IBM Software Group

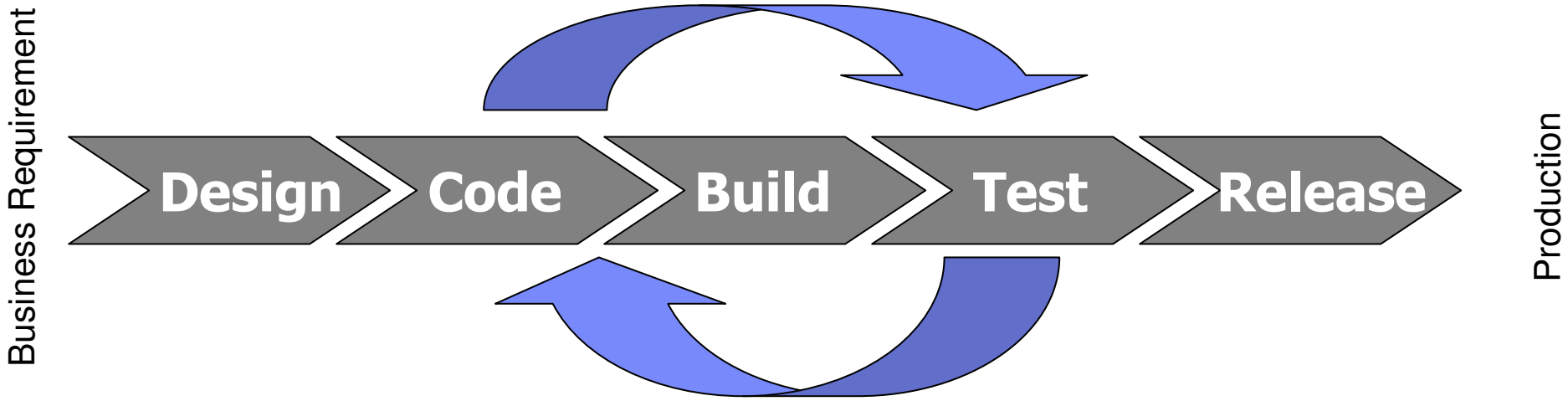
Rational Build Forge Introduction

Cheri Bergeron
Rational Build Forge Evangelist

Rational. software

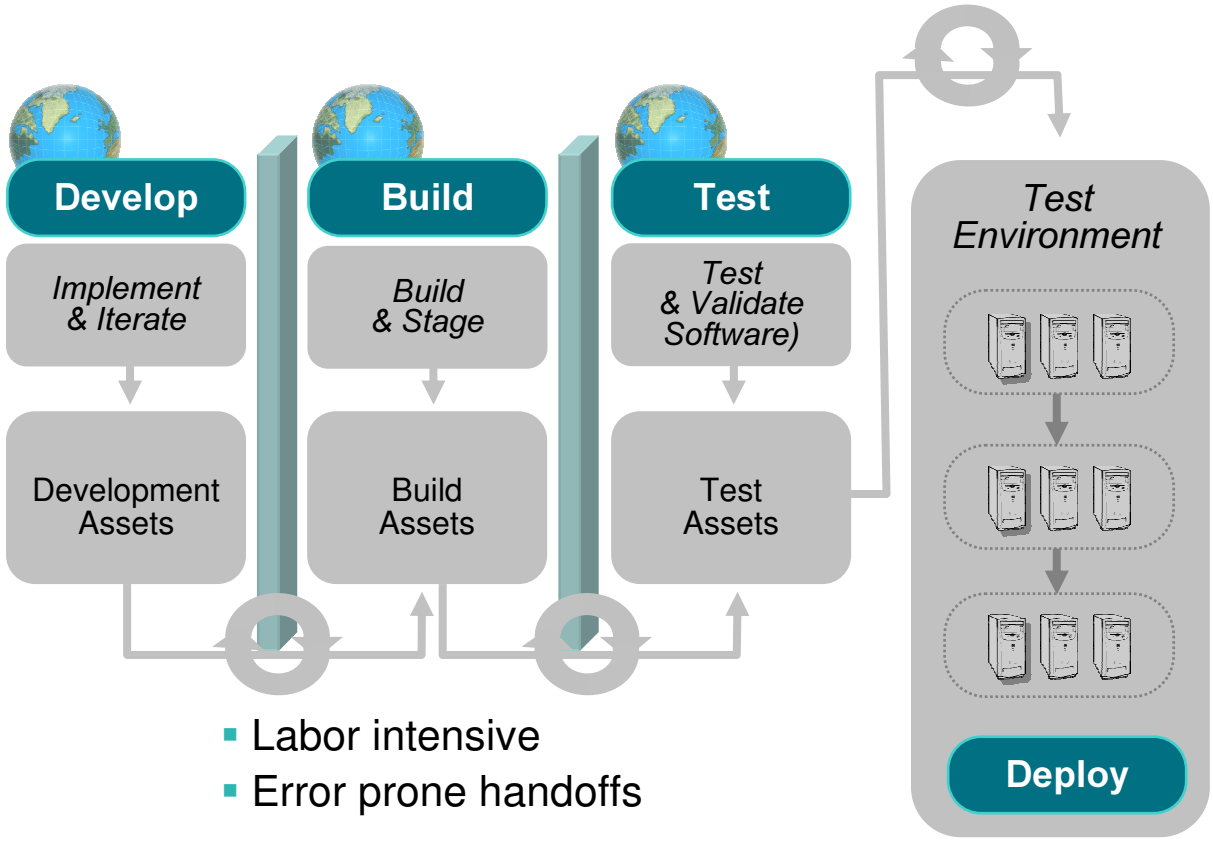
ON DEMAND BUSINESS

The Typical Development Cycle



Challenge: Silos between development, build, test and deployment

- No shared processes, artifacts or controls
- "Over the wall" communication
- Difficulty tracking and testing status of multiple builds
- Minimal reuse



- Labor intensive
- Error prone handoffs

The Silos

Analyst

Model the business & identify the business services

Architect

Design the services architecture

Analyze assets for reuse

Developer

Construct the services

Iterate and integrate the services

Build / Release Engineer

Assemble, package & deploy the composite application

Tester

Test the individual services & composite application

Deployment Manager

Provision, configure, tune and troubleshoot composite applications

Project Manager

- Follow a service-oriented process
- Manage requirements
- Manage software assets
- Manage quality

