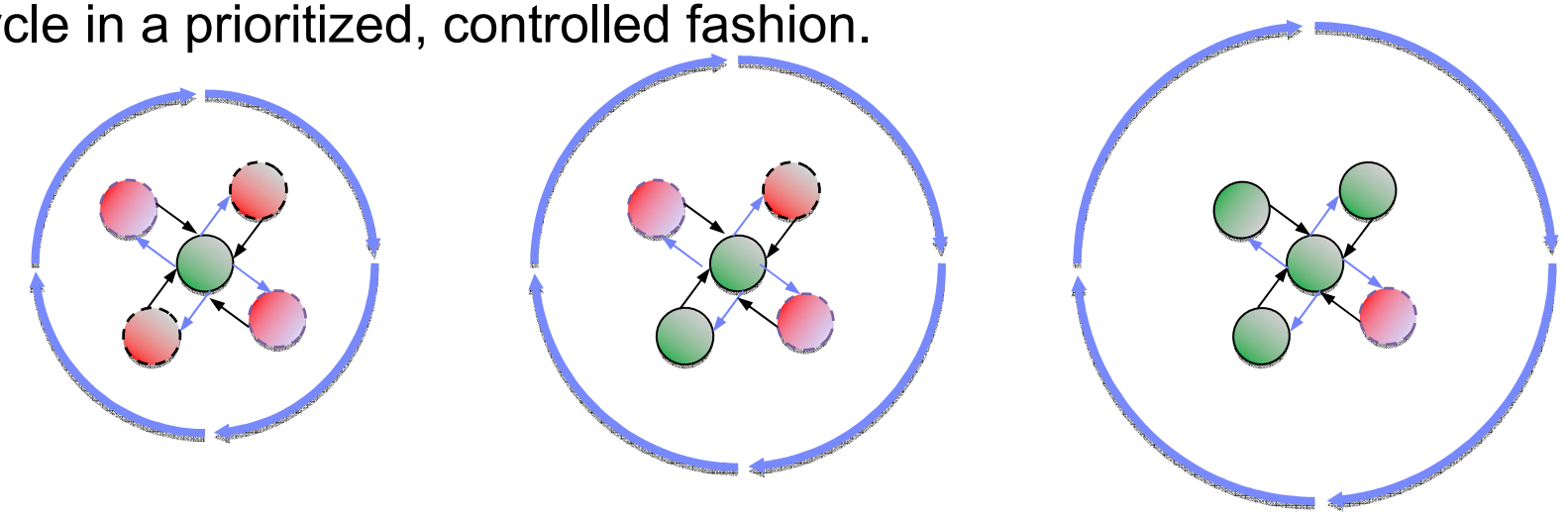
IBM Rational Test Virtualization Solution is a key enabler for Continuous Integration Testing

✓Test Virtualization is an enabler for continuous Integration Testing

Actual Service/App

Virtual Service/App

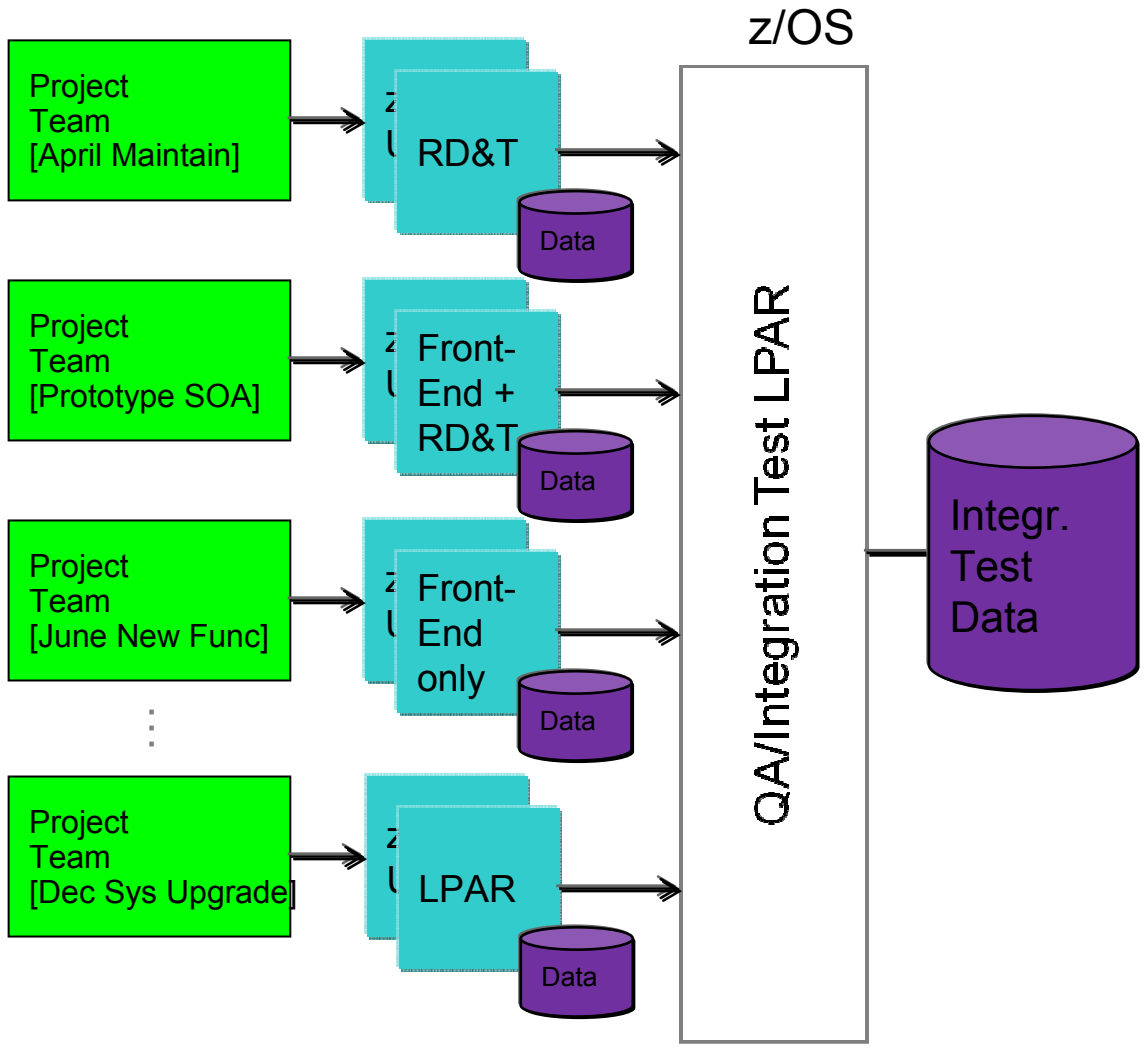
✓Services, applications, systems are introduced into the continuous integration cycle in a prioritized, controlled fashion.



