Gestión de configuración multiplataforma con RTC (or: *Accelerate delivery, reduce costs, with the IBM Integrated Solution for System z Development)*

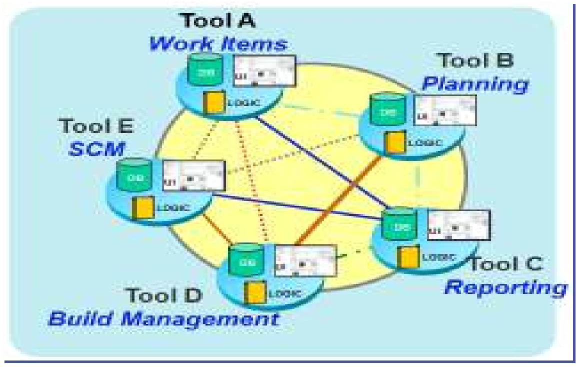
Keith Allen
Arquitecto Especialista en Soluciones de Modernización de la Empresa

Agenda

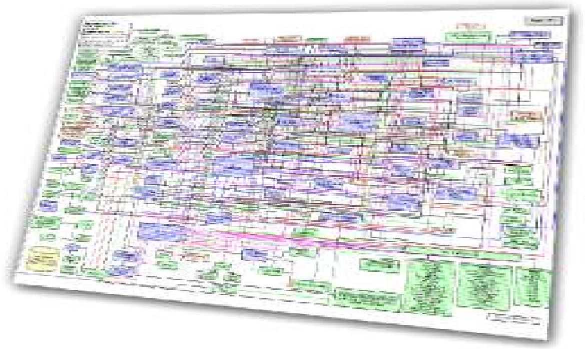
- Today’s Mainframe Development Challenges
- Addressing these challenges with IBM Integrated Solution for System z Development (ISDz)
- One customer’s story
  - Overview of current client environment
  - Proposed solution (ISDz)
  - Road to deployment
  - Results
- Summary

Customer Challenges

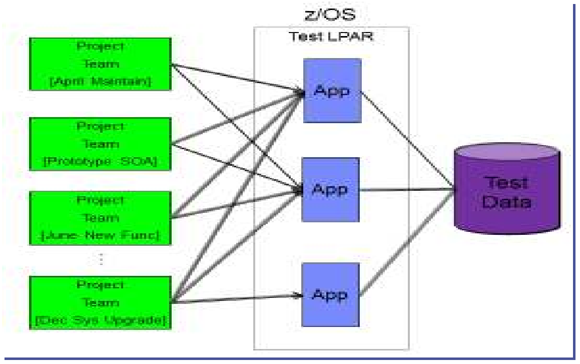
"Our skills gap keeps growing.
How do we stay current with all the language and technology changes?"



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

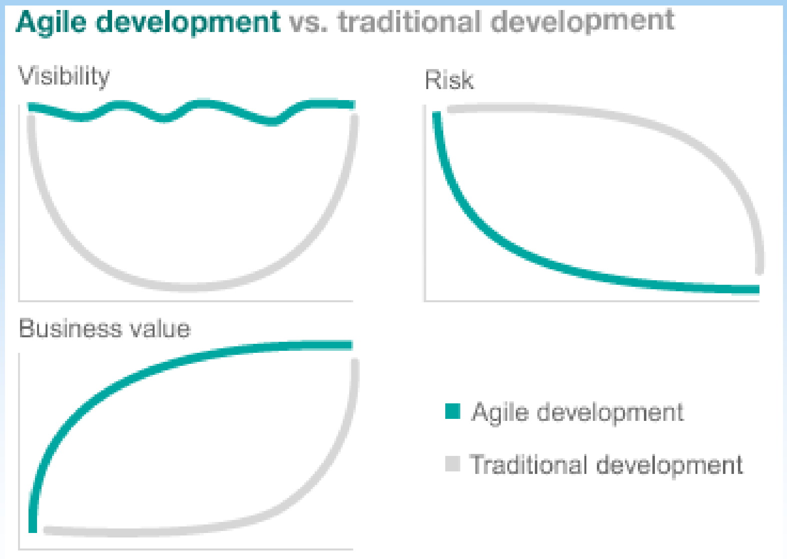


"We don't understand the effort, risk and impact of modernizing our legacy applications."


z/OS
Test LPAR
Project Team [April Maintain]
Project Team [Prototype SOA]
Project Team [June New Func]
Project Team [Dec Sys Upgrade]
App
App
App
Test Data

"We need a cost effective way to improve our infrastructure efficiency and free up capacity to handle more workload."

And Mainframe development is under pressure to change….



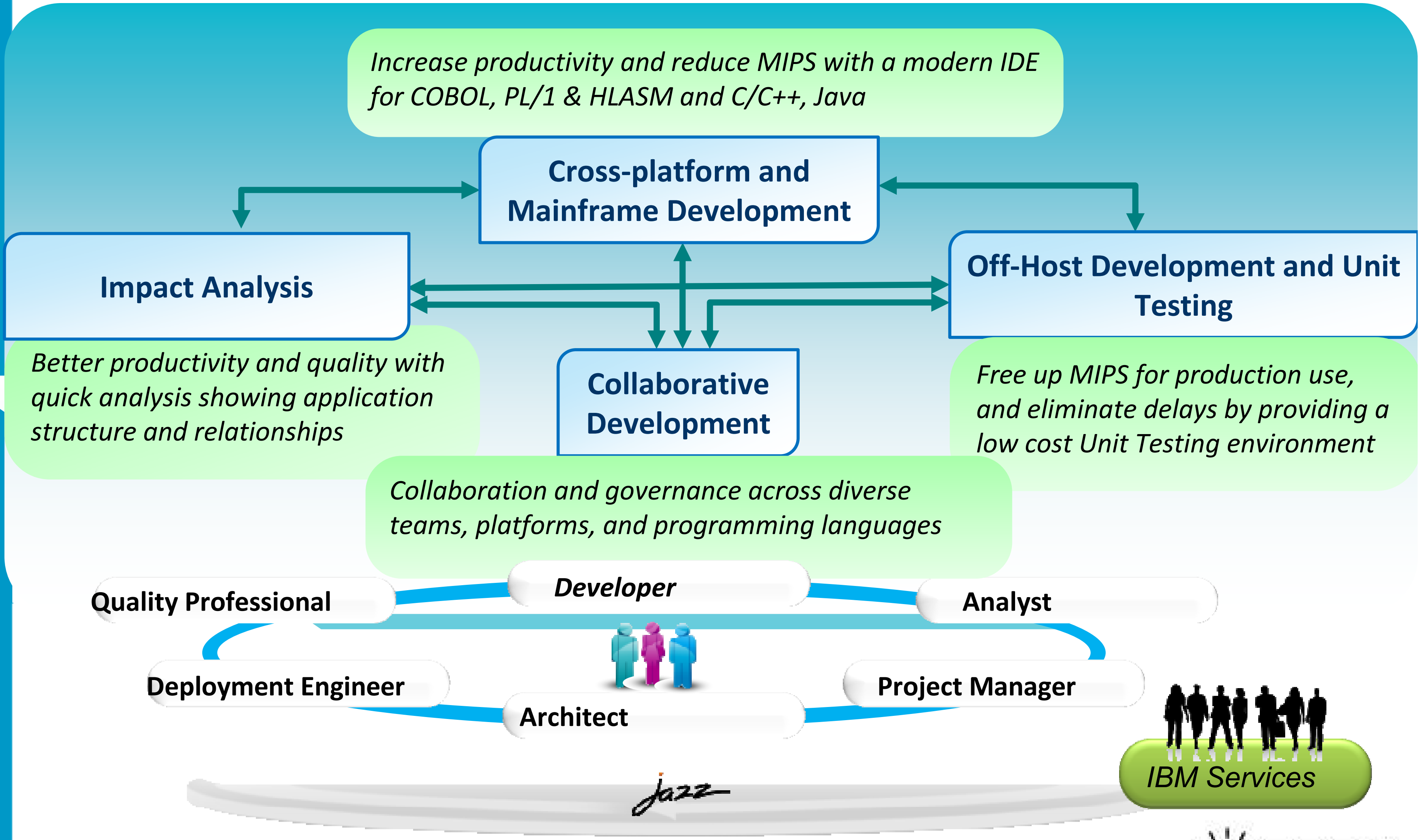
Challenges

- Agile and Lean development is working
- Legacy infrastructure is restrictive
- Hybrid and collaborative applications

Agenda

- Today's Mainframe Development Challenges
- Addressing these challenges with IBM Integrated Solution for System z Development (ISDz)
- One customer's story
  - Overview of current client environment
  - Proposed solution (ISDz)
  - Road to deployment
  - Results
- Summary

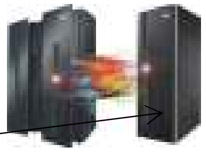
The IBM Integrated Solution for System z Development


Increase productivity and reduce MIPS with a modern IDE for COBOL, PL/1 & HLASM and C/C++, Java
Cross-platform and Mainframe Development
Impact Analysis
Better productivity and quality with quick analysis showing application structure and relationships
Off-Host Development and Unit Testing
Free up MIPS for production use, and eliminate delays by providing a low cost Unit Testing environment
Collaborative Development
Collaboration and governance across diverse teams, platforms, and programming languages
Quality Professional
Developer
Analyst
Deployment Engineer
Architect
Project Manager
jazz
IBM Services

Complete set of System z Development and Test capabilities from an integrated development environment

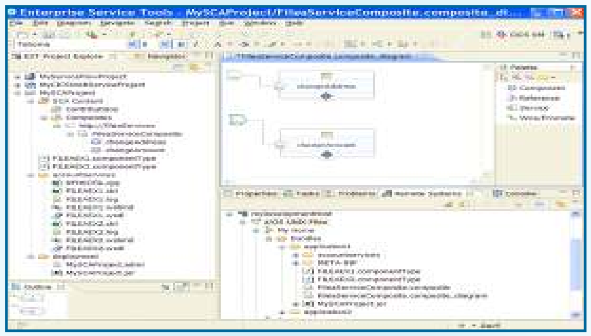
Integration with Lifecycle and Source Management

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



ISDz

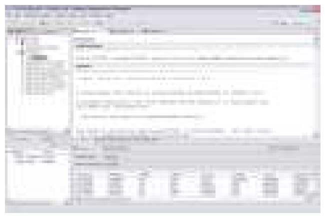
Integration with Application Understanding and Impact Analysis

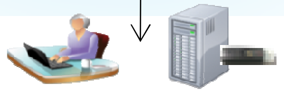

Enterprise Service Tools - MySCAProject/FilesServiceComposite.composite_di...

Integration with Development and Testing areas

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

Integration with Dump Analysis



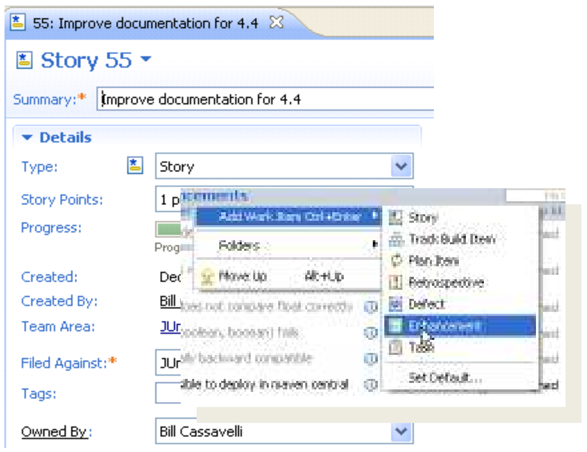


Integration with off host for flexible access to System z environment

Integration with file and test data handling

A single platform for IT Application Planning

- Work Items



55: Improve documentation for 4.4

Story 55

Summary:* Improve documentation for 4.4

Details

Type: Story

Story Points: 1 p

Progress:

Created: Dec

Created By: Bill

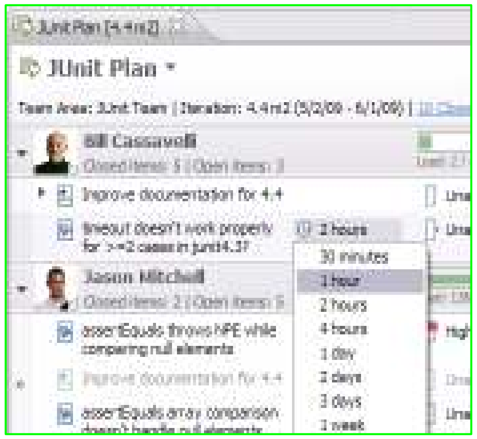
Team Area: JUn

Filed Against:* JUn

Tags:

Owned By: Bill Cassavelli

- Planning



JUnit Plan

Team Area: JUnit Team | Iteration: 4.4m2 (5/2/09 - 6/1/09)

Bill Cassavelli — Closed Items: 5 | Open Items: 3

Improve documentation for 4.4

timeout doesn't work properly for >=2 cases in junit4.3?

