Continuous Testing: DevOps for the Enterprise

Delivering Quality Software at the Speed of Technology

David Lawrence
Lead, IBM DevOps for Enterprise Community of Practice

Virtual CICS User Group Meeting – 1030 EST (1530 GMT) 12 January 2016

Reducing your CICS Development Costs with IBM DevOps for Enterprise Systems

David Lawrence – Lead, IBM DevOps for Enterprise Systems Community of Practice

Investing in the mainframe and System z

Investment in mainframe capabilities is critical to both current and future aspirations.

Future:

- Organisations unlikely to build new large global System z based applications in medium future.
- The challenge is to rapidly and efficiently expose systems of record to new systems of engagement.
- Business demand for reduced time to market and global solutions drives the need to rapidly deliver to common API's.
- Investing in current applications will
  - increase profit today by enabling you to deliver new features to market more rapidly.
  - reduce future maintenance costs and maintain your ability to continue rapid delivery by reducing end to end complexity
  - By moving disparate Core application systems of record to a common set of API's, position you for future consolidation.

8 Key Practices Accelerate Delivery

1. Minimum Viable Product
2. Deliver in Small Batches
3. Minimize Hand-offs, Maximize Flow
4. Eliminate Overhead
5. Automate Testing using APIs
6. Dedicate Teams
7. Practice Transparency
8. Loosely Coupled Architectures

Base: 600 IT professionals with app development responsibilities from US, Canada, UK, France, & Germany
Source: A commissioned study conducted by Forrester Consulting on behalf of IBM, May 2014

The unicorns (born on the web companies) set the bar for DevOps.
Some examples:

11.6 seconds mean time between weekday deployments, 1079 max deployments in an hour[1]

15000 engineers working on 4000+ projects, 5500 code commits/day,

75M testcases run daily[2]

>100 releases/day[3]

6419 deployments to production/year, 25/day, by 196 different people [4]

[1] http://www.slideshare.net/Dynatrace/why-everyone-needs-devops-now-gene-kim
[2] http://www.slideshare.net/realgenekim/why-everyone-needs-devops-now
[3] http://www.slideshare.net/jedberg/devops-at-netflix-reinvent
[4] http://www.slideshare.net/beamrider9/continuous-deployment-at-etsy-a-tale-of-two-approaches

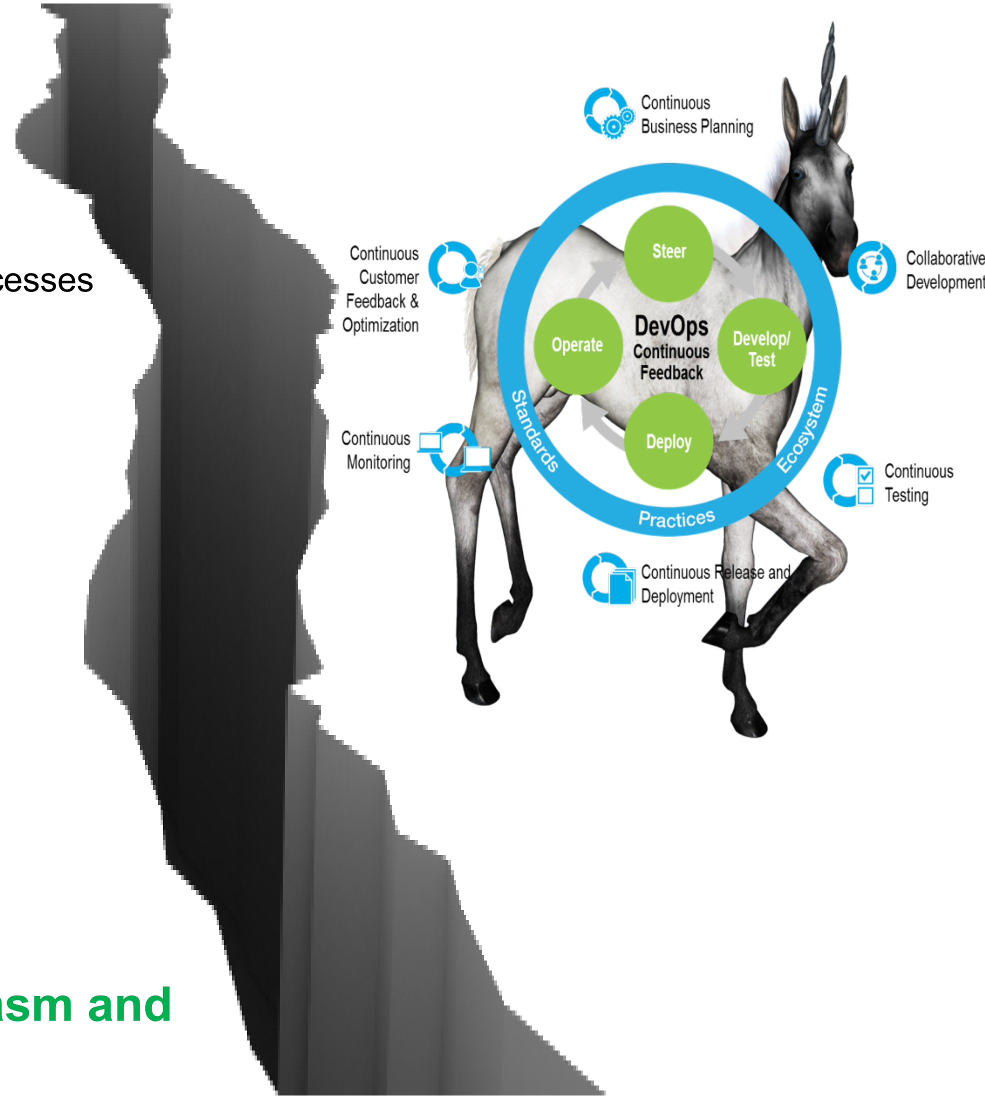
Reality: Most enterprise companies are not unicorns

Ancient Infrastructure and Beliefs Remain

- ❑ Outdated developer and team tools
- ❑ Aging developer population
- ❑ Disconnected teams, silos
- ❑ FUD: “millennials can’t code COBOL”, “manual processes exist for a reason”, “SoR dev can’t be as nimble as distributed dev”

Ancient Practices Need Overhauling

- ❑ Manual testing
- ❑ Availability of entire system is required to test
- ❑ Mainframe availability required (if some z)
- ❑ Reluctance to move test data off mainframe
- ❑ Cross-platform coordination required
- ❑ Manual project prioritization, status tracking


Continuous Business Planning
Collaborative Development
Continuous Customer Feedback & Optimization
Steer
Operate
DevOps
Continuous Feedback
Develop/ Test
Deploy
Standards
Ecosystem
Practices
Continuous Monitoring
Continuous Testing
Continuous Release and Deployment

So, is it possible to cross this chasm and become a unicorn?

Utopia: Enterprise unicorn fun facts

Yes!!! And, many large companies are leading the way.
Some examples:

80 deploys/week, <10 incidents/month[1]

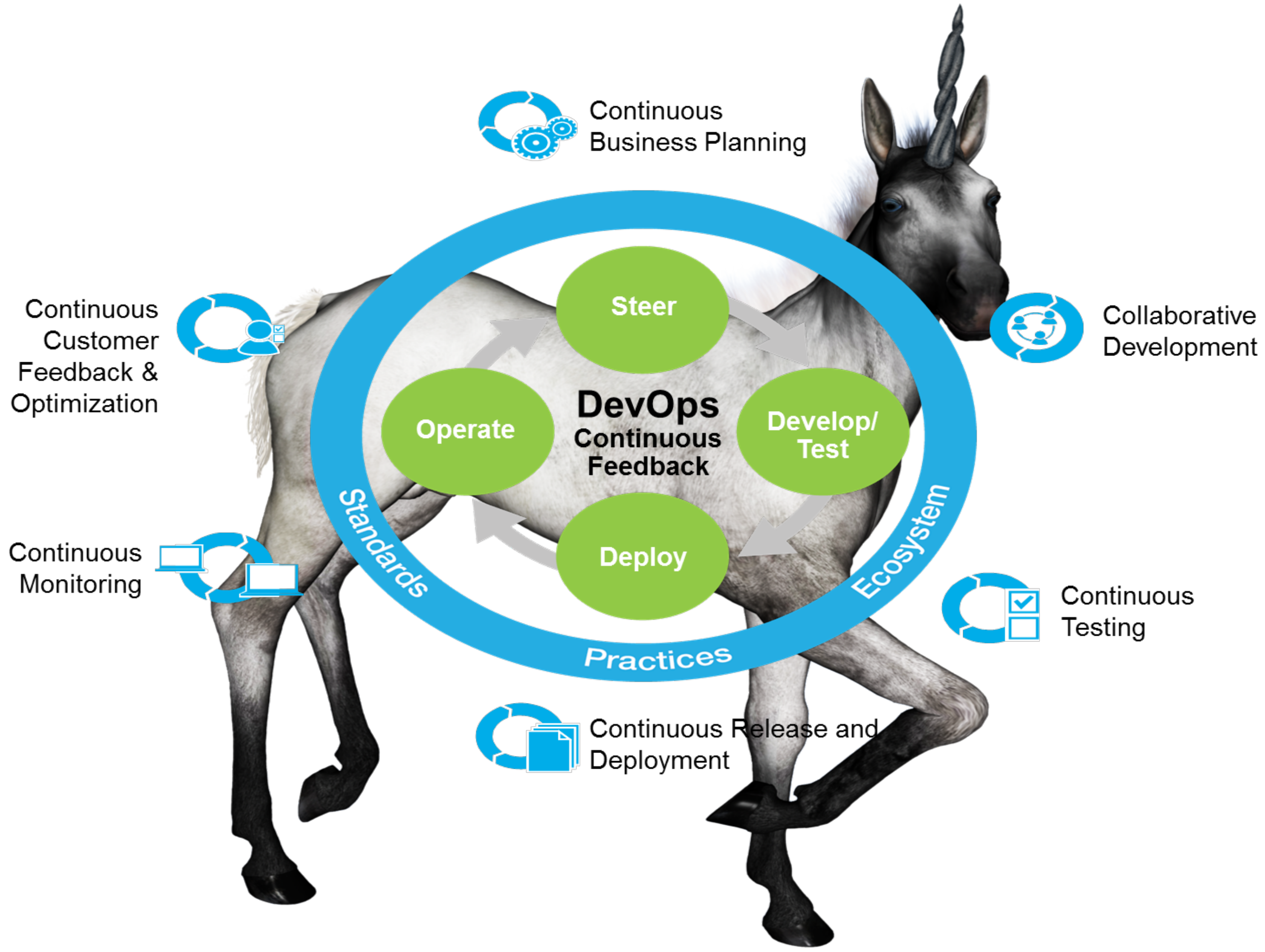
80% reduction in critical defects, 70% increase in system availability, 90% on-time delivery vs. 60% previously[2]

reduced dev cost from 100M to 55M/year, 140% increase in number of products under development[3]

resale up 30% first half of 2014, 24% YoY increase in customer service rating[4]

[1] http://www.slideshare.net/DevOpsEnterpriseSummit/does14-ross-clanton-and-heather-mickman-devops-at-target-41869677
[2] http://www.slideshare.net/DevOpsEnterpriseSummit/tuesday-400-hayden-lindsey-and-carmen-de-ardo-final?
[3] http://www.slideshare.net/DevOpsEnterpriseSummit/does14-gary-gruver-macys-transforming-traditional-enterprise-software-development-processes
[4] http://www.slideshare.net/DevOpsEnterpriseSummit/tuesday-330-shakeel-sorathia-final?qid=d758c122-8df0-4e03-b2da-4ba4c7271897&v=qf1&b=&from_search=11

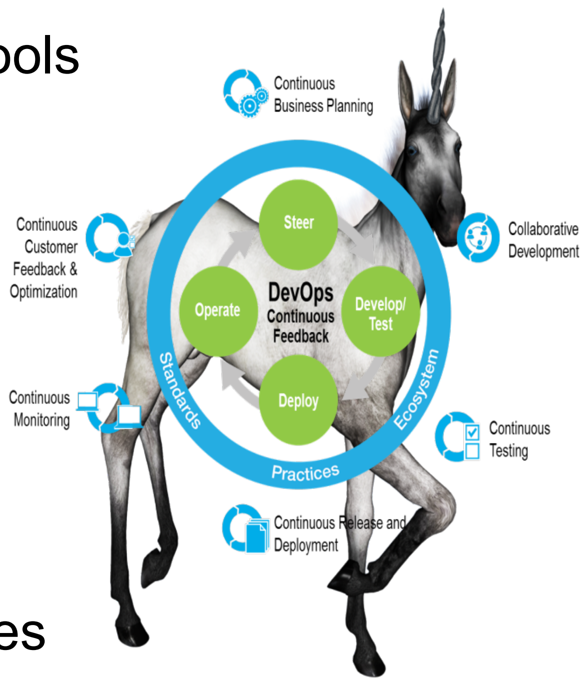
Delivering on the DevOps promise


Continuous Business Planning
Continuous Customer Feedback & Optimization
Collaborative Development
Steer
Operate
DevOps
Continuous Feedback
Develop/ Test
Deploy
Standards
Ecosystem
Practices
Continuous Monitoring
Continuous Testing
Continuous Release and Deployment

Evolving towards the unicorns

B
es
t
Pr
ac
tic
es

- ❑ Use modern multi-platform developer and team tools
- ❑ Automate deployment, configuration, and testing
- ❑ Use virtualized services to enable earlier testing
- ❑ Offload testing from the mainframe
- ❑ Build and deploy in small batches
- ❑ Start with small pilot projects to build confidence
- ❑ Use real-time dashboards
- ❑ Consolidate SCMs
- ❑ Build a staged rollout plan
- ❑ Train the teams in tool usage and process changes
- ❑ Organize with cross functional teams
- ❑ Gain executive buy-in and sponsorship up front
- ❑ Hire and train millennials on enterprise applications, tools, and languages
- ❑ Employ a loosely coupled architecture

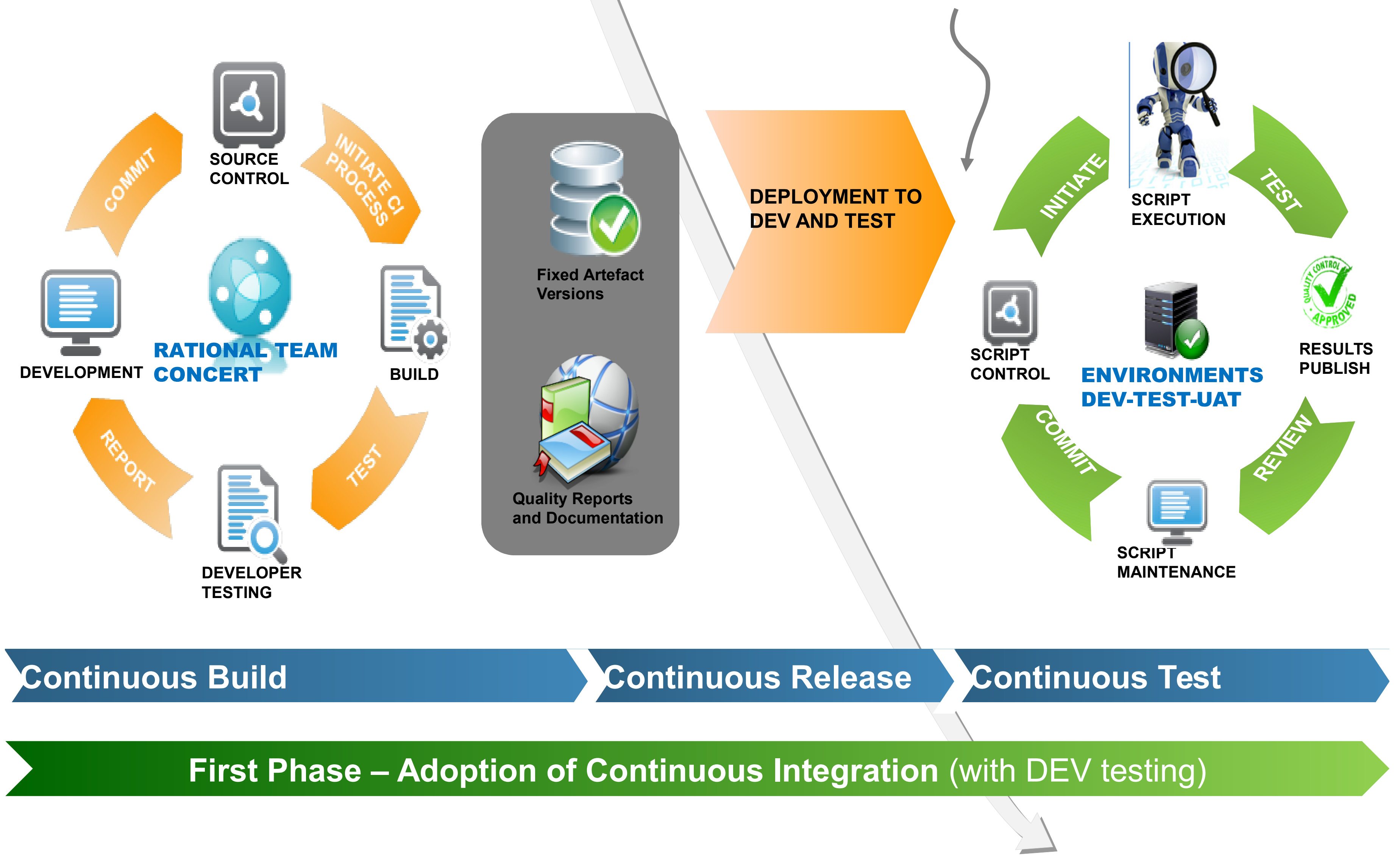

Continuous Business Planning
Continuous Customer Feedback & Optimization
Collaborative Development
Steer
Operate
DevOps Continuous Feedback
Develop/ Test
Deploy
Standards
Ecosystem
Practices
Continuous Monitoring
Continuous Testing
Continuous Release and Deployment

Proposed solution based around proven technology

Deliver a continuous integration software stack that enables application development for the mainframe and beyond.