Incremental Integration Testing

Testing Organized for Flexibility and Quick Delivery

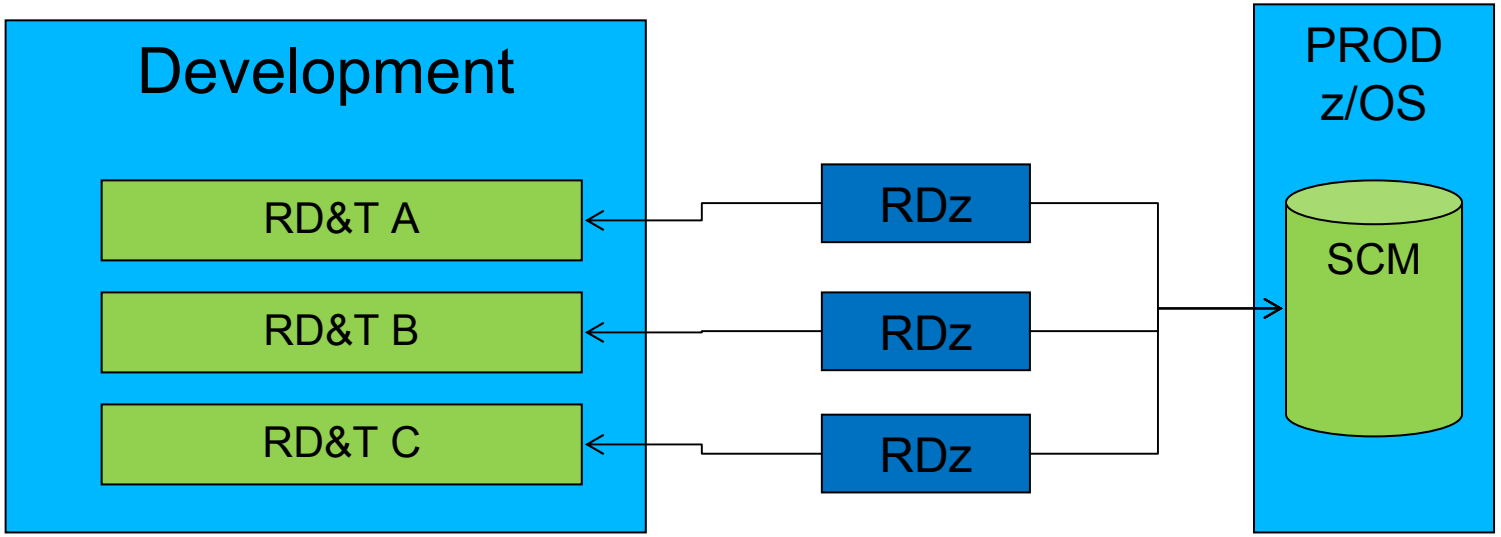
Organized by application team, horizontally sliced, dedicated resources, highly automated



Problems Encountered

1. Shared resources combi[illegible] over[illegible] schedules can [illegible] impe[illegible] slow code delivery
2. Coordination of environmental changes and releases cause bottlenecks, delays and additional overhead
3. Share[illegible] is diffi[illegible] and can [illegible] testing or inc[illegible] results
4. Provisioning, managing, and synchronizing project test environments including data

Customer example

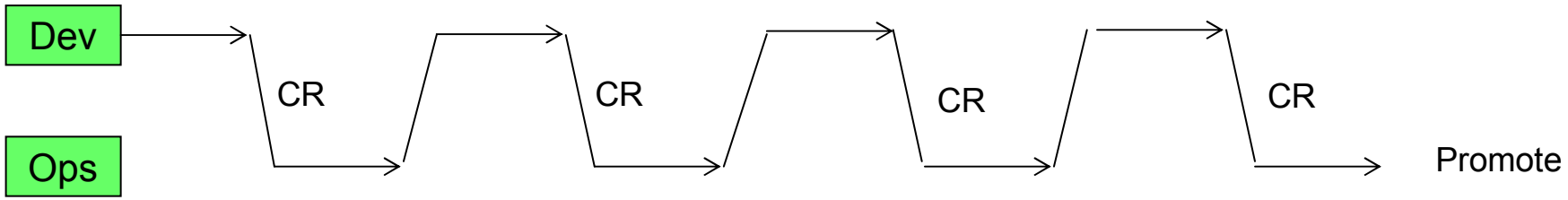


Large US Financial Customer - Implementation

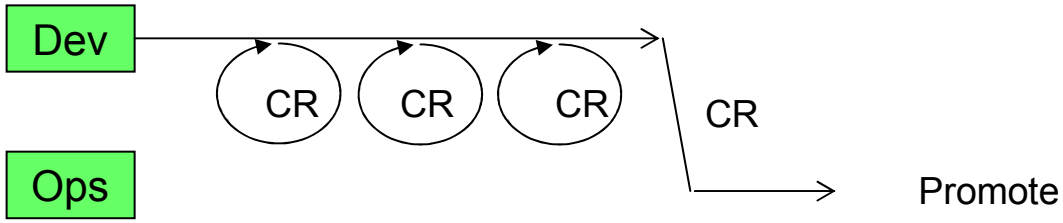
- Requirement
  - Provide more responsive System z access for application developers
  - Reduce application software delivery through faster test cycle
  - Provide each application group their own unit test facility
- RD&T Solution
  - Application developers have use of a uniquely defined RD&T feature 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 (few months)



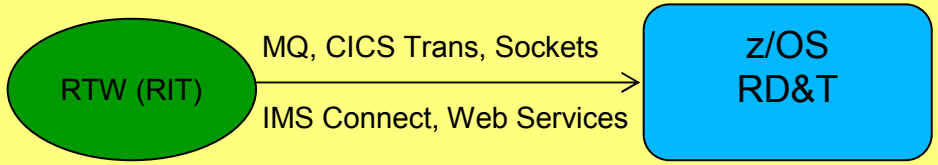
CIz process using RD&T (few weeks)



"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!"