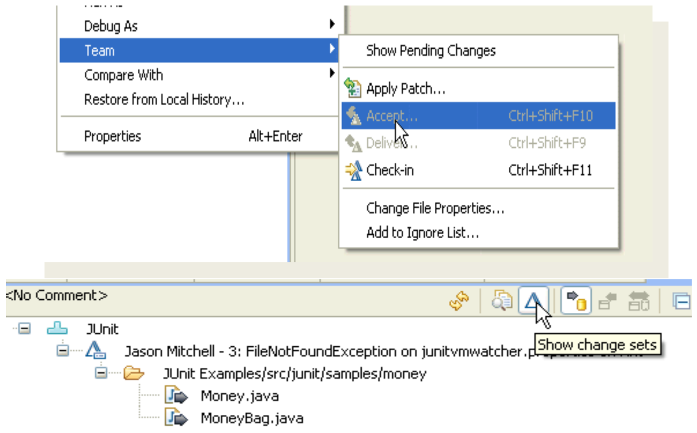
Jason Mitchell — Closed Items: 2 | Open Items: 5

assertEquals throws NPE while comparing null elements

Improve documentation for 4.4

assertEquals array comparison doesn't handle null elements

- Source Control



JUnit

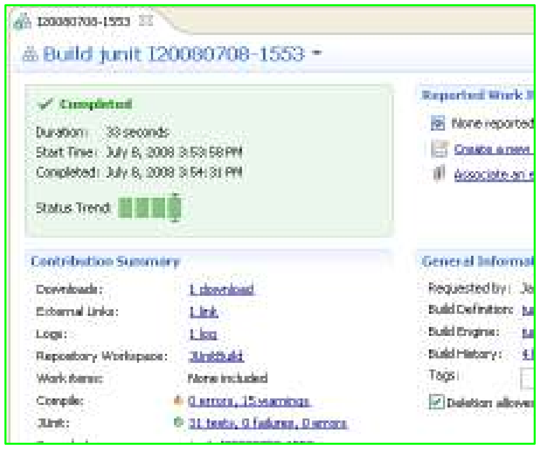
Jason Mitchell - 3: FileNotFoundException on junitvmwatcher.p

JUnit Examples/src/junit/samples/money

Money.java

MoneyBag.java

- Builds – Continuous



Build junit I20080708-1553

Completed

Duration: 33 seconds

Start Time: July 8, 2008 3:53:58 PM

Completed: July 8, 2008 3:54:31 PM

Status Trend:

Contribution Summary

Downloads:	1 download
External Links:	1 link
Logs:	1 log
Repository Workspace:	JUnitBuild
Work items:	None included
Compile:	0 errors, 15 warnings
JUnit:	31 tests, 0 failures, 0 errors

Reported Work Items

None reported

Create a new

Associate an

General Information

Requested by:

Build Definition:

Build Engine:

Build History:

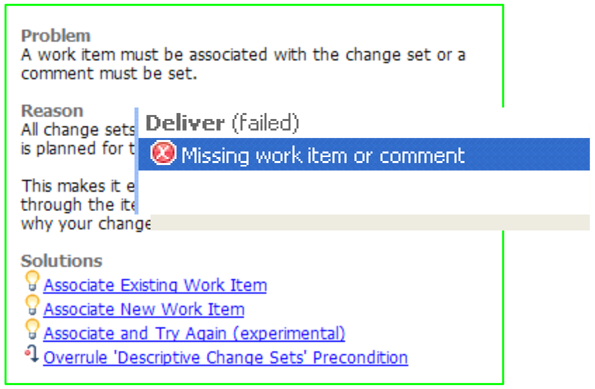
Tags:

Deletion allowed

- Dashboards & Reporting

JUnit Project

- Method Enforcement and Automation



Problem

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

Reason

All change sets is planned for t

This makes it e through the ite why your change

Deliver (failed)

Missing work item or comment

Solutions

Associate Existing Work Item

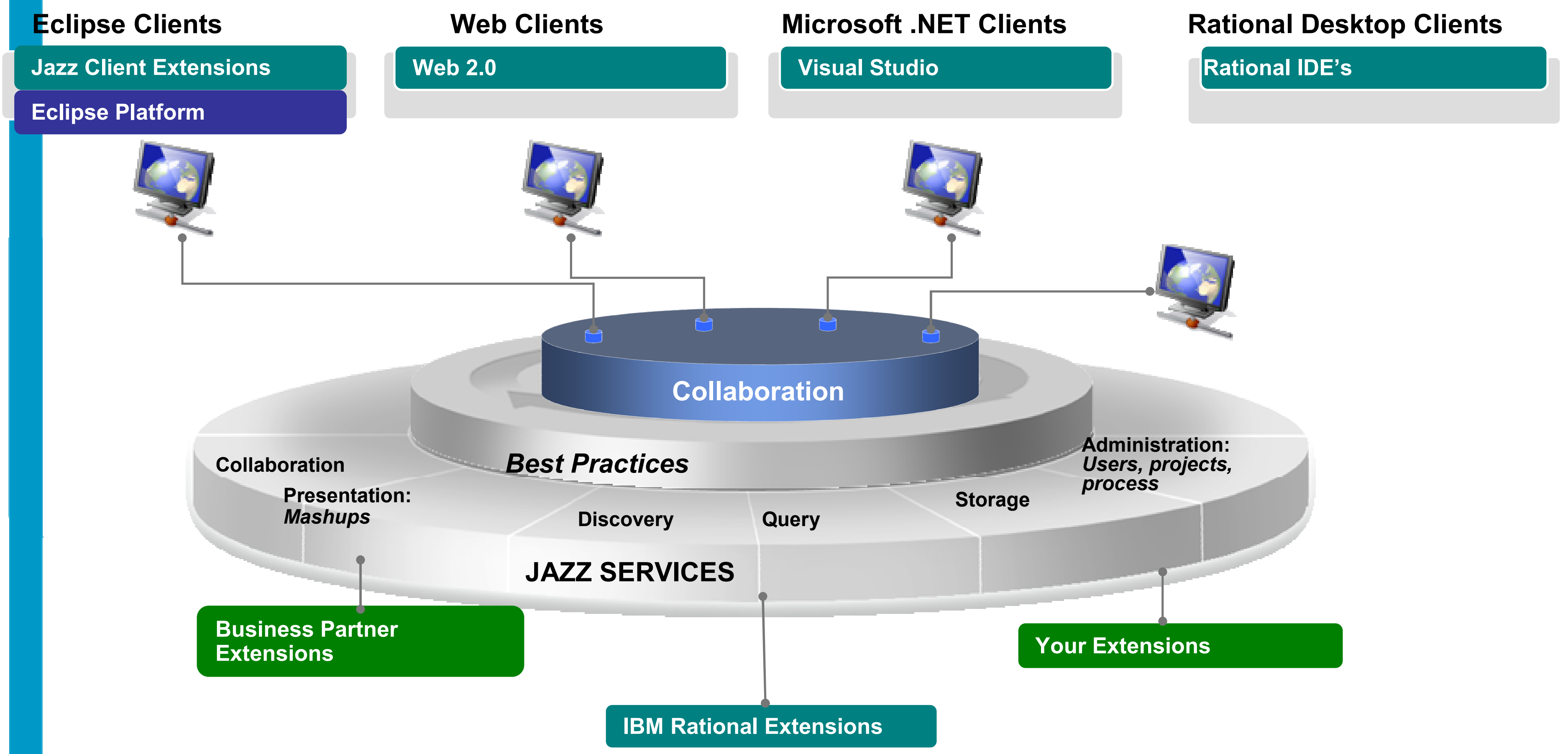
Associate New Work Item

Associate and Try Again (experimental)

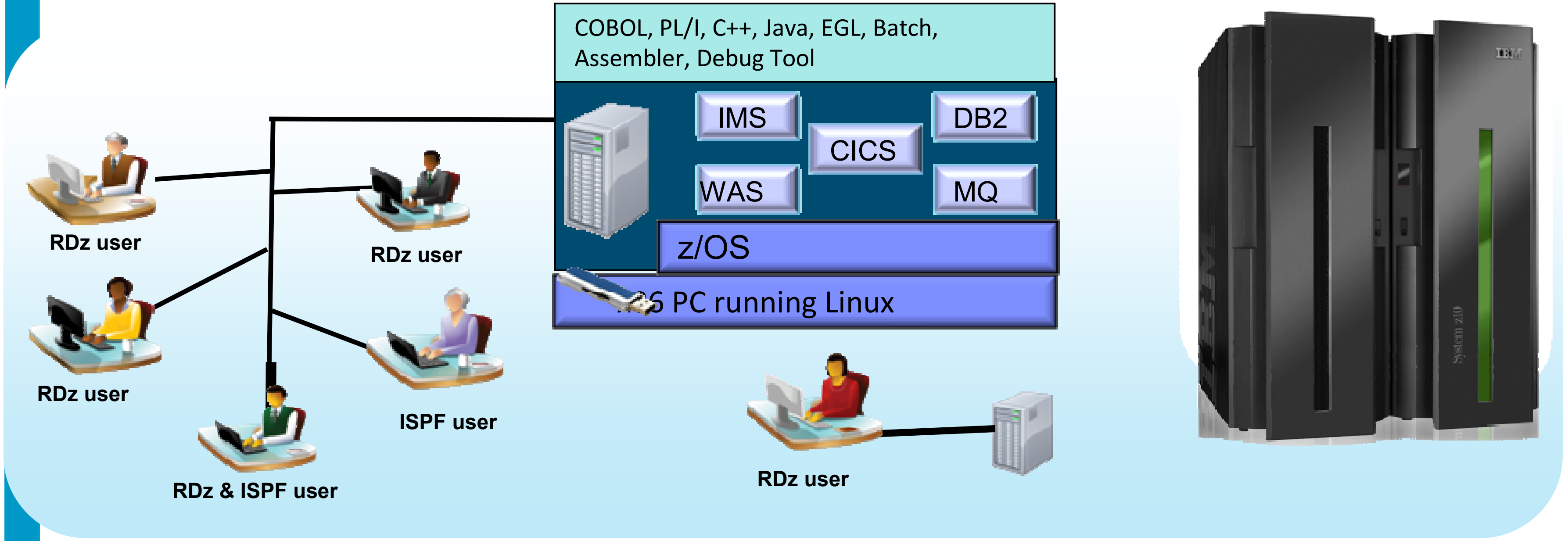
Overrule 'Descriptive Change Sets' Precondition

ISDz Built on an open, Web 2.0 platform

Supporting a broad range of desktop clients, IDE's and languages



Off Host System Z - *The ultimate in modern application development for System z*



- Increase availability of z/OS testing environment and resources
  - Liberate developers to rapidly prototype new applications
  - Develop and test System z applications anywhere, anytime!
  - Eliminate costly delays by reducing dependencies on operations staff
- Improve quality and lower risk via automation, measurement, and collaboration
- Focus on what is required for the change at hand, then scale