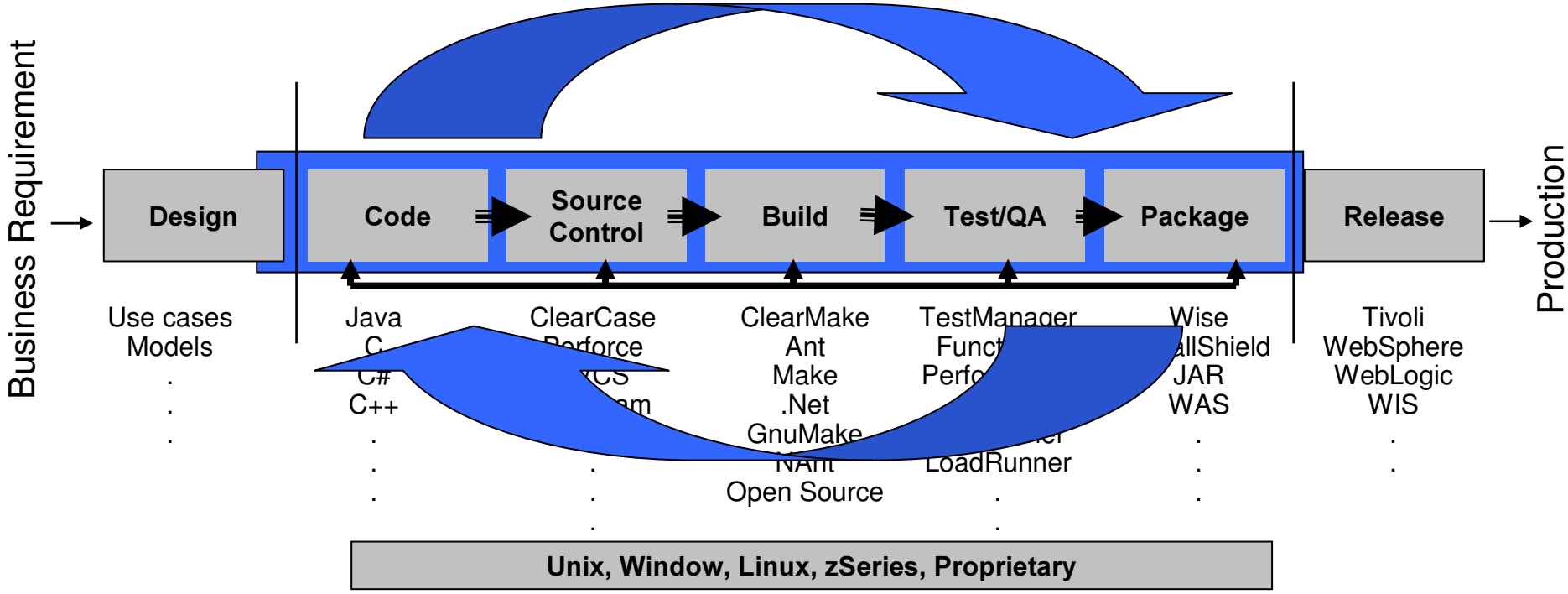
IT Executive

- Align business strategy with IT execution
- Govern Software Development – align, execute and control investments

Build Forge Business Drivers

- Governance-repeatable/traceable/self-documented/audit-ready
- Globally Distributed Development-Secure/Centralized management/ Automated handoffs
- Application Lifecycle Optimization-Automation increases iterations/eliminate manual handoffs increases efficiency/server management reduces cost/increased iterations improve quality

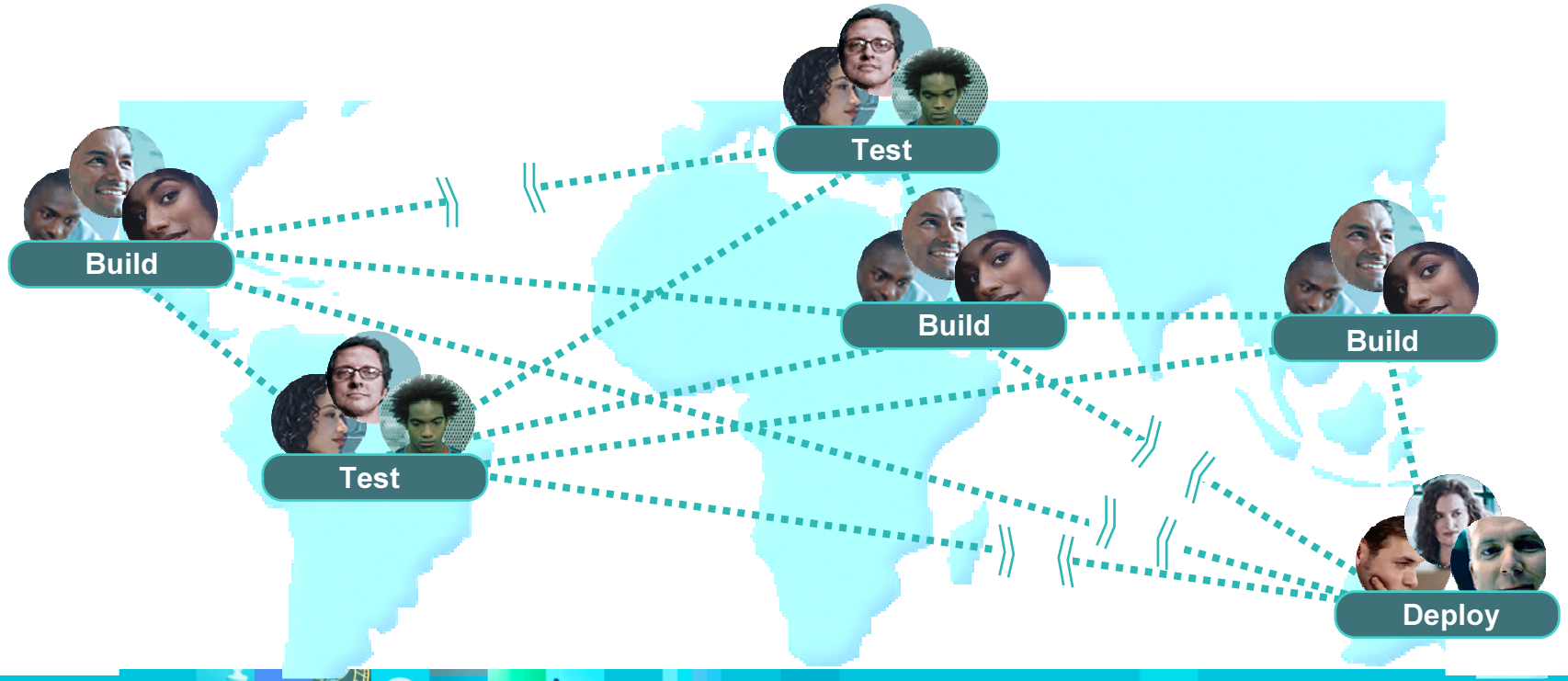
1. Retrieve source code
2. Set Config spec
3. Baseline project
4. Build main app
5. Run functional tests
6. Package application