Capabilities:	**Proposed Solution (and TRM status)**
Developer IDE	RDz or Rde for full Enterprise IDE facilities
Automated Unit Test	zUnit (fundamental capability of RDz)
Off mainframe z/OS environment	Rational Developer & Test
Collaboration and integration	RTC
Environment mirroring	Optim (TDM), Urban Code and GhreenHat
Continuous Delivery	Urban Code
Quality Dashboard	RTC

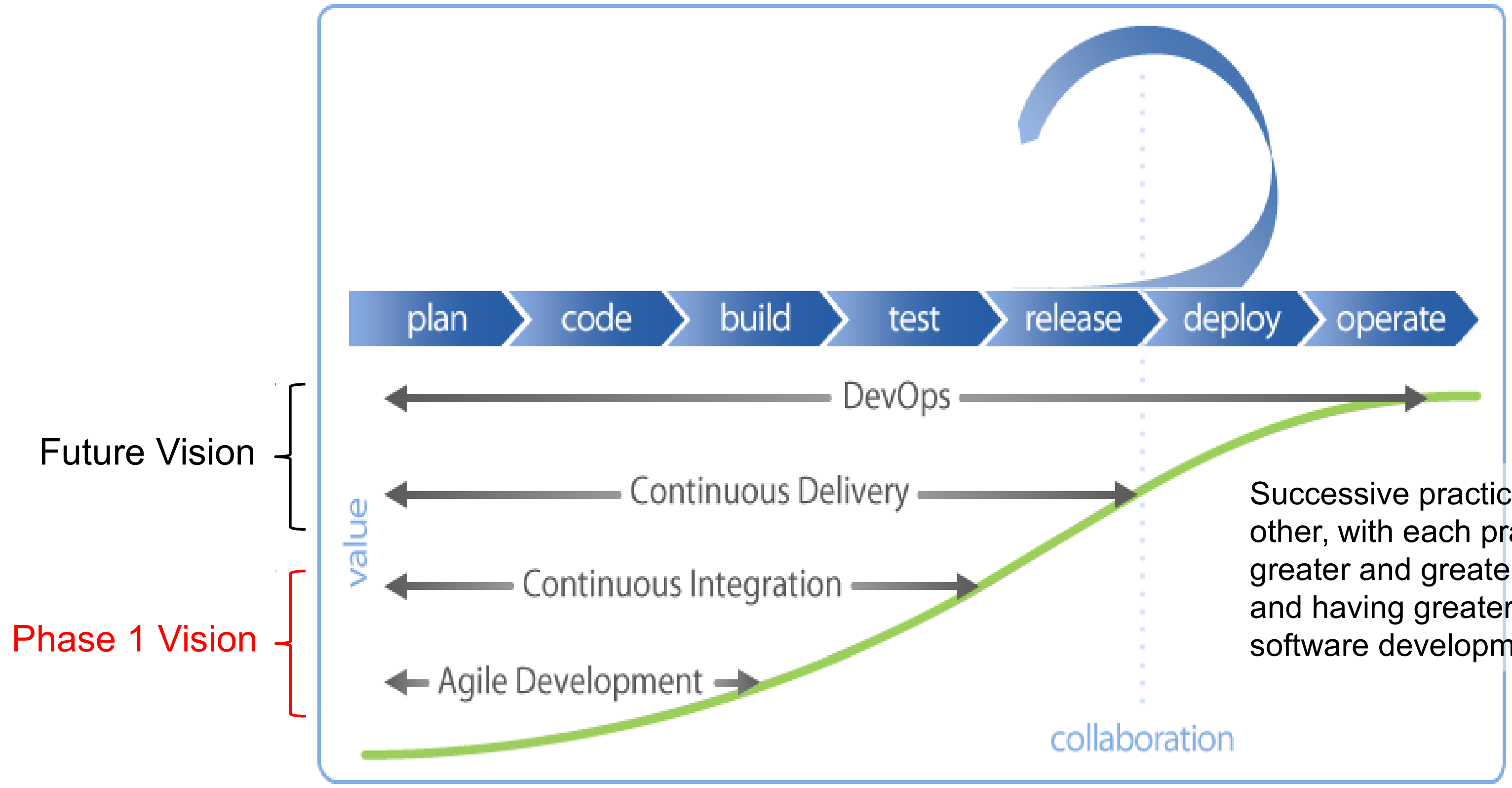
Common Vision for Development Practices



Why start with Continuous Integration?

Improves Quality
Improves Productivity
* if done correctly

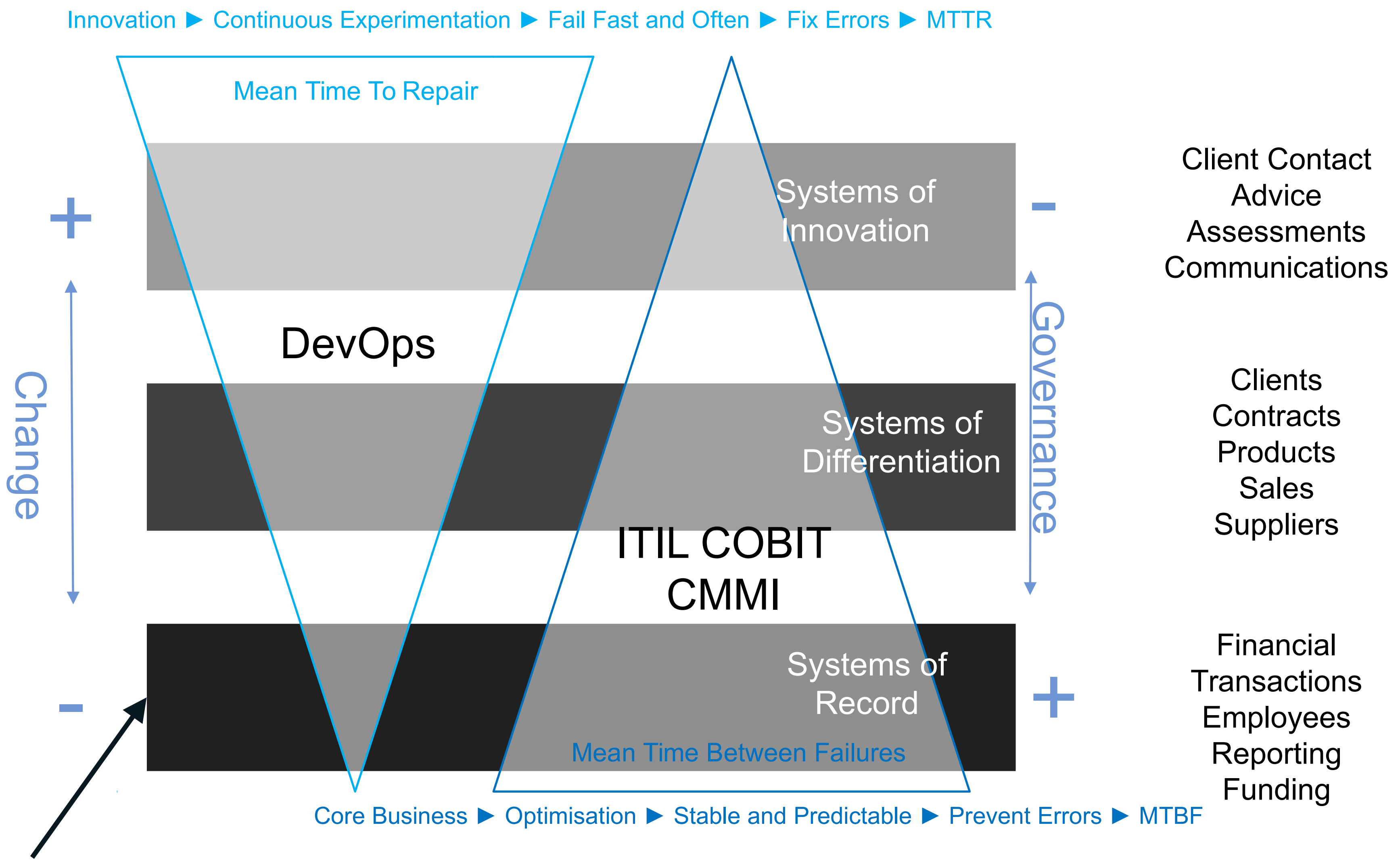
- Instant feedback to developers on quality issues
- Aids unit test automation <u>on every build</u>
- Supports Agile development
- Pre-cursor to Continuous Delivery and DevOps



Successive practices build on each other, with each practice adding greater and greater business value and having greater scope over the software development process.

ALM Modernisation

Pace layer approach and mainframe ALM modernisation



Mainframe applications tend to be 'systems of record' – stability is a core consideration.
Stability and Agility should not be mutually exclusive

System z and CI Through DevOps

Risks and Mitigation

Continuous Integration
Known Risks and proposed mitigation

The following key risks have been identified;

- Environmental mismatch.
  - If the z/OS Linux environment mismatched the mainframe environment we introduce delay not savings.
    - Mitigation – Review UrbanCode to minimise the work involved
- Increased demand on high cost resource removes any financial savings.
  - Sysprogs maintain the environments. This resource is more expensive than application developer resource.
    - Mitigation - Need to validate the Sysprog overhead is not too onerous as above UrbanCode investigated.
- Robust RTC to Endevor or RTC SCM link required
  - Need to handle automated integration between the strategic SCM and the virtual environment.
- Cost and availability of high quality training.
  - Any solution will need to be capable of global deployment and consumption.
- New skills required to drive efficiency
  - Need to focus on technical skills across the community.
    - Mitigation - Looking at global technical training opportunities in parallel with this initiative.
- Test driven development capabilities immature.
  - Deliver a means to carry out unit testing of CICS and DB2 programs.
    - Mitigation –IBM working on a solution and investigate potential test harness capabilities as an interim measure.
- Security considerations need to be evaluated.
  - Impacts of the new platform, ISR involvement critical.

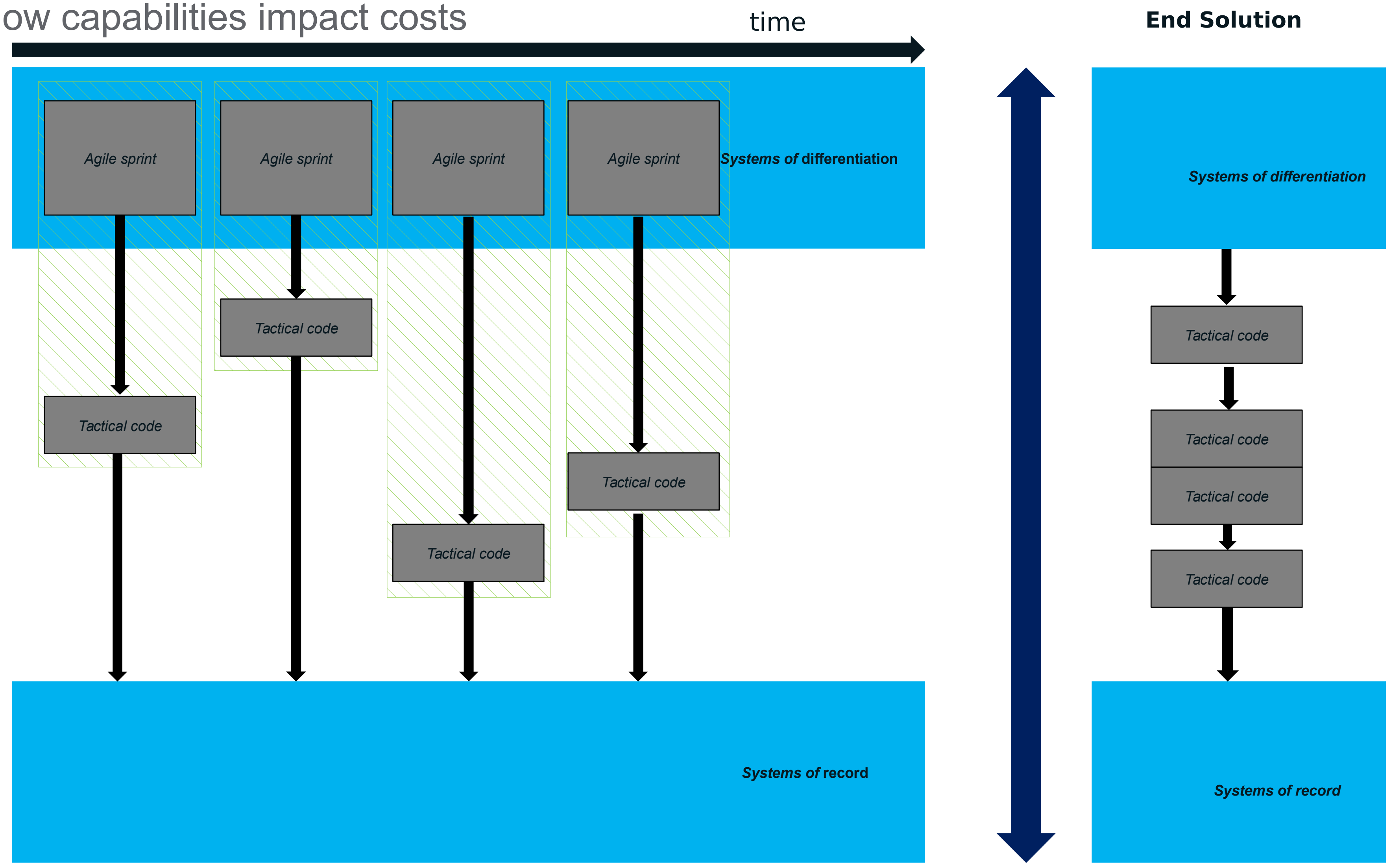
Delivery Pressures
Time versus maintainability

- Do you have already have continuous integration capabilities in the distributed languages.
- Pressures to deliver in an Agile manner against perceived slower pace in mainframe development lead to screen driven development.
- Tactical code written within the sprint to meet delivery dates whilst minimising impact on the system of record.
  - Screen based design can lead to earlier delivery.
  - True end to end architecture, design and development leads to more easily maintained services.

- By providing continuous integration capabilities to mainframe developers we maximise the potential for true end to end delivery.
- Coupled with a definition of common API’s for global systems of record we create the possibility of moving toward our future state aspirations.

Current process

How capabilities impact costs



Technical debt, e2e complexity and eventual loss of agility imposed

Code Quality

Moving quality to the left

The adoption of automated unit testing introduces fundamental new opportunities:

- Enables a move to test driven development on the mainframe.
- Developers build tests with expected results directly from the requirements. zUnit supports this, the tests are mainframe artefacts.
- Run the tests to validate they fail – (if they pass your code already supports the business requirements!)
- Build code to fulfil the tests.
- Use the quality dashboard to validate % successful tests over time - progress reports are % of function delivered.
- At the end of the project the tests and known results are an asset stored in the SCM. Future regression tests are stored with the code.
- The quality dashboard stores unit test line and branch coverage, and you can click to individual lines.
- All this capability is automated as part of deliverable.