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

Flexible and Incremental Adoption*

Entry Point	Add Capability	Add Capability	Add Capability
• Increase developer productivity to reduce maintenance backlog • Quickly modernize System z apps with coding assists and service creation and refactoring wizards • Improve code quality with code review, automated UT, and code coverage	• More rapid, flexible developer testing • Reduce development MIPS	• Reduce delivery time by understanding the impact of change, upfront • Shortened learning curve for new team members	• Unified status, change management, process, and SCM across tools, teams, and platforms • Reduce risks and meet audit an compliance mandates with automated process enforcement • Reduce the cost of System z SCM
RDz Modern IDE for applications that include System z components	RD&T Add z/OS development and unit test environment on an z86 Linux Server	RAA Add rapid application understanding	RTC Add collaboration and governance across diverse teams, platforms, and programming languages

*Elements of the solution may be adopted any order based on your needs

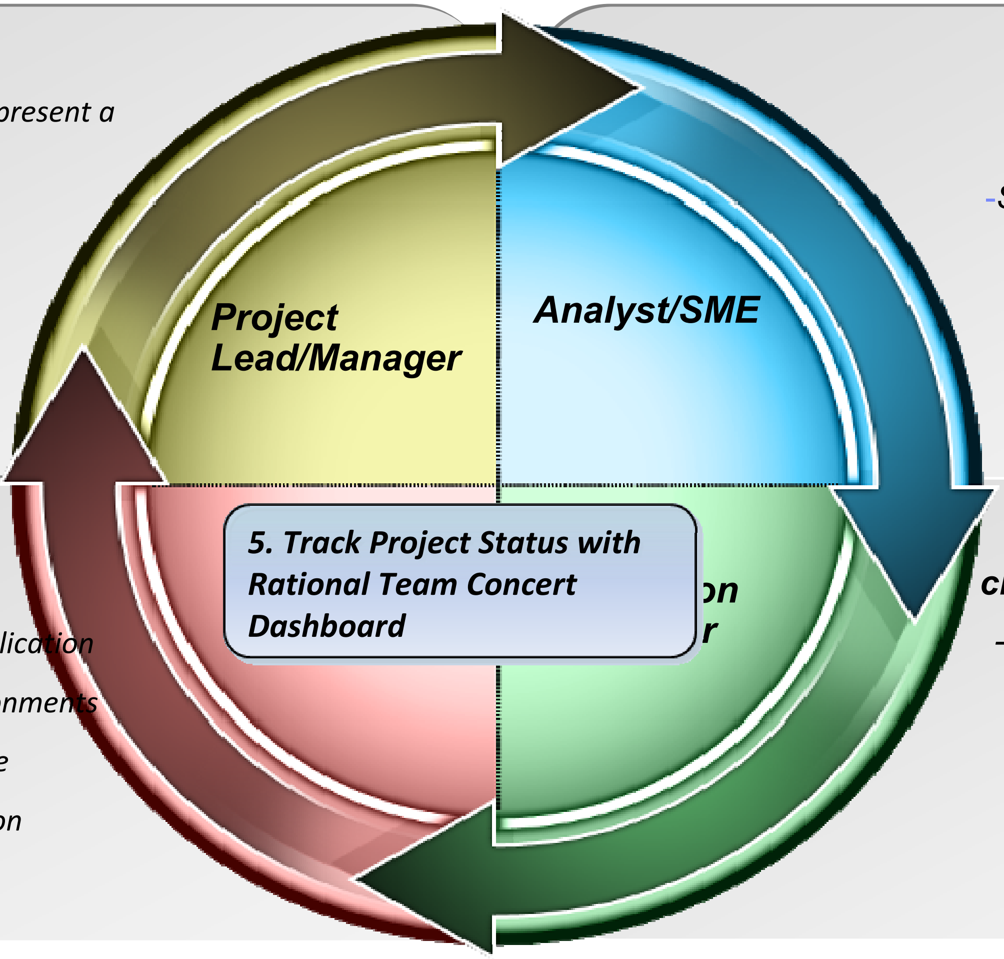
Overview of Supported Production Scenario

1: Initiate Change Request

- *Submit new work item to represent a change request*
- *Assign to Analyst*

2: Analyze

- *Analyze Application to be changed*
- *Size/Scope the effort for the change*
- *Create Bill of Materials*



5. Track Project Status with Rational Team Concert Dashboard

3: Implement required changes, build and deliver

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

4: Promote and deploy enhancement

- *Create 'official' build of application*
- *Promote through test environments*
- *Build formal release package*
- *Deploy package to production*

Overview of Supported **Production** Scenario

Objective: Implement change request

Analyst: Analyze to scope size and risk of change request

1. IT has received a request that requires a change to an existing COBOL application. A requirement (work item) is created using the ISDz browser interface.

2. The Analyst receives the request and uses ISDz to understand application structure and complexity as well as determine the set of application artifacts involved in the change. A link to the impact analysis is added to the original requirement (work item) and optionally a "Bill of Materials" (BOM) is generated and added as an attachment.

3. Based on analysis, the Analyst refines original requirement into one or more work items for development based on the scope of the change request, using ISDz to update the initial requirement, create more fine-grained work items, and update the work item(s) with analysis information.

4. The Analyst creates work items for formal test, alerting the test team that some test cases will be impacted by the change request (or new test cases will be required).

Overview of Supported **Production** Scenario

Objective: **Implement change request**

Developer: **Implement changes, build, and deliver**

1. The Developer views his work using the ISDz client to verify the development level tasks.

2. The Developer reviews the analysis information in the work item and uses ISDz via the browser interface within the ISDz workspace as well as the ISDz plugin capabilities to better understand the programs and other assets to be modified.

3. The Developer uses ISDz to make the changes and deploys and tests the updates on the Off Host Z platform using ISDz build capabilities.

4. The Developer completes his changes by marking the work item ready for formal test and delivers the changes to the development stream, again using ISDz build capabilities.

Overview of Supported **Production** Scenario

Objective: Implement change request

Release Engineer: Promote and deploy the enhancement

1. The Release Engineer is notified that changes are ready for promotion to the formal test system and uses ISDz to move (promote) the set of changes to the QA stream.

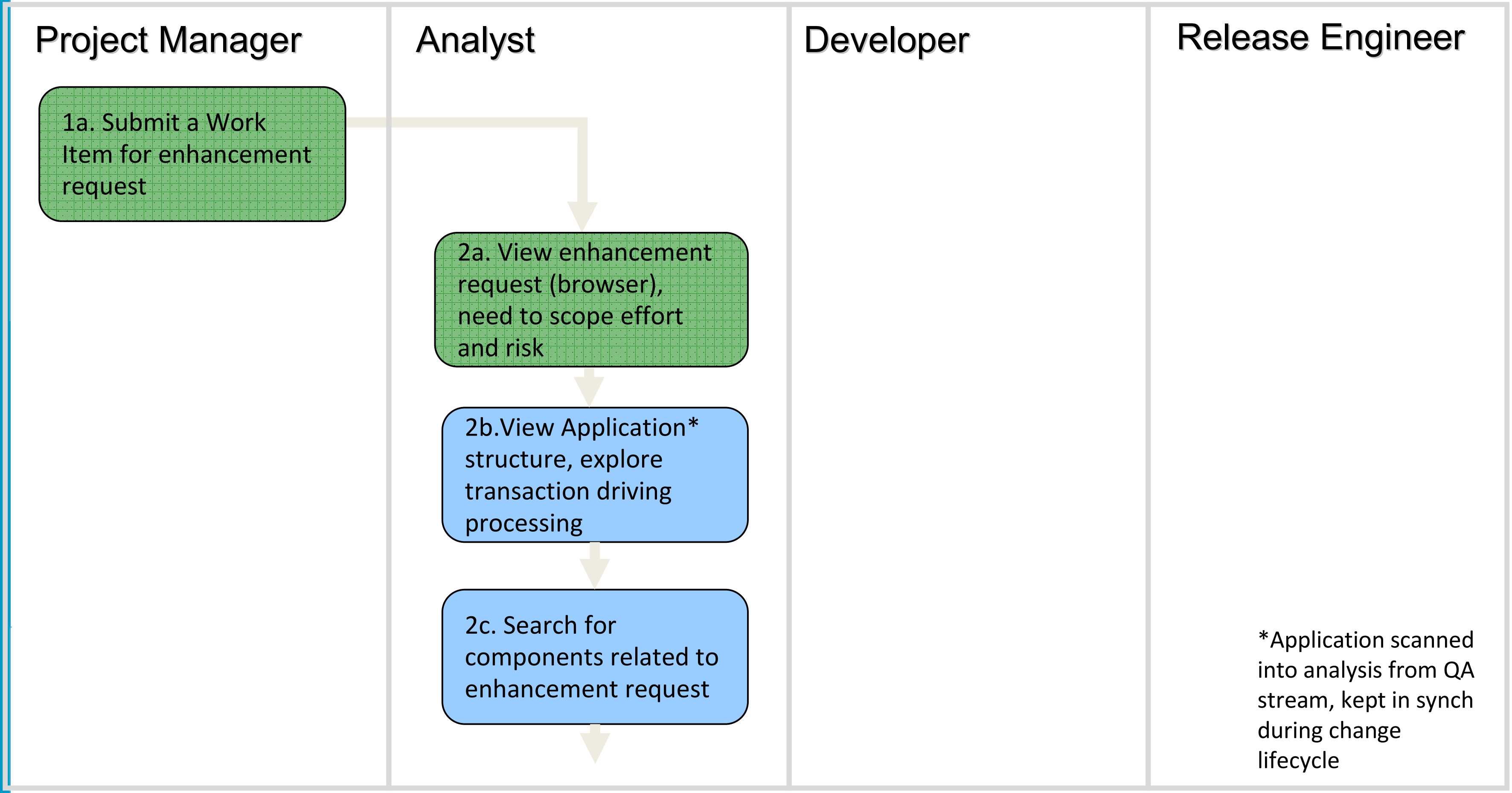
2. The Release Engineer alerts the QA team that testing can be started on the set of work items promoted to the QA stream by updating the associated test work items.

3. Once the testing is complete, the Release Engineer is notified via the test work items and uses, then uses ISDz to build the formal release package. If problems occur during formal testing, the Release Engineer or test resource opens new work items that are routed back to the development team.

4. The Release Engineer uses ISDz to deploy the package to production.

Note: The formal testing role is intentionally omitted from this high-level scenario for simplicity but is covered in the Detailed Scenarios document.