Manual, error prone process.....Inconsistent
Limited automation.....Dependence on individual knowledge and scripts
Different projects and platforms.....Requires different knowledge and tools
Not traceable and poorly documented.....Audit/compliance risk
Time consuming.....Long waits "between" steps due to "over the wall" process

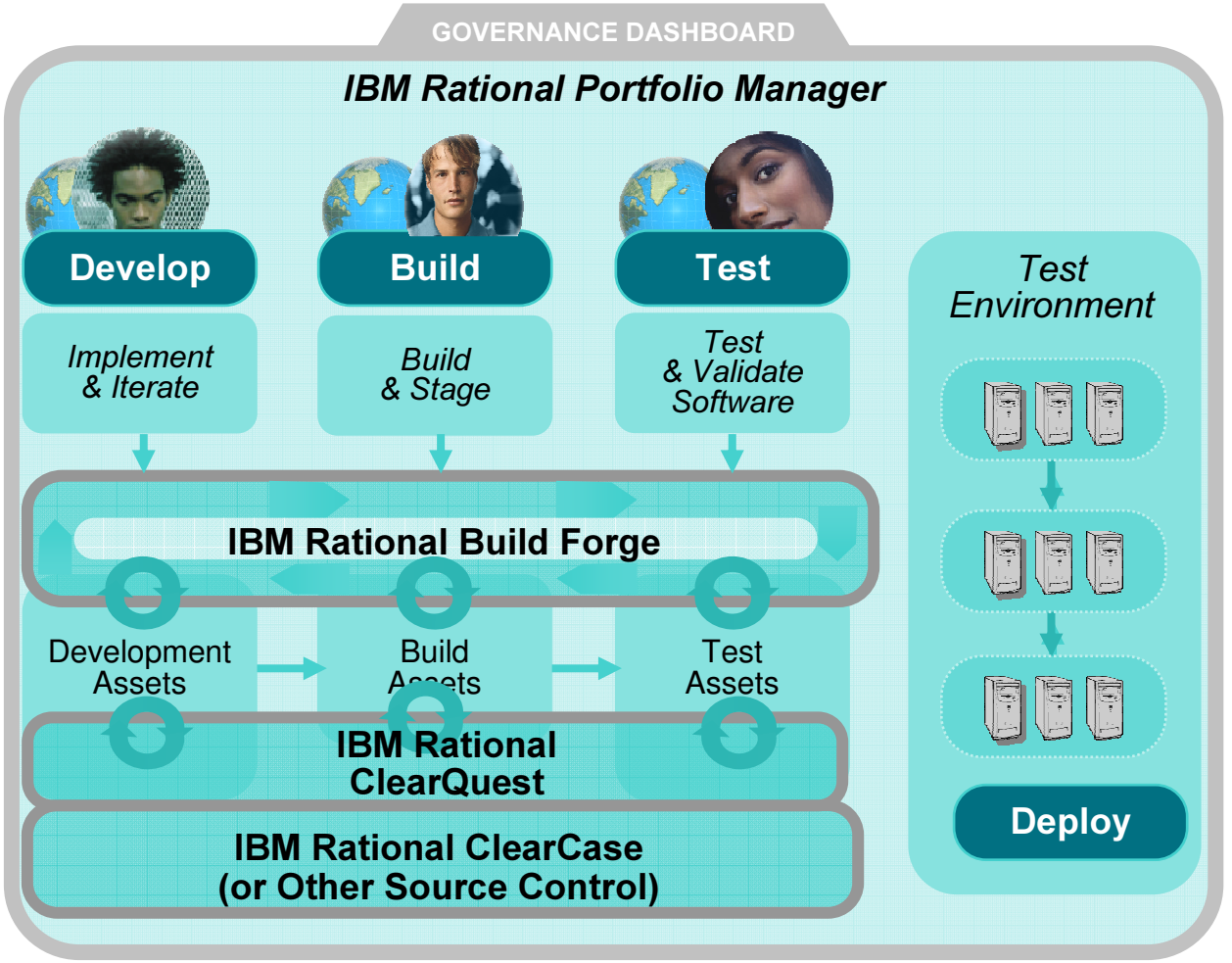
Challenge: Geographic silos

- Local and global project management pressures
- Participate in a project that's not using your language
- Work in preferred language
- Flexibility in product access with strong remote access capabilities



Solution: Automation Improves Software Delivery Performance

- Streamline development by automating handoffs
- IBM Rational Build Forge automates software delivery handoffs and captures detailed audit records
- IBM Rational ClearCase is the repository of record for all artifacts
- IBM Rational ClearQuest captures relevant state information from each phase including build records and deploy records from Build Forge
- Automate testing on build success and log test results in build record