Modern and open tools for z Systems DevOps

Modern and open tools for z Systems

Java 8 and z13

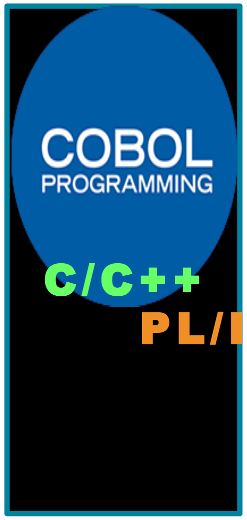
Optimized CICS, IMS and DB2 transactions

COBOL, PL/I, & C/C++ Compilers

z13 exploitation for increased performance

Up to 50%
improvement for generic applications

Up to 2X
improvement in throughout per core for security enabled applications



Up to 17%
performance improvement

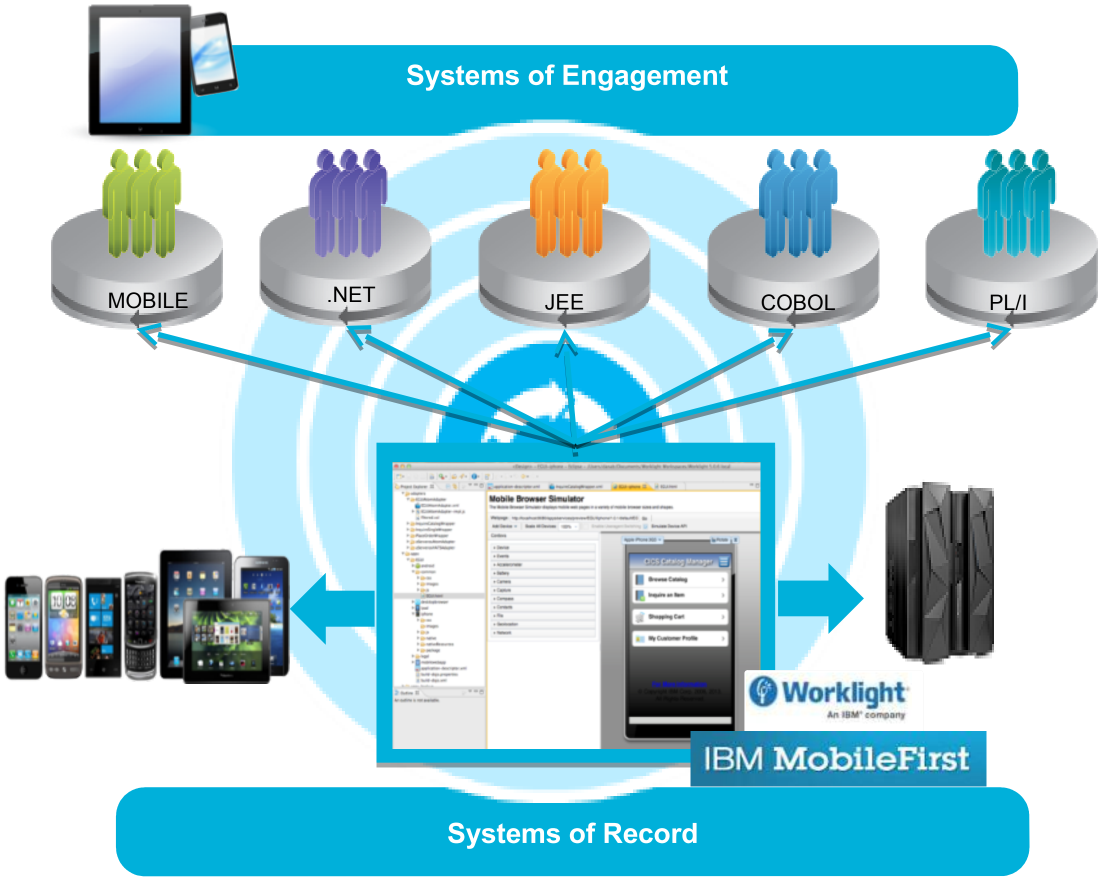
1.5x performance gain for COBOL apps using packed decimal

30x performance gain for COBOL stmts with SIMD instructions

Results based on internal IBM lab measurements. Results for specific applications will vary, depending on the source code, the compiler options specified, and other factors

Modern multi-platform developer and team tools

✓ Rational Developer Enterprise Edition (RDz) – modern IDE

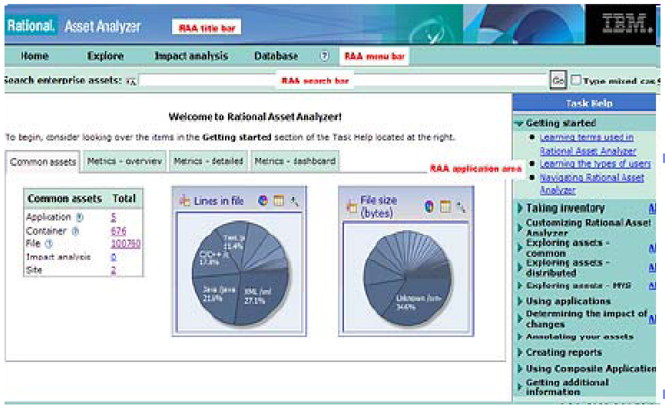


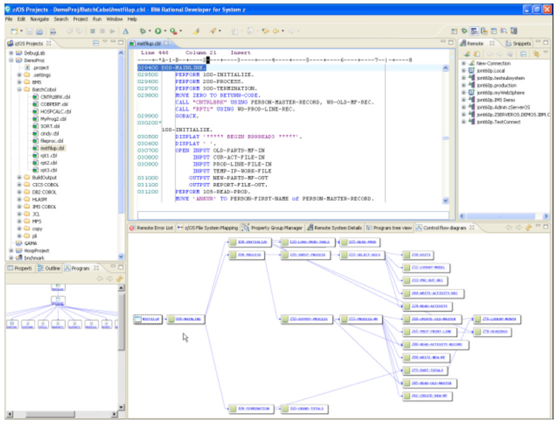
Key practices:

- MVP
- Dedicated Teams
- Loosely Coupled Arch.
- Minimizing Hand-offs Maximizing Flow
- Small Batch Delivery
- Transparency
- Eliminate Overhead
- Automate Testing

Revitalize

Analyze and Understand Application

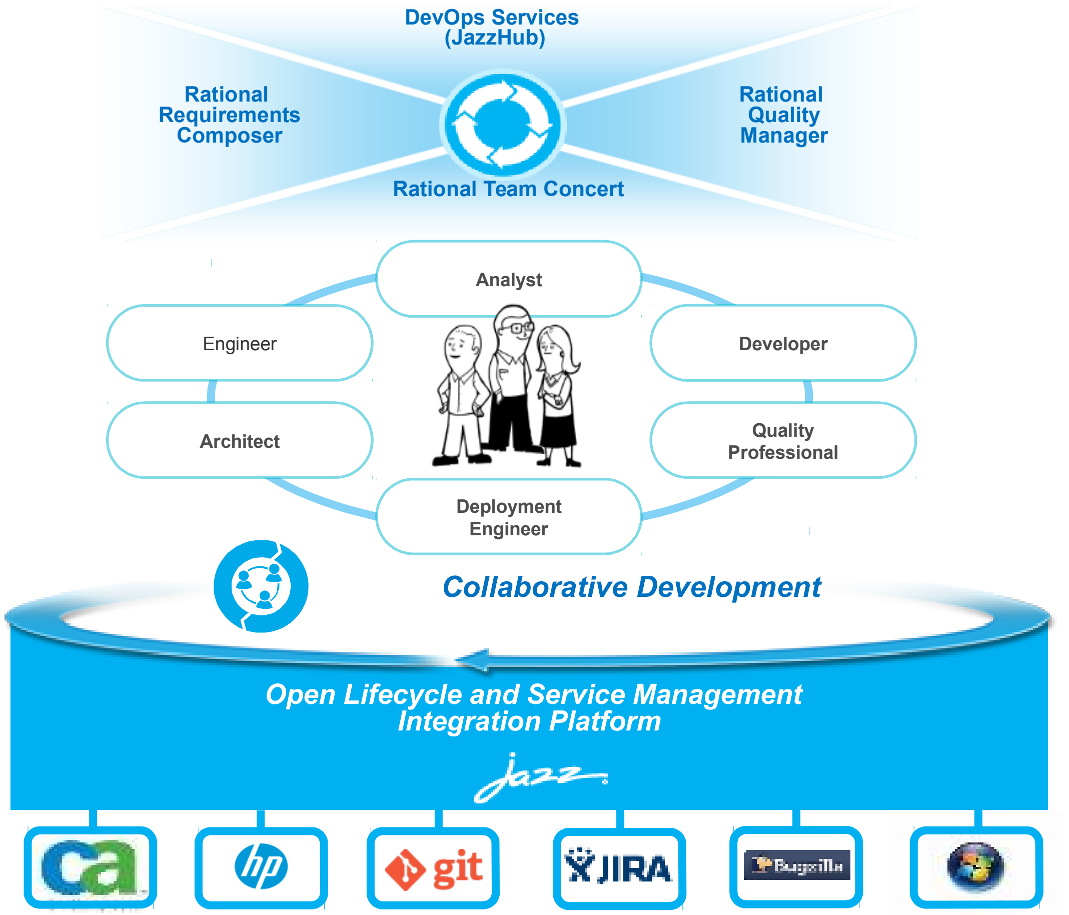

Rational. Asset Analyzer
RAA title bar
Home
Explore
Impact analysis
Database
RAA menu bar
Search enterprise assets:
RAA search bar
Go
Type mixed case
Task Help
Getting started
Learning terms used in Rational Asset Analyzer
Learning the types of users
Navigating Rational Asset Analyzer
Taking inventory
Customizing Rational Asset Analyzer
Exploring assets - common
Exploring assets - distributed
Exploring assets - MVS
Using applications
Determining the impact of changes
Annotating your assets
Creating reports
Using Composite Application
Getting additional information
Welcome to Rational Asset Analyzer!
To begin, consider looking over the items in the Getting started section of the Task Help located at the right.
Common assets
Metrics - overview
Metrics - detailed
Metrics - dashboard
RAA application area
Common assets
Total
Application
Container
File
Impact analysis
Site
Lines in file
File size (bytes)



- **Use Rational Asset Analyzer to quickly understand flow and relationships across the enterprise even with little or no documentation**
  – Analyze, understand, and navigate complex application source code, including COBOL, PL/I, Assembler, C/C++, Java/JEE, etc…
- **Reduce time to market & risk of resource shortage by understanding the impact of change, upfront**
  – Understand source code complexity/fragility
  – Analyze impact of potential code changes or database changes
  – Find "dead code" for deletion from source base
- **Choose from two user interfaces for ease of access and use**
  – Integration with Rational Developer for System z for IDE users
  – Browser-based user interface for dashboard and complex query construction
- **Supports enhanced usage Scenarios**
  – COBOL Business Rule identification and capture
    - Extend RAA "vocabulary" to map business terms and properties to those used by developers
    - Leverage RAA's capabilities to find where rules are encoded in the COBOL source
    - Export results in formats consistent with WODM BRMS technologies
  – Healthcare Industry ICD Migrations

Modern multi-platform developer and team tools

- ✓ Rational Team Concert Enterprise Edition (RTCee) – collaborative team environment across platforms and the lifecycle

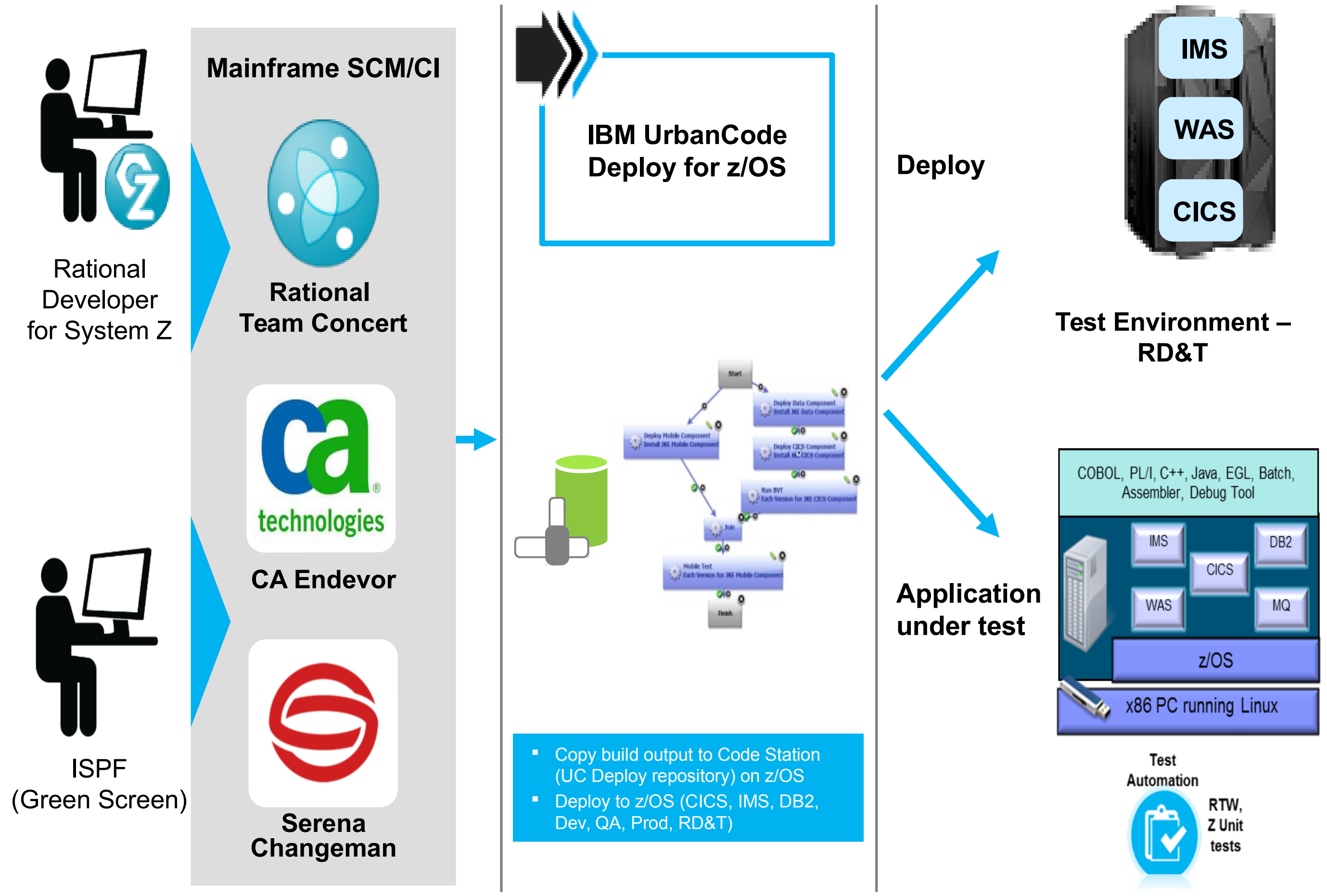


Key practices:

- •MVP
- •Dedicated Teams
- •Loosely Coupled Arch.
- •Minimizing Hand-offs
- •Maximizing Flow
- •Small Batch Delivery
- •Transparency
- •Eliminate Overhead
- •Automate Testing

Automated deployment and configuration

✓ UrbanCode Deploy – multi-platform applications and middleware



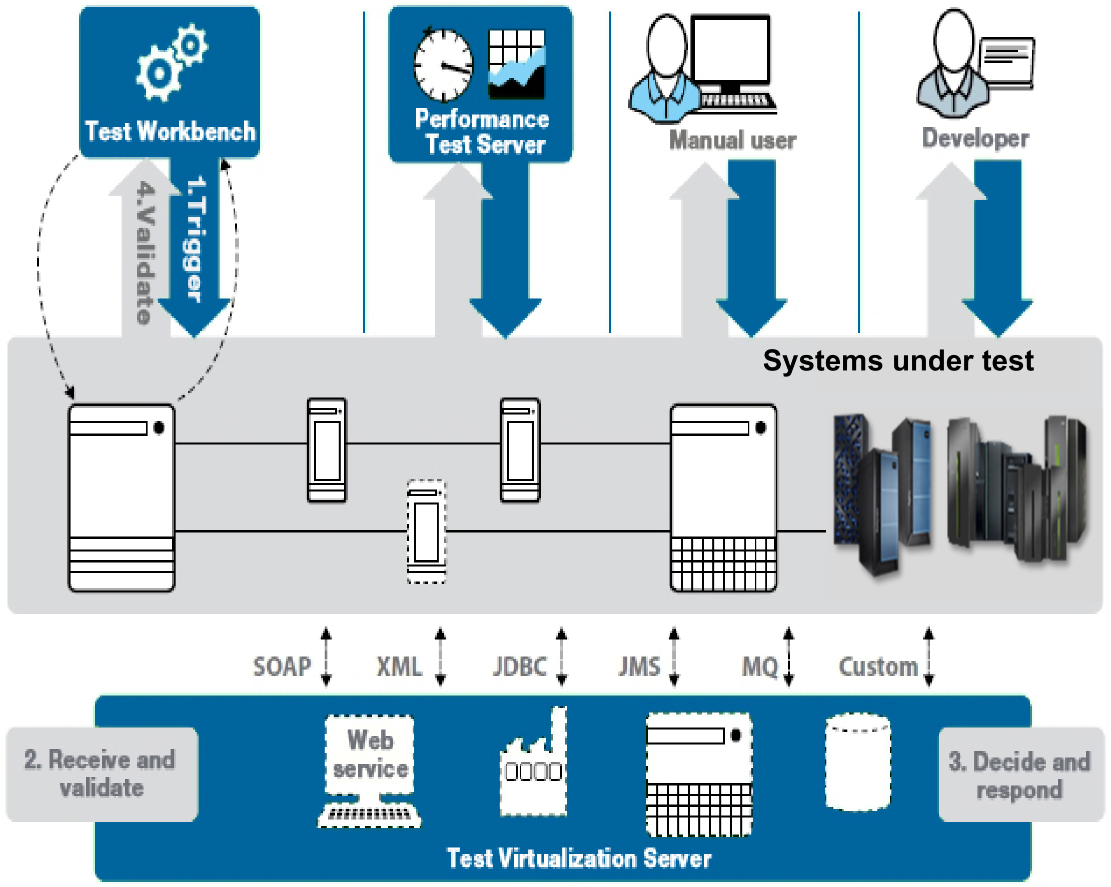
- Copy build output to Code Station (UC Deploy repository) on z/OS
- Deploy to z/OS (CICS, IMS, DB2, Dev, QA, Prod, RD&T)

Key practices:

- MVP
- Dedicated Teams
- Loosely Coupled Arch.
- Minimizing Hand-offs Maximizing Flow
- Small Batch Delivery
- Transparency
- Eliminate Overhead
- Automate Testing