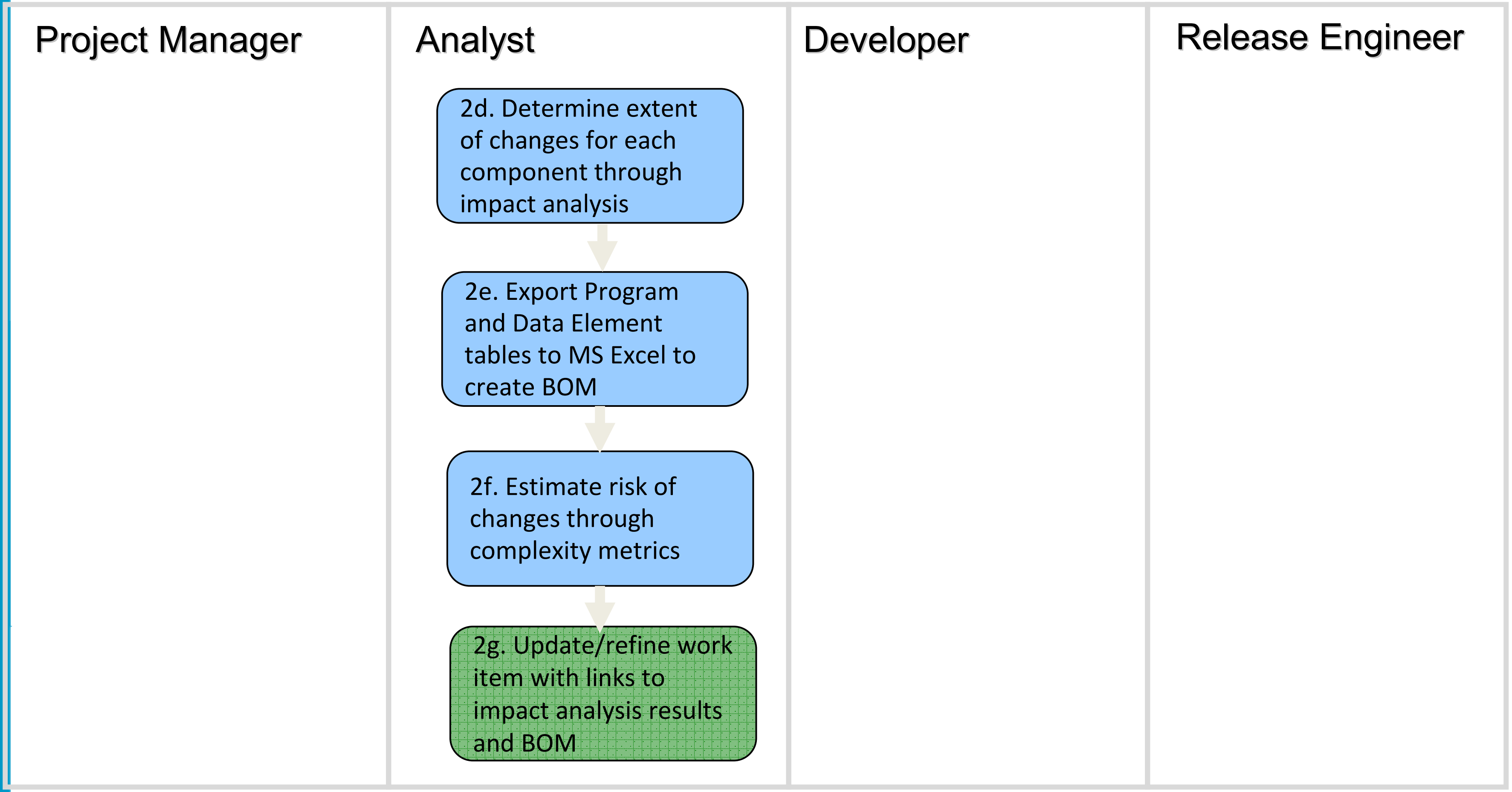
Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
1a. Submit a Work Item for enhancement request
2a. View enhancement request (browser), need to scope effort and risk
2b.View Application* structure, explore transaction driving processing
2c. Search for components related to enhancement request

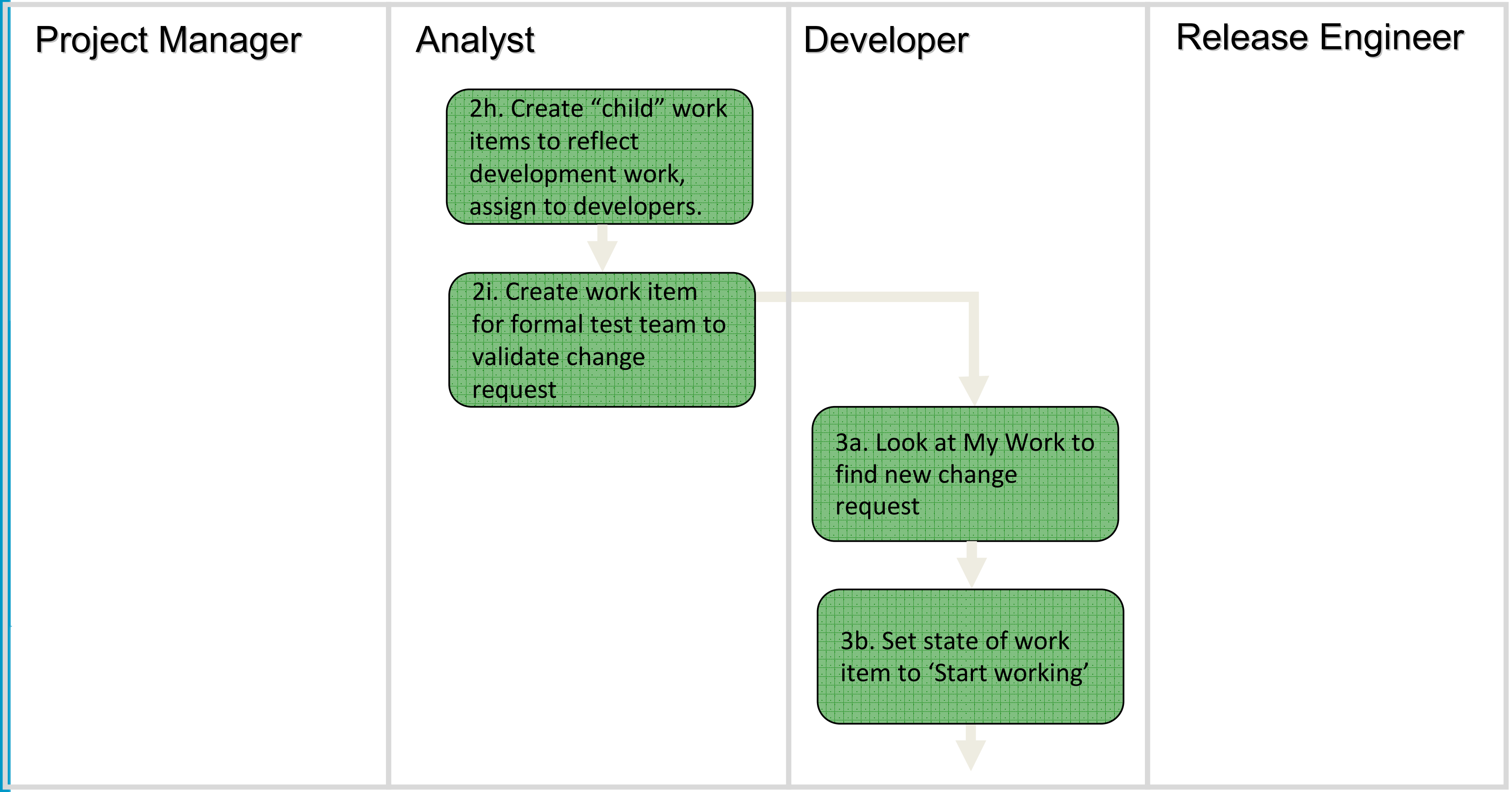
*Application scanned into analysis from QA stream, kept in synch during change lifecycle

Rational Solution: ISDz

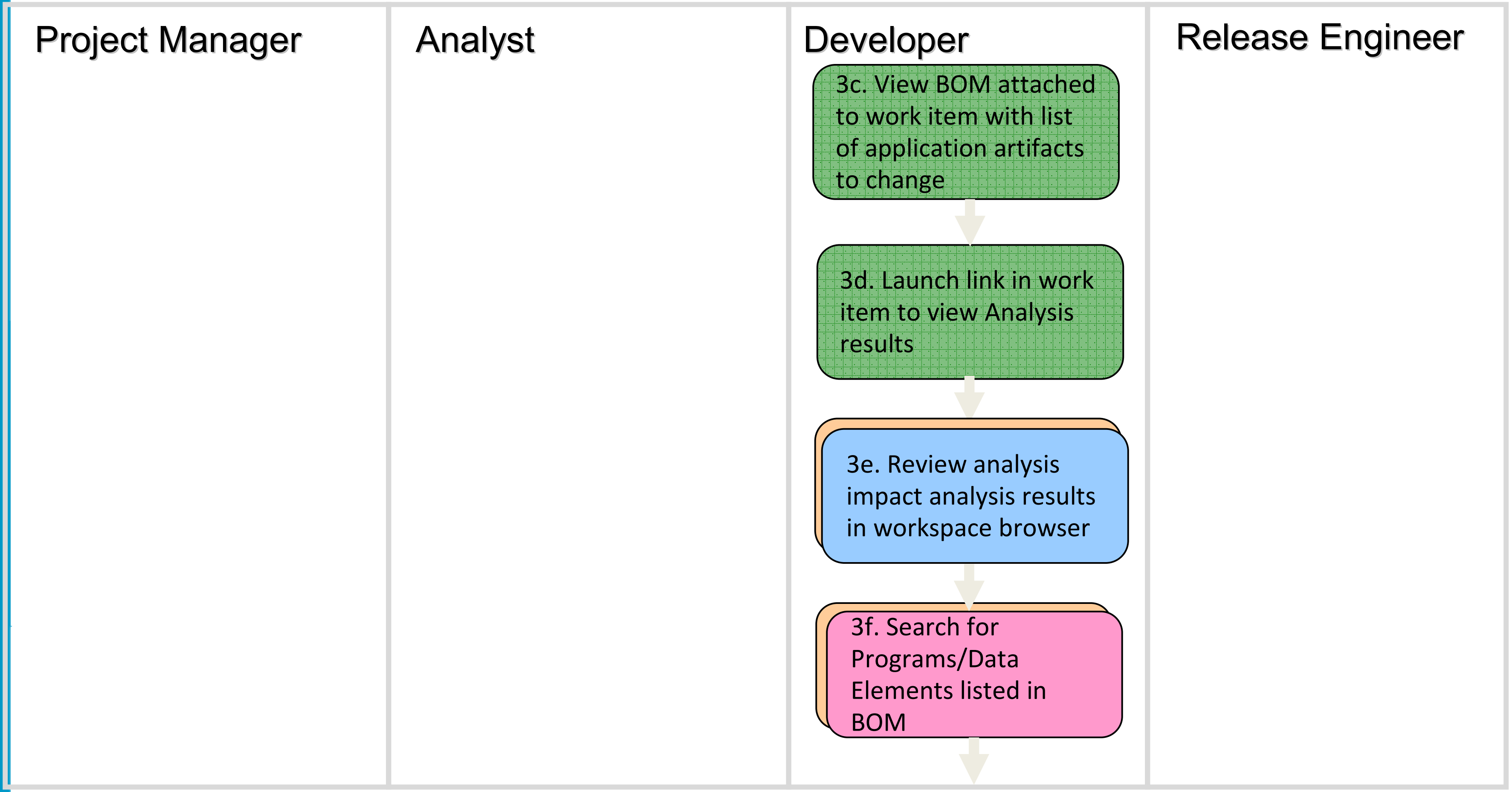
Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
2d. Determine extent of changes for each component through impact analysis
2e. Export Program and Data Element tables to MS Excel to create BOM
2f. Estimate risk of changes through complexity metrics
2g. Update/refine work item with links to impact analysis results and BOM