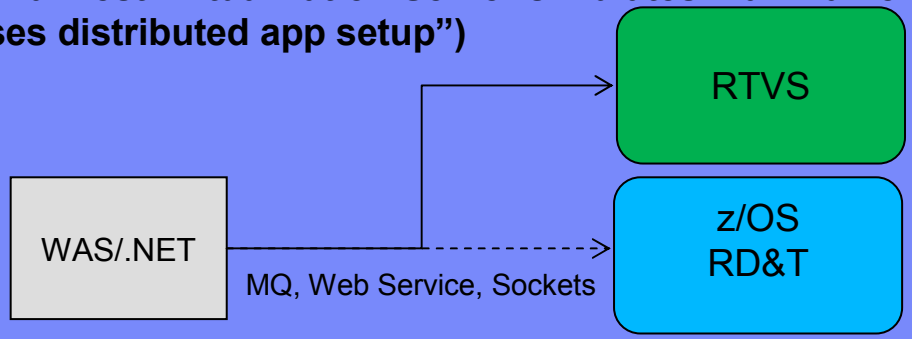
Test Automation and Virtualization for System z

1. **Rational Test Workbench drives mainframe interfaces ("Automate Testing")**

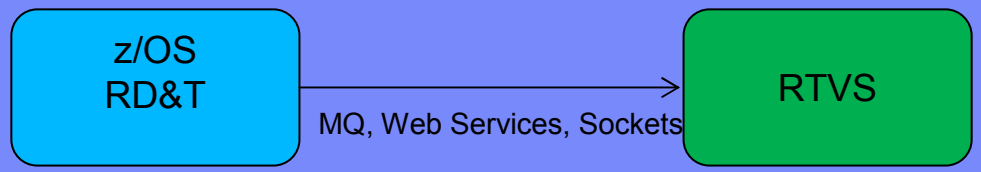


CIz Scenario

2. **Rational Test Virtualization Server simulates mainframe services ("eases distributed app setup")**



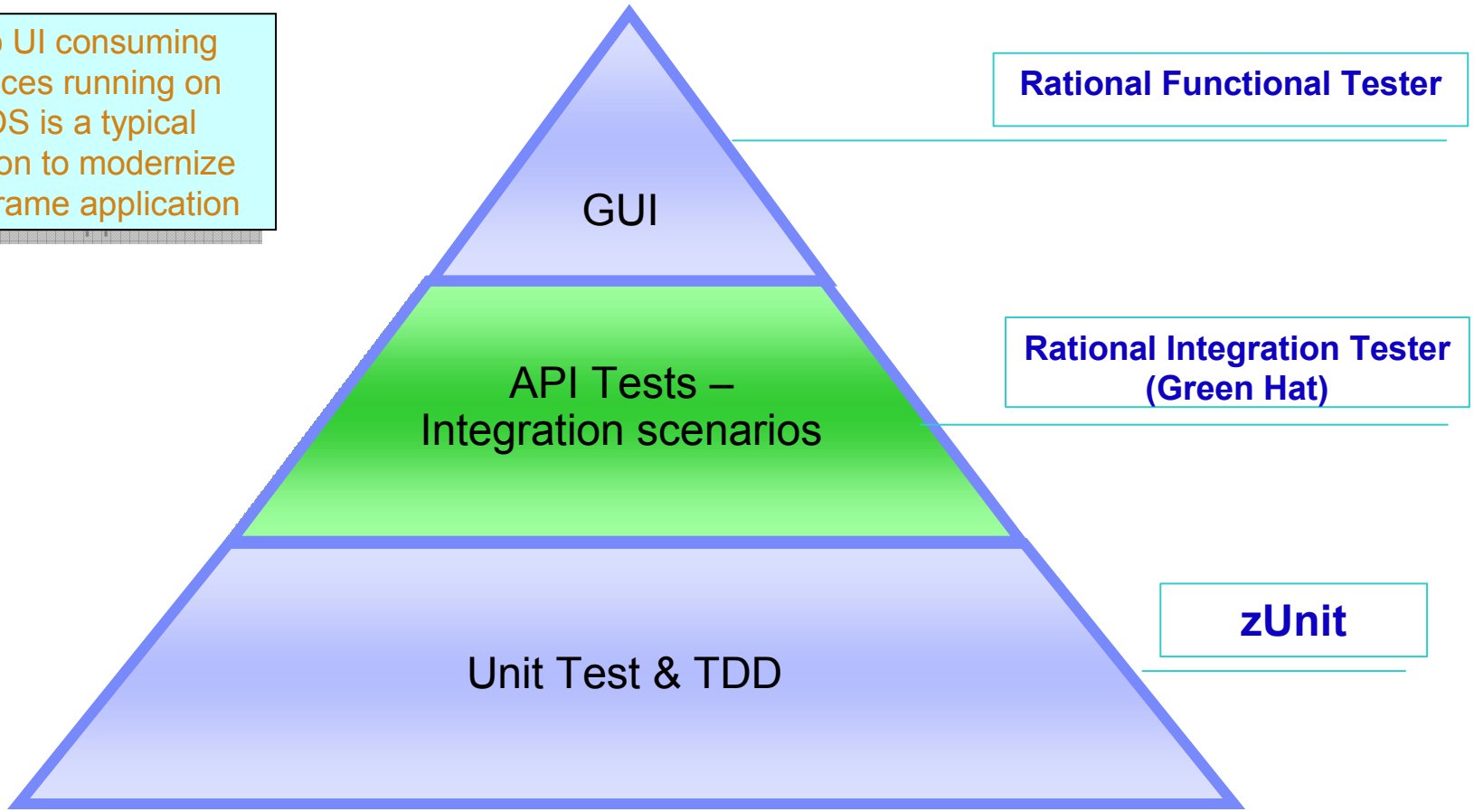
3. **Rational Test Virtualization Server simulates subsystem or external services ("eases z/OS app setup")**