New IBM Rational Software Development Platform

Analyst

WebSphere Business Modeler

WebSphere Business Monitor

Architect

Rational Software Architect

Rational Data Architect

WebSphere Studio Asset Analyzer

Developer

WebSphere Developer for Z

EGL

Rational Application Developer

WebSphere Integration Developer

Build / Release Engineer

Rational Build Forge

Rational ClearMake

Tester

Rational Functional & Manual Tester

Rational Performance Tester

Deployment Manager

Tivoli Configuration Manager

Tivoli Monitoring

Customer Extensions

ECLIPSE

3rd Party ISV Tools

Project Manager

Rational Team Unifying Platform

Rational Unified Process

Rational Requisite

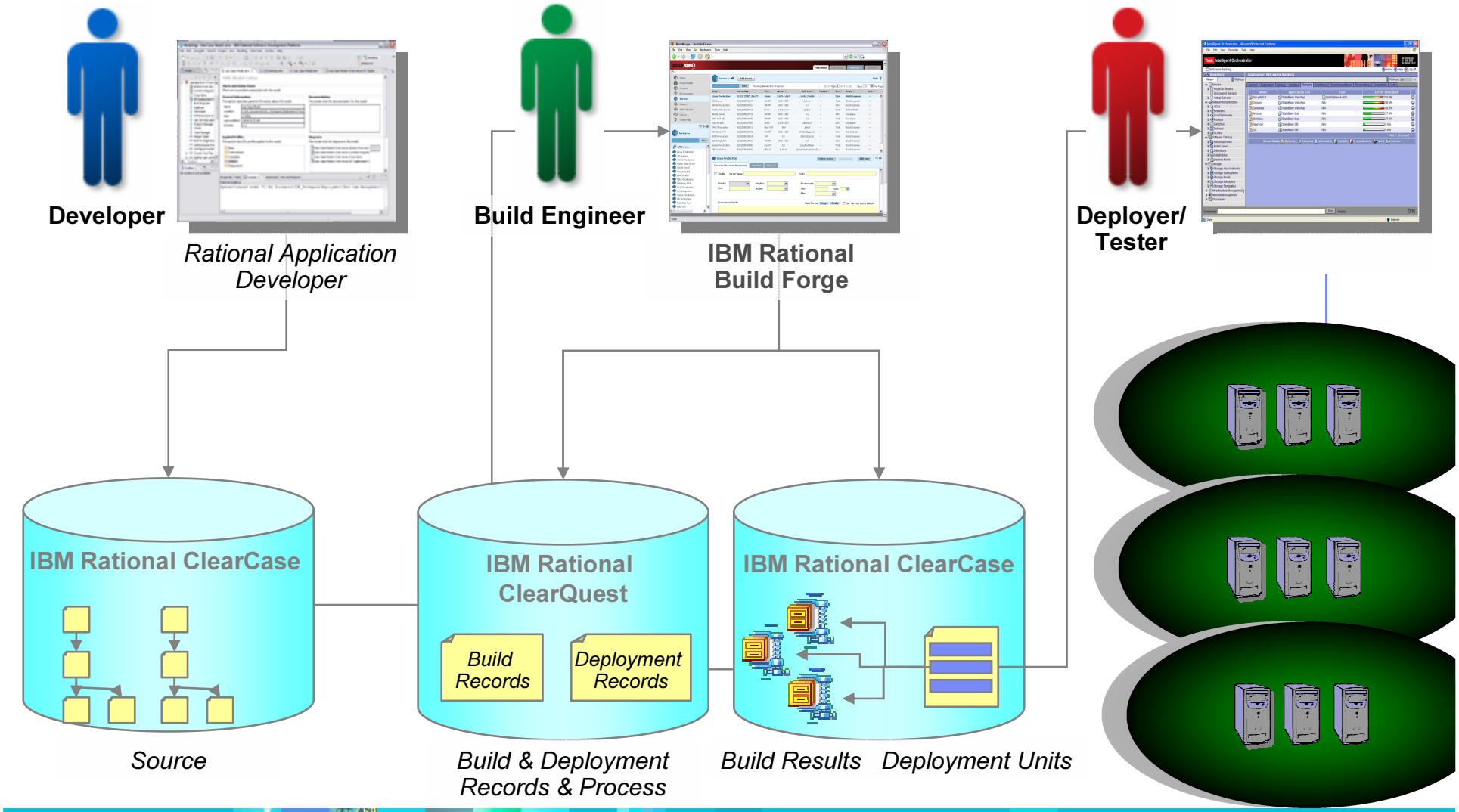
Rational ClearCase

Rational ClearQuest

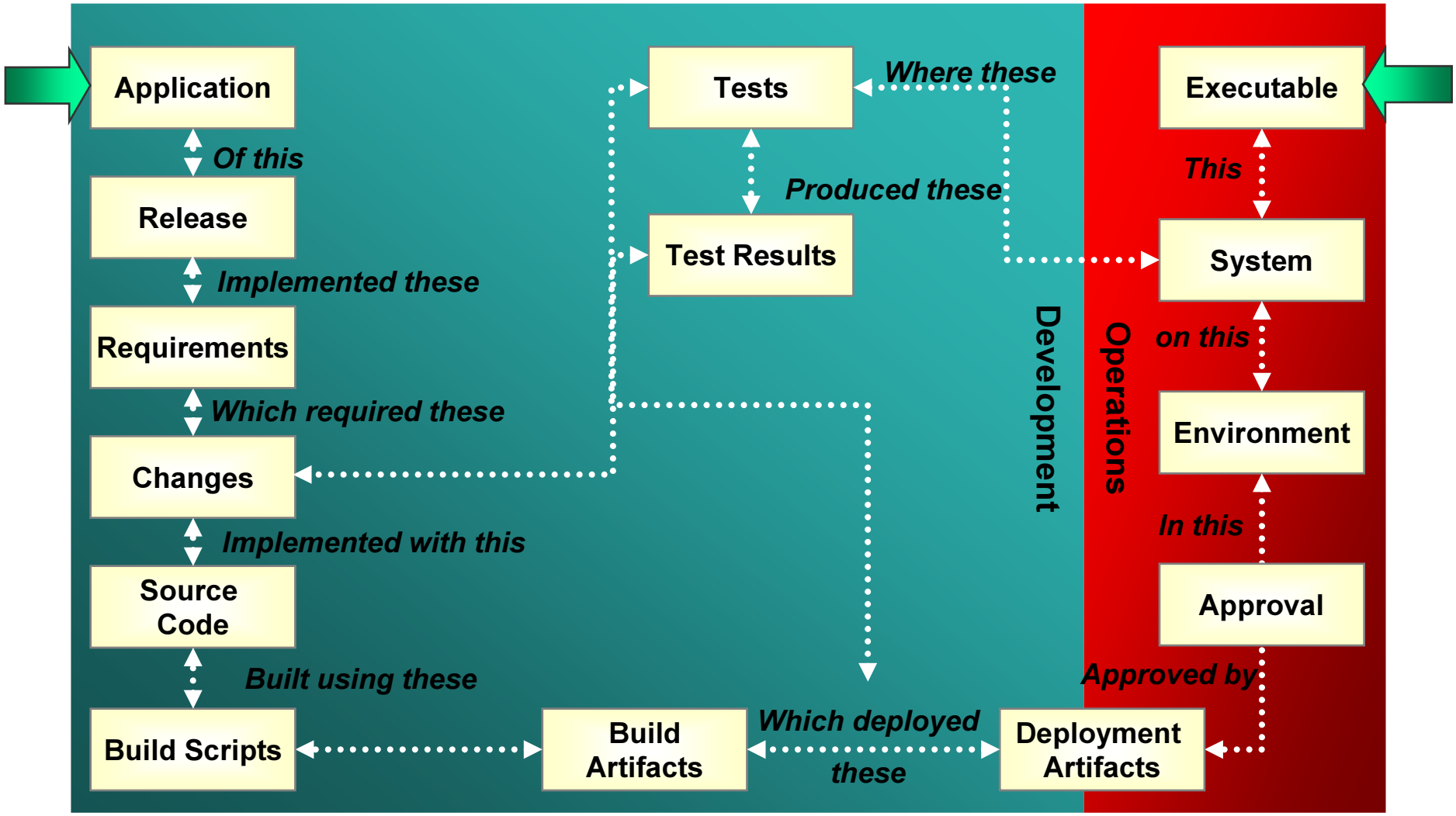
Executive

Rational Portfolio Manager

Managing and Automating the Process



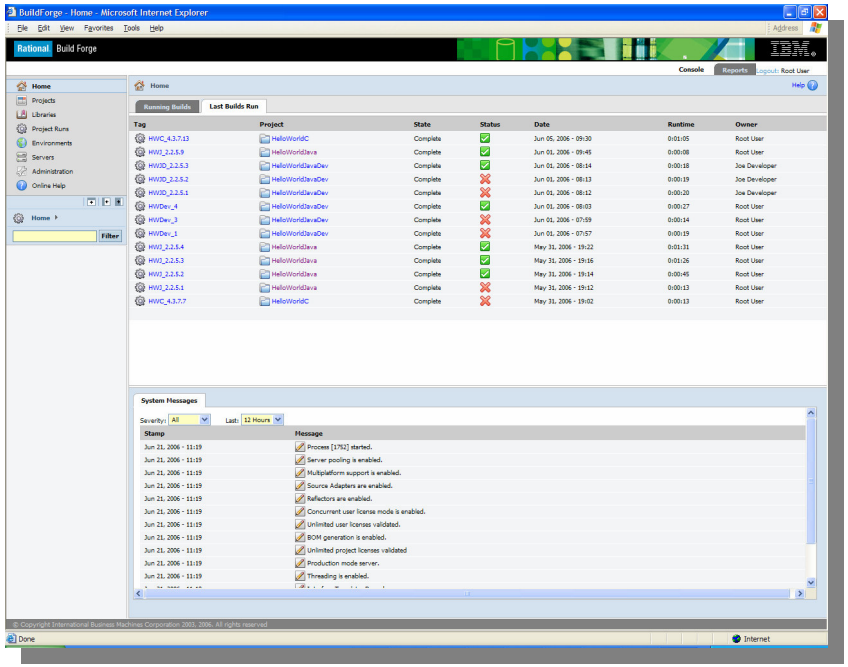
Tracing Your Development Artifacts



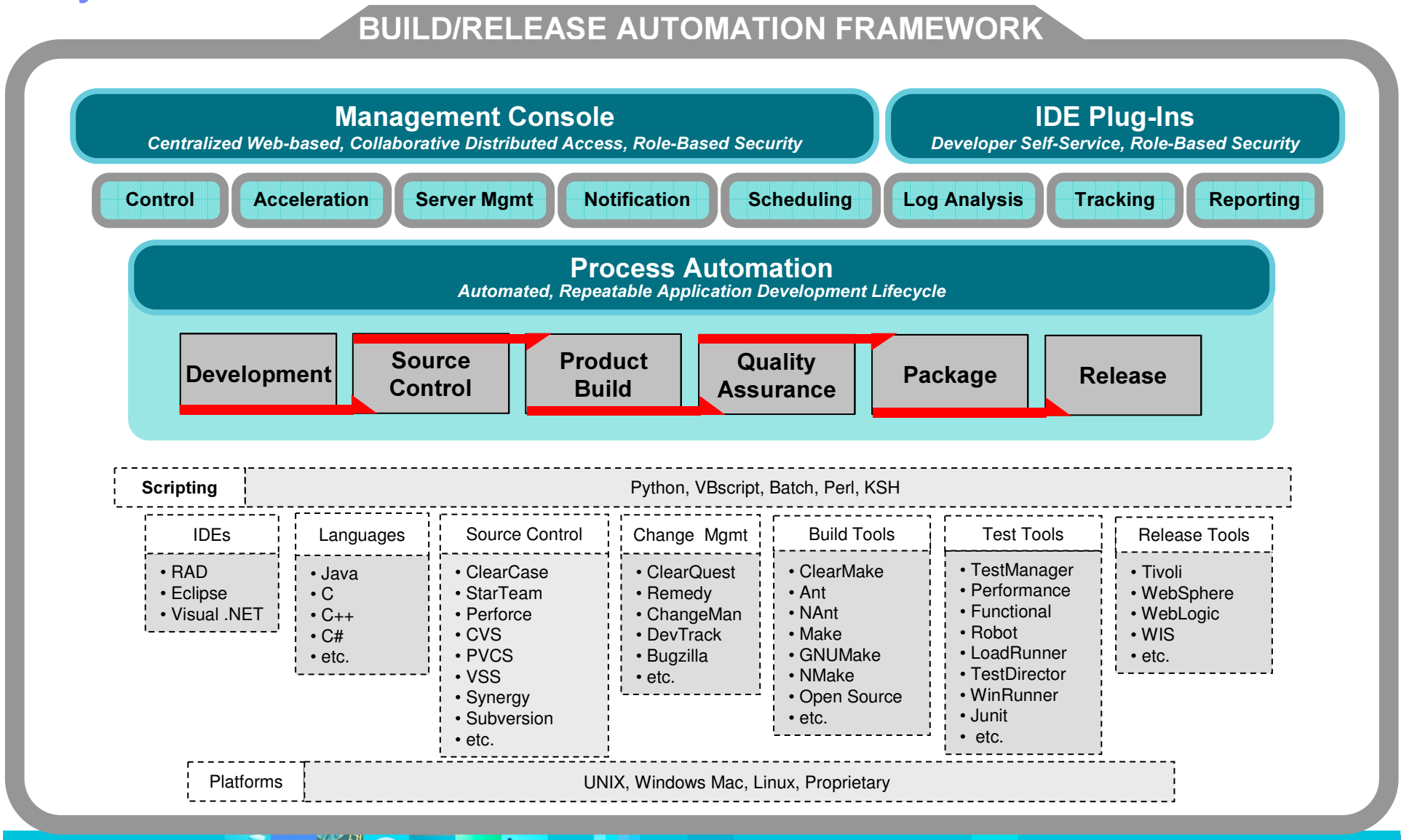
Why Rational Build Forge?