Automated testing and virtualized services

✓ Rational Test Workbench – automated testing of all aspects of the product

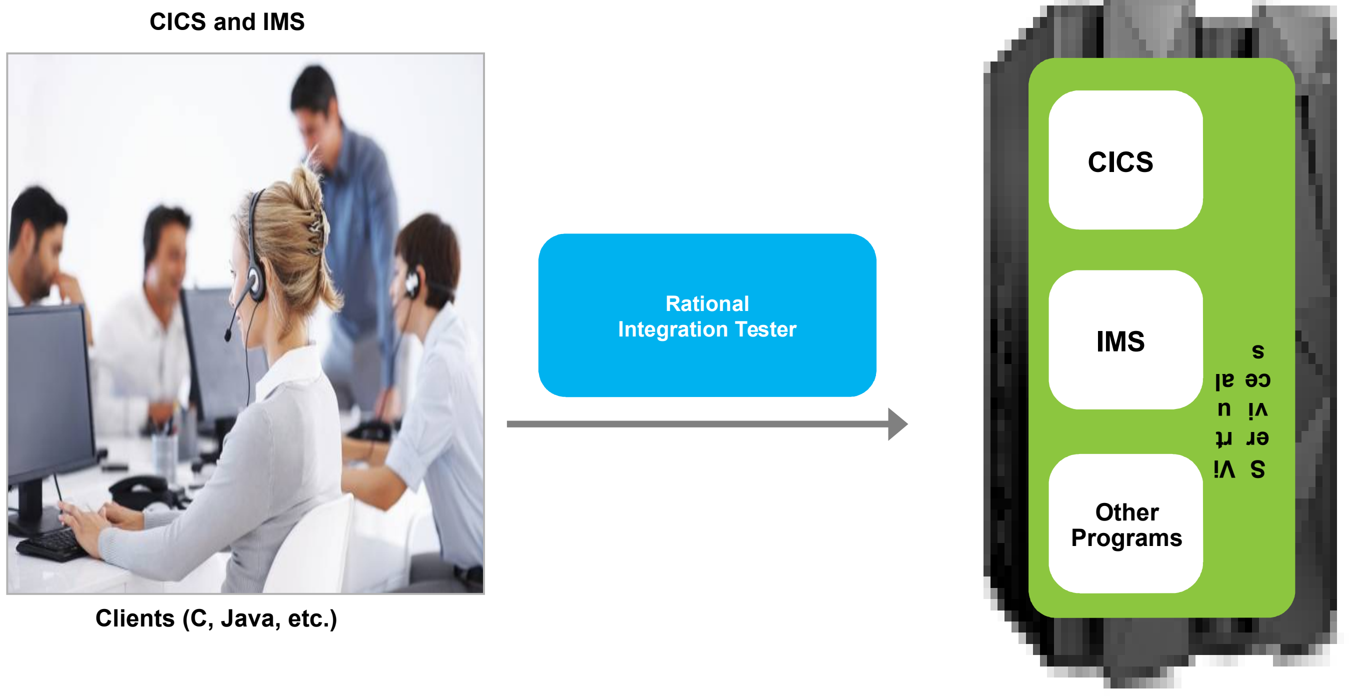


Key practices:

- •MVP
- •Dedicated Teams
- •Loosely Coupled Arch.
- •Minimizing Hand-offs Maximizing Flow
- •Small Batch Delivery
- •Transparency
- •Eliminate Overhead
- •Automate Testing

Automated testing and virtualized services

✓ Rational Test Workbench – virtual testing of systems and middleware

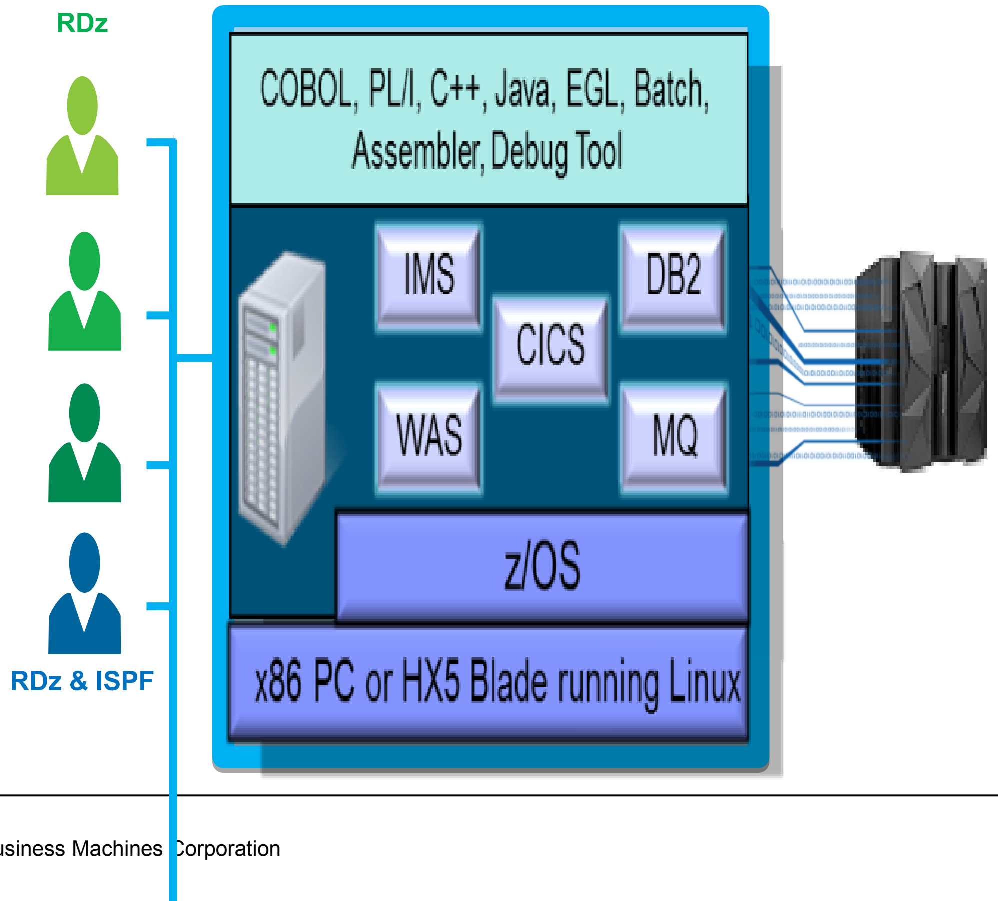


Key practices:

- MVP
- Dedicated Teams
- Loosely Coupled Arch.
- Minimizing Hand-offs Maximizing Flow
- Small Batch Delivery
- Transparency
- Eliminate Overhead
- Automate Testing

Testing off the mainframe

- ✓ Rational Development and Test Environment for System z – test z/OS software on Intel platforms without using z System hardware


RDz
COBOL, PL/I, C++, Java, EGL, Batch, Assembler, Debug Tool
IMS
DB2
CICS
WAS
MQ
z/OS
x86 PC or HX5 Blade running Linux
RDz & ISPF

Key practices:

- MVP
- Dedicated Teams
- Loosely Coupled Arch.
- Minimizing Hand-offs
- Maximizing Flow
- Small Batch Delivery
- Transparency
- Eliminate Overhead
- Automate Testing

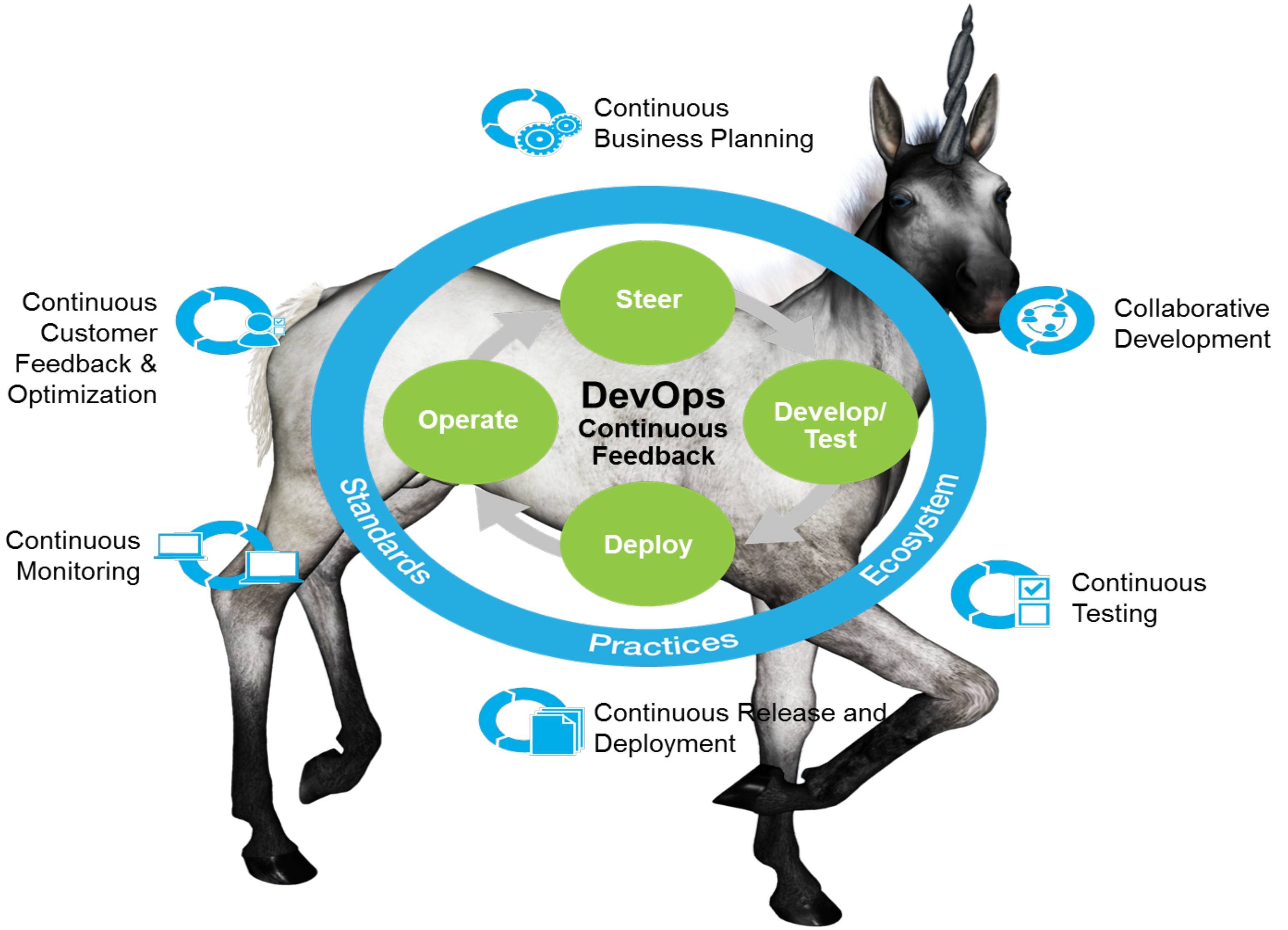
Build and deploy in small batches[1]

While not specific to a product, this is a critical best practice

- ✓ Reduces project risk
- ✓ Encourages automation
- ✓ Simplifies problem determination
- ✓ Speeds up feedback – “reduces queue size”
- ✓ Improves flow
- ✓ Reduces cycle time
- ✓ Increases efficiency
- ✓ Lowers overhead
- ✓ Improves project visibility
- ✓ Encourages decoupled architectures

[1] http://dev2ops.org/2012/03/devops-lessons-from-lean-small-batches-improve-flow/

Delivering on the DevOps promise


Continuous Business Planning
Collaborative Development
Continuous Testing
Continuous Release and Deployment
Continuous Monitoring
Continuous Customer Feedback & Optimization
Steer
Develop/ Test
Deploy
Operate
DevOps Continuous Feedback
Standards
Practices
Ecosystem

Continuously delivering you more value

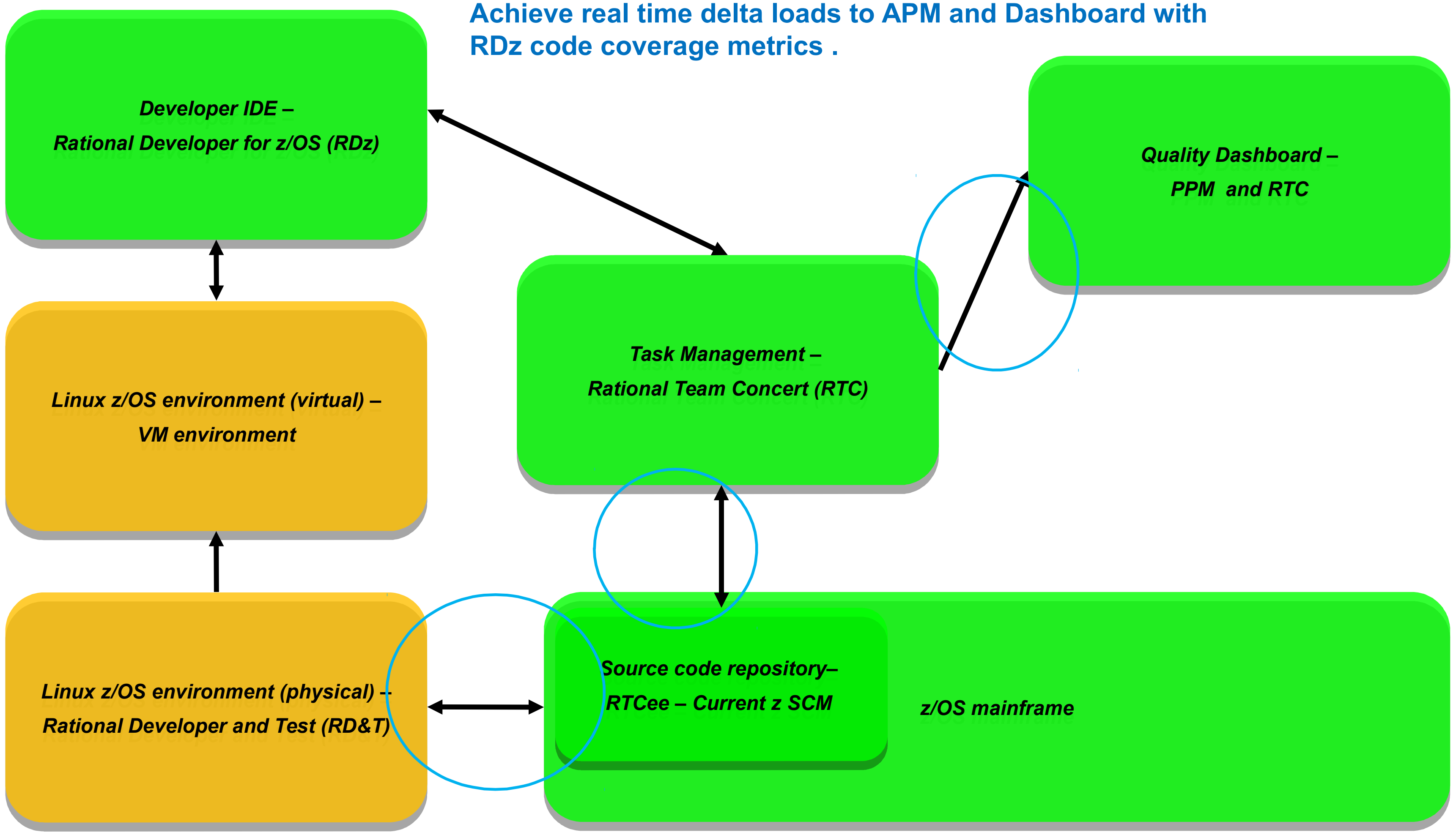
Key new capabilities

Product	What's New?
Rational Developer for System z	Adds zUnit test capability to test COBOL and PL/I apps at a module level, including ability to drive unit tests for continuous integration builds
Collaborative Lifecycle Management as a Managed Service	Reduces cost with pay-for-use managed services with 99% (SLO) availability
Rational Development & Test for System z	Exploits latest middleware; now runs as a managed service reducing time to value and minimizing ongoing admin and capital expense
Rational Test Workbench	Virtualizes DB2 database access from CICS COBOL programs, tests/virtualizes CICS transactions over IPIC protocol, supports PL/I data structures
Urban Code Deploy	Enhances support for z/OS deployments with SMP/E install; supports JCL submission and job monitoring
Compilers	Exploits z13 and latest z middleware, gains up to 17+%[1] performance improvement with new optimizations in Enterprise COBOL; supports XL C/C++ compiler for Linux on z
PD TOOLS	Simplifies ordering with new PD TOOLS Modernization Solution Pack which bundles together the most commonly requested tools

Continuous Integration for System z development

Optional Physical architecture and key challenges

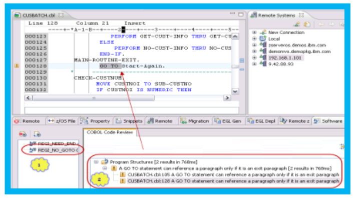
Achieve real time delta loads to APM and Dashboard with RDz code coverage metrics .


