IBM Rational Software
Development Conference
2008

IBM Rational Build Forge
Extra Value Use Cases for Software Development

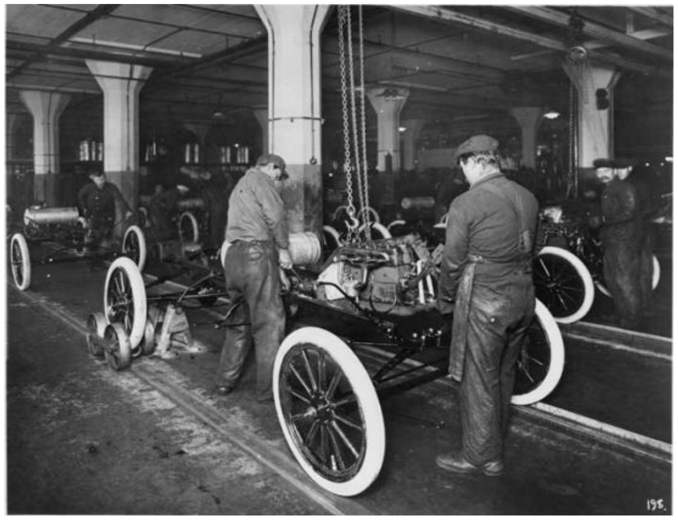
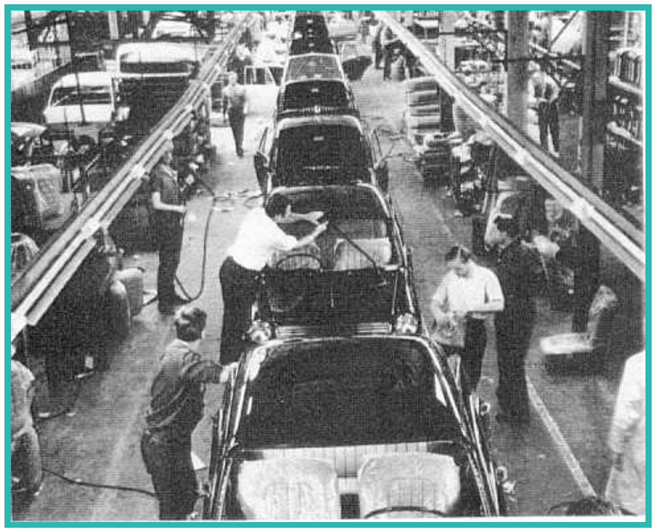
Leigh Williamson
IBM Distinguished Engineer, Rational Software Architecture & Development
leighw@us.ibm.com

C&RM Track – IBM Build Forge Extra Value Use Cases

Rational. software

Go to IBM

Manual → Assembly Line → Automated

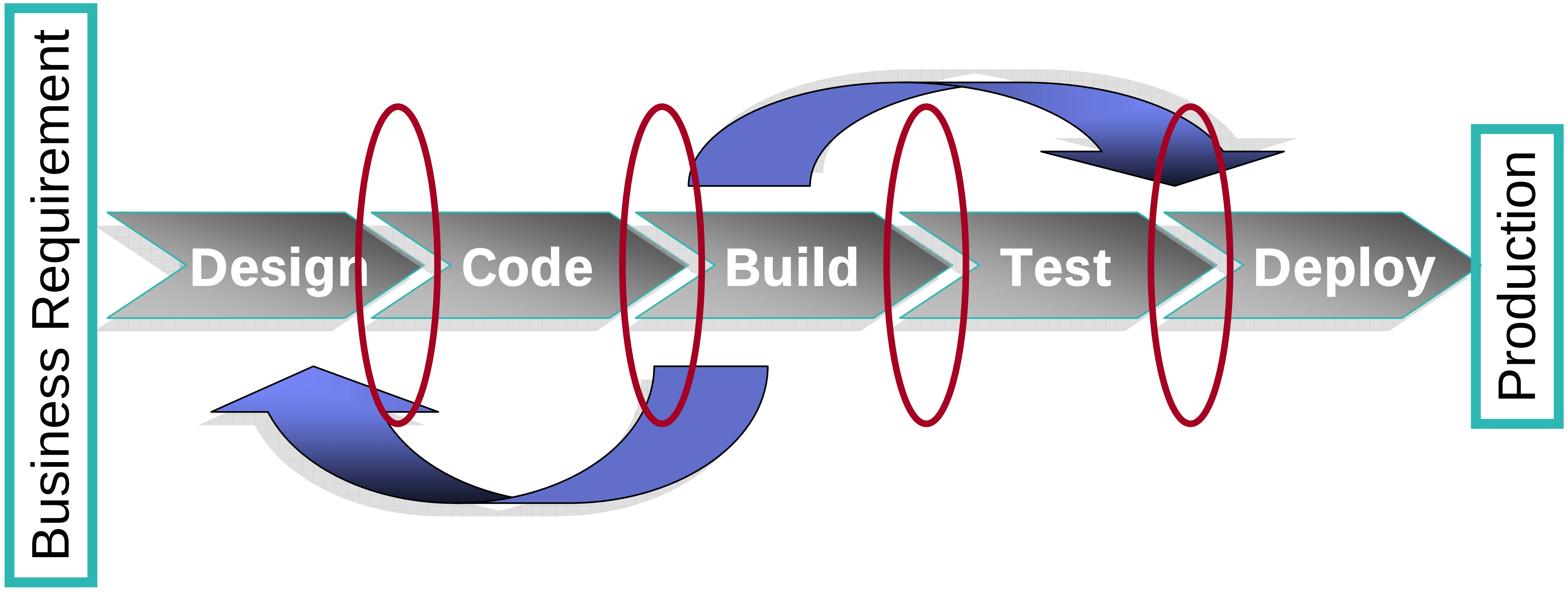



Expensive
Low Productivity
Error Prone
Inconsistent
Resource intensive
Manual Governance

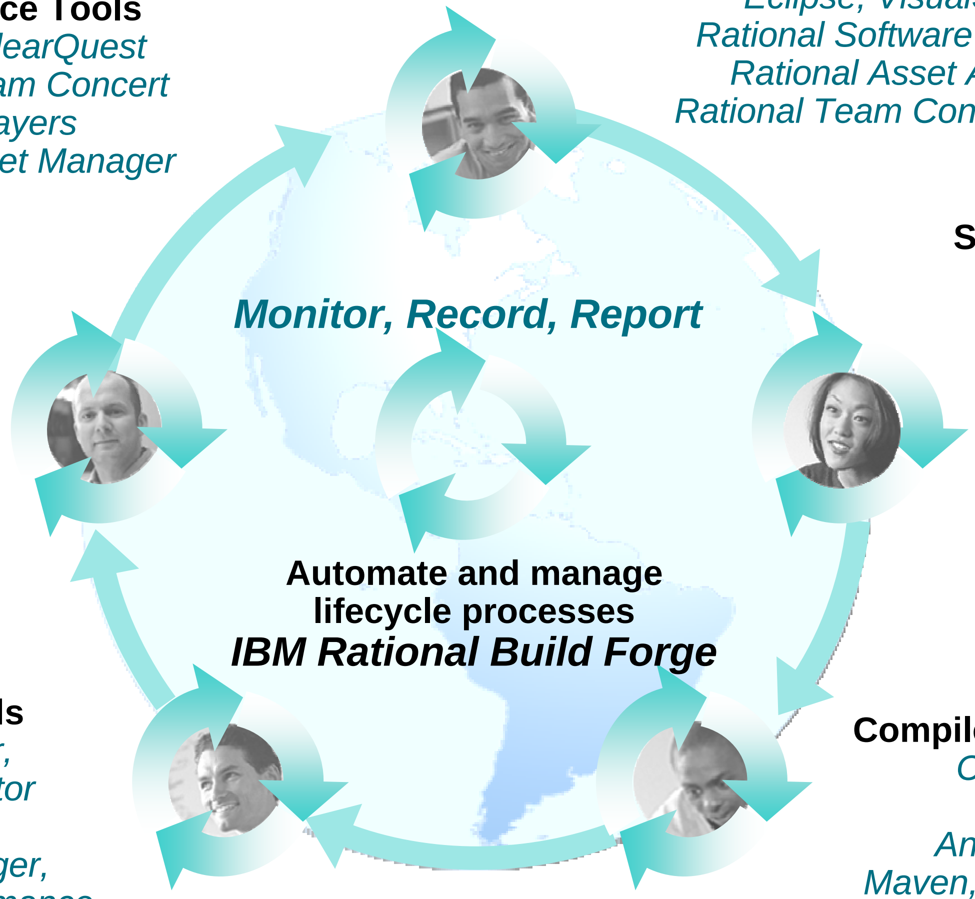
Efficient/Cheaper
High Productivity
High Quality
Consistent/Repeatable
Self Documenting
Automated Governance

Typical Application Development Lifecycle

Automating Scheduling Notifying Logging
Tracking Controlling Analyzing Steps



Build Forge Extensive Tool Integration



Development Tools
Rational Application Developer
Eclipse, VisualStudio
Rational Software Analyzer
Rational Asset Analzer
Rational Team Concert (Jazz)

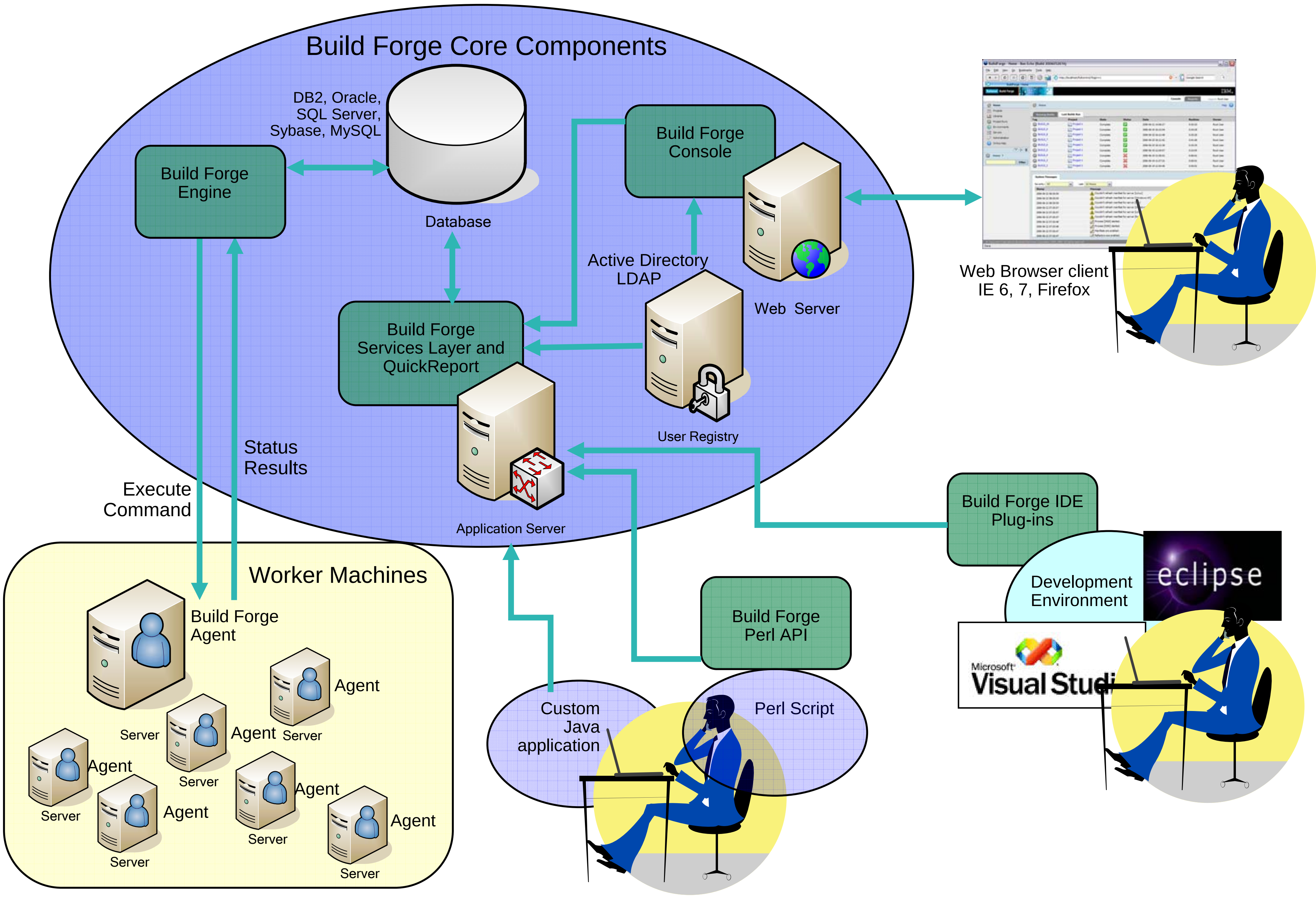
Governance Tools
Rational ClearQuest
Rational Team Concert
WebLayers
Rational Asset Manager