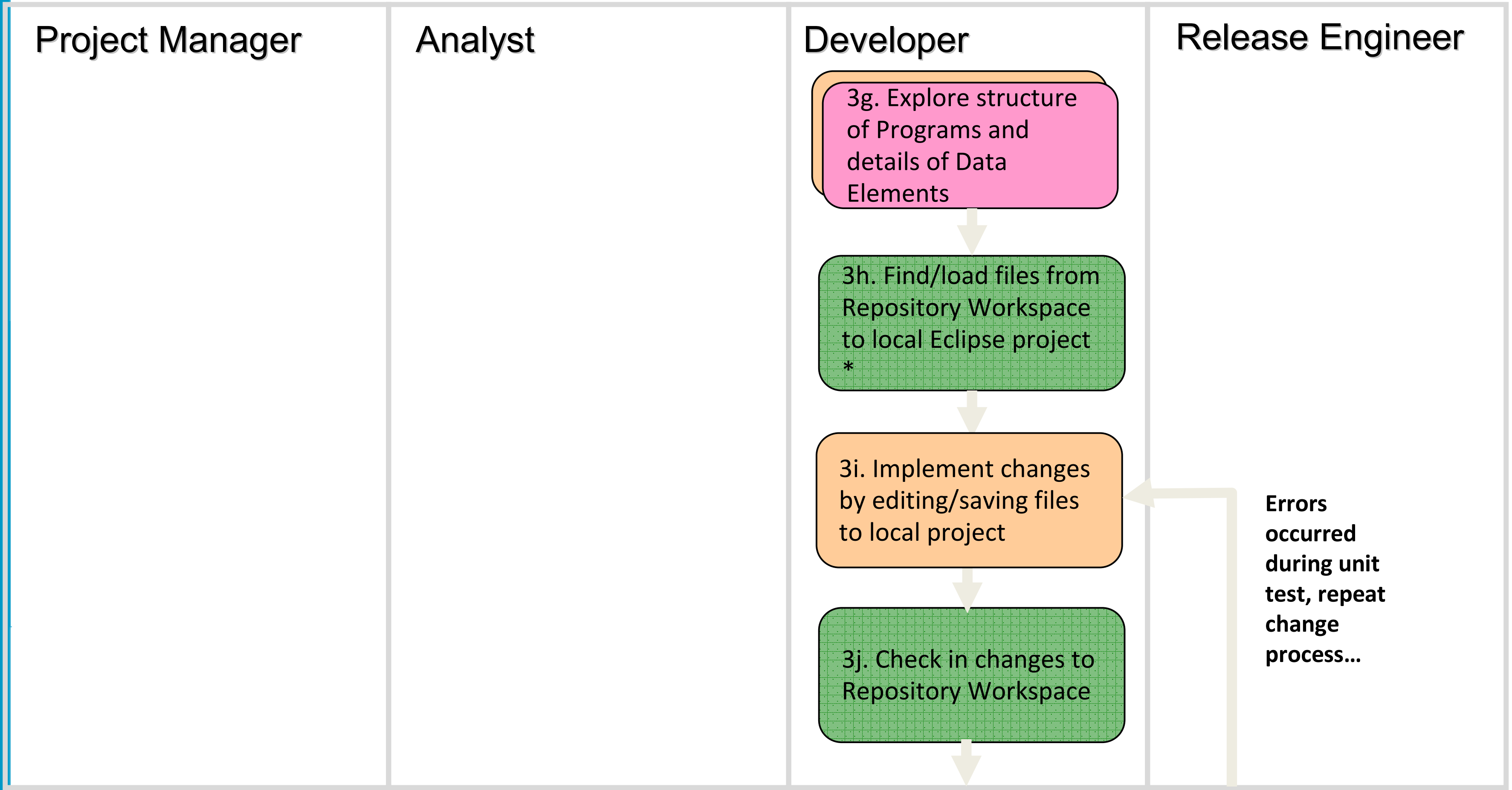
Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
2h. Create “child” work items to reflect development work, assign to developers.
2i. Create work item for formal test team to validate change request
3a. Look at My Work to find new change request
3b. Set state of work item to ‘Start working’

Sequence of Events

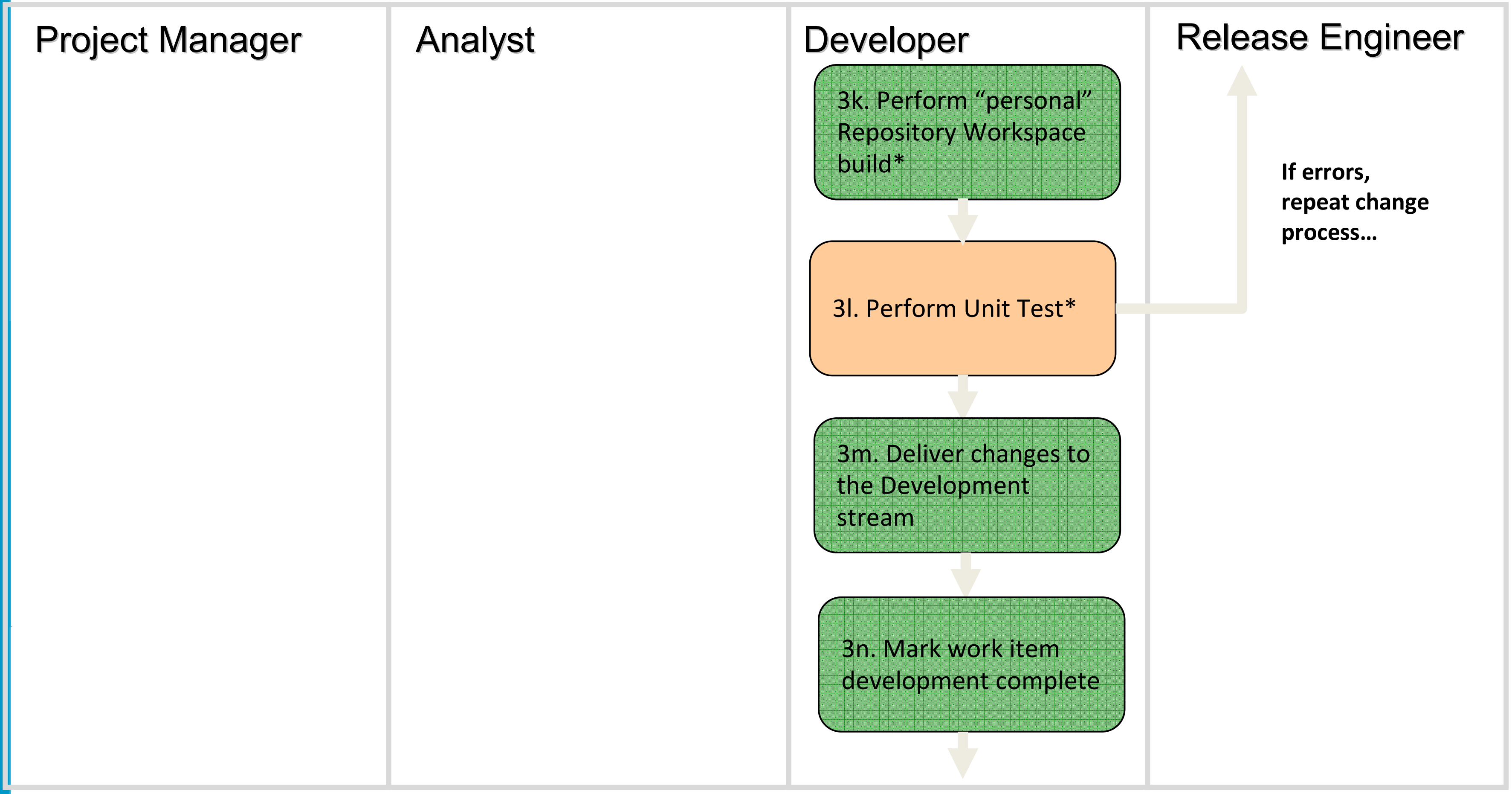

Project Manager
Analyst
Developer
Release Engineer
3c. View BOM attached to work item with list of application artifacts to change
3d. Launch link in work item to view Analysis results
3e. Review analysis impact analysis results in workspace browser
3f. Search for Programs/Data Elements listed in BOM

Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
3g. Explore structure of Programs and details of Data Elements
3h. Find/load files from Repository Workspace to local Eclipse project *
3i. Implement changes by editing/saving files to local project
3j. Check in changes to Repository Workspace
Errors occurred during unit test, repeat change process…

*Assume Dev workspace from Dev stream

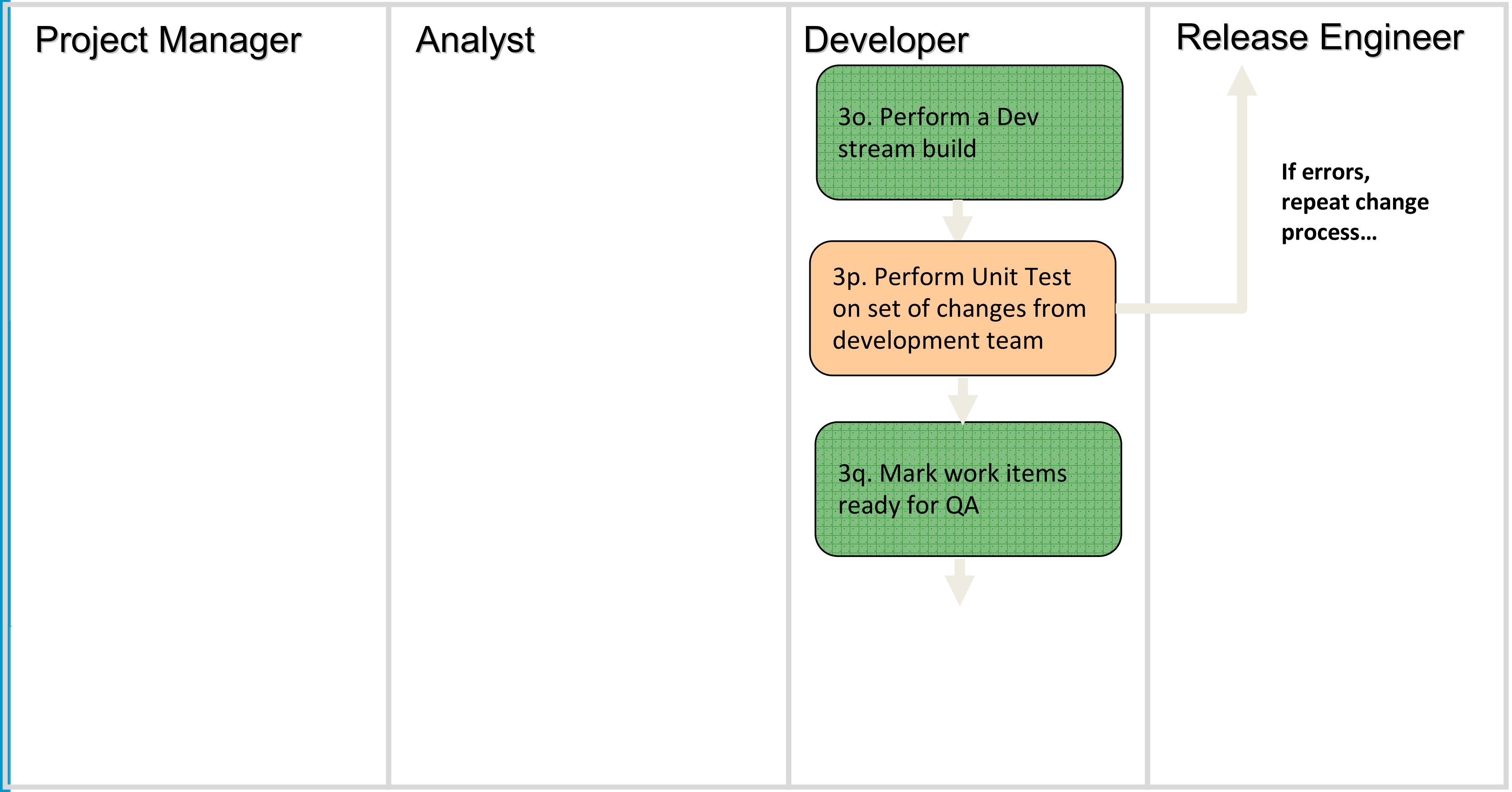
Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
3k. Perform "personal" Repository Workspace build*
3l. Perform Unit Test*
3m. Deliver changes to the Development stream
3n. Mark work item development complete
If errors, repeat change process…