New Mainframe capability in 4Q12

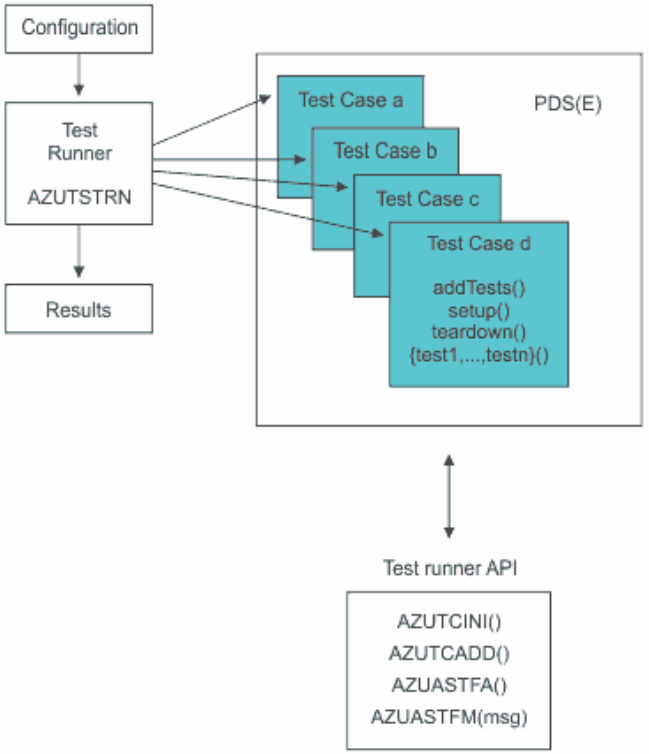
Levels of testing

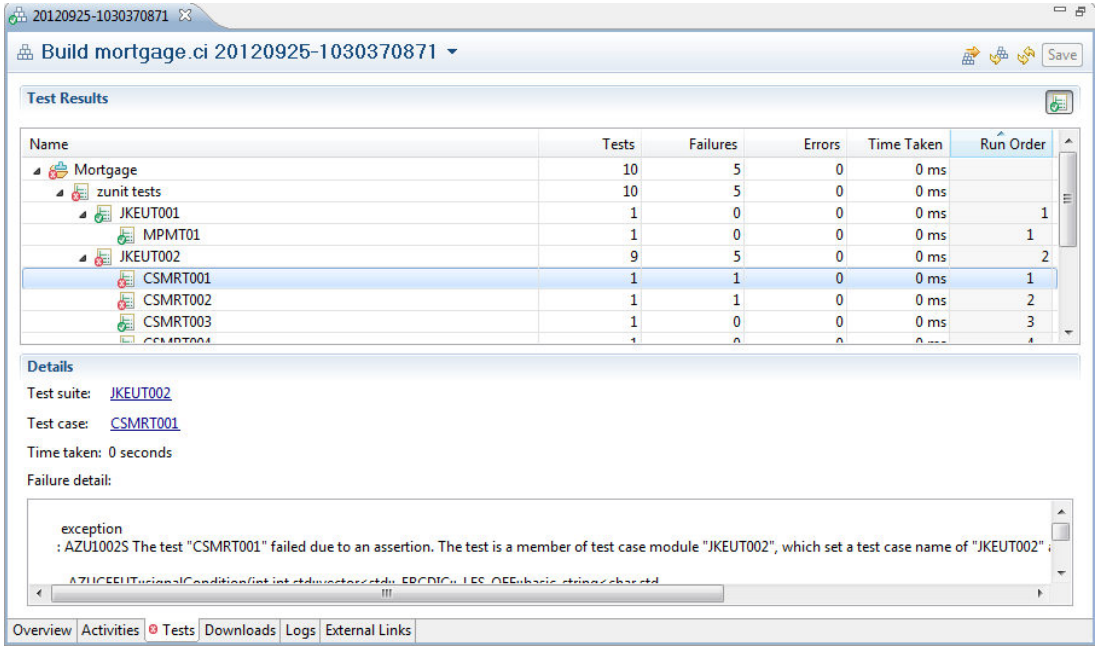
Web UI consuming services running on z/OS is a typical solution to modernize mainframe application