Source Control & Change Management Tools
Rational ClearCase
Rational ClearQuest
StarTeam, Perforce
CVS, PVCS, VSS
Synergy
Subversion
Remedy, Bugzilla
Endevor

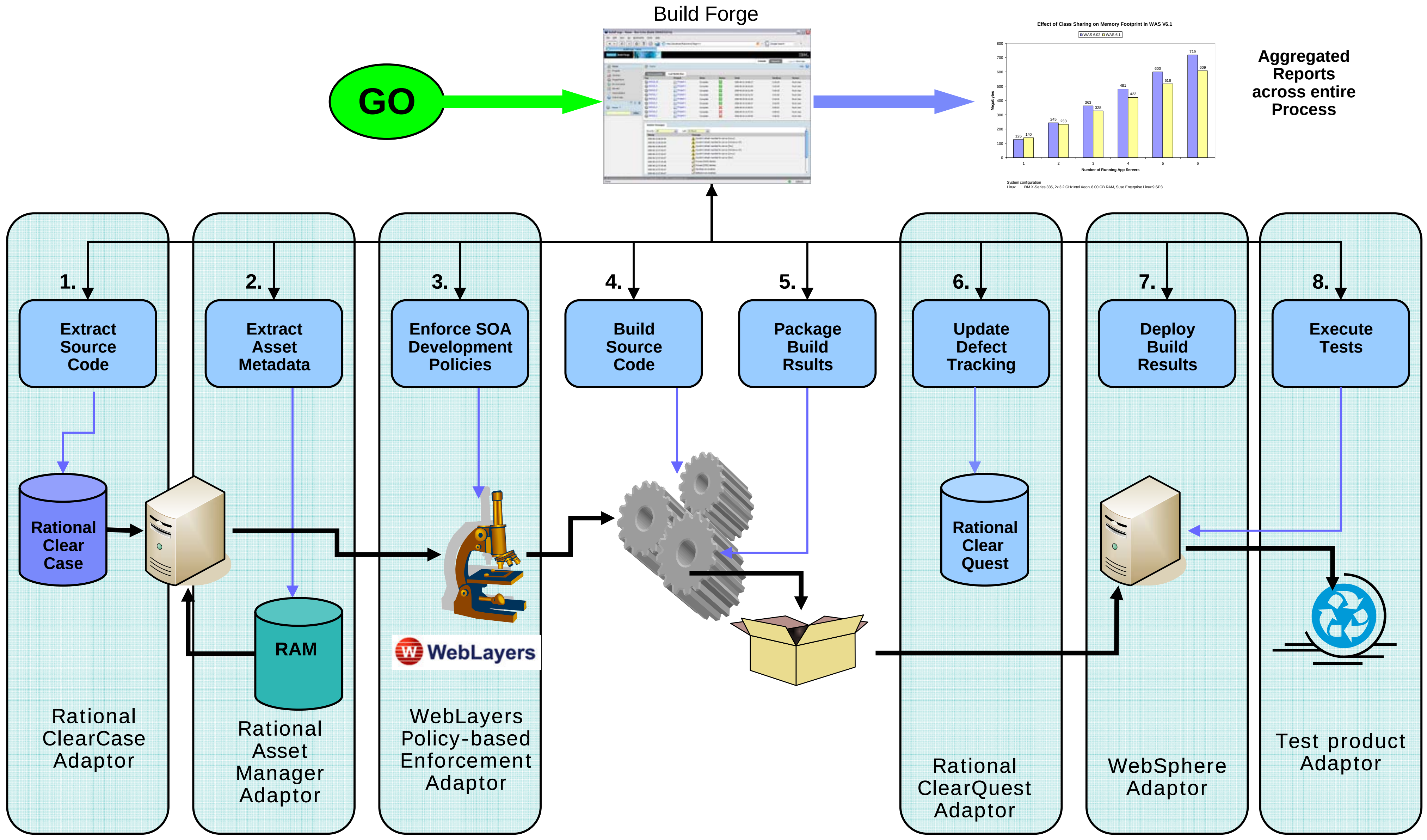
Deployment & Provisioning Tools
Tivoli Provisioning Manager
WebSphere Admin (wsadmin)
Phurnace, VMWare, VMLogix
Ascendant WICA

Compile & Assemble Tools
Compilers (any)
Linkers (all)
Ant, Make, NMake
Maven, Rational ClearMake
Jar, rpm, zip,

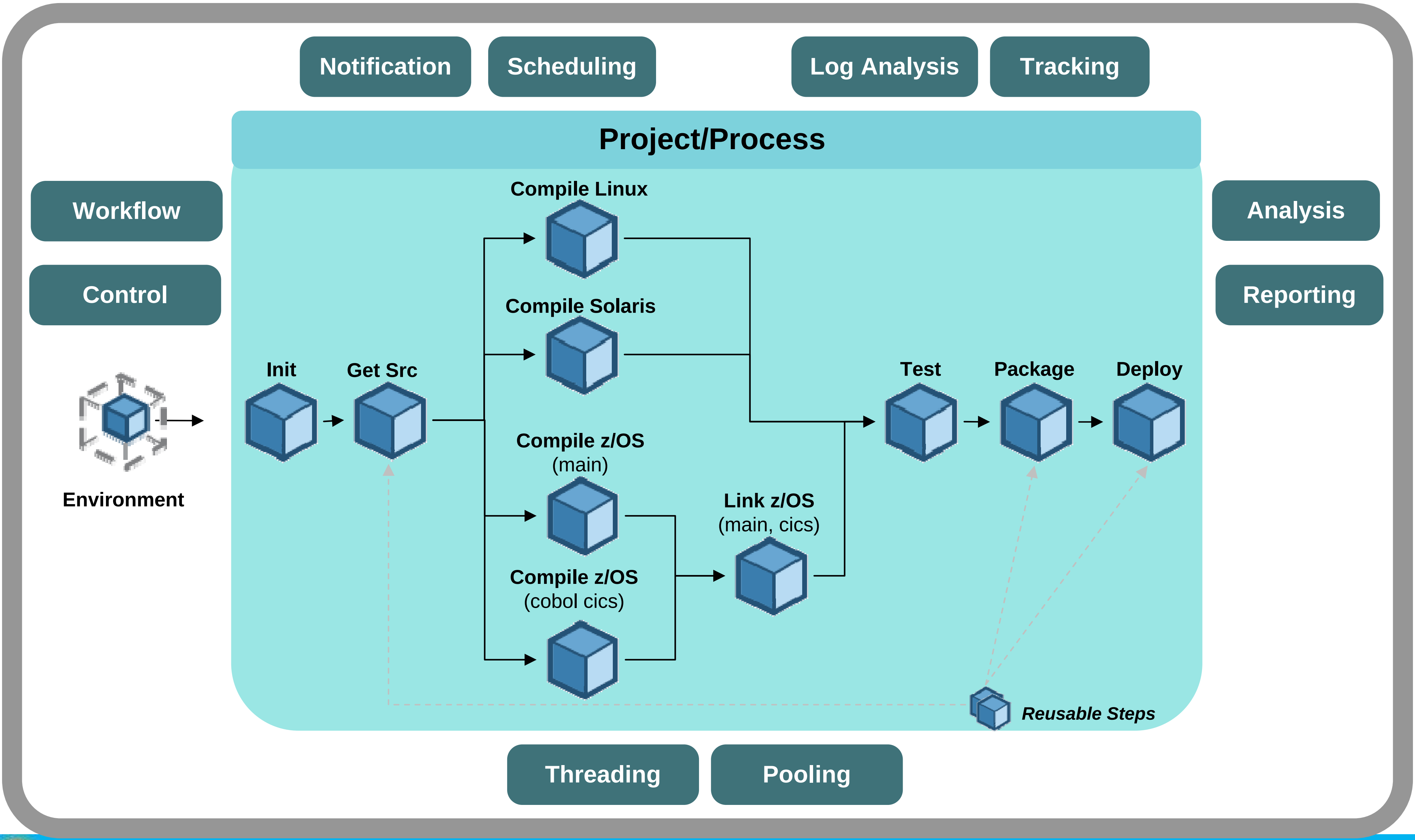
Software Quality Tools
Mecury Quality Center,
LoadRunner, TestDirector
WinRunner, Junit
Rational Test Lab Manager,
Rational Functional, Performance
& Manual Tester


Build Forge Core Components
DB2, Oracle, SQL Server, Sybase, MySQL
Database
Build Forge Engine
Build Forge Console
Web Server
Active Directory LDAP
User Registry
Build Forge Services Layer and QuickReport
Application Server
Status Results
Execute Command
Web Browser client IE 6, 7, Firefox
Build Forge IDE Plug-ins
Development Environment
eclipse
Microsoft Visual Studi
Build Forge Perl API
Custom Java application
Perl Script
Worker Machines
Build Forge Agent
Agent
Server