- **Process efficiency:** Reliable, repeatable, and scalable process
  - *Delivers automation, speed, productivity*
- **Geographically distributed teams:** Integrated and in sync
  - *Delivers automated handoffs, streamlined communication, secure access*
- **Sustainable compliance:** Auditing and IT controls “baked in”
  - *Delivers audit trails, complete bill of materials, reproducibility*
- **Implementing agile methods:** Developer self-service and continuous integration
  - *Delivers predictable quality, fewer errors, faster troubleshooting*

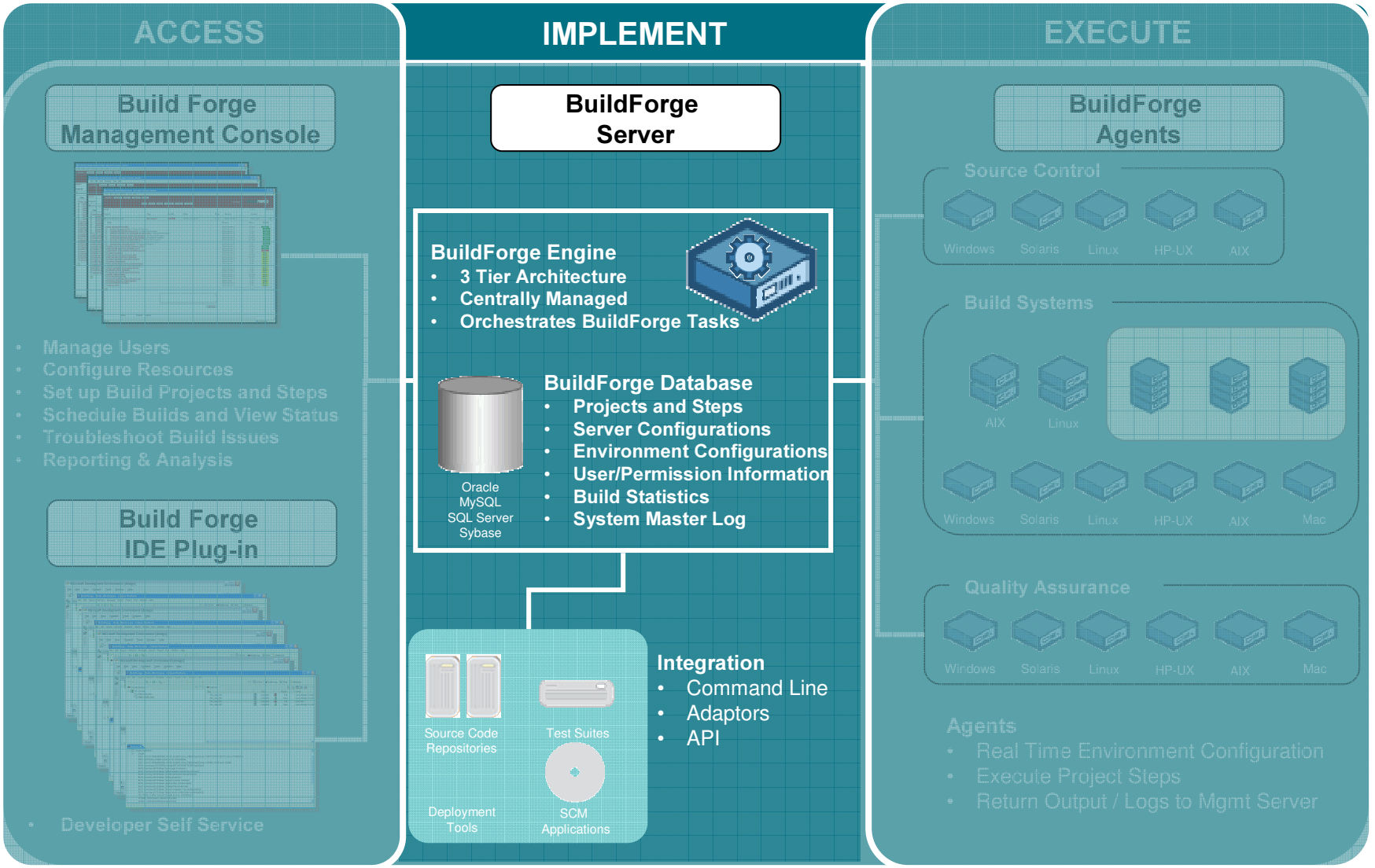
Product Demonstration
Build Forge Management Console



System Overview



How Build Forge Works



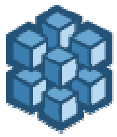
Building Blocks…

Steps

```
1a. cleartool mkview -tag $BF_TAG \\host\ccstg\views\$BF_TAG.vws
1b. cleartool setcs -tag $BF_TAG config.spec
```

```
2a. gcc main.c -o main.o
2b. gcc main.c -o ui.o
2c. gcc main.o ui.o -o HelloWorld.exe
```

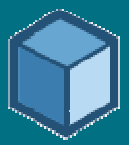
```
3a. testscript.sh -run -r $RELEASE -module HelloWorld.exe
```