Developer IDE –
Rational Developer for z/OS (RDz)
Quality Dashboard –
PPM and RTC
Task Management –
Rational Team Concert (RTC)
Linux z/OS environment (virtual) –
VM environment
Linux z/OS environment (physical) –
Rational Developer and Test (RD&T)
Source code repository–
RTCee – Current z SCM
z/OS mainframe

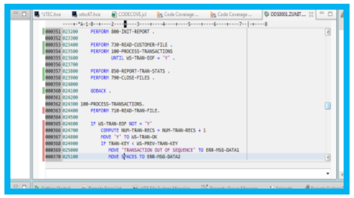
Environment mismatch:
High cost resource cost (DBA's, sysprogs) greater than lower cost resource saving (App programmer)

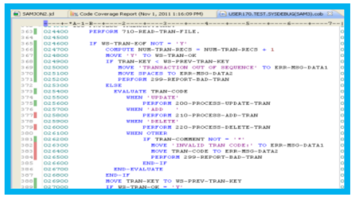
Retain current z SCM prior to migration as single source of truth whilst exposing code via RTC.
Dependency on IFO ALM.

Continuous Integration Process

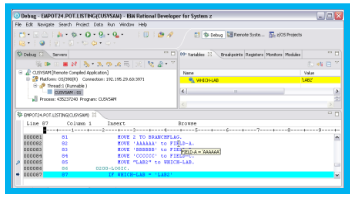


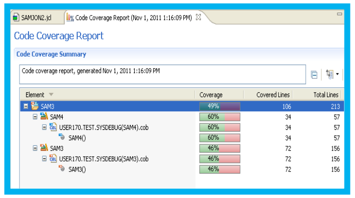
Mainframe 3rd Part SCM Link





Application Quality Feedback From Endevor





1 **Developer Checks Out Source Code/Quick Edit.**

2 **Make code change via RDz editor.**

3 **Check In / Quick Edit Save to Endevor using CARMA.**

4 **By default Endevor runs Compile and Link**

(Optional) Developer can additional select and run Sub Processors

5 **Automated tests triggered in Endevor via Sub Processors provide feedback on application quality to developer.**

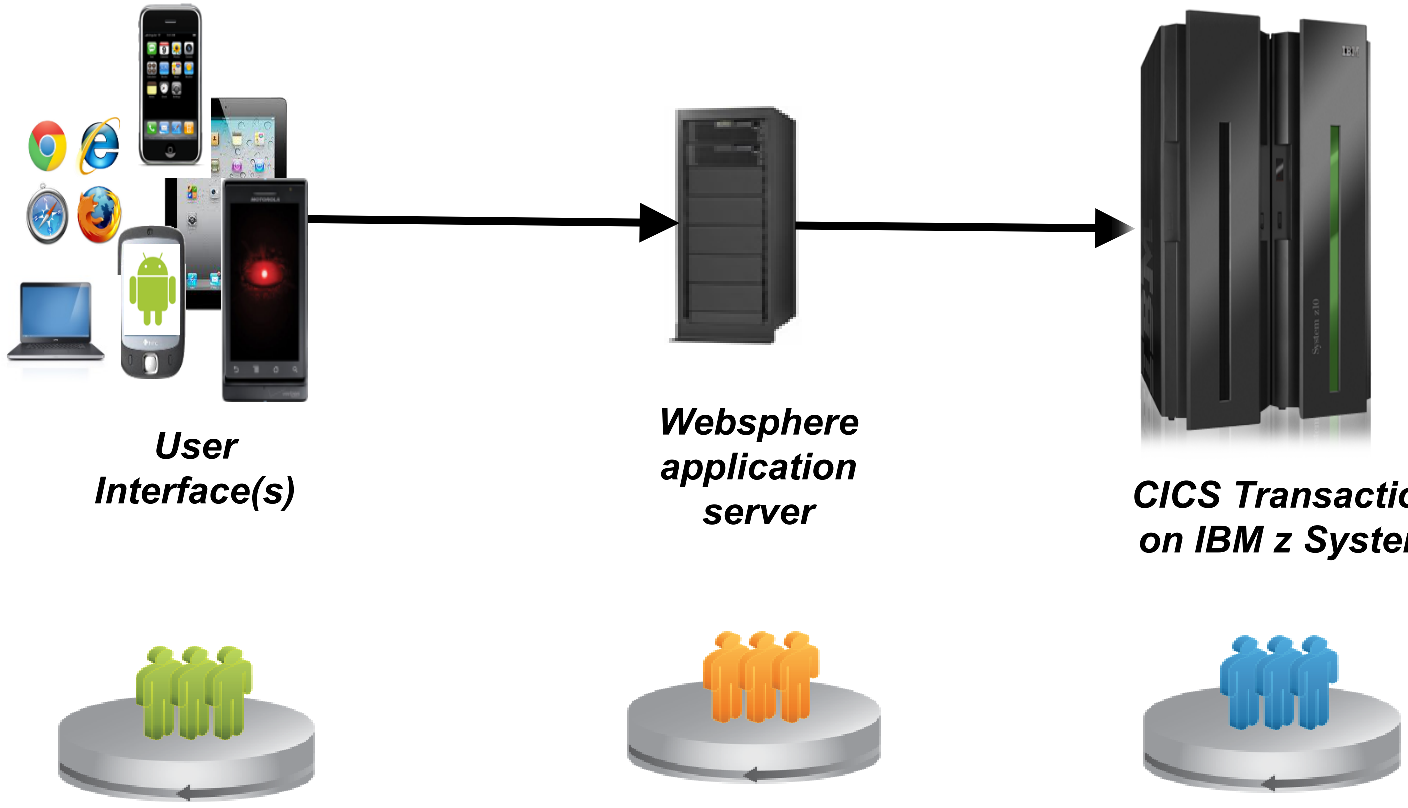
Compile and Link

Code Review *RUNCR*	Debugging *RUNDB*	Code Coverage *RUNCC*	Automated Unit Testing *RUNZU*

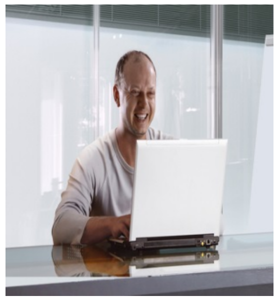
Sale and Solution Plays to help get statred

Backups and Solution Review

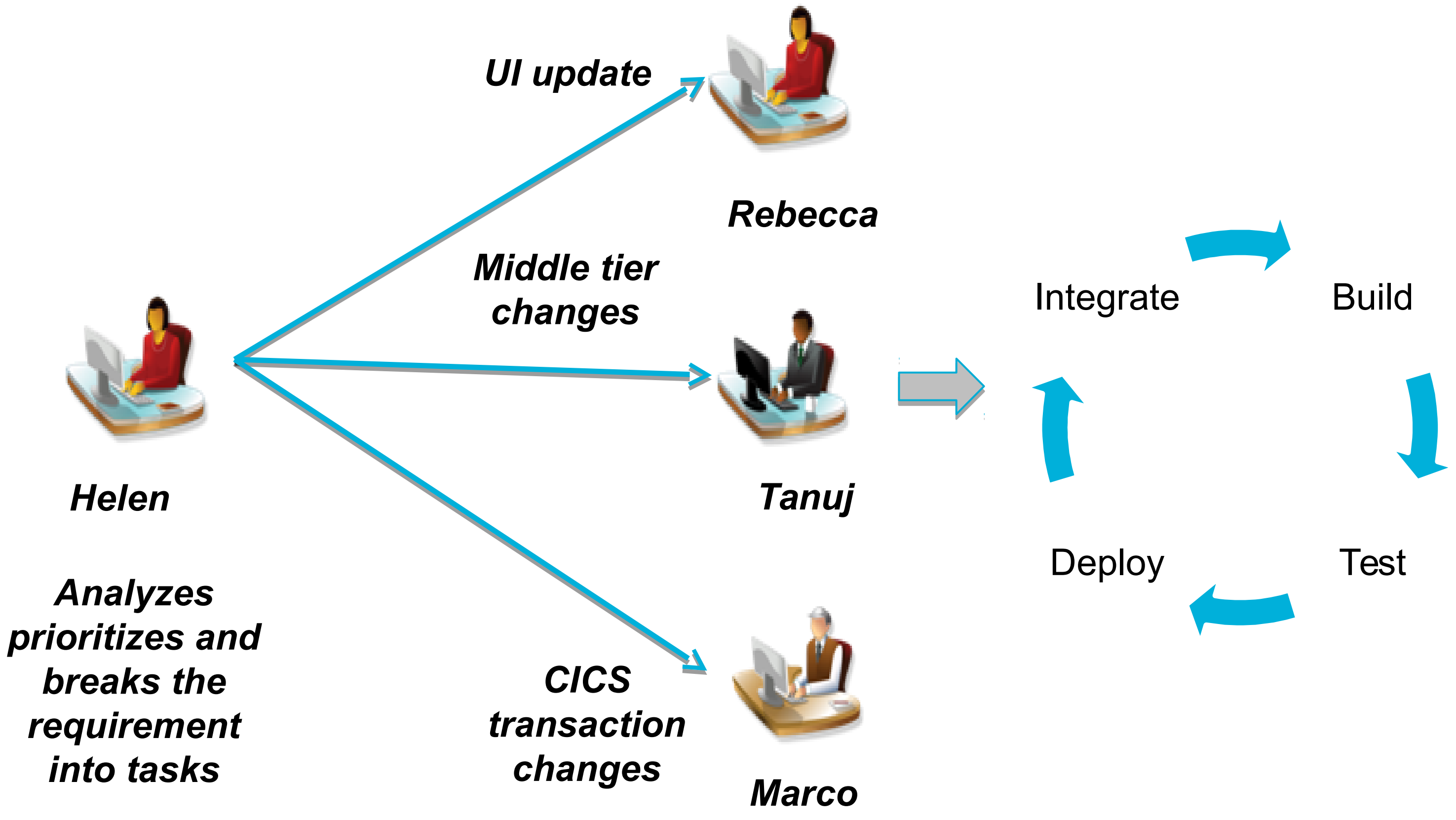
Application and Team Overview


User Interface(s)
Websphere application server
CICS Transaction on IBM z System

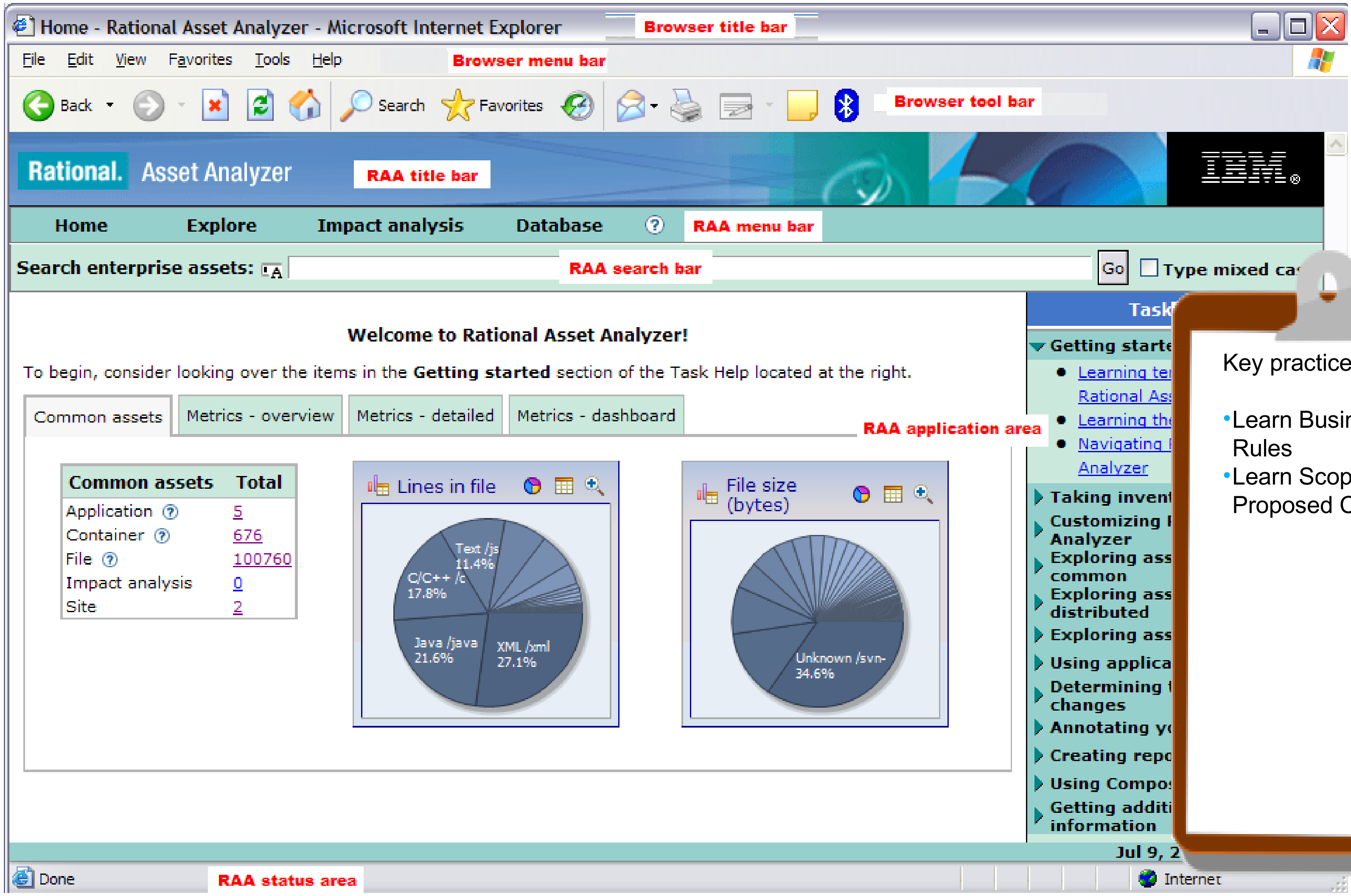
Scenario Overview



Client opens a requirement



Understand the Application Structure: RAA



Welcome to Rational Asset Analyzer!

To begin, consider looking over the items in the **Getting started** section of the Task Help located at the right.

Common assets | Metrics - overview | Metrics - detailed | Metrics - dashboard

Common assets	Total
Application	5
Container	676
File	100760
Impact analysis	0
Site	2

Task
- Getting starte
  - Learning ter Rational Ass
  - Learning the
  - Navigating Analyzer
- Taking invent
- Customizing Analyzer
- Exploring ass common
- Exploring ass distributed
- Exploring ass
- Using applica
- Determining changes
- Annotating y
- Creating repo
- Using Compo
- Getting additi information

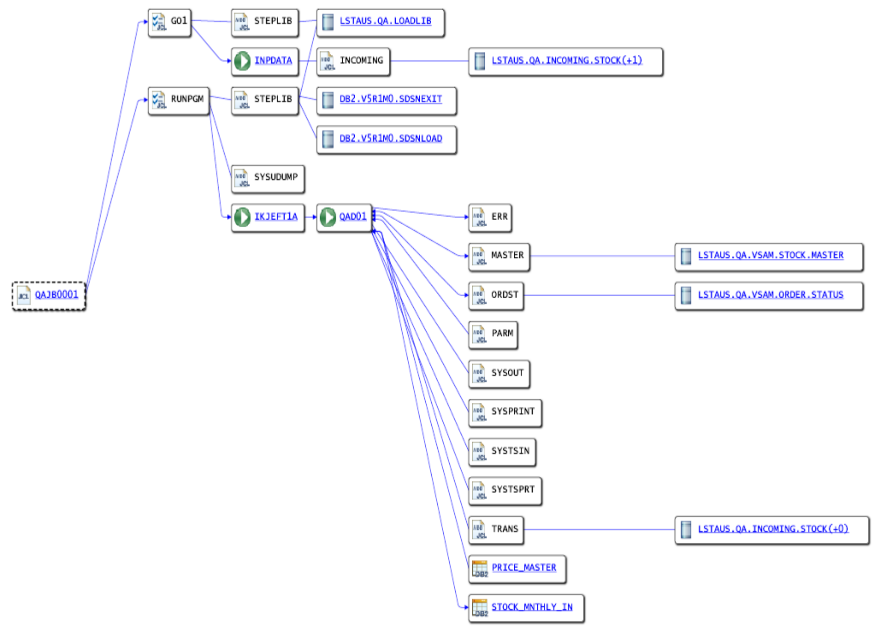
Key practices:

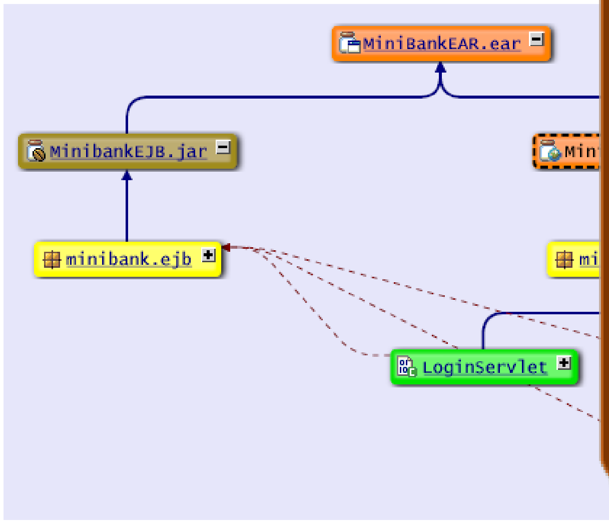
- Learn Business Rules
- Learn Scope of Proposed Changes