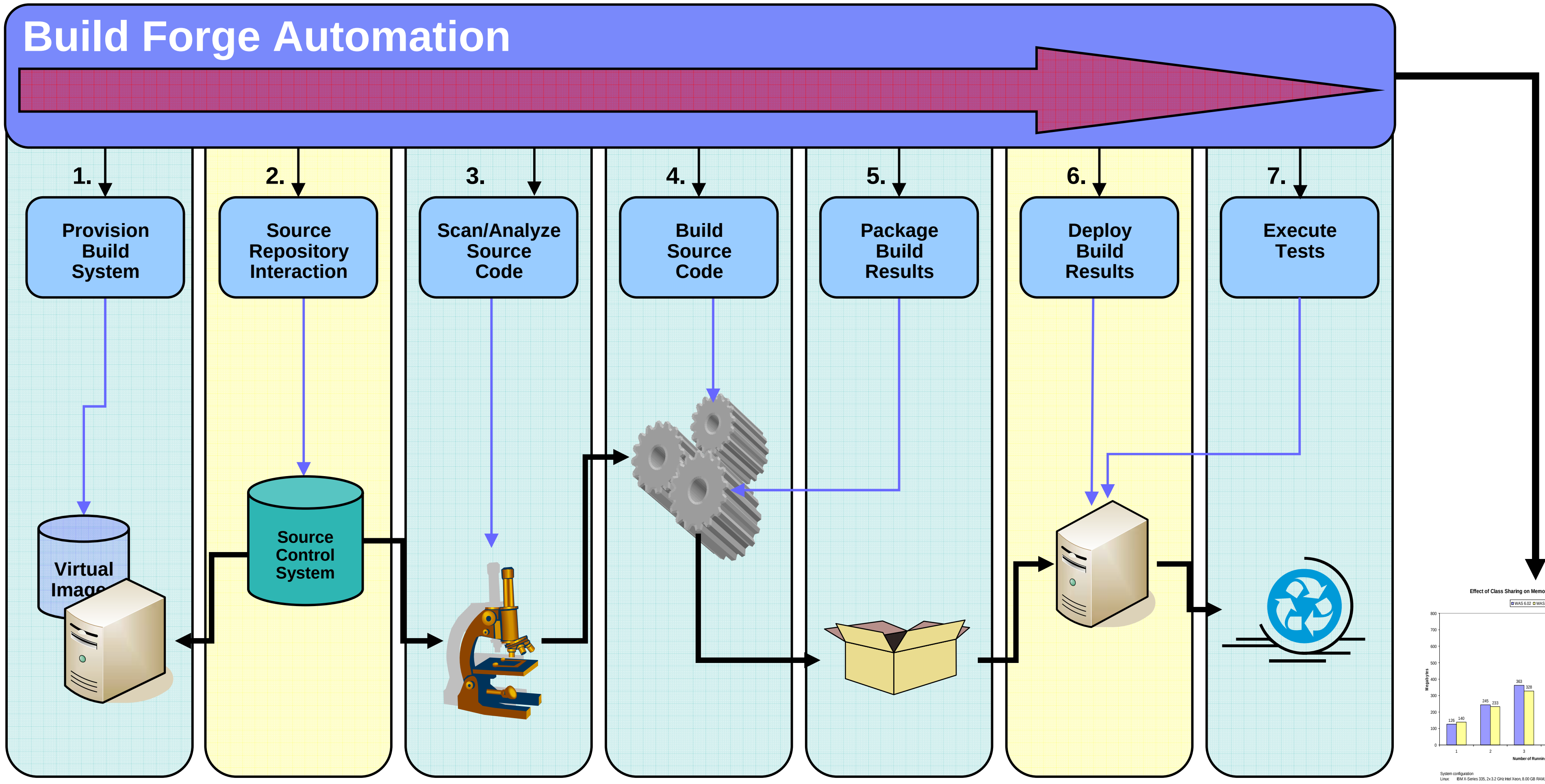
SOA Example Build Forge Automated Process

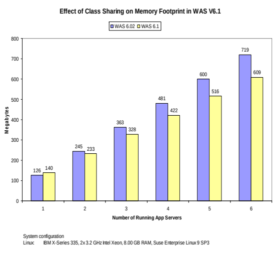

Build Forge
GO
Effect of Class Sharing on Memory Footprint in WAS V6.1
WAS 6.02
WAS 6.1
Megabytes
Number of Running App Servers
1: 126, 140
2: 245, 233
3: 363, 328
4: 481, 422
5: 600, 516
6: 719, 609
System configuration
Linux: IBM X-Series 335, 2x 3.2 GHz Intel Xeon, 8.00 GB RAM, Suse Enterprise Linux 9 SP3
Aggregated Reports across entire Process
1. Extract Source Code
2. Extract Asset Metadata
3. Enforce SOA Development Policies
4. Build Source Code
5. Package Build Rsults
6. Update Defect Tracking
7. Deploy Build Results
8. Execute Tests
Rational Clear Case
RAM
WebLayers
Rational Clear Quest
Rational ClearCase Adaptor
Rational Asset Manager Adaptor
WebLayers Policy-based Enforcement Adaptor
Rational ClearQuest Adaptor
WebSphere Adaptor
Test product Adaptor

Example: SOA Composite Application



Development Automation with Build Forge





System configuration
Linux: IBM X-Series 335, 2x 3.2 GHz Intel Xeon, 8.00 GB RAM, Suse Enterprise Linux 9 SP3

Build Forge Agents or Virtualization Support

Execute ClearCase, CVS, SVN, Synergy, etc. commands

Rational Software Analyzer, etc.

ObjectMake ClearMake, Ant, Maven, etc.

Wise, Install Shield, RPM, JAR, WAR, etc.

Simple deploy or BF WebSphere Framework, etc.

Test Manager, Robot, Functional, etc.

Aggregated Reports across entire process

Types of Static Software Analysis

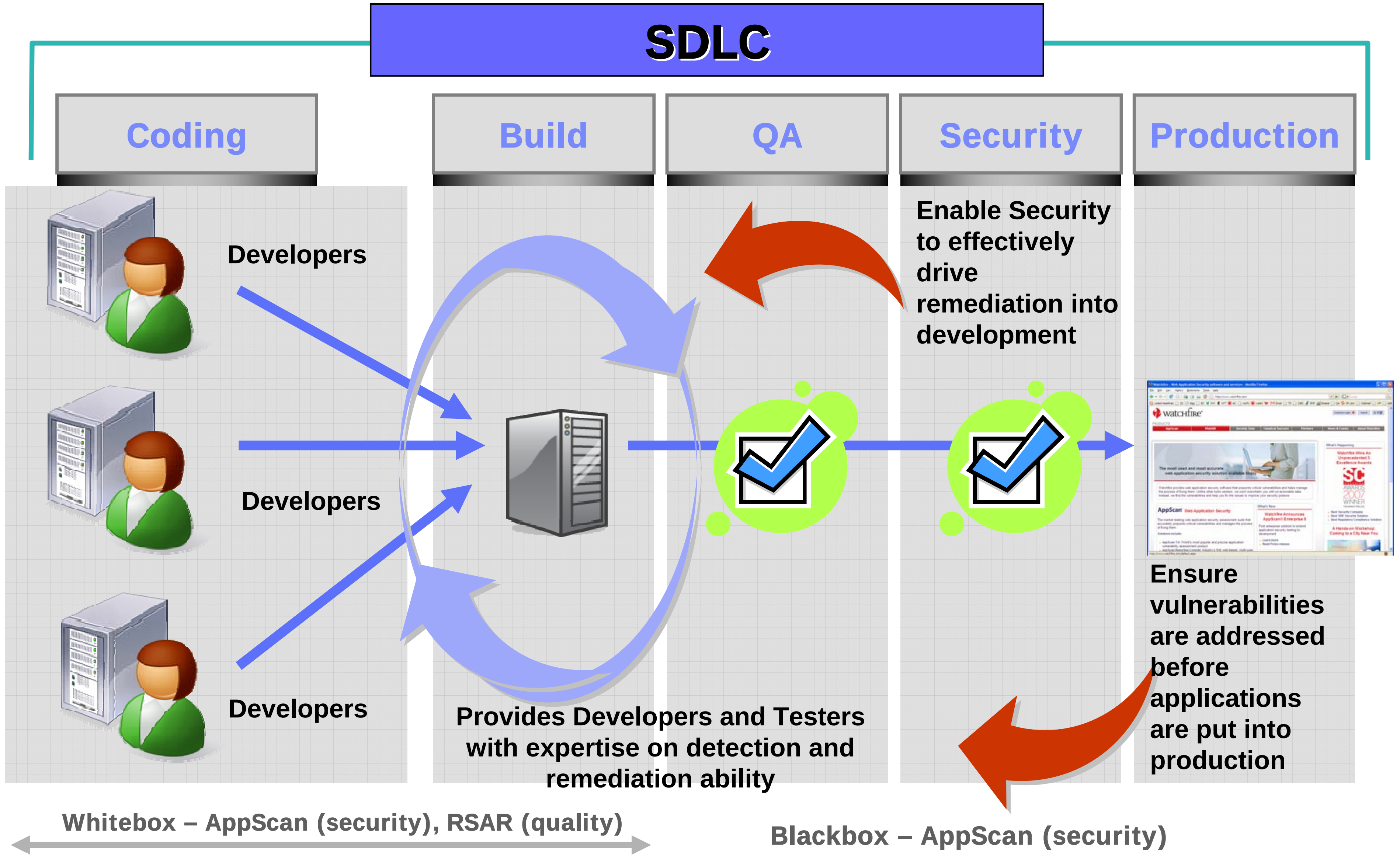
- **Code review**
  - Generally concerned with coding style
  - Find rudimentary bugs on a per class basis
- **Structural analysis**
  - Inter-class dependency
  - Find cyclical dependencies, hubs, etc.
- **Software metrics**
  - Measures the complexity of software
  - Lots of standard metrics (line counting, McCabbe, Halstead)
- **Trend Analysis**
  - Measure quality over time
  - Is the code getting better or worse?