Project

Source

Build



Test

Package

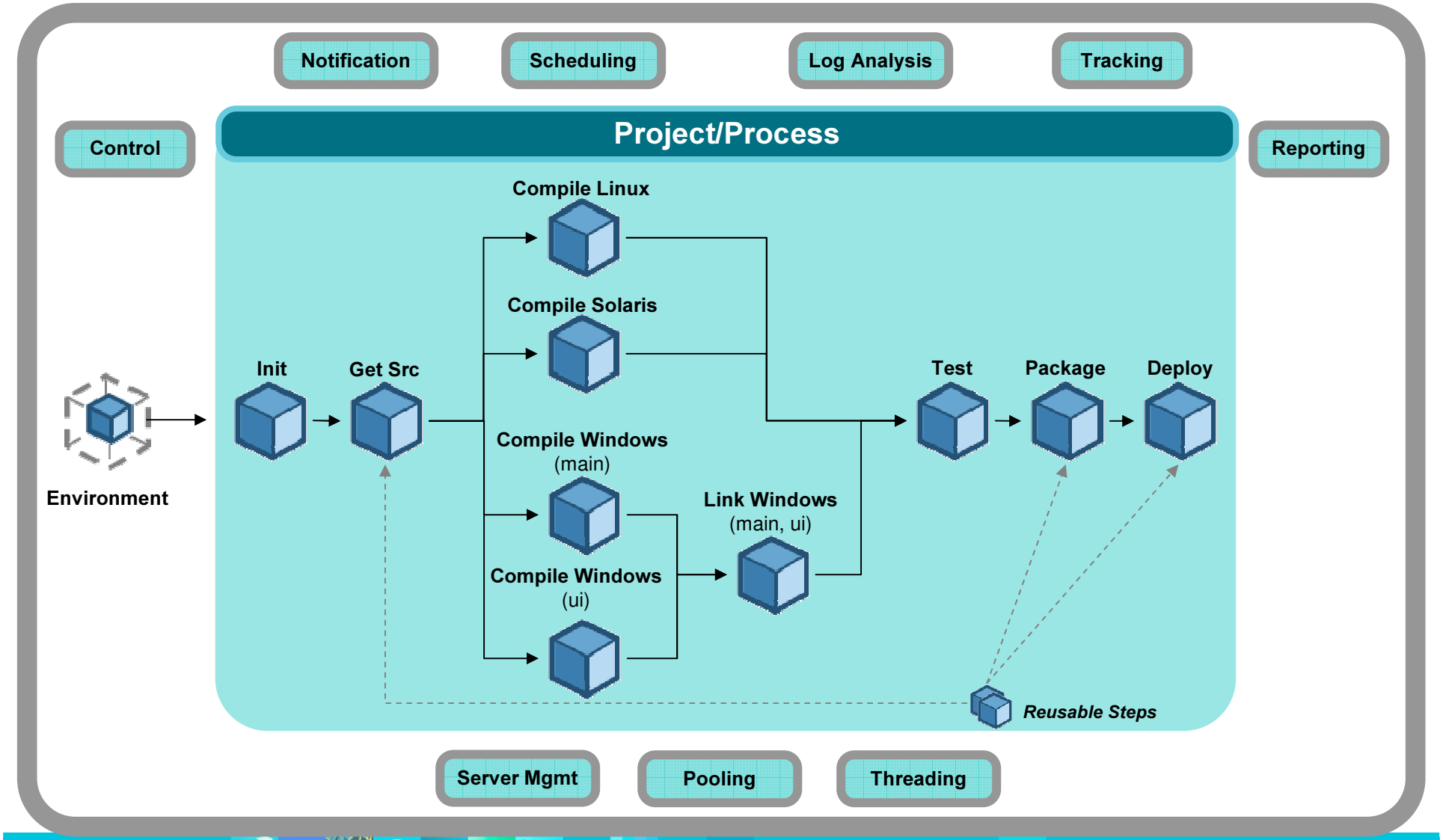
Deploy

…

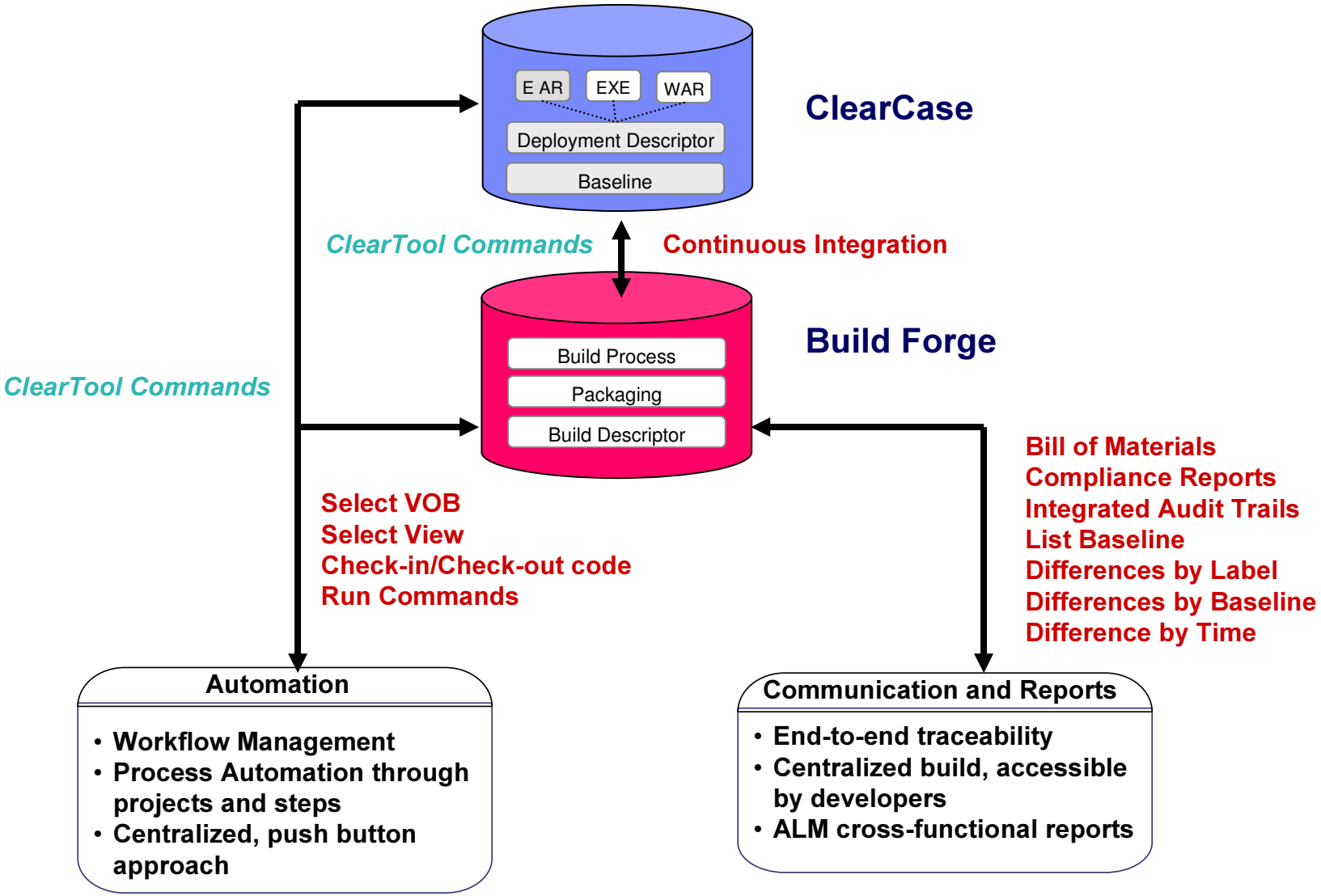
Environment

```
1. RELEASE=Release_1.1
2. JAVA_HOME=C:\Program Files\Java\jdk1.5.0_06
3. PATH=C:\Program Files\Java\jdk1.5.0_06\bin
4. …
```

Putting it all together…



ClearCase Integration Today



Who Uses Rational Build Forge

What's The Value?