Helen: Analyst Exploring Application Structure

- Group artifacts into **user-defined groups** called Applications to limit scope to area of interest
- Use **various types of diagrams** for understanding how the application “hangs together”

- Use **annotations** to capture knowledge from SMEs e.g. Business function, description, etc.
- Create **user-defined relationships** for situations where relationships cannot be determined through static analysis
- Perform **enterprise-level keyword searches**

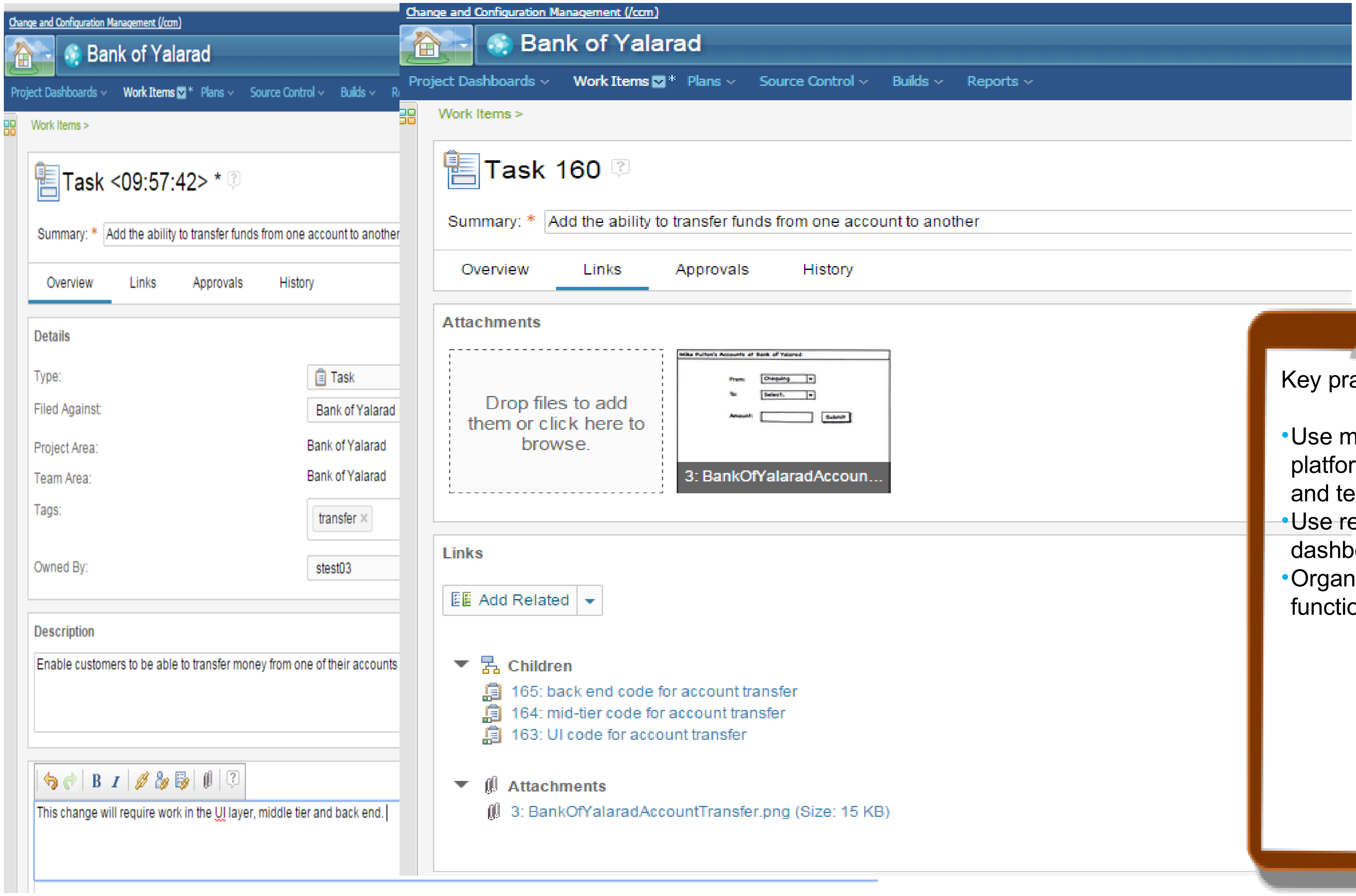




Key practices:

- Decompose work into small deliverables
- Identify parallel tasks

Enhancement creation and Task breakdown: RTC



Change and Configuration Management (/ccm)

Bank of Yalarad

Project Dashboards | Work Items * | Plans | Source Control | Builds | R

Work Items >

Task <09:57:42> *

Summary: * Add the ability to transfer funds from one account to another

Overview | Links | Approvals | History

Details

Type: Task

Filed Against: Bank of Yalarad

Project Area: Bank of Yalarad

Team Area: Bank of Yalarad

Tags: transfer

Owned By: stest03

Description

Enable customers to be able to transfer money from one of their accounts

This change will require work in the UI layer, middle tier and back end.

Change and Configuration Management (/ccm)

Bank of Yalarad

Project Dashboards | Work Items * | Plans | Source Control | Builds | Reports

Work Items >

Task 160

Summary: * Add the ability to transfer funds from one account to another

Overview | Links | Approvals | History

Attachments

Drop files to add them or click here to browse.

3: BankOfYalaradAccoun...

Links

Add Related

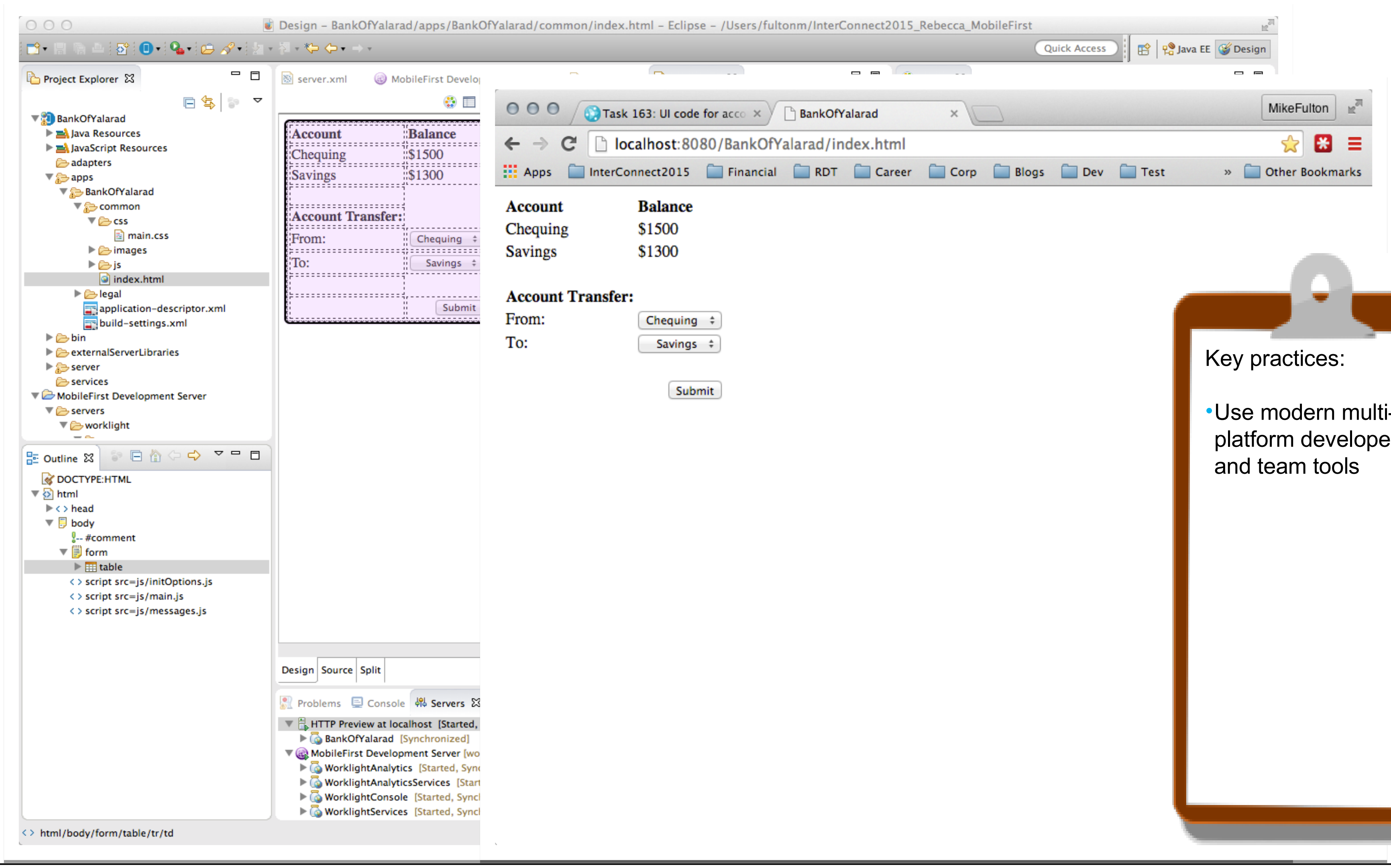
Children
- 165: back end code for account transfer
- 164: mid-tier code for account transfer
- 163: UI code for account transfer

Attachments
- 3: BankOfYalaradAccountTransfer.png (Size: 15 KB)

Key practices:

- Use modern multi-platform developer and team tools
- Use real-time dashboards
- Organize with cross functional teams

Rebecca: UI Developer using IBM MobileFirst Studio



```
BankOfYalarad
  Java Resources
  JavaScript Resources
  adapters
  apps
    BankOfYalarad
      common
        css
          main.css
        images
        js
        index.html
      legal
      application-descriptor.xml
      build-settings.xml
  bin
  externalServerLibraries
  server
  services
MobileFirst Development Server
  servers
    worklight
```

```
DOCTYPE:HTML
html
  head
  body
    #comment
    form
      table
    script src=js/initOptions.js
    script src=js/main.js
    script src=js/messages.js
```

html/body/form/table/tr/td

```
HTTP Preview at localhost [Started,
  BankOfYalarad [Synchronized]
MobileFirst Development Server [wo
  WorklightAnalytics [Started, Sync
  WorklightAnalyticsServices [Start
  WorklightConsole [Started, Sync
  WorklightServices [Started, Sync
```

Account	Balance
Chequing	$1500
Savings	$1300

Account Transfer:

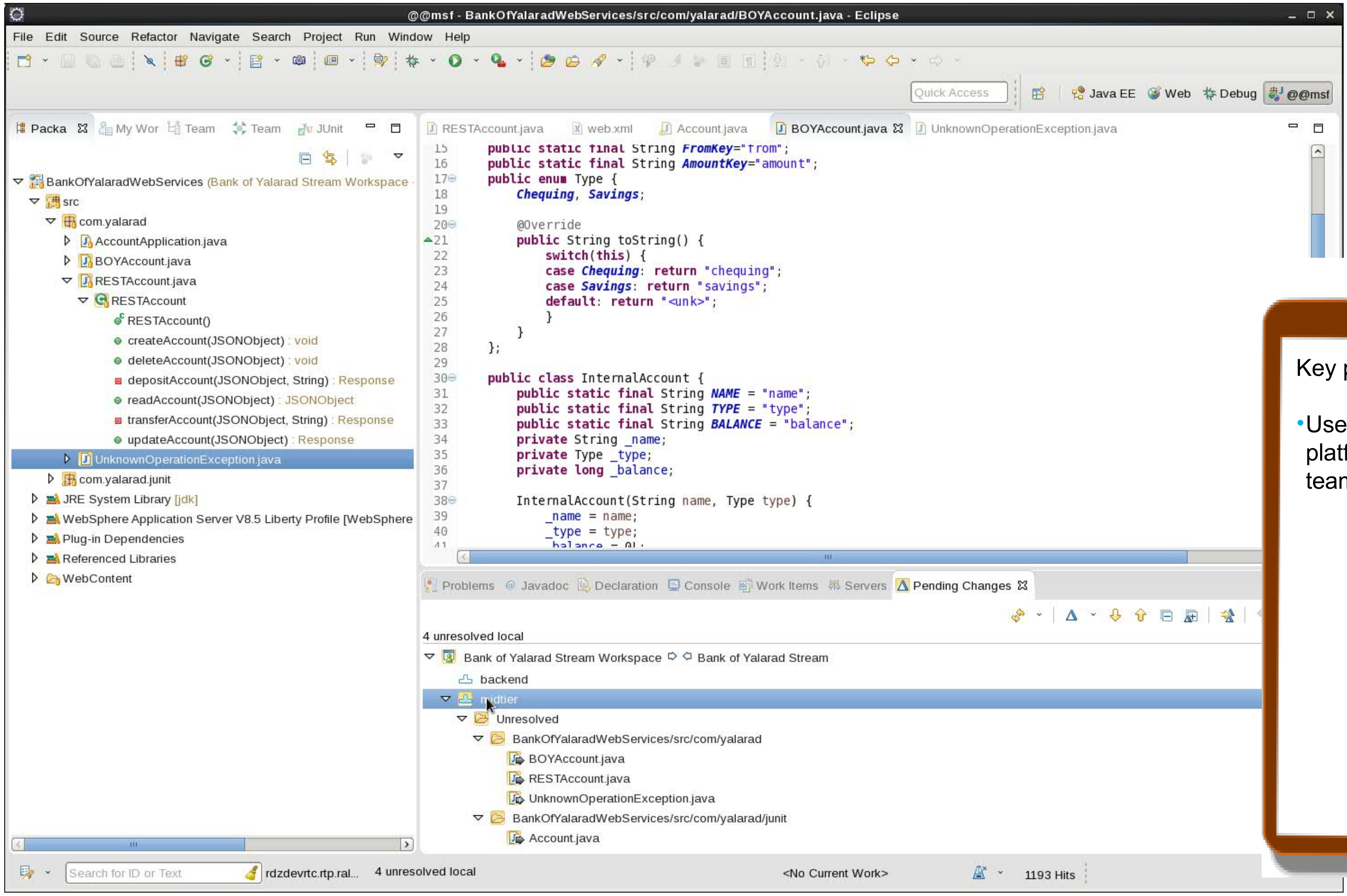
From:	Chequing
To:	Savings

Submit

Key practices:

- Use modern multi-platform developer and team tools

Tanuj: Mid-Tier Developer using RAD for WebSphere



```
BankOfYalaradWebServices (Bank of Yalarad Stream Workspace
  src
    com.yalarad
      AccountApplication.java
      BOYAccount.java
      RESTAccount.java
        RESTAccount
          RESTAccount()
          createAccount(JSONObject) : void
          deleteAccount(JSONObject) : void
          depositAccount(JSONObject, String) : Response
          readAccount(JSONObject) : JSONObject
          transferAccount(JSONObject, String) : Response
          updateAccount(JSONObject) : Response
      UnknownOperationException.java
    com.yalarad.junit
  JRE System Library [jdk]
  WebSphere Application Server V8.5 Liberty Profile [WebSphere
  Plug-in Dependencies
  Referenced Libraries
  WebContent
```

```java
    public static final String FromKey="from";
    public static final String AmountKey="amount";
    public enum Type {
        Chequing, Savings;

        @Override
        public String toString() {
            switch(this) {
            case Chequing: return "chequing";
            case Savings: return "savings";
            default: return "<unk>";
            }
        }
    };

    public class InternalAccount {
        public static final String NAME = "name";
        public static final String TYPE = "type";
        public static final String BALANCE = "balance";
        private String _name;
        private Type _type;
        private long _balance;

        InternalAccount(String name, Type type) {
            _name = name;
            _type = type;
            _balance = 0L;
```