Lots of things to analyze in the source code

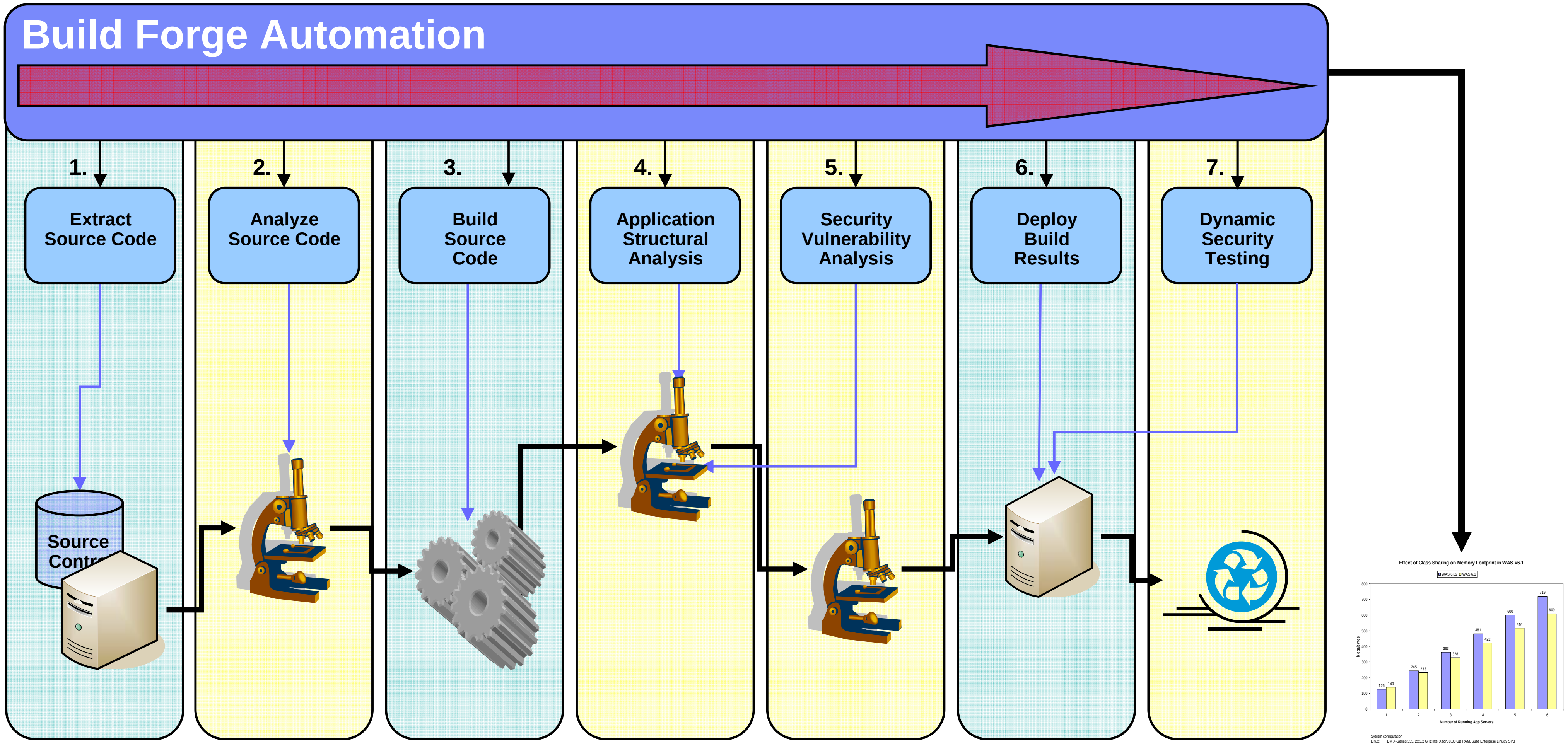
- **Defects in the code**
  - **Different languages – Java, C, C++, C#, COBOL, …**
- **Corporate coding conventions & policies**
- **IP issues, open source content**
- **Security exposures**
- **Discover program structure**
  - **Build dependencies, data flow, call graphs**
- **Program specifications**
- **Test coverage & test generation**

Building security & compliance into the SDLC

Leveraging Whitebox and Blackbox technologies


SDLC
Coding
Build
QA
Security
Production
Developers
Developers
Developers
Enable Security to effectively drive remediation into development
Ensure vulnerabilities are addressed before applications are put into production
Provides Developers and Testers with expertise on detection and remediation ability
Whitebox – AppScan (security), RSAR (quality)
Blackbox – AppScan (security)

Application Intelligence & Analysis Automation with Build Forge


Build Forge Automation
1. Extract Source Code
2. Analyze Source Code
3. Build Source Code
4. Application Structural Analysis
5. Security Vulnerability Analysis
6. Deploy Build Results
7. Dynamic Security Testing
Source Control
Effect of Class Sharing on Memory Footprint in WAS V6.1
WAS 6.02
WAS 6.1
Megabytes
Number of Running App Servers
126
140
245
233
363
328
481
422
600
516
719
609
System configuration
Linux: IBM X-Series 335, 2x 3.2 GHz Intel Xeon, 8.00 GB RAM, Suse Enterprise Linux 9 SP3

Rational ClearCase Adaptor

Rational Software Analyzer

Ant, Maven, Make, etc.

Rational Asset Analyzer

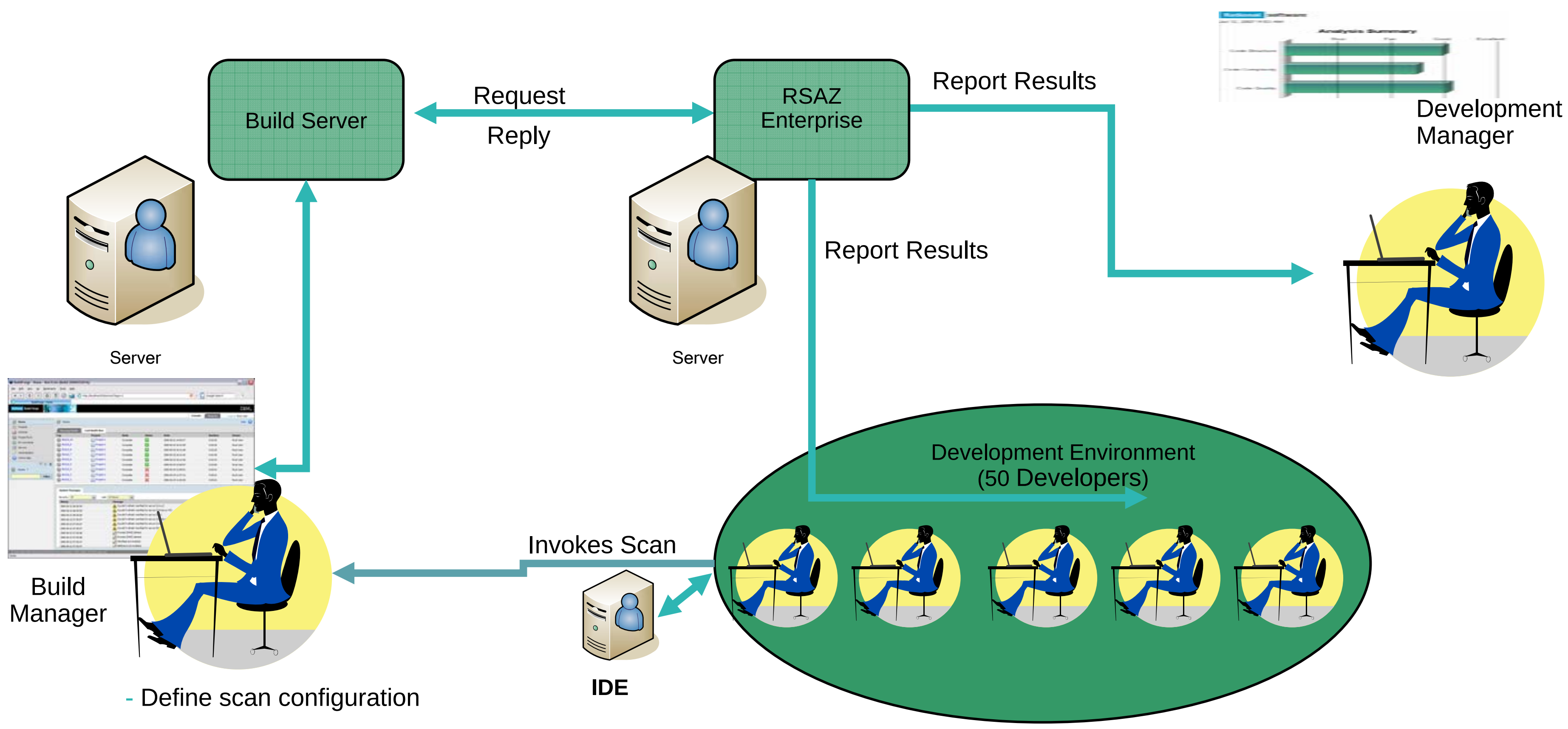
Rational AppScan Build Edition

Simple deploy, Tivoli tools, BFWS Framework, etc.

Rational Test tools, Rational AppScan

Aggregated Reports across entire process

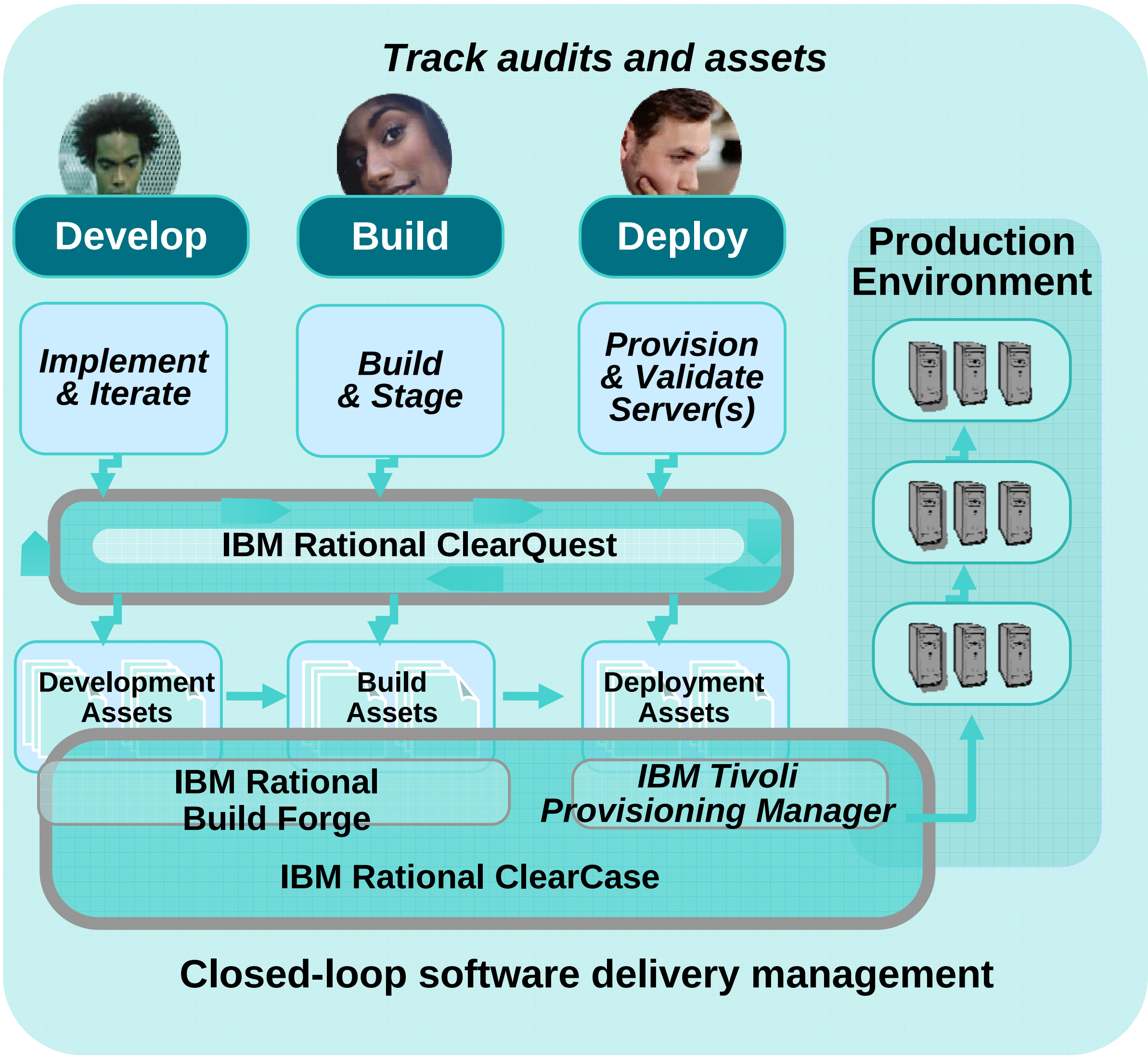
Software Analyzer and Build Forge Integration