Rational Solution: ISDz

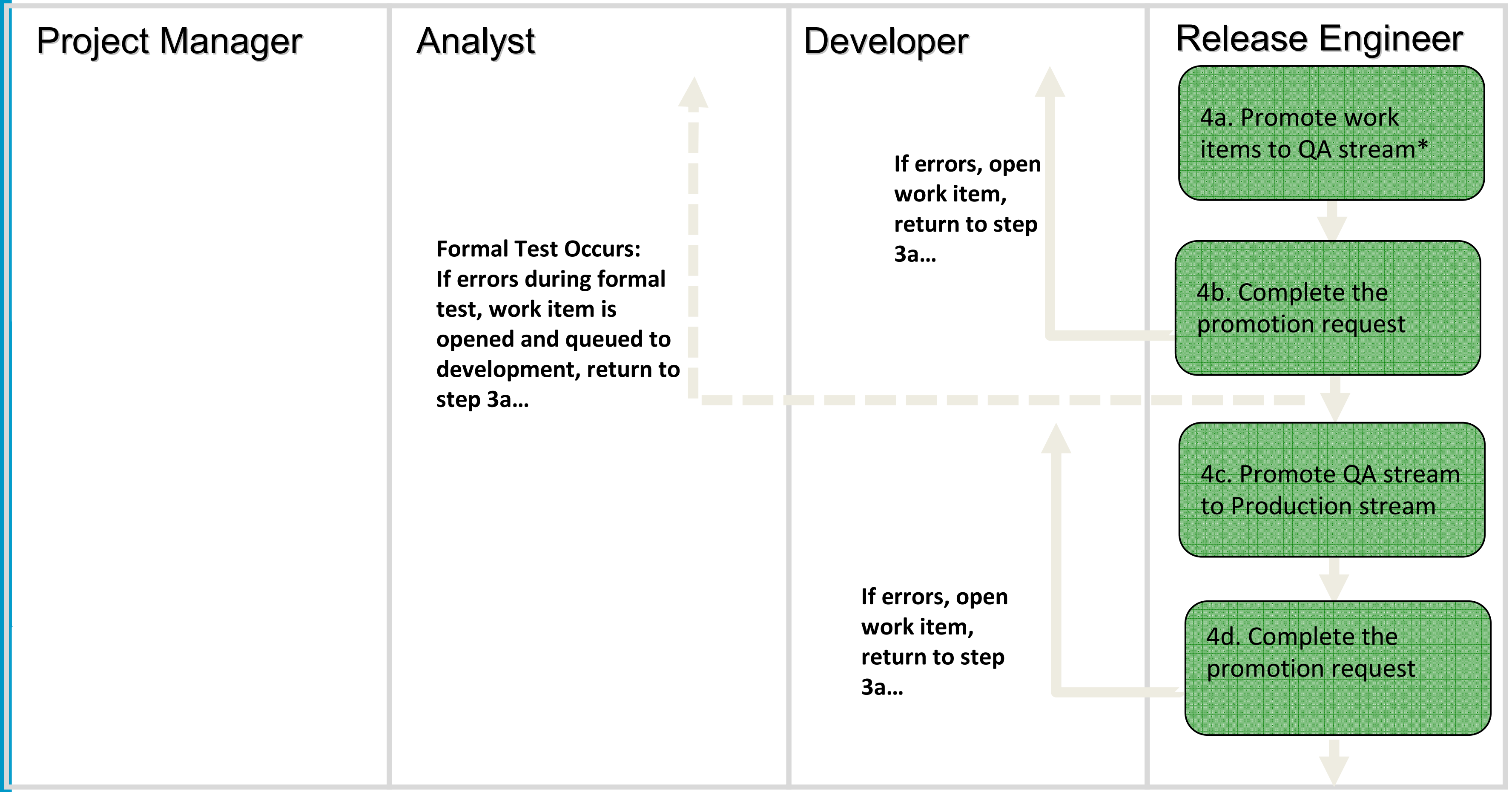
*Build and UT performed on MVS or RDz UT

Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
3o. Perform a Dev stream build
3p. Perform Unit Test on set of changes from development team
3q. Mark work items ready for QA
If errors, repeat change process…

Rational Solution: ISDz

Sequence of Events



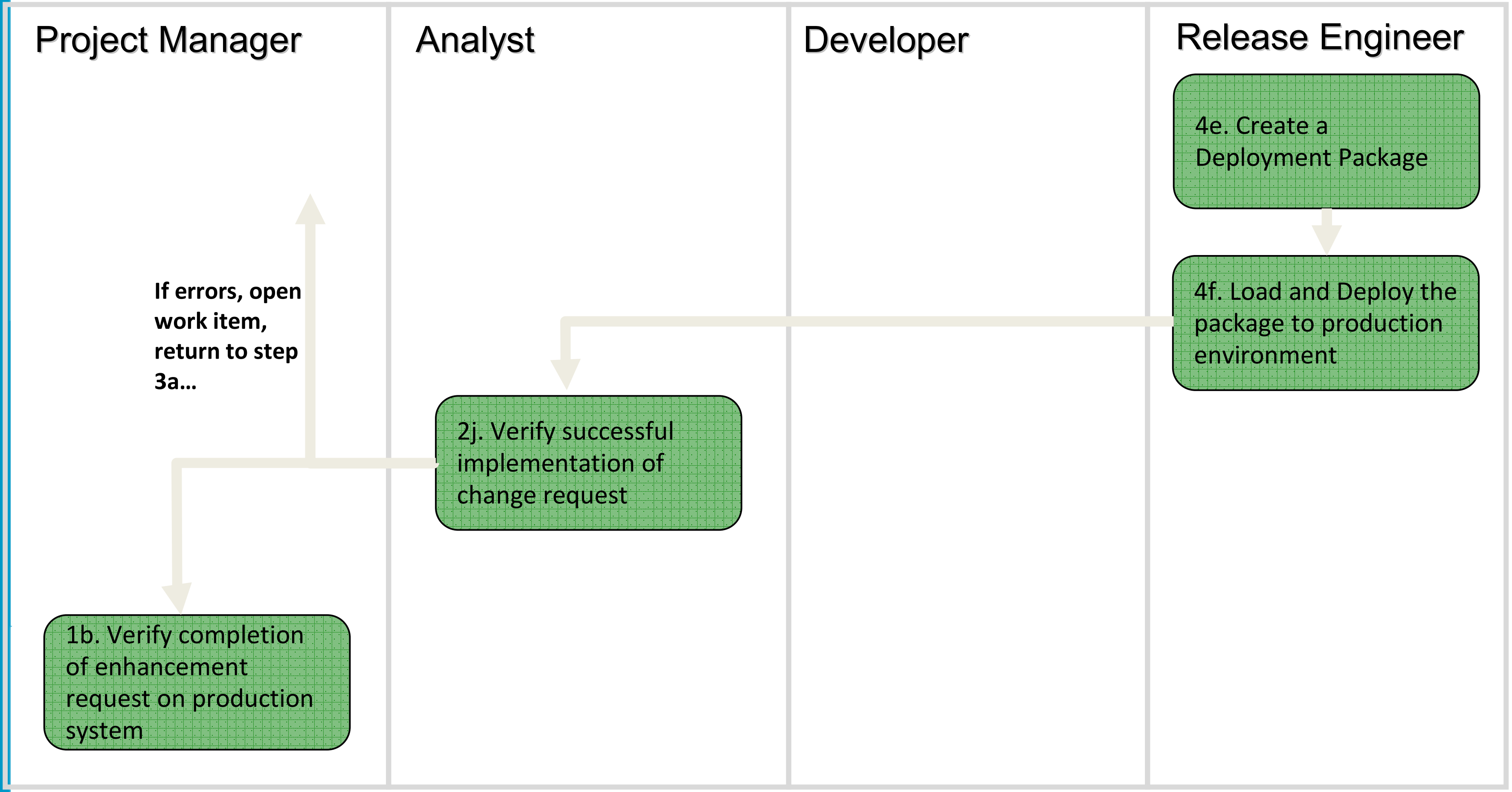
Project Manager
Analyst
Developer
Release Engineer
4a. Promote work items to QA stream*
4b. Complete the promotion request
4c. Promote QA stream to Production stream
4d. Complete the promotion request
If errors, open work item, return to step 3a…
Formal Test Occurs: If errors during formal test, work item is opened and queued to development, return to step 3a…
If errors, open work item, return to step 3a…

Rational Solution

Rational Developer for System z	Rational Team Concert	Rational Asset Analyzer	Rational Asset Analyzer Integraiton Plugin for RDz

*Promote does auto-rescan into RAA of updated source

Sequence of Events


Project Manager
Analyst
Developer
Release Engineer
4e. Create a Deployment Package
4f. Load and Deploy the package to production environment
2j. Verify successful implementation of change request
If errors, open work item, return to step 3a…
1b. Verify completion of enhancement request on production system

Agenda

- Today's Mainframe Development Challenges
- Addressing these challenges with IBM Integrated Solution for System z Development (ISDz)
- One customer's story
  - Overview of current client environment
  - Proposed solution (ISDz)
  - Road to deployment
  - Results
- Summary

Customer Profile

Main Goal Improve Productivity

- Approach: Modernize software development infrastructure

Software Production Software

- Largely mainframe-based applications
- Mostly COBOL
- ChangeMan for source code management

Team In-house and outsourced development

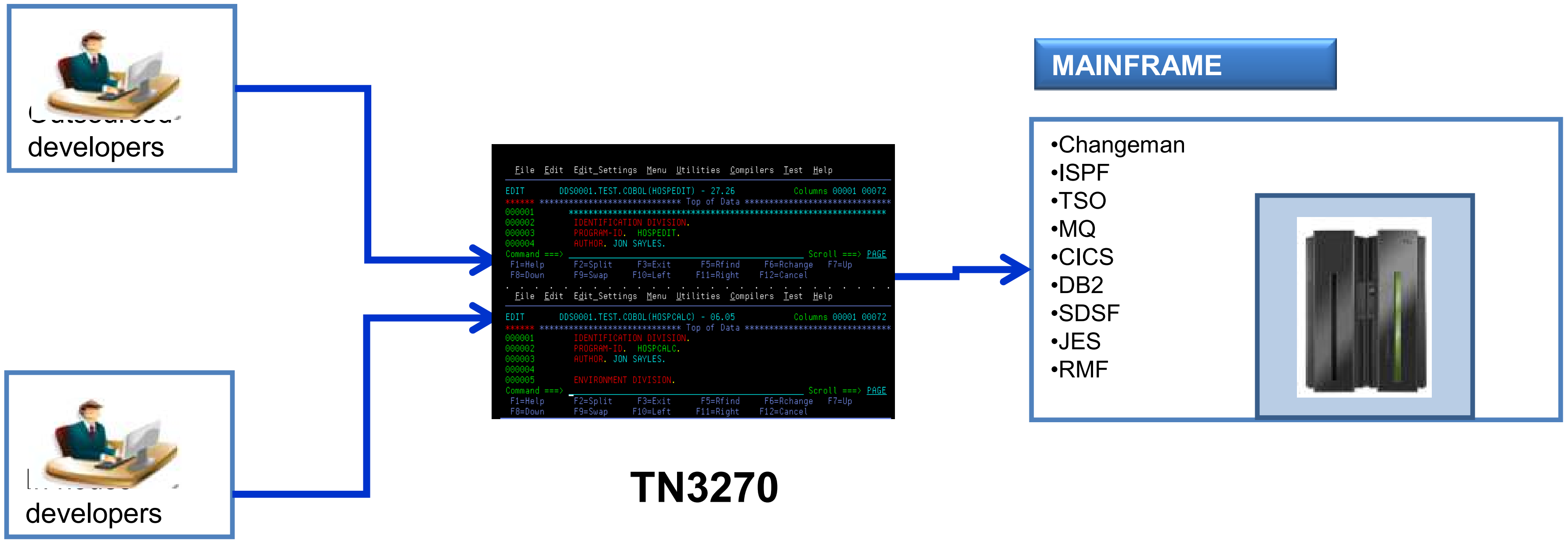
- Development on TSO through ISPF
- 3270 emulators
- Hundreds of developers (at times, thousands of developers)