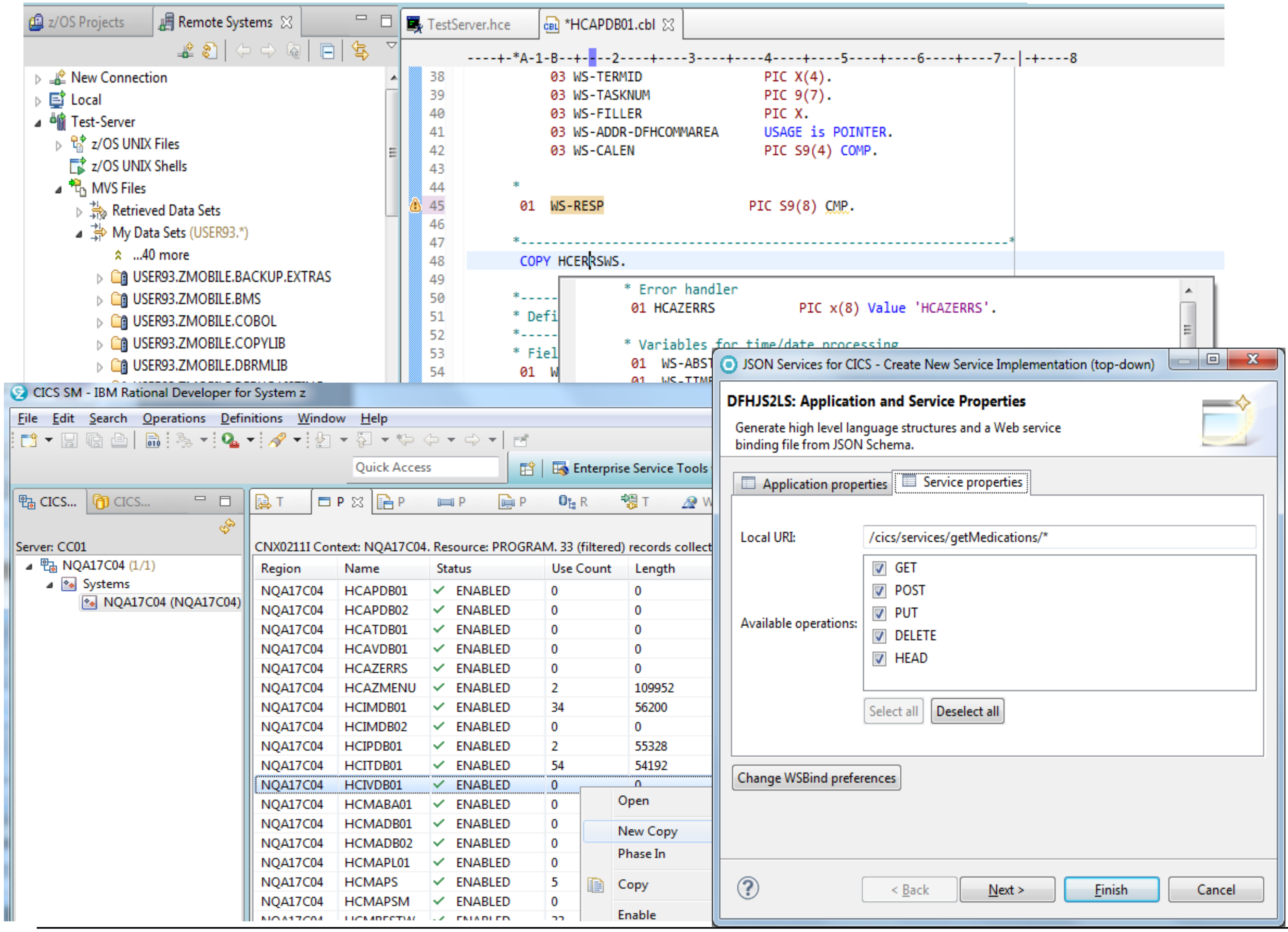
```
4 unresolved local
Bank of Yalarad Stream Workspace ⇨ ⇦ Bank of Yalarad Stream
  backend
  midtier
    Unresolved
      BankOfYalaradWebServices/src/com/yalarad
        BOYAccount.java
        RESTAccount.java
        UnknownOperationException.java
      BankOfYalaradWebServices/src/com/yalarad/junit
        Account.java
```

```
Search for ID or Text   rdzdevrtc.rtp.ral...   4 unresolved local   <No Current Work>   1193 Hits
```

Key practices:

- Use modern multi-platform developer and team tools

Marco: CICS Developer using RDz



```
----+-*A-1-B--+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
           03 WS-TERMID              PIC X(4).
           03 WS-TASKNUM             PIC 9(7).
           03 WS-FILLER              PIC X.
           03 WS-ADDR-DFHCOMMAREA    USAGE is POINTER.
           03 WS-CALEN               PIC S9(4) COMP.

       *
        01  WS-RESP                PIC S9(8) CMP.

       *-----------------------------------------------------------------*
        COPY HCERRSWS.
       *-----
       * Defi
       *-----
       * Fiel
        01  W
```

```
      * Error handler
       01 HCAZERRS           PIC x(8) Value 'HCAZERRS'.

      * Variables for time/date processing
       01  WS-ABST
       01  WS-TIME
```

CNX0211I Context: NQA17C04. Resource: PROGRAM. 33 (filtered) records collect

Region	Name	Status	Use Count	Length
NQA17C04	HCAPDB01	✓ ENABLED	0	0
NQA17C04	HCAPDB02	✓ ENABLED	0	0
NQA17C04	HCATDB01	✓ ENABLED	0	0
NQA17C04	HCAVDB01	✓ ENABLED	0	0
NQA17C04	HCAZERRS	✓ ENABLED	0	0
NQA17C04	HCAZMENU	✓ ENABLED	2	109952
NQA17C04	HCIMDB01	✓ ENABLED	34	56200
NQA17C04	HCIMDB02	✓ ENABLED	0	0
NQA17C04	HCIPDB01	✓ ENABLED	2	55328
NQA17C04	HCITDB01	✓ ENABLED	54	54192
NQA17C04	HCIVDB01	✓ ENABLED	0	0
NQA17C04	HCMABA01	✓ ENABLED	0	
NQA17C04	HCMADB01	✓ ENABLED	0	
NQA17C04	HCMADB02	✓ ENABLED	0	
NQA17C04	HCMAPL01	✓ ENABLED	0	
NQA17C04	HCMAPS	✓ ENABLED	5	
NQA17C04	HCMAPSM	✓ ENABLED	0	

JSON Services for CICS - Create New Service Implementation (top-down)

DFHJS2LS: Application and Service Properties

Generate high level language structures and a Web service binding file from JSON Schema.

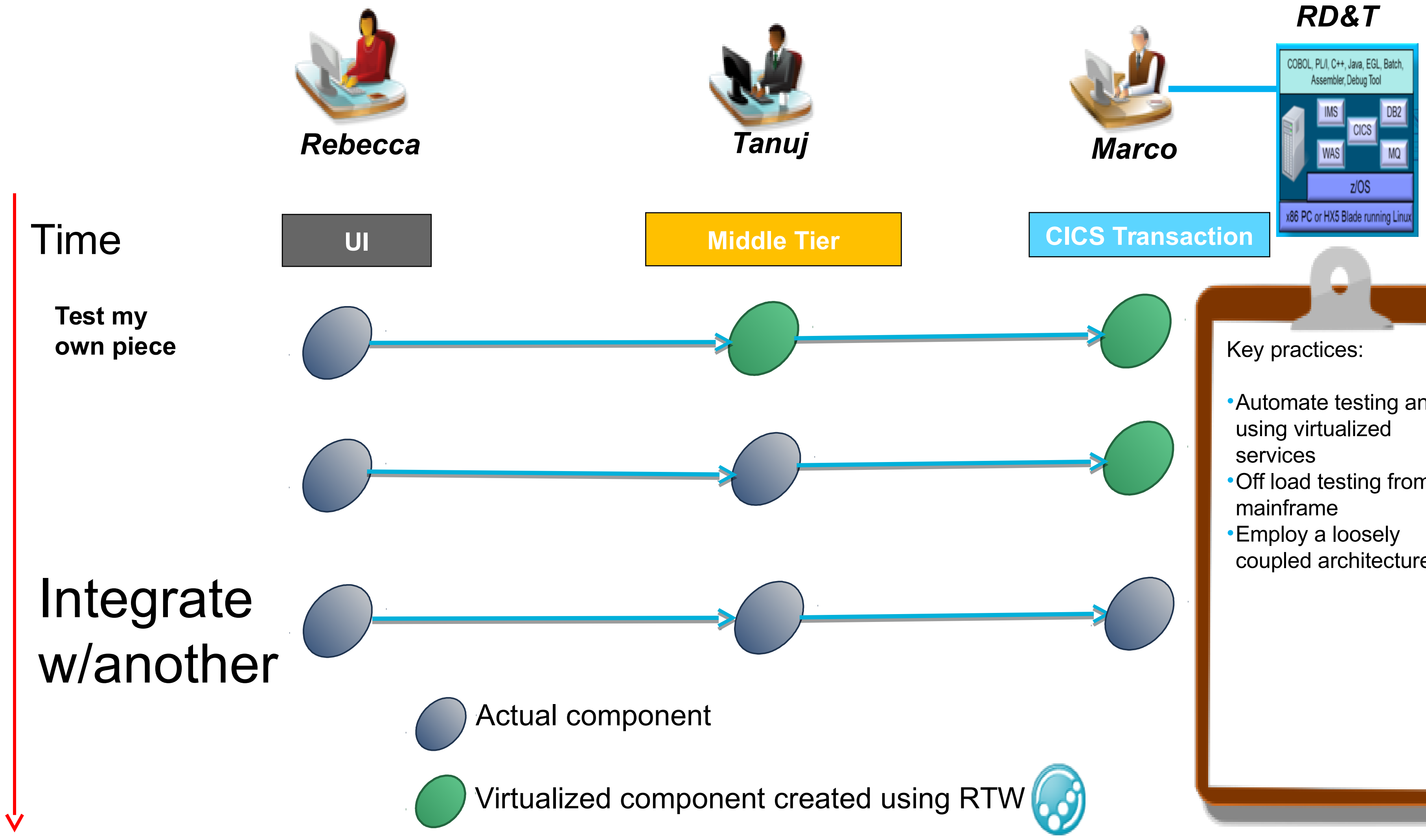
Local URI: /cics/services/getMedications/*

Available operations:
- [✓] GET
- [✓] POST
- [✓] PUT
- [✓] DELETE
- [✓] HEAD

Key practices:

- Use modern multi-platform developer and team tools

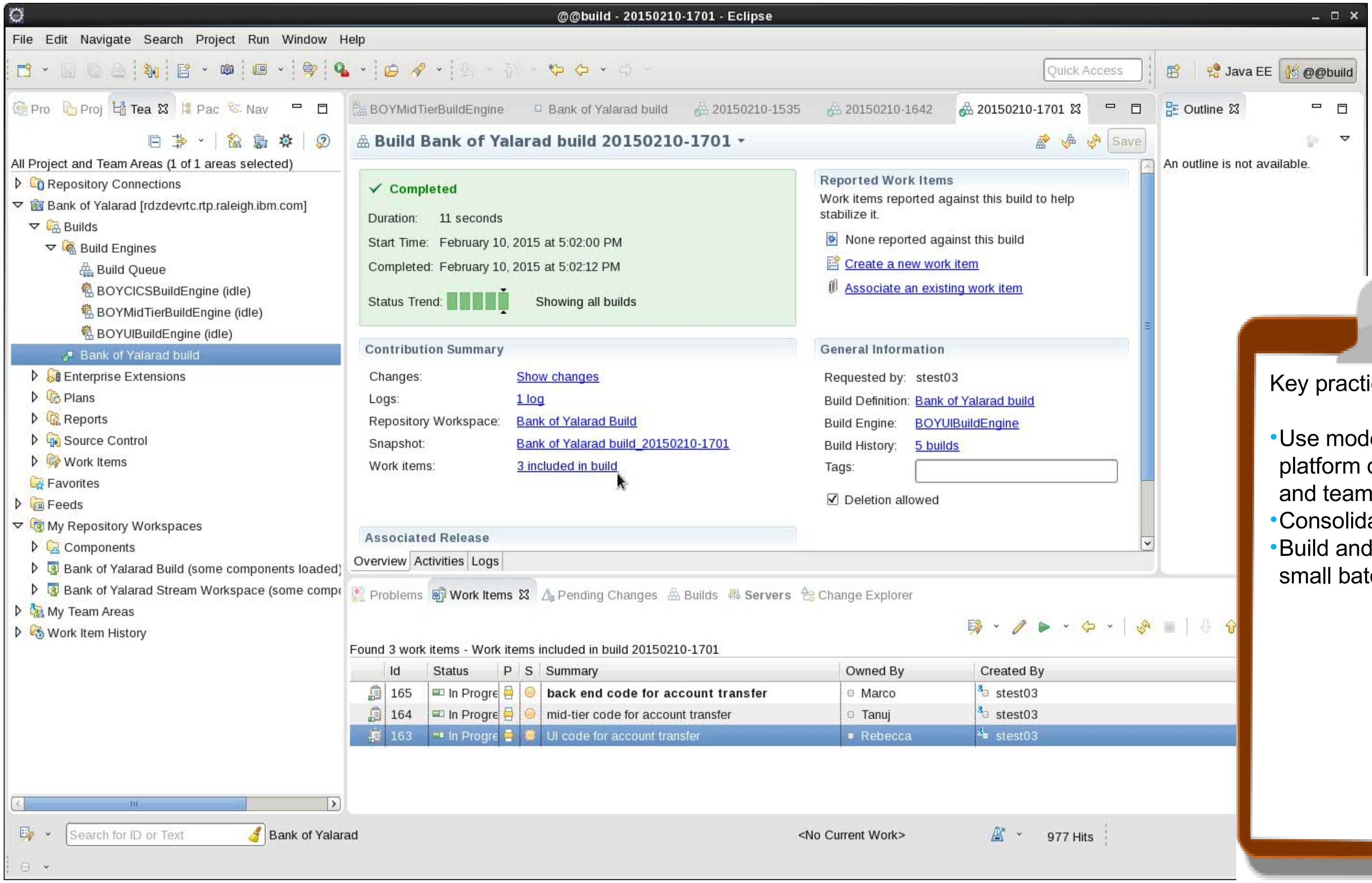
Continuous Integration Testing


RD&T
COBOL, PL/I, C++, Java, EGL, Batch, Assembler, Debug Tool
IMS
DB2
CICS
WAS
MQ
z/OS
x86 PC or HX5 Blade running Linux
Rebecca
Tanuj
Marco
Time
UI
Middle Tier
CICS Transaction
Test my own piece
Integrate w/another
Actual component
Virtualized component created using RTW

Key practices:

- Automate testing and using virtualized services
- Off load testing from mainframe
- Employ a loosely coupled architecture

Automated Build: Base Code Built with RTC



@@build - 20150210-1701 - Eclipse

File Edit Navigate Search Project Run Window Help

Build Bank of Yalarad build 20150210-1701

✓ Completed

Duration:	11 seconds
Start Time:	February 10, 2015 at 5:02:00 PM
Completed:	February 10, 2015 at 5:02:12 PM
Status Trend:	Showing all builds

Reported Work Items

Work items reported against this build to help stabilize it.

None reported against this build

Create a new work item

Associate an existing work item

Contribution Summary

Changes:	Show changes
Logs:	1 log
Repository Workspace:	Bank of Yalarad Build
Snapshot:	Bank of Yalarad build_20150210-1701
Work items:	3 included in build

General Information

Requested by:	stest03
Build Definition:	Bank of Yalarad build
Build Engine:	BOYUIBuildEngine
Build History:	5 builds
Tags:	

☑ Deletion allowed

Associated Release

All Project and Team Areas (1 of 1 areas selected)

- Repository Connections
- Bank of Yalarad [rdzdevrtc.rtp.raleigh.ibm.com]
  - Builds
    - Build Engines
      - Build Queue
      - BOYCICSBuildEngine (idle)
      - BOYMidTierBuildEngine (idle)
      - BOYUIBuildEngine (idle)
    - Bank of Yalarad build
  - Enterprise Extensions
  - Plans
  - Reports
  - Source Control
  - Work Items
- Favorites
- Feeds
- My Repository Workspaces
  - Components
  - Bank of Yalarad Build (some components loaded)
  - Bank of Yalarad Stream Workspace (some comp
- My Team Areas
- Work Item History

Found 3 work items - Work items included in build 20150210-1701

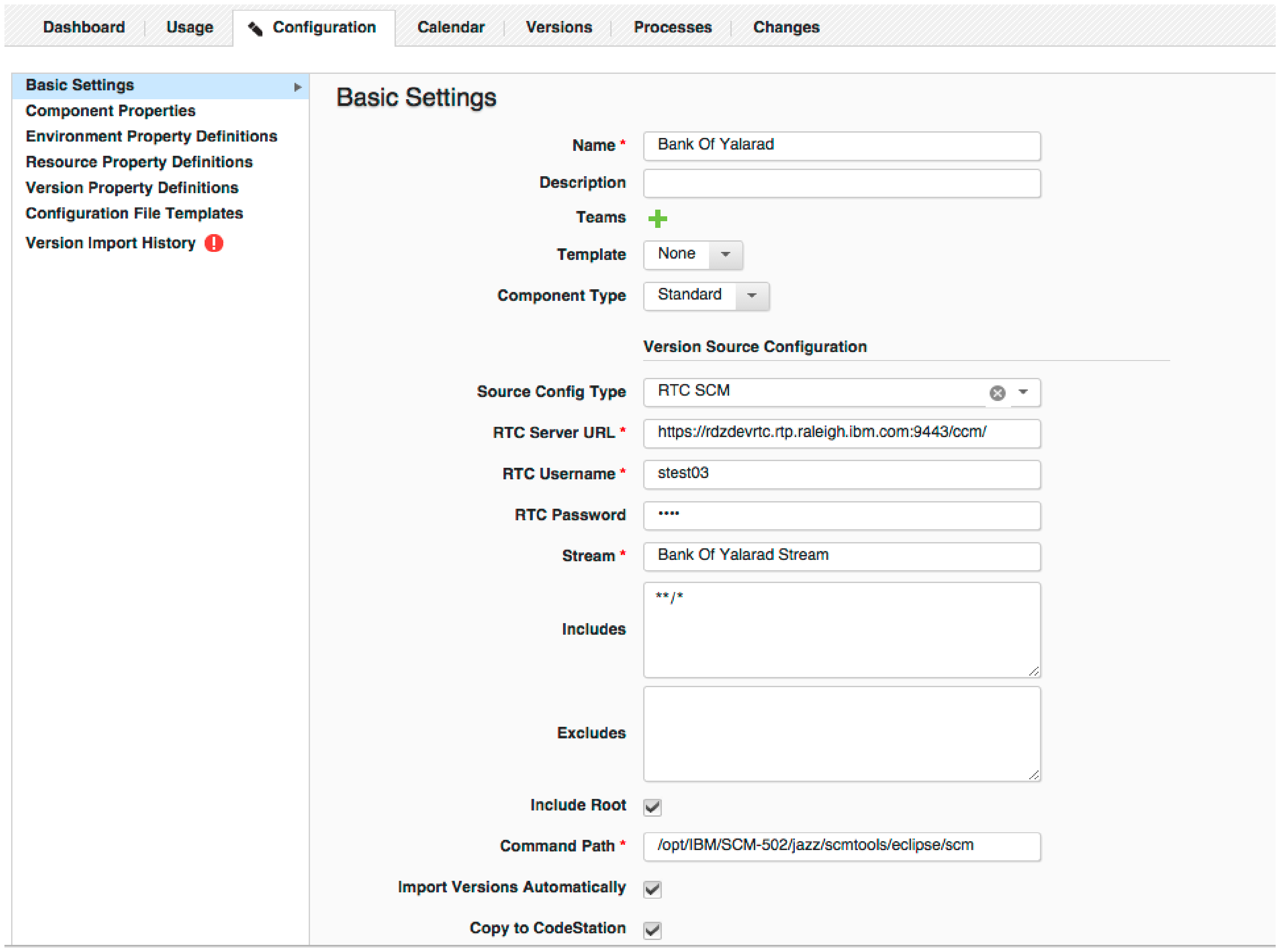
Id	Status	P	S	Summary	Owned By	Created By
165	In Progre			back end code for account transfer	Marco	stest03
164	In Progre			mid-tier code for account transfer	Tanuj	stest03
163	In Progre			UI code for account transfer	Rebecca	stest03