Run Unit Testing (zUnit) as part of the build

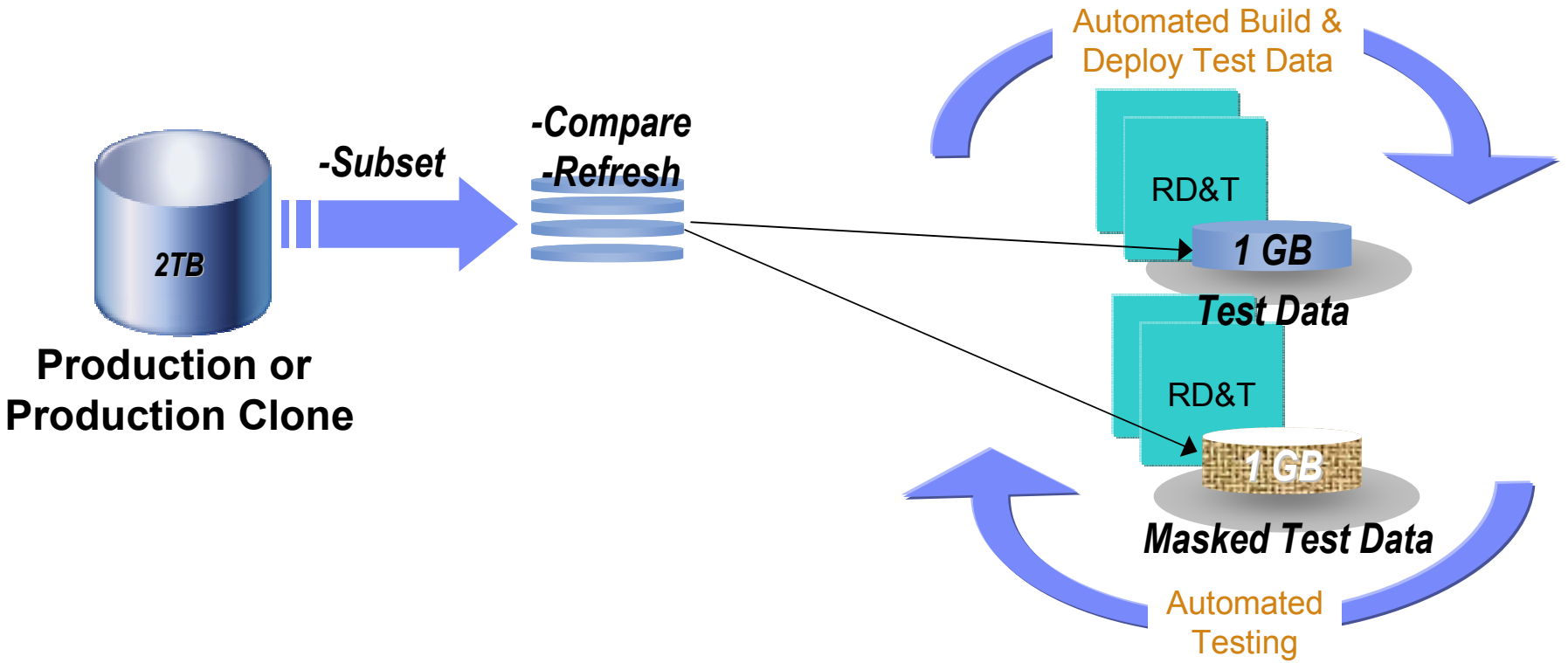
zUnit is an adaptation of the xUnit framework for writing code to run repeatable, self-checking unit tests. The ideas and framework developed in xUnit for unit testing object-oriented code are adapted by zUnit for testing Enterprise COBOL and PL/I code.





Running zUnit as part of the build

Create , secure, and deploy test data



Create "right-size" production-like environments for application testing

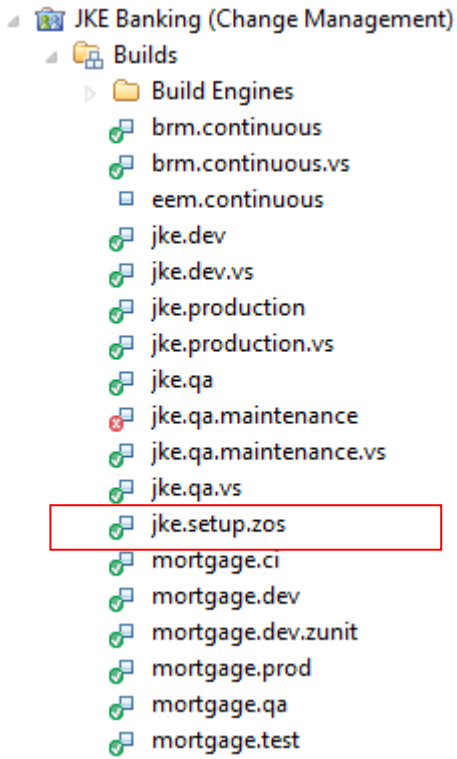
De-identify sensitive information with realistic *but fictional* data for testing & development purposes

Optim

Automated test data deployment for each build and test

RTC

Automatically setup region infrastructure



Automated setup infrastructure

in a test environment

1. Data Sets, USS Files
2. CICS Resource
3. VSAM Data
4. System configurations
5. …

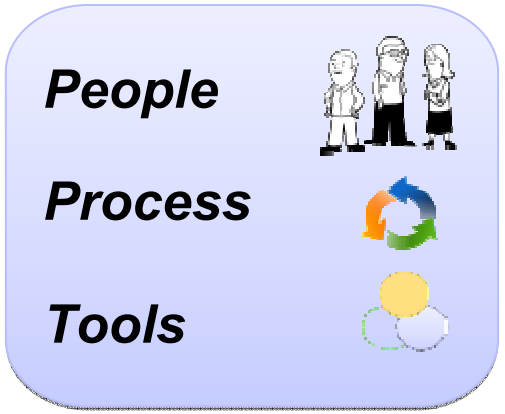
WHAT'S NEXT?

DevOps

A set of principles and values that facilitate collaboration across disciplines to…