- Define scan configuration
- Initiate build or analysis job
- Report results

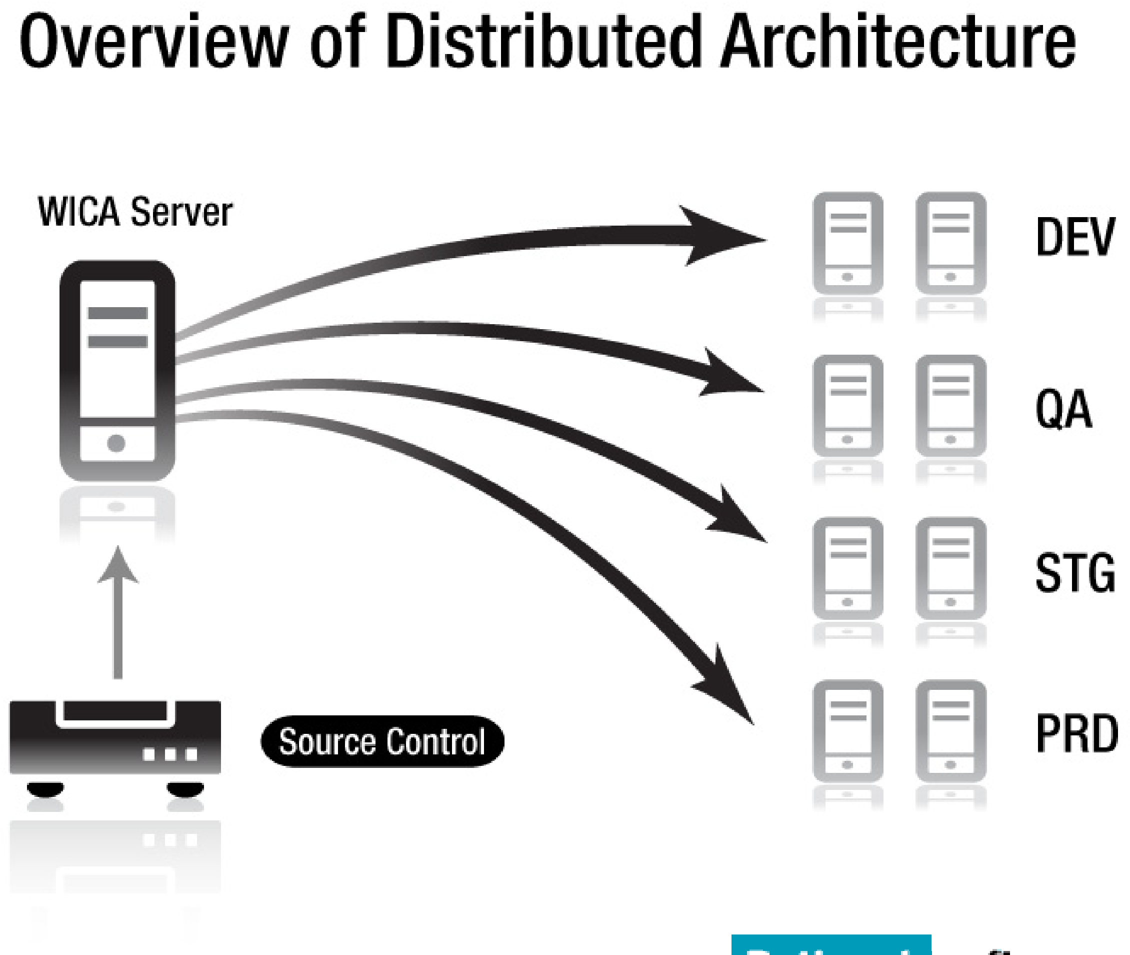
Bridging Between Development & Operations

- **Manage** with ClearQuest (activities) and Clear Case (artifacts)
  - Define the test environments, roles and approvals needed prior to deployment
  - Manage builds and deployments for each "Release" of the application
  - Retain build artifacts in a secure repository
  - Create deployment units – which managed artifacts to deploy
- **Automate** with Build Forge and Tivoli Provisioning Manager
  - Build Forge orchestration & automation of deployment artifact creation
  - Build Forge automates deployment to developer test environments
  - Tivoli provisioning of deployment artifacts to production test environments using deployment record and deployment unit
- **Track** with ClearQuest
  - Link build results to activities in ClearQuest
  - Link Deployment units to deployment records in ClearQuest
  - Provides audit trail of all defined build, test, repair, deployment & approval processes



Build Forge Framework for WebSphere Installation, Configuration, Administration

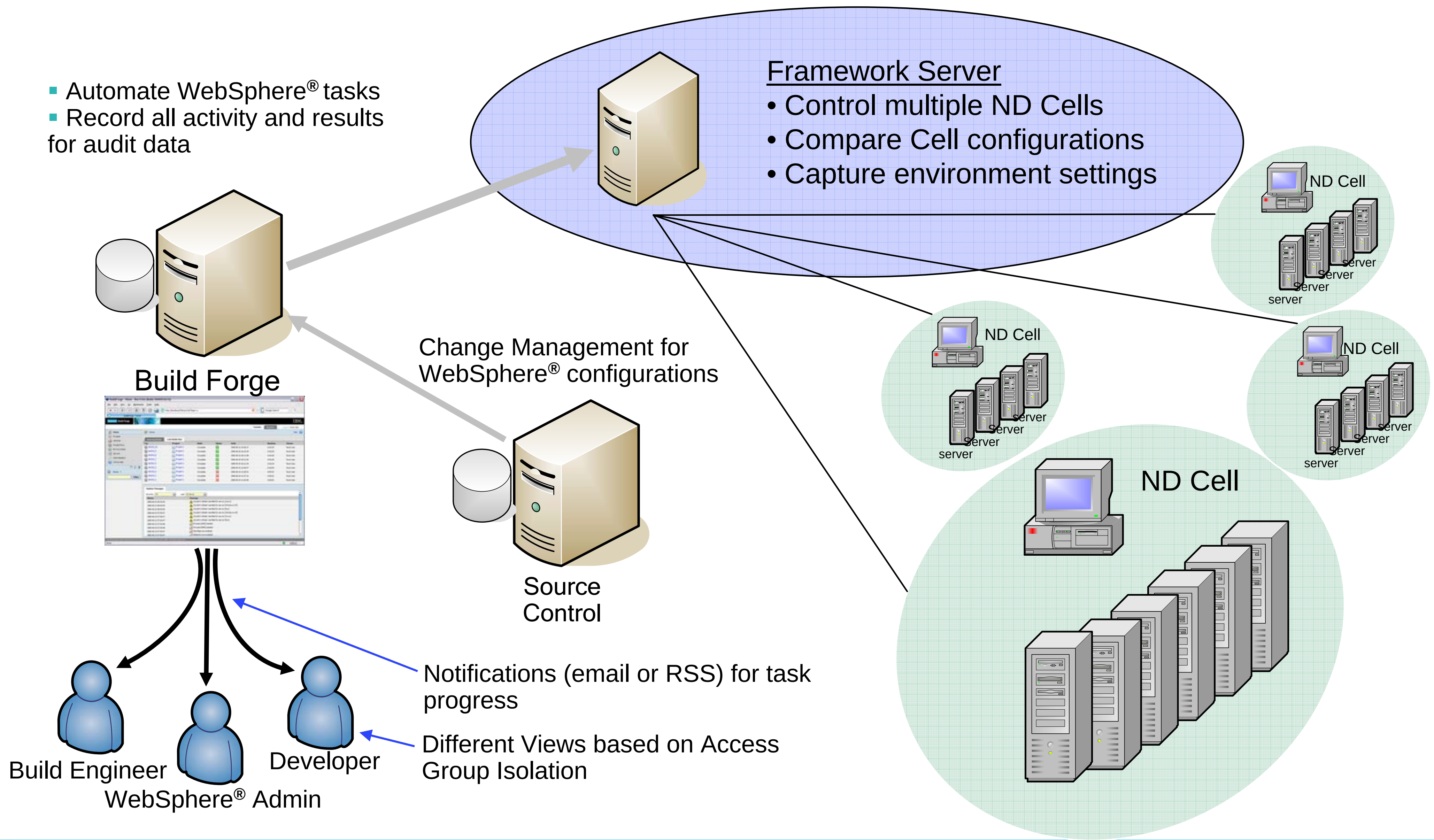
- Multi-cell WebSphere tool
- Environment build out
- Configuration capture
- Application deployment
- Change management for WebSphere configuration
- Configuration comparison



WebSphere software

Rational software

Build Forge Framework for WebSphere® Topology



▪ Automate WebSphere® tasks
▪ Record all activity and results for audit data
Framework Server
• Control multiple ND Cells
• Compare Cell configurations
• Capture environment settings
Build Forge
Change Management for WebSphere® configurations
Source Control
Notifications (email or RSS) for task progress
Different Views based on Access Group Isolation
Build Engineer
WebSphere® Admin
Developer
ND Cell
server
Server
Server
server
ND Cell
server
Server
Server
server
ND Cell
server
Server
Server
server
ND Cell

Build Forge / RAM Integration Use Case

Build artifacts are collected from Rational Asset Manager

a. Build is triggered through some mechanism (manual or programmatic)