Environment Typical mainframe development

- Concurrent access
- Development
- Testing
- Quality Management

Current Software Development Environment…

- –Mainframe-based SCM
- –ISPF for development
- –Formal process for change management
- –Customized front end



```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       DDS0001.TEST.COBOL(HOSPEDIT) - 27.26            Columns 00001 00072
****** ***************************** Top of Data ******************************
000001        ****************************************************************
000002         IDENTIFICATION DIVISION.
000003         PROGRAM-ID.  HOSPEDIT.
000004         AUTHOR. JON SAYLES.
Command ===> ________________________________________________ Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       DDS0001.TEST.COBOL(HOSPCALC) - 06.05            Columns 00001 00072
****** ***************************** Top of Data ******************************
000001         IDENTIFICATION DIVISION.
000002         PROGRAM-ID.  HOSPCALC.
000003         AUTHOR. JON SAYLES.
000004
000005         ENVIRONMENT DIVISION.
Command ===> ________________________________________________ Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Business Challenge

Lengthy Software Delivery Life Cycle

Significant degradation in system response time of the development environment during peak hours leads to:

- ✓ Considerable delays in building COBOL components → Slow compilation times
- ✓ Lack of availability of the development environment
- ✓ Slow execution of batch processes

Lack of Quality

Testing process are shortened and the number of unit and functional tests executed is reduced because of:

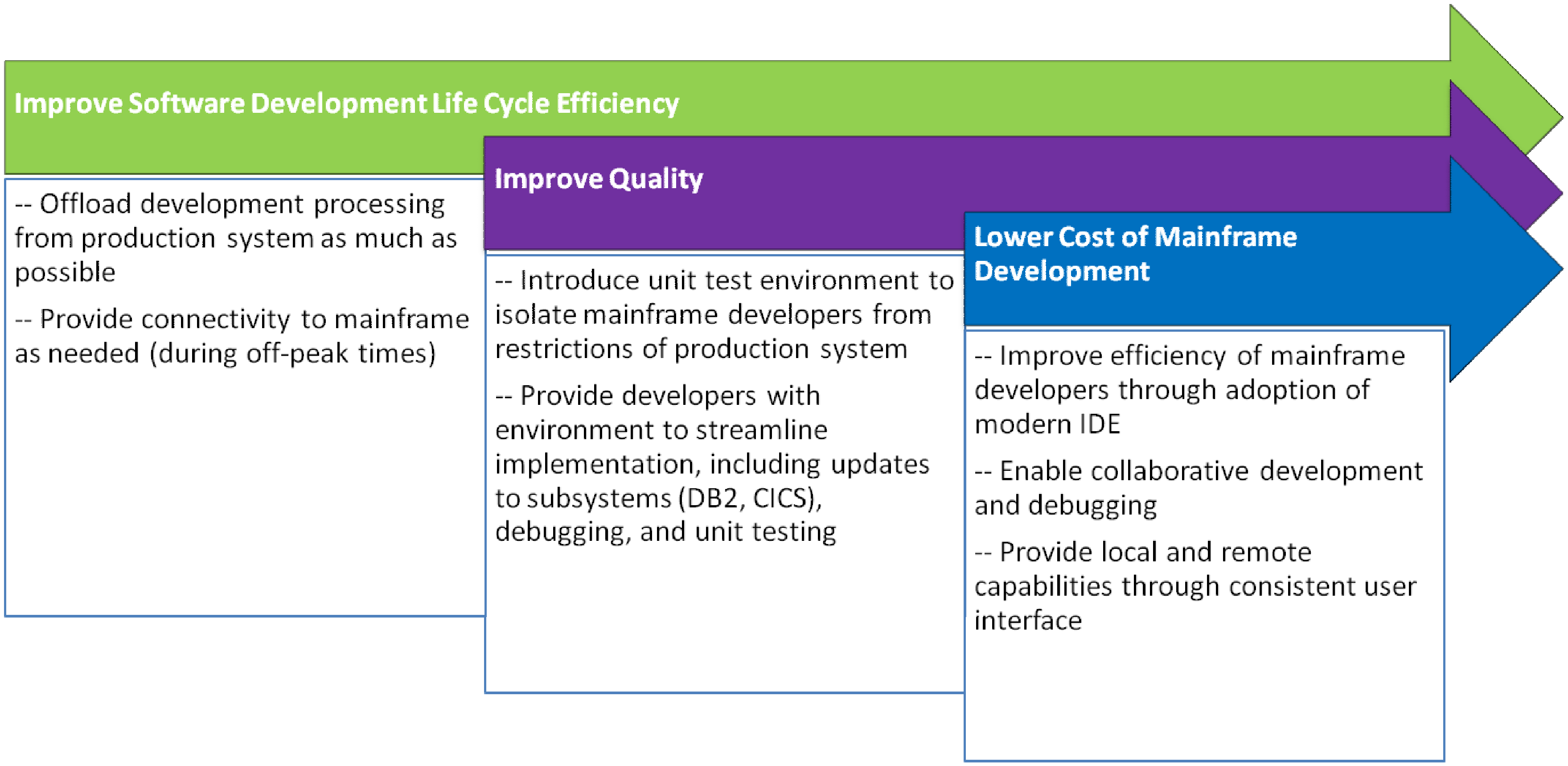
✓Too much time spent during implementation so there is less time to run tests

✓System availability, especially during peak hours, leads to degradation in response time to run tests

High Development Cost / Low Development Productivity:

Overall development cost is increased because it takes more time and resources to complete implementation of the COBOL components

Plan for Improvement



Improve Software Development Life Cycle Efficiency

-- Offload development processing from production system as much as possible

-- Provide connectivity to mainframe as needed (during off-peak times)

Improve Quality

-- Introduce unit test environment to isolate mainframe developers from restrictions of production system

-- Provide developers with environment to streamline implementation, including updates to subsystems (DB2, CICS), debugging, and unit testing

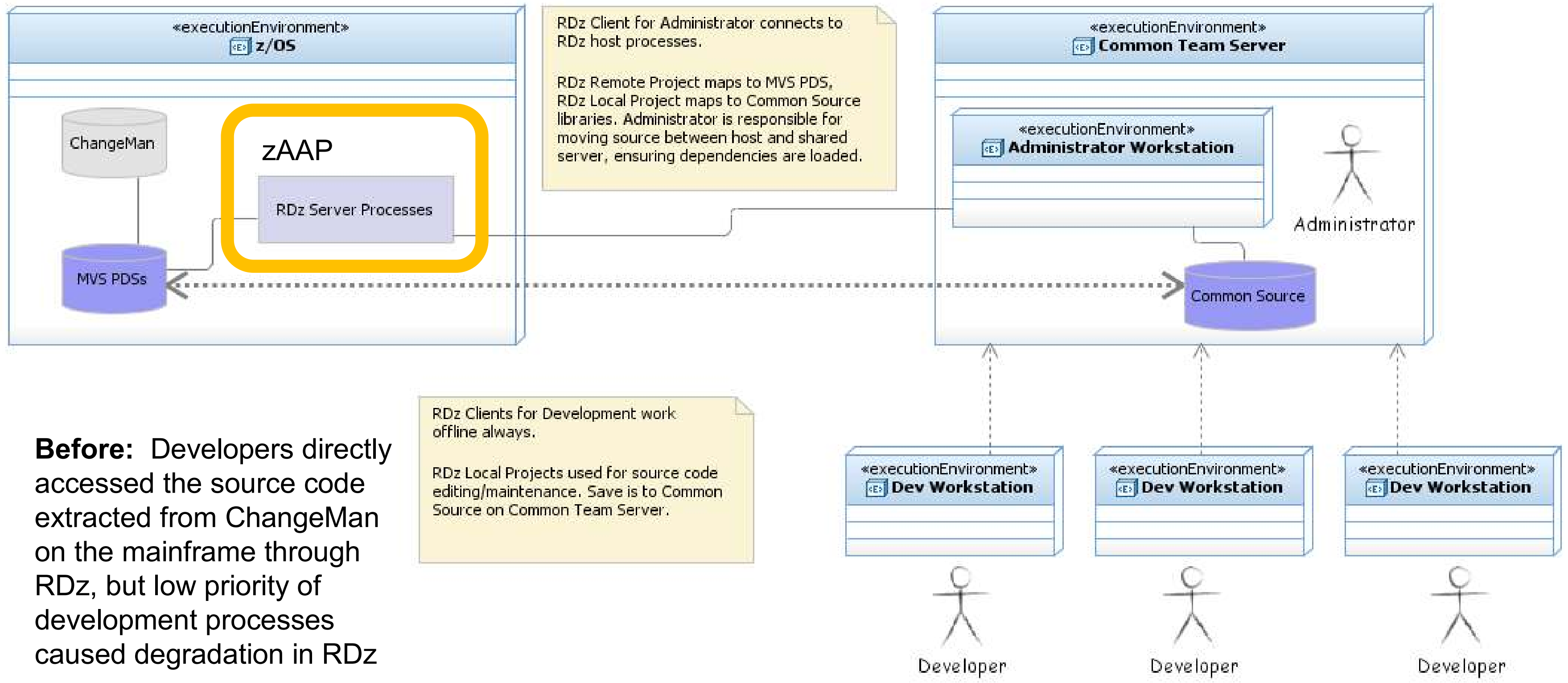
Lower Cost of Mainframe Development

-- Improve efficiency of mainframe developers through adoption of modern IDE

-- Enable collaborative development and debugging

-- Provide local and remote capabilities through consistent user interface

Phase 1 – Adoption of a Enterprise ISDz



RDz Client for Administrator connects to RDz host processes.

RDz Remote Project maps to MVS PDS, RDz Local Project maps to Common Source libraries. Administrator is responsible for moving source between host and shared server, ensuring dependencies are loaded.

RDz Clients for Development work offline always.

RDz Local Projects used for source code editing/maintenance. Save is to Common Source on Common Team Server.

Before: Developers directly accessed the source code extracted from ChangeMan on the mainframe through RDz, but low priority of development processes caused degradation in RDz

After: Administrator accesses the source code on the mainframe, downloads to a Common Source directory accessible by the mainframe developers

Phase 1 –Deployment

Objective:

Implement ISDz to improve efficiency

Benefits Realized:

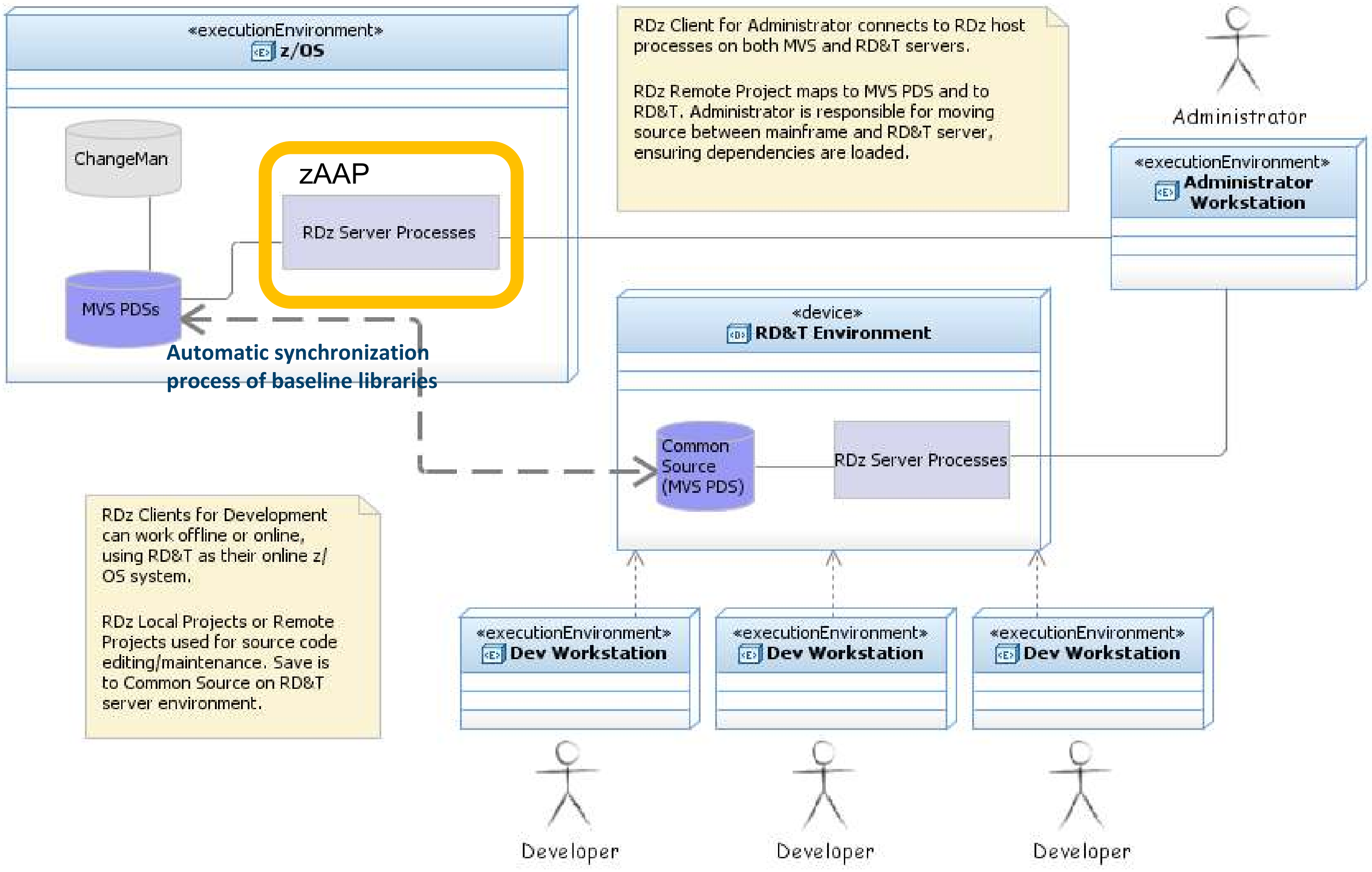
✓Improvements in productivity, specifically in COBOL development measured through:

- Lower actual processing time (MIPS usage)
- Reduced number of days/hours spent on development activities

✓Quality Improvements measured by:

- Reduction in number of defects

Phase 2 – Deployment of off host Z


«executionEnvironment»
z/OS
ChangeMan
zAAP
RDz Server Processes
MVS PDSs
Automatic synchronization process of baseline libraries
RDz Client for Administrator connects to RDz host processes on both MVS and RD&T servers.
RDz Remote Project maps to MVS PDS and to RD&T. Administrator is responsible for moving source between mainframe and RD&T server, ensuring dependencies are loaded.
Administrator
«executionEnvironment»
Administrator Workstation
«device»
RD&T Environment
Common Source (MVS PDS)
RDz Server Processes
RDz Clients for Development can work offline or online, using RD&T as their online z/OS system.
RDz Local Projects or Remote Projects used for source code editing/maintenance. Save is to Common Source on RD&T server environment.
«executionEnvironment»
Dev Workstation
«executionEnvironment»
Dev Workstation
«executionEnvironment»
Dev Workstation
Developer
Developer
Developer

Phase 2 – ISDz and off Host

Objective:

Implement Off Host Z as an additional component of the overall development and build process

Benefits Realized:

✓**Improve delivery**: Developers can apply changes to the databases structures and CICS transactions in the local RD&T environment to complete builds and unit testing