1. Enable rapid evolution of deployed business services

2. Reduce risk, decrease cost, and improve quality across the portfolio

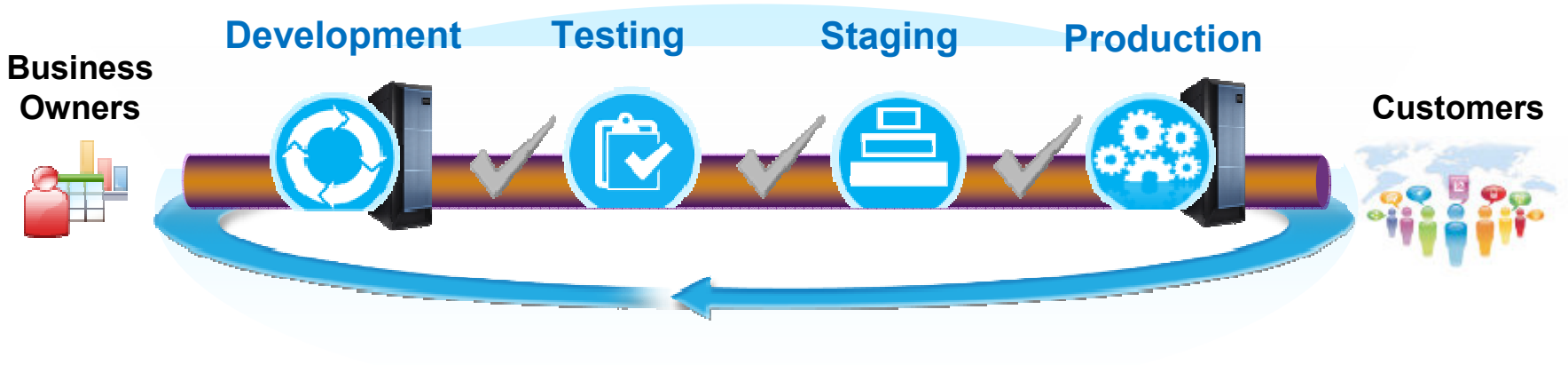


DevOps Principles

- **Collaborate across disciplines**
- **Develop and test against a production-like system**
- **Deploy frequently using repeatable and reliable processes**
- **Continuously monitor and validate operational quality characteristics**
- **Amplify feedback loops**

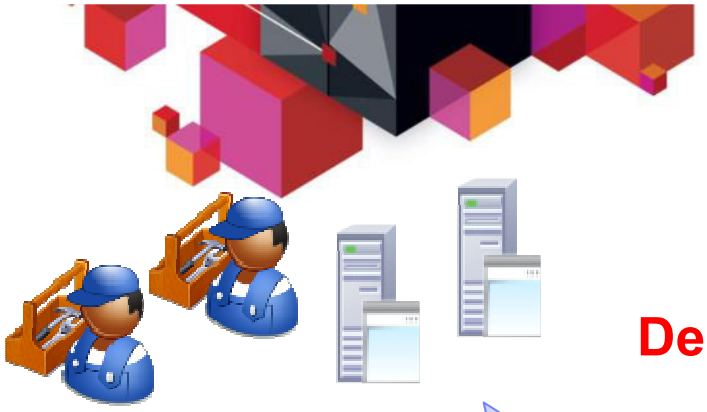
Continuous Delivery Pipeline

An iterative set of quality checks and verifications that each piece of application code must pass during lifecycle phases before being released to production.



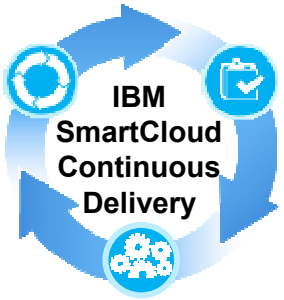
Ensures applications are production-ready throughout the lifecycle
and can be released at any time while minimizing rollback due to quality issues

Deploy application and configuration changes to test environments in minutes

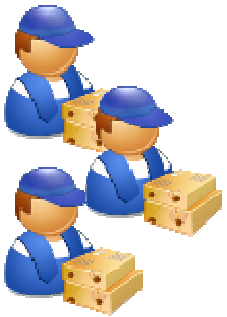


Operations team establishes deployment patterns for use by development and test teams

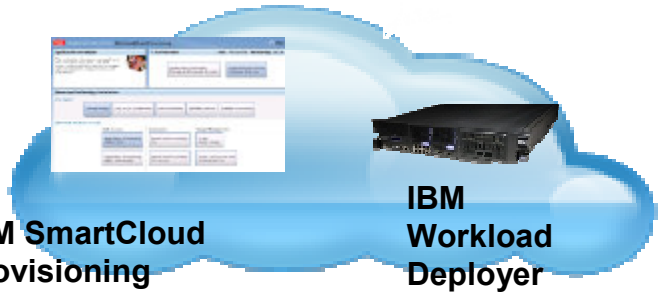
Deliver Environments 3x Faster!!



Development team makes application code and configuration changes

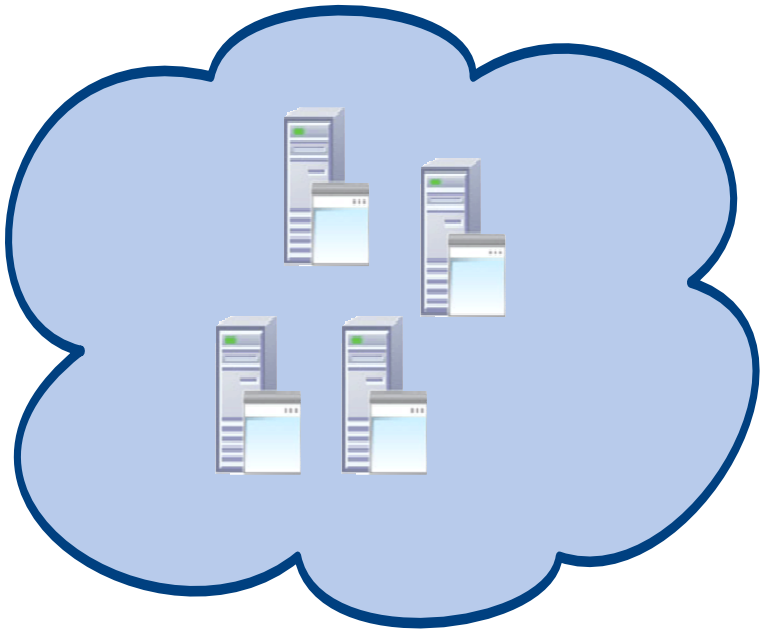


Test against production-like environments with Applications & middleware AUTOMATICALLY installed and properly configured