"We were able to improve from 18 builds per week to over 360 builds per week! Across 50 other projects, that will save us $25 Million annually!" -- Adobe

Customer results: higher productivity, improved quality, faster delivery, reduced cost

- **Reduced cost of software delivery** through standardized processes, team efficiency, and effective asset leverage, and hardware usage
- **Increased quality of products** delivered through reliable, repeatable processes and rapid error detection.
- **Decreased overall time to market** through more frequent, iterative development cycles
- **Integrated management decision support** and compliance by providing critical information about your build and release life cycle

Quantifying the Value

Annual savings ranging from $250,000-$25M

Customer	CM Efficiency	Developer Efficiency	Quality / Error Reduction	Return on Investment
EA.com ELECTRONIC ARTS	Over 90%	10-15%	51-70%	< 3 months
symantec.	Over 90%	10-15%	51-70%	< 3 months
EMC² where information lives	80-90%	20-25%	80-90%	< 6 months
Fidelity Investments	21-25%	5-10%	26-50%	< 6 months
AVAYA	51-70%	26-50%	51-70%	< 3 months

Rational Case Study

"BuildForge helped us improve our turnaround times, quality and overall process by giving us a continuous integration system that allows us to notify developers of project status"

Rational. software

Environment

- 47 Active Projects / Products – 3 Locations
- 8 Platforms, 124 Build Machines
- Rational Products
- Windows, All Unix Flavors

Problem

- Release Team is bottleneck. No developer capabilities
- Serial and manual work effort
- 24 Hour "Suite" Build, 14 Hour Product Build

Solution

- implemented Developer self-service in 3 mos.
- "Suite" and point product builds reduced to 3 hours
- Parallel processes implemented.
- Automated packaging

Siemens Medical Case Study

SIEMENS

"We were interested to adopt Agile Development, but were limited by an inflexible, non-standard build process. Each team did their own thing, and there were multiple points of failure on each project."

Environment

- 1000+ users
- Build machines around the world (US, EMEA, India)
- Continuous unit testing (Cactus and Junit)
- ClearCase, ClearQuest, Test Director

Problem

- No standards
- No global access
- Multiple points of failure
- Low developer productivity
- No continuous integration

Solution

- # of build cycles increased 3X
- Build times reduced by 65%
- Secure developer self-service established
- $6M savings over 3 years

Adobe Case Study

"Our products were late, our communication and processes were poor, and we were without any centralized repository or auditing capabilities."

Environment

- 1500 Developers
- 150 Products
- 150 CM's
- Multi-platform
- Geographically Dispersed
- Perforce,Java, C++, Python, Perl

Problem

- Different Tools across the company – no control
- Teams unable to work together
- Minimally 1 CM per Product – limited # of builds
- No visibility across lifecycle

Solution

- 20X Build Improvement - Removed need for additional CM's
- Initial deployment in 2 months – 50 products
- Using BuildForge for lifecycle visibility.

Avaya Case Study

AVAYA

"We wanted to create a development platform that all of our global teams and projects could use – giving developers enough access to be effective while maintaining strict quality standards."

Environment

- Developers in Multiple Locations
- 12 CM in 5 offices
- Unix, Windows
- Variety of languages
- ClearCase, ClearQuest

Problem

- No centralized release mgmt
- Underutilized server farm
- No effective way for development teams to share components
- CM bottleneck

Solution

- Build times reduced from 12 hours to push button operation
- 30-120 minutes developer time saved per day
- $3+ million savings over 3 years

Bank of Nova Scotia Case Study

Scotiabank

"The environment necessary for a successful product build is extremely complex, and is different for every bus. Build Forge has helped us to become data-driven, instead of reactionary."

Environment

- 1500 Developers – 35 CM's
- Bus-driven development – 70 products
- ANT; Batch; Korn Shell; Perl; Ruby;
- CC, CQ, .NET, etc
- Windows, All Unix Flavors

Problem

- Manual process for Ant script updating.
- Defect fixes not always making proper "bus"
- Overlap in releases causing too many errors

Solution

- Dramatically reduced build-related errors
- Data-driven process Libraries help show what is reused and what isn't between builds.

Electronic Arts Case Study

EA.com ELECTRONIC ARTS

"The environment necessary for a successful build is very complex, and is different for every product. This information must be carefully maintained and consistently used."

Environment

- 500 Developers – 30 CM's
- 20 Products
- C++, .NET, Perl, Python
- Perforce, DevTrack
- Windows, Xbox, Playstation

Problem

- No centralized release mgmt.
- Underutilized server farm
- 60,000+ graphic files built daily that take 30 minutes to 60 hours

Solution

- Build times reduced 20X. from 60 to 3 hours.
- Machine usage improving – no additional HW buys.
- Management has new intelligence with dashboards.

In Summary

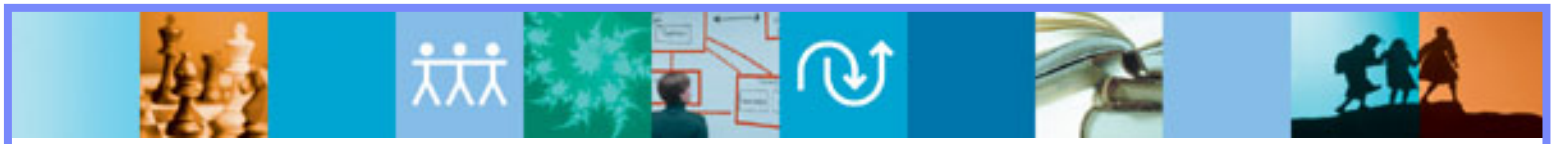
- Automate software delivery processes
  - Streamline overall development
  - Reduce manual handoff time
  - Remove bottlenecks
- Improve resource utilization – hardware and people
- Rapid time to value
  - Implement quickly
  - Reuse existing scripts and processes
  - Refine over time

What to Do Next

- Customized Demo for Your Team
- Build and Release Process Assessment
  - Examine your build and release process
  - Determine what areas could be impacted by Rational Build Forge
  - Recommend specific course of action for Proof of Concept
- Proof of Concept
  - Implement Build Forge on your system
  - Experience results first hand
  - Determine potential return
- Buy Build Forge to Extend Your Software Development Platform

Learn More About Rational Build Forge

- More Information on IBM Rational Build Forge
  http://www-306.ibm.com/software/awdtools/buildforge/index.html
- Additional white papers and recorded webcasts
  http://www.buildforge.com/resources/



Questions


Thank
You