<No Current Work> 977 Hits

Key practices:

- Use modern multi-platform developer and team tools
- Consolidate SCMs
- Build and deploy in small batches

Automated Deployment from RTC to UCD


Dashboard
Usage
Configuration
Calendar
Versions
Processes
Changes
Basic Settings
Component Properties
Environment Property Definitions
Resource Property Definitions
Version Property Definitions
Configuration File Templates
Version Import History

Basic Settings

Name *	Bank Of Yalarad
Description	
Teams	
Template	None
Component Type	Standard

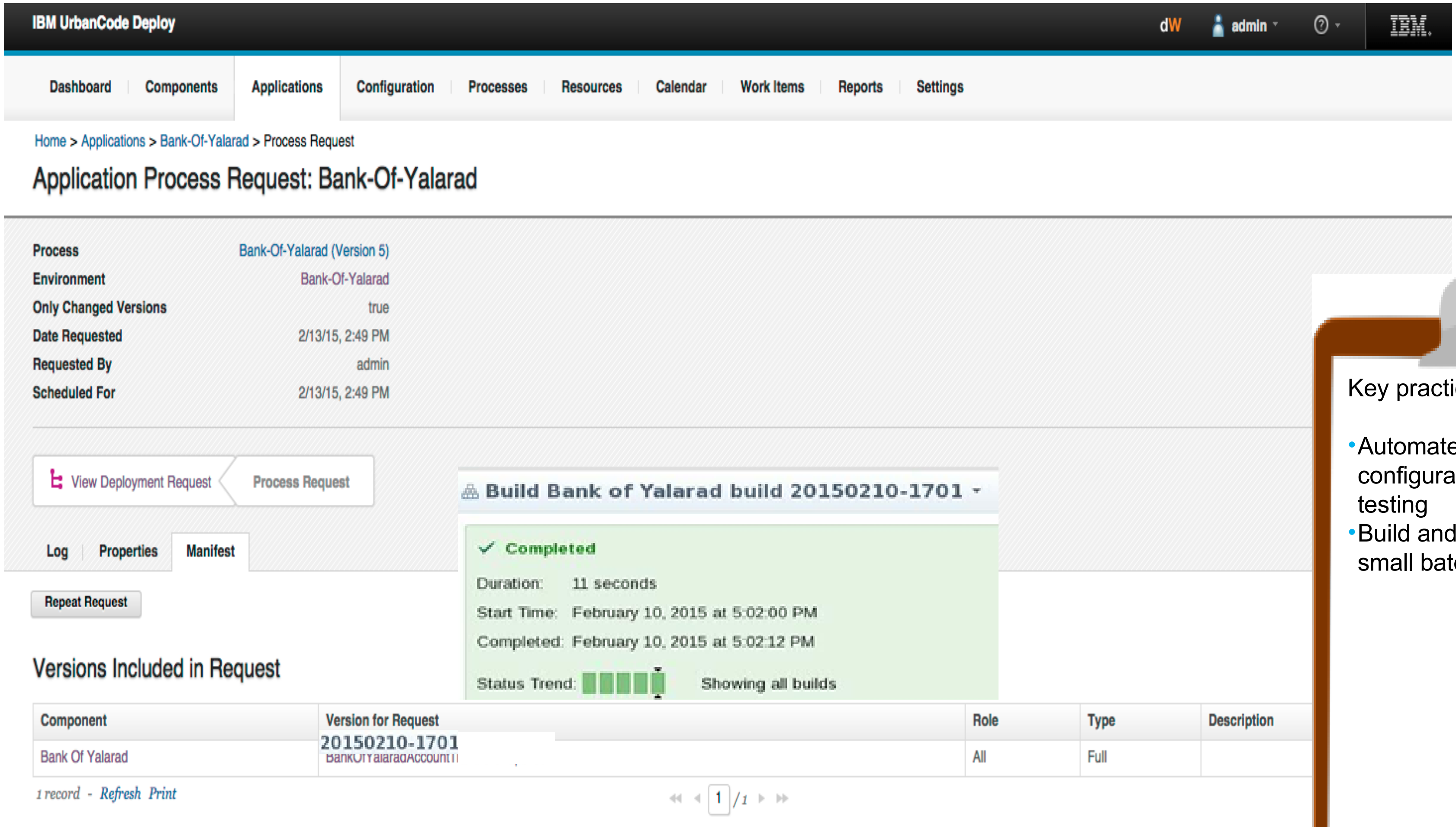
Version Source Configuration

Source Config Type	RTC SCM
RTC Server URL *	https://rdzdevrtc.rtp.raleigh.ibm.com:9443/ccm/
RTC Username *	stest03
RTC Password	••••
Stream *	Bank Of Yalarad Stream
Includes	**/*
Excludes	
Include Root	✓
Command Path *	/opt/IBM/SCM-502/jazz/scmtools/eclipse/scm
Import Versions Automatically	✓
Copy to CodeStation	✓

Key practices:

- Automate deployment, configuration, and testing
- Build and deploy in small batches

Traceability: Trace deploy to build with UCD



Home > Applications > Bank-Of-Yalarad > Process Request

Application Process Request: Bank-Of-Yalarad

Process	Bank-Of-Yalarad (Version 5)
Environment	Bank-Of-Yalarad
Only Changed Versions	true
Date Requested	2/13/15, 2:49 PM
Requested By	admin
Scheduled For	2/13/15, 2:49 PM

Build Bank of Yalarad build 20150210-1701

✓ Completed

Duration: 11 seconds

Start Time: February 10, 2015 at 5:02:00 PM

Completed: February 10, 2015 at 5:02:12 PM

Status Trend: 🟩🟩🟩🟩🟩 Showing all builds

Versions Included in Request

Component	Version for Request	Role	Type	Description
Bank Of Yalarad	20150210-1701 BankOfYalaradAccount1	All	Full	

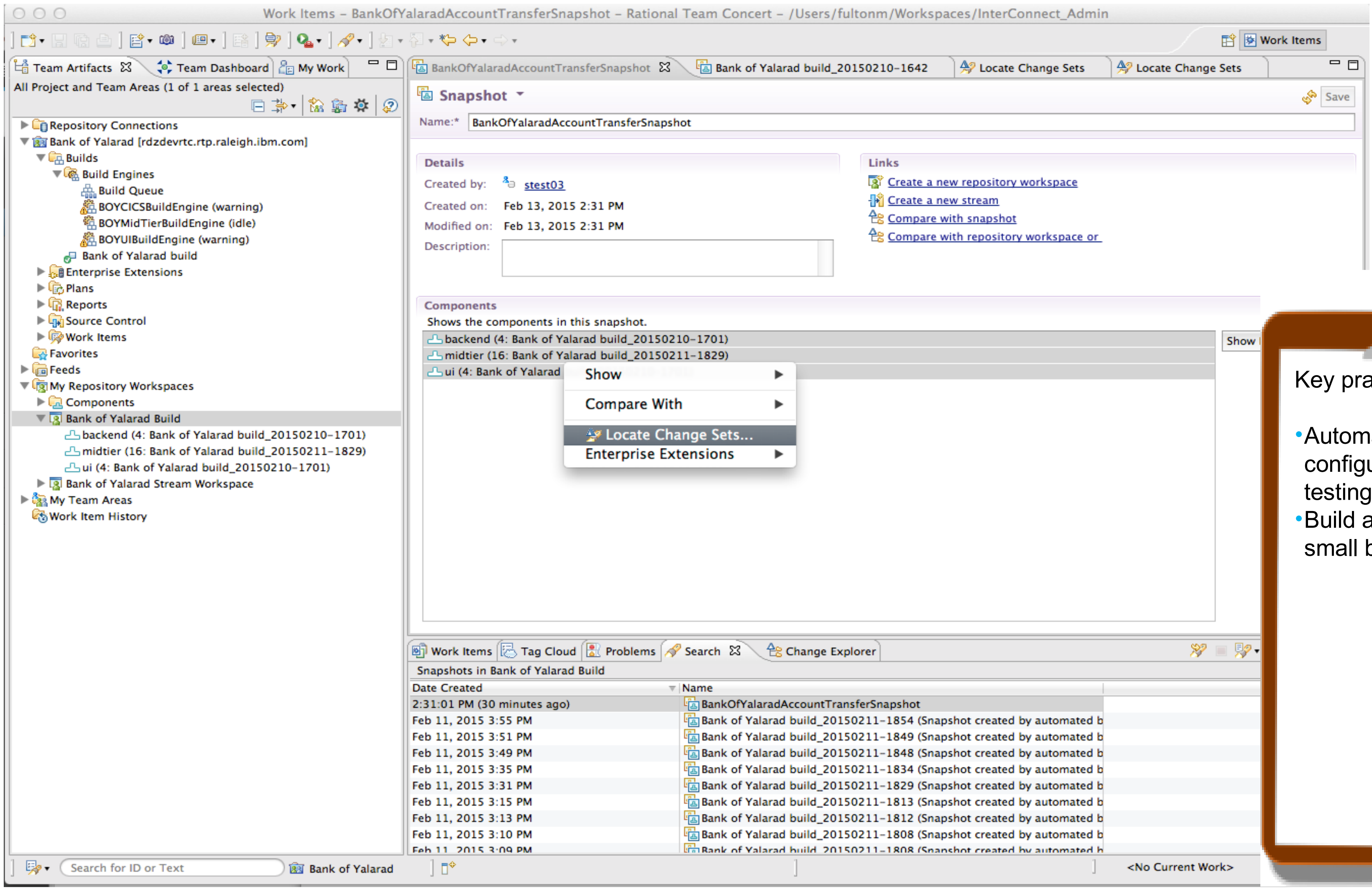
1 record - Refresh Print

1 /1

Key practices:

- Automate deployment, configuration, and testing
- Build and deploy in small batches

Traceability: Trace build to components with RTC



Work Items – BankOfYalaradAccountTransferSnapshot – Rational Team Concert – /Users/fultonm/Workspaces/InterConnect_Admin

All Project and Team Areas (1 of 1 areas selected)

```
▶ Repository Connections
▼ Bank of Yalarad [rdzdevrtc.rtp.raleigh.ibm.com]
  ▼ Builds
    ▼ Build Engines
        Build Queue
        BOYCICSBuildEngine (warning)
        BOYMidTierBuildEngine (idle)
        BOYUIBuildEngine (warning)
      Bank of Yalarad build
  ▶ Enterprise Extensions
  ▶ Plans
  ▶ Reports
  ▶ Source Control
  ▶ Work Items
  Favorites
▶ Feeds
▼ My Repository Workspaces
  ▶ Components
  ▼ Bank of Yalarad Build
      backend (4: Bank of Yalarad build_20150210-1701)
      midtier (16: Bank of Yalarad build_20150211-1829)
      ui (4: Bank of Yalarad build_20150210-1701)
  ▶ Bank of Yalarad Stream Workspace
▶ My Team Areas
  Work Item History
```

BankOfYalaradAccountTransferSnapshot
Bank of Yalarad build_20150210-1642
Locate Change Sets
Locate Change Sets

Snapshot

Name:* BankOfYalaradAccountTransferSnapshot

Details

Created by:	stest03
Created on:	Feb 13, 2015 2:31 PM
Modified on:	Feb 13, 2015 2:31 PM
Description:	

Links

- Create a new repository workspace
- Create a new stream
- Compare with snapshot
- Compare with repository workspace or

Components

Shows the components in this snapshot.

- backend (4: Bank of Yalarad build_20150210-1701)
- midtier (16: Bank of Yalarad build_20150211-1829)
- ui (4: Bank of Yalarad

Snapshots in Bank of Yalarad Build

Date Created	Name
2:31:01 PM (30 minutes ago)	BankOfYalaradAccountTransferSnapshot
Feb 11, 2015 3:55 PM	Bank of Yalarad build_20150211-1854 (Snapshot created by automated b
Feb 11, 2015 3:51 PM	Bank of Yalarad build_20150211-1849 (Snapshot created by automated b
Feb 11, 2015 3:49 PM	Bank of Yalarad build_20150211-1848 (Snapshot created by automated b
Feb 11, 2015 3:35 PM	Bank of Yalarad build_20150211-1834 (Snapshot created by automated b
Feb 11, 2015 3:31 PM	Bank of Yalarad build_20150211-1829 (Snapshot created by automated b
Feb 11, 2015 3:15 PM	Bank of Yalarad build_20150211-1813 (Snapshot created by automated b
Feb 11, 2015 3:13 PM	Bank of Yalarad build_20150211-1812 (Snapshot created by automated b
Feb 11, 2015 3:10 PM	Bank of Yalarad build_20150211-1808 (Snapshot created by automated b
Feb 11, 2015 3:09 PM	Bank of Yalarad build_20150211-1808 (Snapshot created by automated b

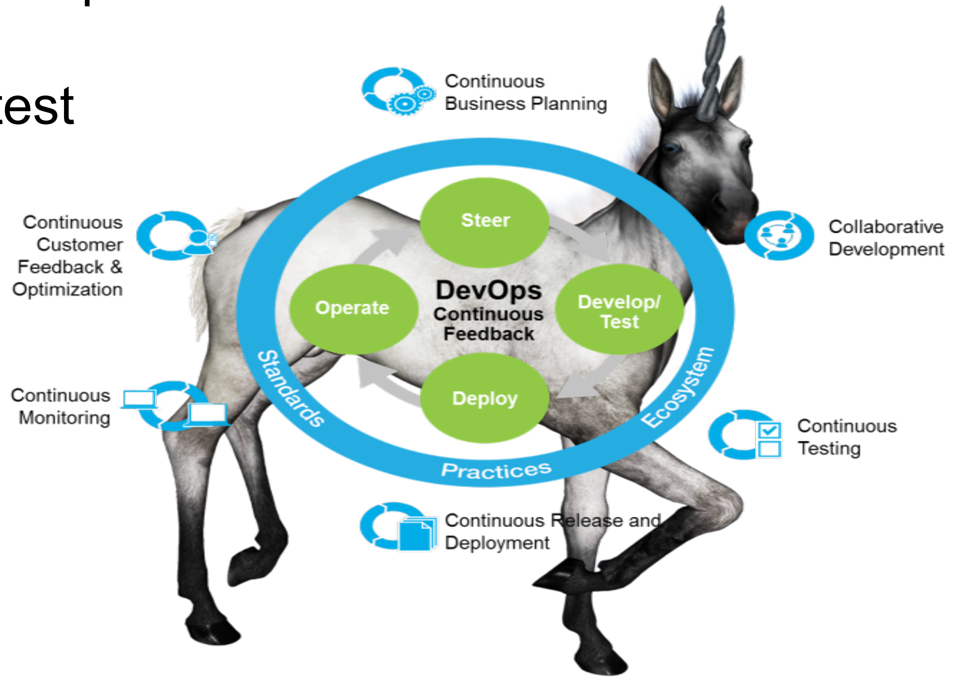
Key practices:

- Automate deployment, configuration, and testing
- Build and deploy in small batches

Summary

Tools **are** available to support this evolution….

- ✓ Modern multi-platform developer and team tools
- ✓ Consolidated SCM
- ✓ Automated build, deploy, test
- ✓ Real-time dashboards


Continuous Business Planning
Collaborative Development
Continuous Testing
Continuous Release and Deployment
Continuous Monitoring
Continuous Customer Feedback & Optimization
Steer
Develop/ Test
Deploy
Operate
DevOps Continuous Feedback
Standards
Practices
Ecosystem

Notices and Disclaimers

Notices and Disclaimers (con’t)