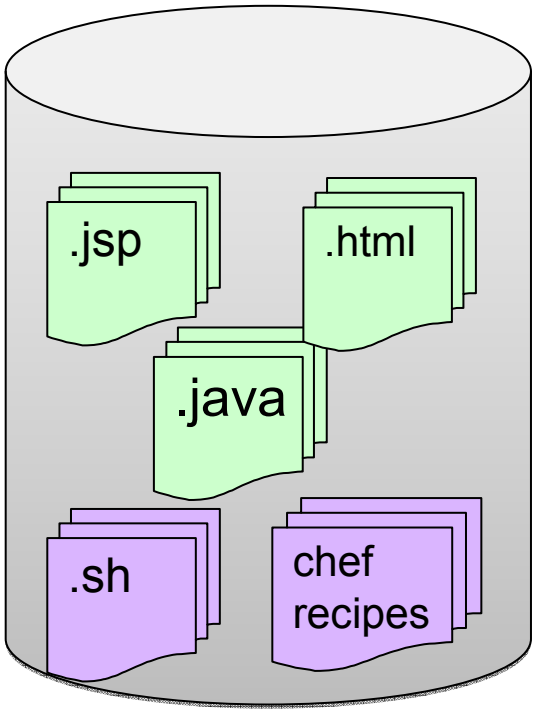


Provisioning using Cloud

```
#!/usr/bin/env ruby

class DevopsDeployer
  def initialize(build_url, build_id)
    @log = Logger.new(LOG_FILE)
    @log.level = LOG_LEVEL

    @iaas_gateway = IaasGateway.new(HsltProvider.new(),
LOG_FILE, LOG_LEVEL)
    @server_instance = nil

    rtc_build_system_provider = RtcBuildSystemProvider.new(
RTC_REPOSITORY_URL, RTC_USER_ID, RTC_PASSWORD_FILE)
    @build = rtc_build_system_provider.resolve_build(
build_url, ENV['buildResultUUID'], build_id)
    @build_system_gateway = BuildSystemGateway.new(
rtc_build_system_provider, LOG_FILE, LOG_LEVEL)
  end

  def add_build_stamp
    template_file = WEB_APP_ROOT +
"/app/templates/pages/page.html"
    @log.info "Adding build ID stamp #{@build.id} to \
#{template_file}"

    # Read in the file's contents as a string, replace
    # the build_id, then overwrite the original contents
    # of the file
    text = File.read(template_file)
    new_text = text.gsub(/\{\{ build_id \}\}/,
"<a href=\"#{@build.uri}\">#{@build.id}</a>")
    File.open(template_file, "w") { |file|
      file.puts new_text
    }
  end

# ...
```



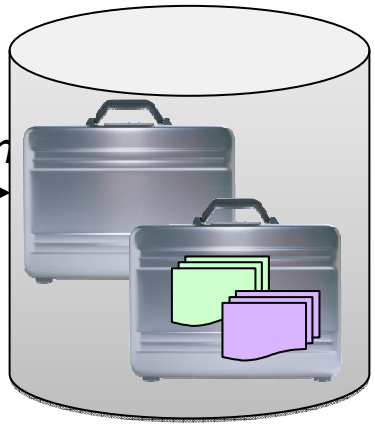
Delivery Pipeline Stage



Source Artifacts

Source Control Management

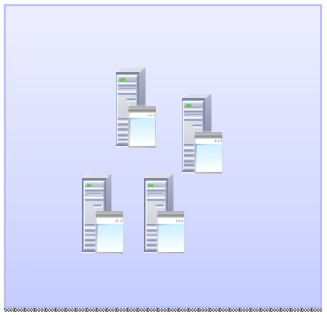
Build, Package, & Unit Test Application Binaries & Platform Configuration