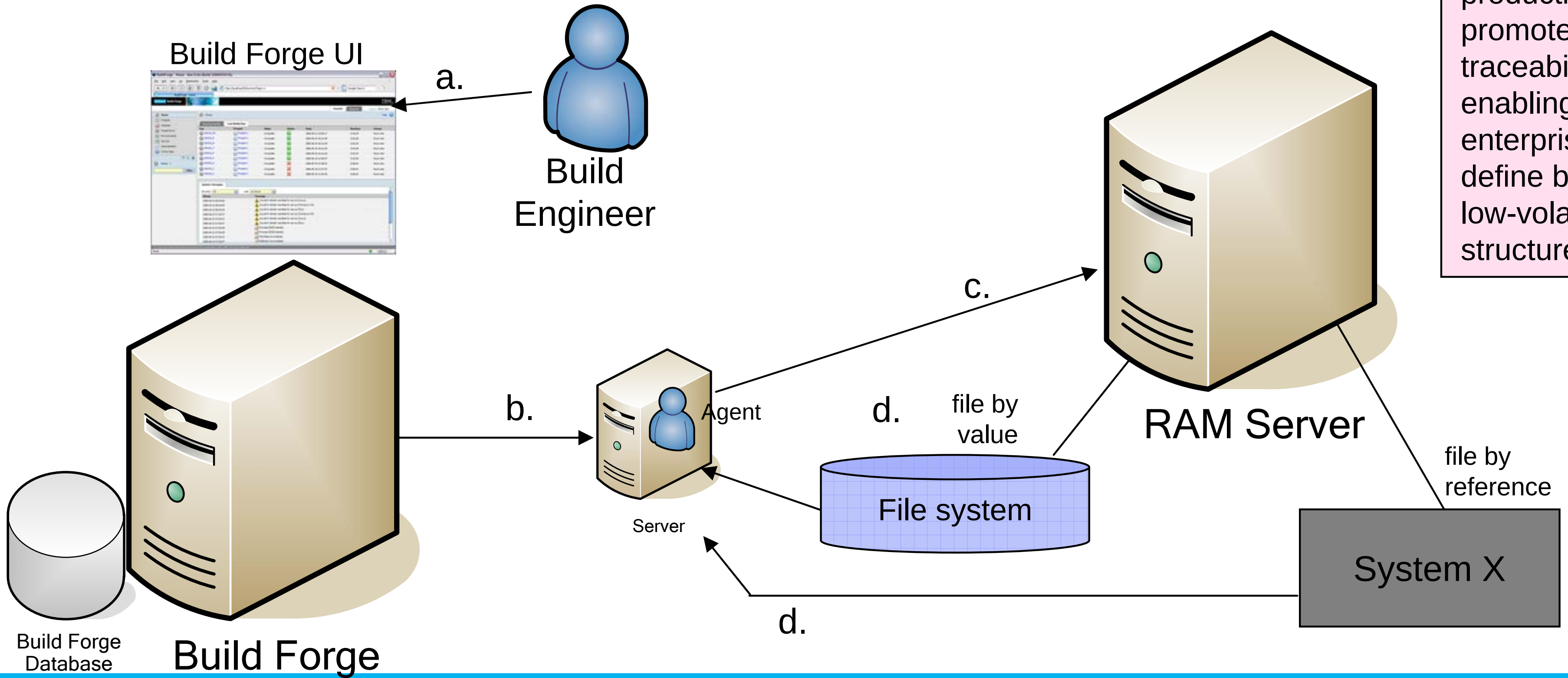
b. Build steps are dispatched to agent system with RAM client

c. Build process begins and first step is querying RAM for files and placing onto a build machine

d. Build artifacts are made available to agent system by RAM

Note that the trigger for a build does not have to be a human action. Builds can be triggered by a change to a monitored file or by programmatic invocation of Build Forge external APIs or CLI.

Rationale: Improves build productivity and promotes traceability by enabling the enterprise to define build for low-volatile structures.



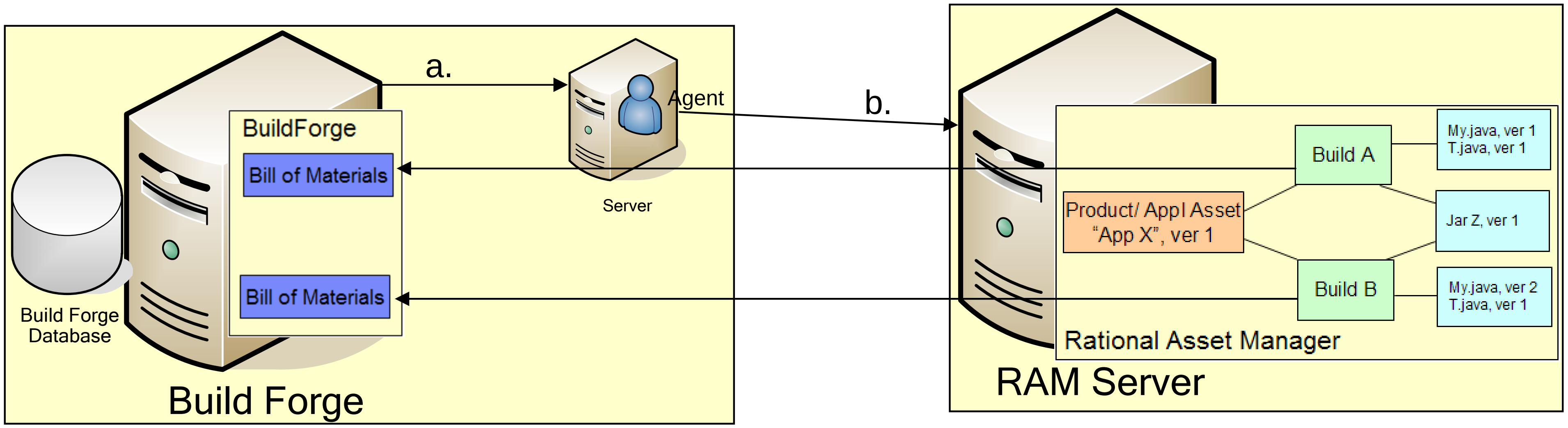
Build Forge / RAM Use Case

Build executes – updating RAM metadata

a. Build executes, steps instruct agent to update information in RAM

b. As build completes, a collection of RAM build assets are created/updated with

1. A relationship to a product/application asset
2. A relationship to a previous build asset
3. Relationships to the contained assets
4. Metadata information
5. Development/runtime context information, tools
6. Reference to BF Bill of Materials (BOM)
7. Code scans



Build Forge Release 7.0.1 Highlighted Feature – New IBM Platform Support

- Added agent support for legacy IBM platforms – System i and System z
  - System z agent runs in Unix System Services (USS) environment under z/OS
  - Built-in REXX support
  - Allows for native system commands to be executed
- Coordinate and execute your software processes across all your platforms from a centralized location, with complete visibility and real-time status

Use cases for System z

Build Forge 7.0.1

- Current Use Cases
  - COBOL
  - COBOL/CICS, COBOL/DB2
  - COBOL/CICS/DB2
  - COBOL/COPYBOOK
  - Java (javac & Ant)
  - WebSphere EAR
  - Combination (Java & COBOL)

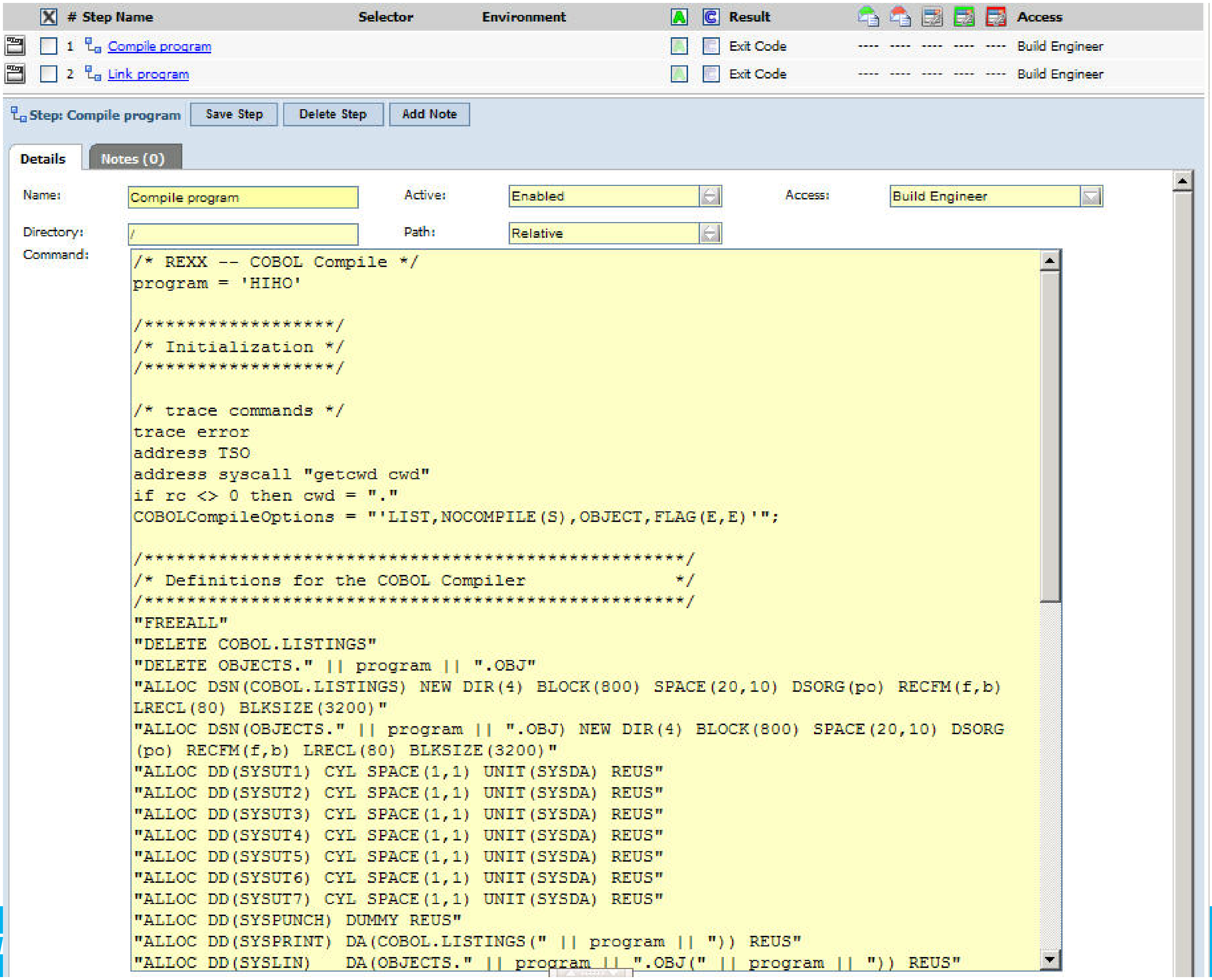
- Post-GA Use Cases
  - C, C++, PL/I, HLASM
  - EGL / COBOL
  - SCLM / DevTk
  - Rational Function Tester
  - WebSphere deploy

IBM Rational Build Forge

Use Cases for IBM System Z Platform

Leigh Williamson, STSM, IBM Rational BuildForge Architect

Native COBOL compile step (REXX script)



	#	Step Name	Selector	Environment	Result	Access
	1	Compile program			Exit Code	---- ---- ---- ---- ---- Build Engineer
	2	Link program			Exit Code	---- ---- ---- ---- ---- Build Engineer

Step: Compile program

Name:	Compile program	Active:	Enabled	Access:	Build Engineer
Directory:	/	Path:	Relative		

Command:

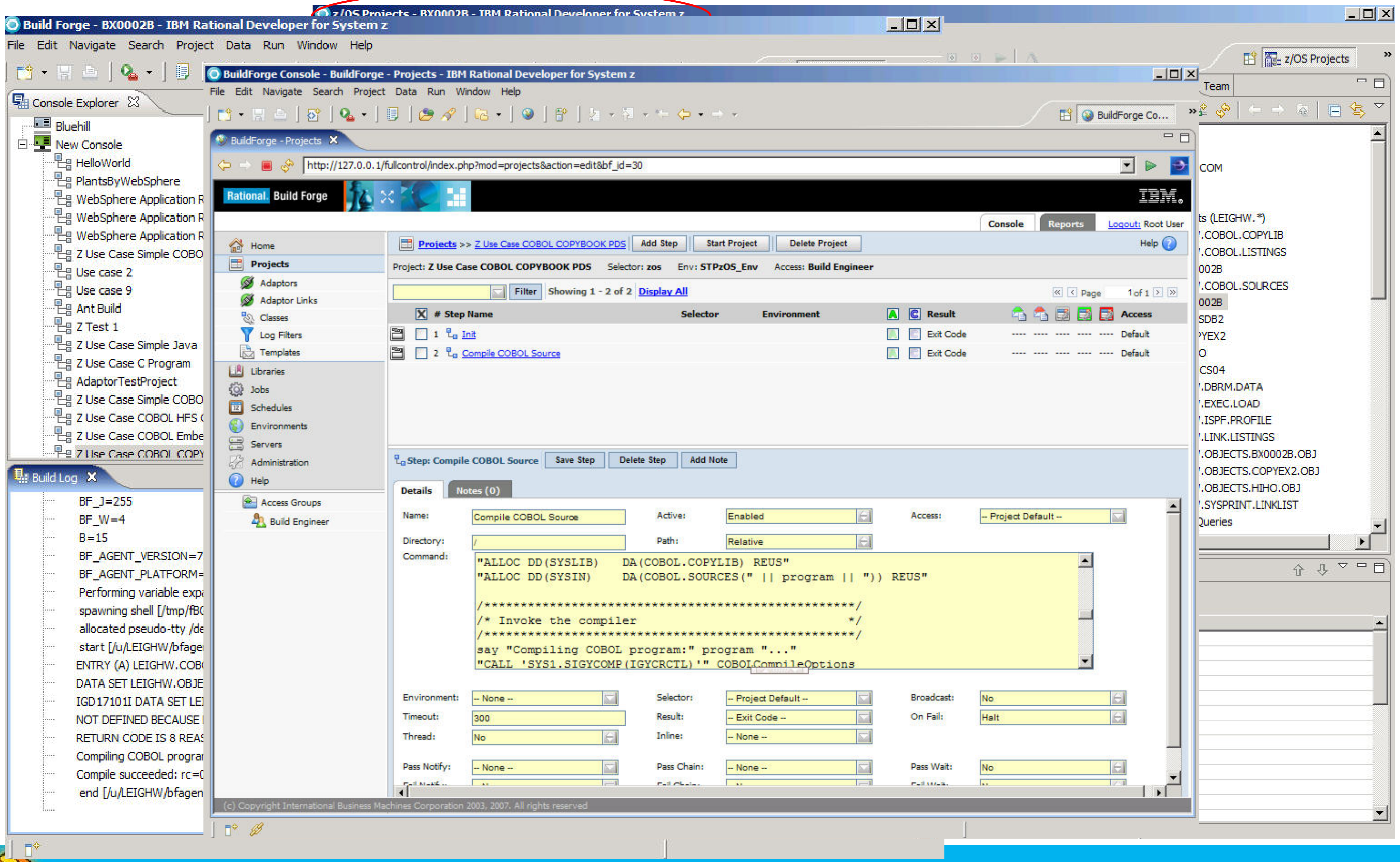
```
/* REXX -- COBOL Compile */
program = 'HIHO'

/******************/
/* Initialization */
/******************/

/* trace commands */
trace error
address TSO
address syscall "getcwd cwd"
if rc <> 0 then cwd = "."
COBOLCompileOptions = "'LIST,NOCOMPILE(S),OBJECT,FLAG(E,E)'";

/****************************************************/
/* Definitions for the COBOL Compiler              */
/****************************************************/
"FREEALL"
"DELETE COBOL.LISTINGS"
"DELETE OBJECTS." || program || ".OBJ"
"ALLOC DSN(COBOL.LISTINGS) NEW DIR(4) BLOCK(800) SPACE(20,10) DSORG(po) RECFM(f,b)
LRECL(80) BLKSIZE(3200)"
"ALLOC DSN(OBJECTS." || program || ".OBJ) NEW DIR(4) BLOCK(800) SPACE(20,10) DSORG
(po) RECFM(f,b) LRECL(80) BLKSIZE(3200)"
"ALLOC DD(SYSUT1) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT2) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT3) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT4) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT5) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT6) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSUT7) CYL SPACE(1,1) UNIT(SYSDA) REUS"
"ALLOC DD(SYSPUNCH) DUMMY REUS"
"ALLOC DD(SYSPRINT) DA(COBOL.LISTINGS(" || program || ")) REUS"
"ALLOC DD(SYSLIN)   DA(OBJECTS." || program || ".OBJ(" || program || ")) REUS"
```

Development Tool Integration



http://127.0.0.1/fullcontrol/index.php?mod=projects&action=edit&bf_id=30

Bluehill
New Console
HelloWorld
PlantsByWebSphere
WebSphere Application R
WebSphere Application R
WebSphere Application R
Z Use Case Simple COBO
Use case 2
Use case 9
Ant Build
Z Test 1
Z Use Case Simple Java
Z Use Case C Program
AdaptorTestProject
Z Use Case Simple COBO
Z Use Case COBOL HFS (
Z Use Case COBOL Embe
Z Use Case COBOL COPY

Projects >> Z Use Case COBOL COPYBOOK PDS

Project: Z Use Case COBOL COPYBOOK PDS Selector: zos Env: STPzOS_Env Access: Build Engineer

Showing 1 - 2 of 2 Display All Page 1 of 1

#	Step Name	Selector	Environment	Result	Access
1	Init			Exit Code	Default
2	Compile COBOL Source			Exit Code	Default

Step: Compile COBOL Source

Name:	Compile COBOL Source	Active:	Enabled	Access:	-- Project Default --
Directory:	/	Path:	Relative		

Command:

```
"ALLOC DD(SYSLIB)   DA(COBOL.COPYLIB) REUS"
"ALLOC DD(SYSIN)    DA(COBOL.SOURCES(" || program || ")) REUS"

/****************************************************/
/* Invoke the compiler                              */
/****************************************************/
say "Compiling COBOL program:" program "..."
"CALL 'SYS1.SIGYCOMP(IGYCRCTL)'" COBOLCompileOptions
```

Environment:	-- None --	Selector:	-- Project Default --	Broadcast:	No
Timeout:	300	Result:	-- Exit Code --	On Fail:	Halt
Thread:	No	Inline:	-- None --		
Pass Notify:	-- None --	Pass Chain:	-- None --	Pass Wait:	No

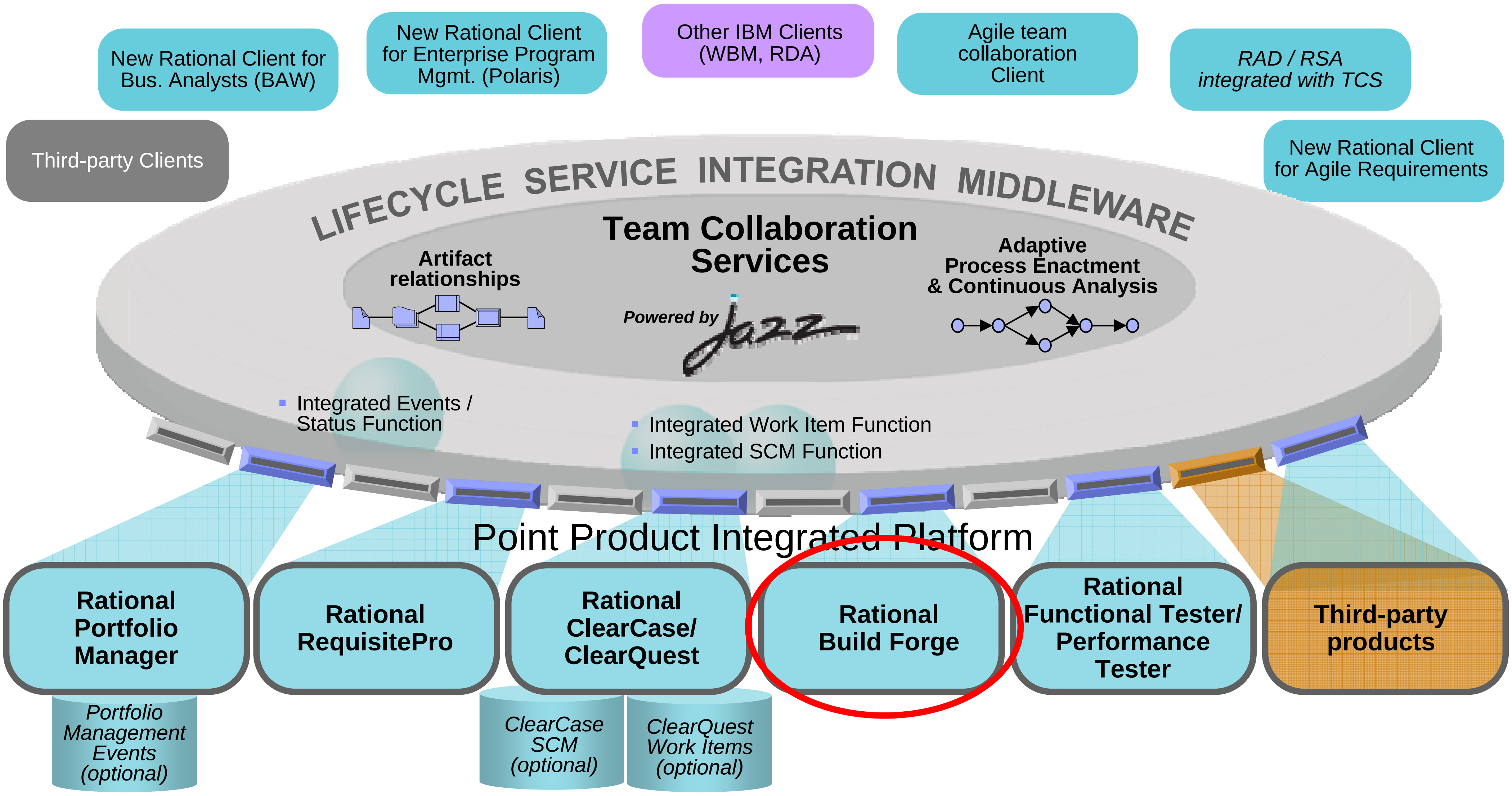
(c) Copyright International Business Machines Corporation 2003, 2007. All rights reserved

```
BF_J=255
BF_W=4
B=15
BF_AGENT_VERSION=7
BF_AGENT_PLATFORM=
Performing variable exp
spawning shell [/tmp/fBC
allocated pseudo-tty /de
start [/u/LEIGHW/bfage
ENTRY (A) LEIGHW.COB
DATA SET LEIGHW.OBJE
IGD17101I DATA SET LE
NOT DEFINED BECAUSE
RETURN CODE IS 8 REAS
Compiling COBOL progra
Compile succeeded: rc=0
end [/u/LEIGHW/bfagen
```

COM
ts (LEIGHW.*)
'.COBOL.COPYLIB
'.COBOL.LISTINGS
002B
'.COBOL.SOURCES
002B
SDB2
'YEX2
O
CS04
'.DBRM.DATA
'.EXEC.LOAD
'.ISPF.PROFILE
'.LINK.LISTINGS
'.OBJECTS.BX0002B.OBJ
'.OBJECTS.COPYEX2.OBJ
'.OBJECTS.HIHO.OBJ
'.SYSPRINT.LINKLIST
Queries