→ No downtime for waiting for systems administration tasks

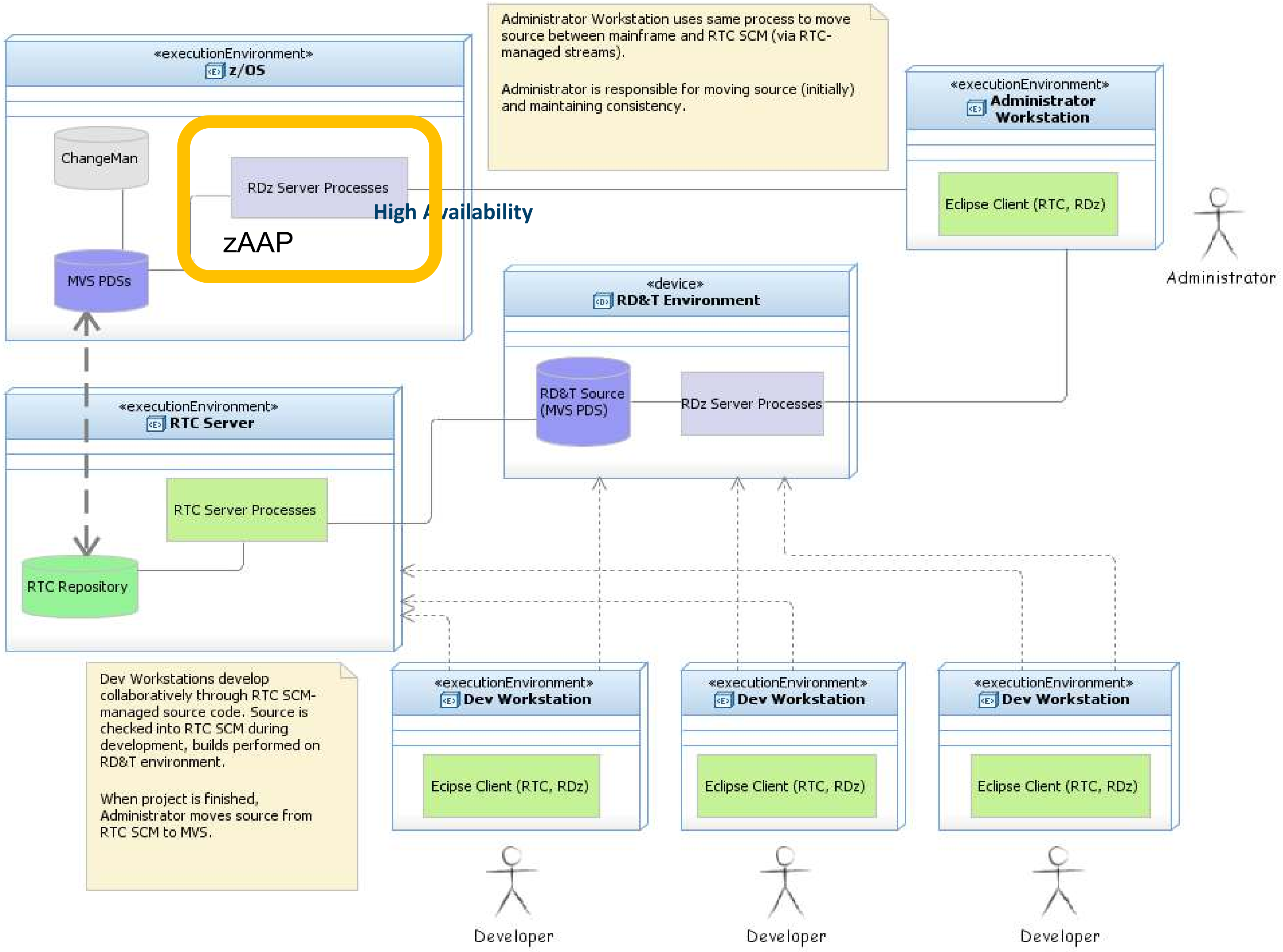
✓**Improve quality of implemented changes:** Developers are available to use debug functionality freely

→ Faster diagnosis of defects

✓**Improve overall quality**: Less space restrictions in the RD&T environment means that larger input files can be used during batch testing

✓**Reduce costs, improve efficiency**: MIPS consumption in mainframe development environment is decreased, leading to lower costs and higher availability of development systems

Phase 3 – Adoption of Team Collaboration



«executionEnvironment» z/OS
ChangeMan
RDz Server Processes
High Availability
zAAP
MVS PDSs
«executionEnvironment» Administrator Workstation
Eclipse Client (RTC, RDz)
Administrator
«device» RD&T Environment
RD&T Source (MVS PDS)
RDz Server Processes
«executionEnvironment» RTC Server
RTC Server Processes
RTC Repository
«executionEnvironment» Dev Workstation
Ecipse Client (RTC, RDz)
«executionEnvironment» Dev Workstation
Eclipse Client (RTC, RDz)
«executionEnvironment» Dev Workstation
Eclipse Client (RTC, RDz)
Developer
Developer
Developer

Administrator Workstation uses same process to move source between mainframe and RTC SCM (via RTC-managed streams).

Administrator is responsible for moving source (initially) and maintaining consistency.

Dev Workstations develop collaboratively through RTC SCM-managed source code. Source is checked into RTC SCM during development, builds performed on RD&T environment.

When project is finished, Administrator moves source from RTC SCM to MVS.

Phase 3 – ISDz with Team Collaboration

Objectives:

✓*Implement source management through Team Collaboration for end to end ISDz environment*

✓*Transform Off Host into a complete testing environment in which developers can run integrated, functional tests using main customer applications and automated test tools*

Expected Benefits:

✓***Improved collaboration**: Developers can work closely, in context, using online reviews, approvals, and threaded discussions*

✓***Shorter delivery cycles**: Developers can work in a flexible, integrated environment without being gated by mainframe availability*

Key Technical/Business Benefits

MIPS Reduction Realized

- ✓ zAAP deployment isolates the Java-based RDz processing, releasing part of the workload of the core mainframe processors
- ✓ Changes in usage model (use of local projects with limited connections to the mainframe) further reduced MIPS consumption

Development Life Cycle Efficiencies

- ✓ Side-by-side comparisons of development scenarios using ISPF and RDz showed significant reduction in development times, demonstrating increased efficiency
- ✓ The impact from the problems due to degradation in response times of the mainframe are lessened
- ✓ Availability of the development environment is improved

Improved Overall Software Delivery Time and Cost

- ✓ The elimination of downtime caused by degradation of the performance of the mainframe leads to a reduction in the total required of man hours
- ✓ Improvements in development efficiency overall leads to improved software delivery times and reduction in development costs

Lessons Learned

- RDz configuration on the mainframe requires planning in order to achieve optimum performance in terms of MIPS consumption
  - Required BOTH a mainframe expert to administer RDz running on the mainframe and an RDz usage expert to teach mainframe developers how to realize the full advantages of the tooling
  - **Key Take-Away: Involve mainframe administrators up front and plan for RDz Education**
- RDz usage alone will not automatically result in reduced development MIPS consumption
  - Use of RDz did not show MIPS savings during periods of high concurrency on the mainframe because the old usage model was still in place although modern tooling was introduced
  - **Key Take-Away: Modernize both the tooling and the working model**
- It is important that developers get support on site during the first days of use of RDz
  - Immediate resolution of questions regarding the use of RDz reinforces the developers confidence in the tools
  - Developers' first instinct is to go back to ISPF rather than wasting time clarifying any questions regarding the use of RDz, hampering RDz adoption
  - **Key Take-Away: Train and assign a group of evangelists to support the wider team early!**

Agenda

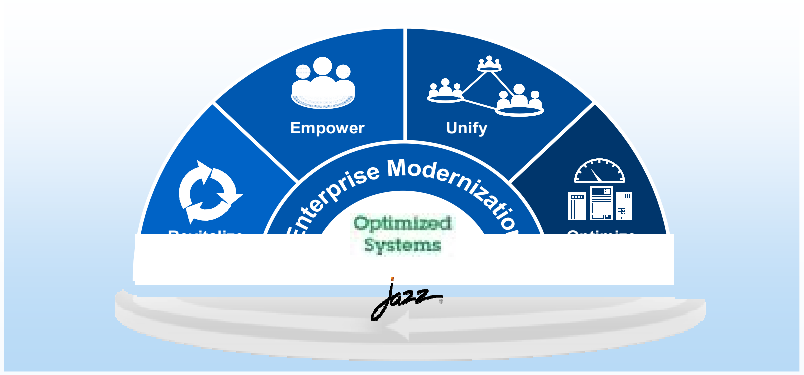
- Today’s Mainframe Development Challenges
- Addressing these challenges with IBM Integrated Solution for System z Development (ISDz)
- One customer’s story
  - Overview of current client environment
  - Proposed solution (ISDz)
  - Road to deployment
  - Results
- Summary

Getting started

Next steps to modernize your enterprise applications

Try the latest System z and Power software for free

Sign up for free web-based training



IBM Rational Cafés
Connecting Communities

Join IBM Rational Cafe Communities

Get prescriptive service solutions

Success stories

Latest news on System z twitter

System z Job Board Your enterprise systems career starts here

Latest skills: System z job board

Latest customer videos

To learn more visit: Rational Enterprise Modernization

QUESTIONS

www.ibm.com/software/rational

IBM.