Deployable Artifacts

Library

Deploy



Environment

Running System

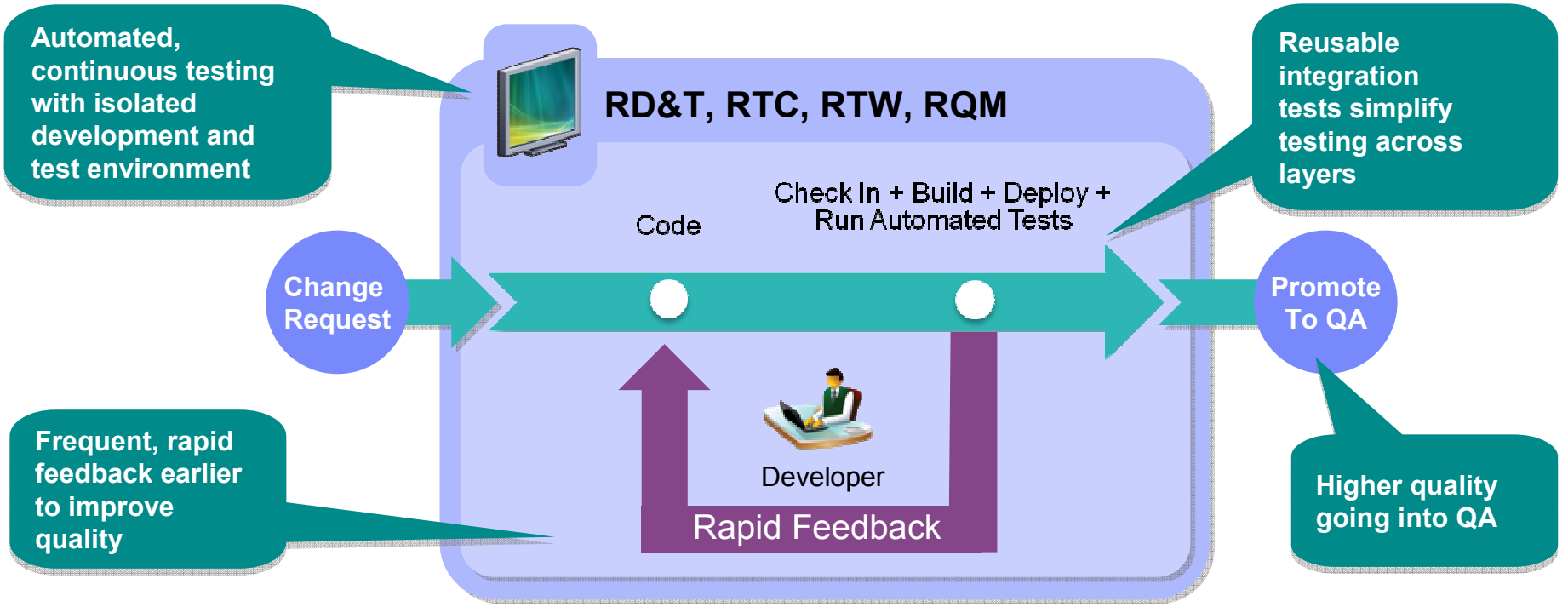
CIz Incremental Adoption*

Entry Point	Add Capability	Add Capability	Add Capability
• **More rapid, flexible developer testing** • **Reduce development MIPS** • **Easier and less costly provisioning of test environments**	• **Traceable deployment of application artifacts to test environment** • **Real-time build health results tied to code check-in** • **Unified status, change management, process, and SCM across tools, teams, and platforms**	• **Increase developer and tester productivity to reduce maintenance backlog** • **Tie automation to existing build process**	• **Improved insight into application testing health** • **Traceability to application errors**
RD&T **Add z/OS development and test environment on an x86 Linux Server**	RTC **Add collaboration and governance across diverse teams, platforms, and programming languages**	RTW **Automated Testing of mainframe application interfaces**	RQM **Tracking and planning of mainframe and cross-platform application tests as a combined suite**

*Follow on elements of the solution can be adopted in any order based on customer needs

IBM Continuous Integration Solution for System z

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



- 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

- Rational Team Concert
- Rational Quality Manager
- Rational Development and Test Environment for System z
- Rational Test Workbench powered by Green Hat Technology

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

www.ibm.com/software/rational

BACKUP

What is new?

Test Automation and Virtualization

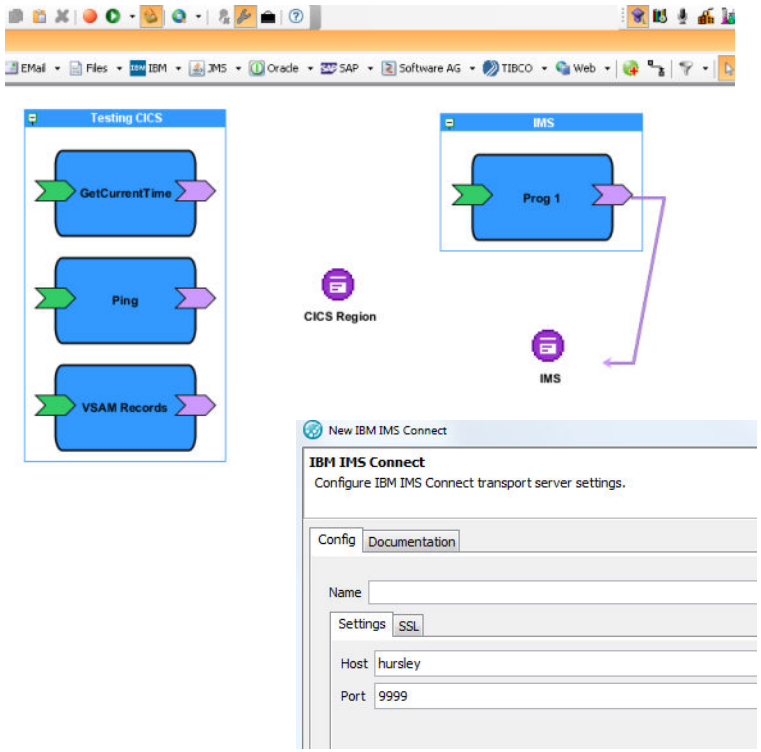
- **New testing capabilities for the mainframe**
  - MQ on System z
  - CICS Transaction Gateway
  - DB2z JDBC connections
  - IMS Connect

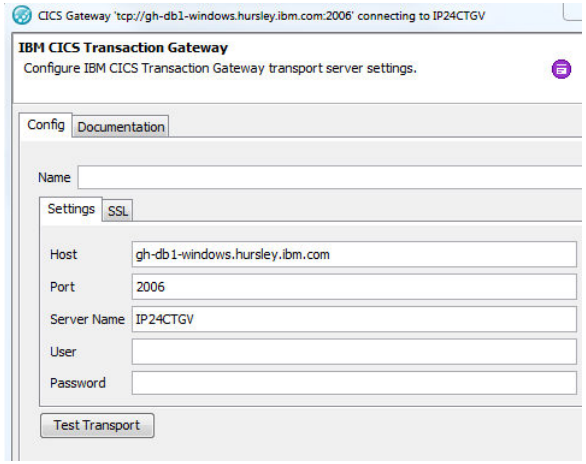
Continuous Integration for System z

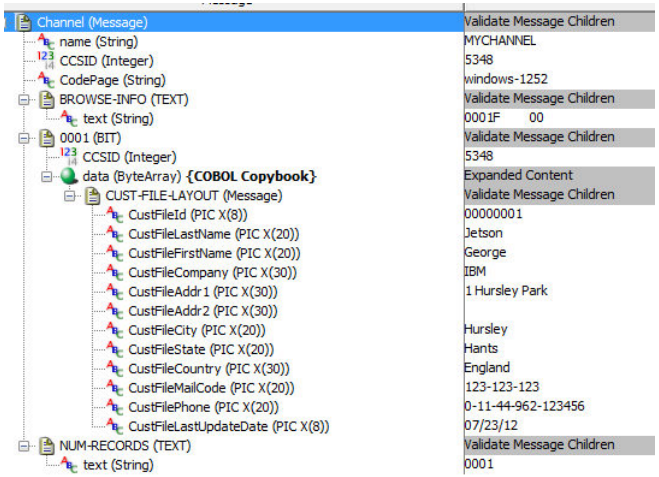
- **New capabilities and proof-of-technology**
  - Using **Optim** TDM and RTC to secure and provision test data
  - **Automatically configuring CICS** using RTC build and JCL/REXX to prepare for region for test
  - Executing and reporting results from **zUnit** using RTC build

Improvements in System z Support

- **Test and Record** CICS & IMS Transactions via CTG and IMS Connect
- Ability for distributed clients to access a virtualized DB2 on z
- Copybook improvements







Improvements in MQ Support

– MQ API Exits for MQ 7.1

– MQ on Z : Interceptor based recording of WebSphere MQ on Z (zero –client configuration)