Evolving the Rational Software Delivery Platform

An open ecosystem based on IBM middleware


New Rational Client for Bus. Analysts (BAW)
New Rational Client for Enterprise Program Mgmt. (Polaris)
Other IBM Clients (WBM, RDA)
Agile team collaboration Client
RAD / RSA integrated with TCS
Third-party Clients
New Rational Client for Agile Requirements
LIFECYCLE SERVICE INTEGRATION MIDDLEWARE
Team Collaboration Services
Artifact relationships
Powered by jazz
Adaptive Process Enactment & Continuous Analysis
Integrated Events / Status Function
Integrated Work Item Function
Integrated SCM Function
Point Product Integrated Platform
Rational Portfolio Manager
Rational RequisitePro
Rational ClearCase/ ClearQuest
Rational Build Forge
Rational Functional Tester/ Performance Tester
Third-party products
Portfolio Management Events (optional)
ClearCase SCM (optional)
ClearQuest Work Items (optional)

Built for development efficiency: Allows developers to innovate rather than duplicating efforts, figuring out who to hand off to, or tracking and reporting status

An Integrated Experience

- **Build Forge becomes a normal resource in RTC**
  - RTC User can pick Build Forge from Build Engines list
  - Connection info stored in Jazz for smooth flow

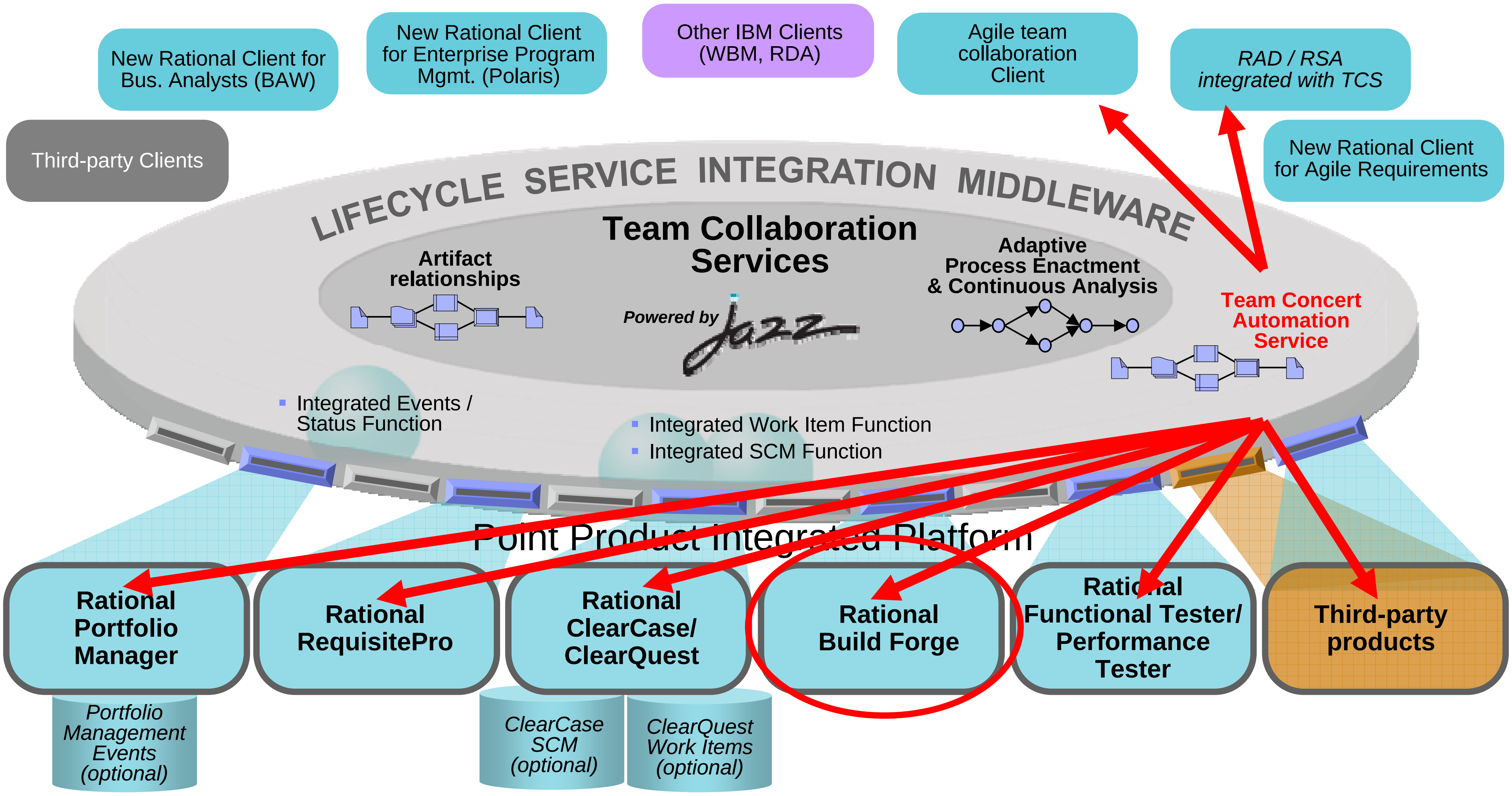
- **Build Forge objects matched to Jazz equivalents**
  - Build Forge Projects have matching Jazz Build Definition
  - Jazz Properties will match Build Forge Project Environments

- **Jazz Build Requests will fire Build Forge Jobs**
- **Build Forge Job results available directly in Jazz**
  - Detail data still in Build Forge with direct link-out from Jazz

Collaborative Builds

Evolving the Rational Software Delivery Platform

An open ecosystem based on IBM middleware


New Rational Client for Bus. Analysts (BAW)
New Rational Client for Enterprise Program Mgmt. (Polaris)
Other IBM Clients (WBM, RDA)
Agile team collaboration Client
RAD / RSA integrated with TCS
New Rational Client for Agile Requirements
Third-party Clients
LIFECYCLE SERVICE INTEGRATION MIDDLEWARE
Team Collaboration Services
Artifact relationships
Powered by jazz
Adaptive Process Enactment & Continuous Analysis
Team Concert Automation Service
Integrated Events / Status Function
Integrated Work Item Function
Integrated SCM Function
Point Product Integrated Platform
Rational Portfolio Manager
Rational RequisitePro
Rational ClearCase/ ClearQuest
Rational Build Forge
Rational Functional Tester/ Performance Tester
Third-party products
Portfolio Management Events (optional)
ClearCase SCM (optional)
ClearQuest Work Items (optional)

Built for development efficiency: Allows developers to innovate rather than duplicating efforts, figuring out who to hand off to, or tracking and reporting status

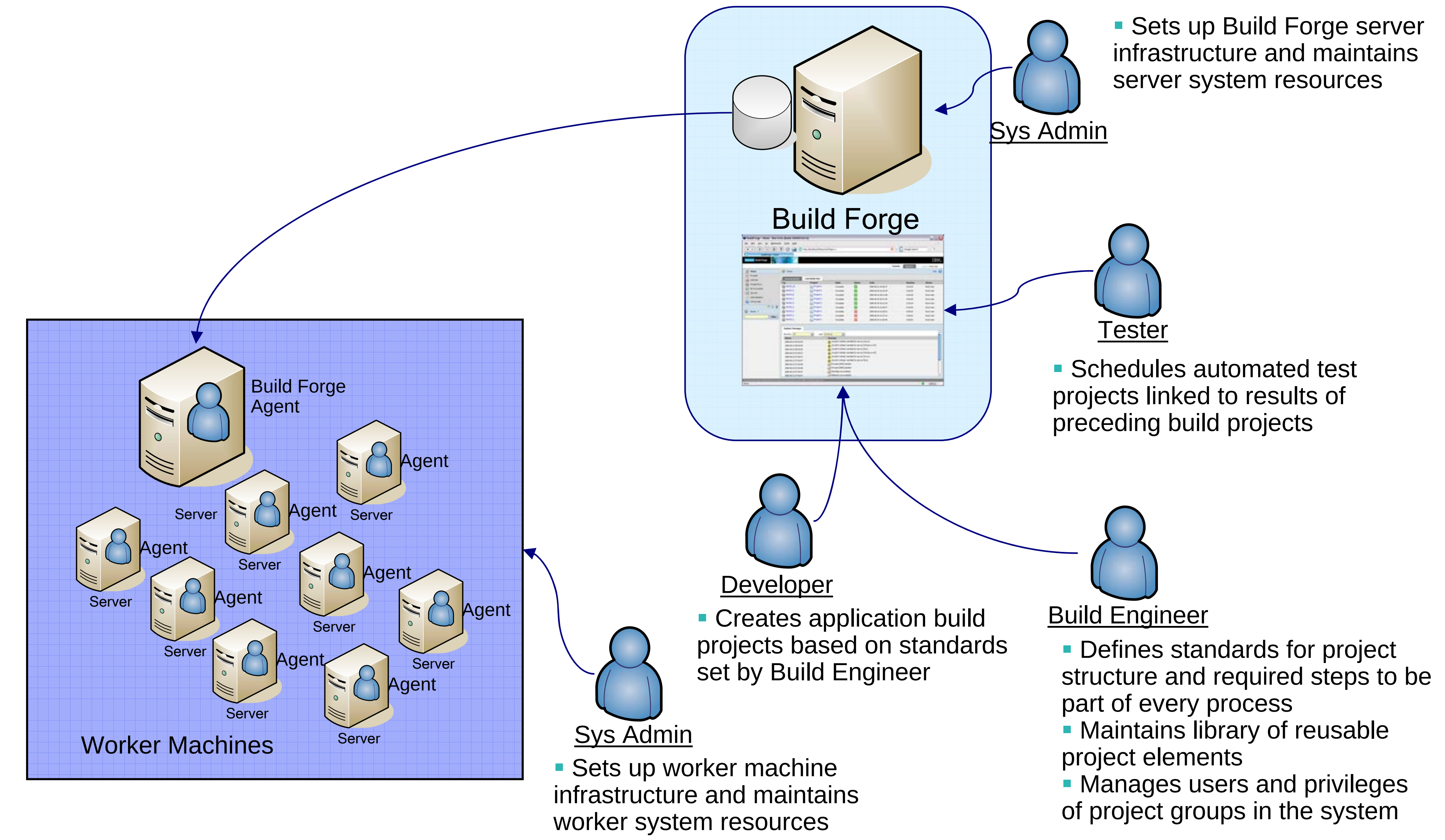
QUESTIONS

THANK YOU

Learn more at:

- IBM Rational software
- IBM Rational Software Delivery Platform
- Process and portfolio management
- Change and release management
- Quality management
- Architecture management
- Rational trial downloads
- Leading Innovation Web site
- developerWorks Rational
- IBM Rational TV
- IBM Rational Business Partners

Roles Involved in Build Forge Solution



- Sets up Build Forge server infrastructure and maintains server system resources
- Schedules automated test projects linked to results of preceding build projects
- Creates application build projects based on standards set by Build Engineer
- Sets up worker machine infrastructure and maintains worker system resources
- Defines standards for project structure and required steps to be part of every process
- Maintains library of reusable project elements
- Manages users and privileges of project groups in the system