An IBM Proof of Technology

Collaborative software development using IBM Rational Team Concert

Presentations
An IBM Proof of Technology

Welcome to the Technical Exploration Center

- Introductions
- Access restrictions
- Restrooms
- Emergency Exits
- Smoking Policy
- Breakfast/Lunch/Snacks – location and times
- Special meal requirements?
Introductions

- Please introduce yourself
- Name and organization
- Current integration technologies/tools in use

What do you want out of this Exploration session?

Agenda

- Introduction to Rational® Team Concert™
- Lab Overview
- Module 1 Setting up the Team
- Module 2 Planning Your Work
- Module 3 Keeping Track of All Our Work
- Module 4 Performing and Sharing Your Work
- Module 5 Remembering Well Known SCM Configurations
- Module 6 User’s View of Build
- Module 7 Exploring Changes and Traceability
- Module 8 Endgame and a Tightened Process
- Module 9 Taking Control of Your Project
- Session Summary
Objectives

• Explore how Rational Team Concert can
  ▸ Enable development teams to collaborate in real time in the context of the work they are doing, especially in globally diverse environments
  ▸ Enable projects to be managed more effectively by providing visibility into accurate project health information drawn directly from actual work
  ▸ Automate traceability and auditability by managing artifacts and their inter-relationships across the lifecycle empowering teams to deliver more value
  ▸ Provide customizable process design and enactment through rule-based process guidance, automation and definable checkpoints

• Provide a hands on experience using Rational Team Concert to automate the software delivery process

Introduction to Rational Team Concert

An IBM Proof of Technology
What if your development tools knew…

… about your teams

… about your artifacts

… who is responsible for what

… about your process
   - Code delivery rules, code quality, traceability, test runs, intellectual property

… how to bootstrap a project

… how to help new team members get started

… your favorite work item types and their state transitions

… when the build runs and what to do if it breaks

Collaborative Application Lifecycle Management

The challenge in enabling it

- Traditionally, each tool came with its own
  - UI - Web and desktop presentations of views and tasks
  - Logic – Workflow, process, search, query, scale, security and collaboration
  - Storage – Availability, traceability
  - Privacy, backup/archive

- Resulting in...
  - Brittle integrations
  - Silos everywhere
  - High cost to maintain and administer
  - Proprietary API's
Rational Team Concert: An open, extensible architecture
Supporting a broad range of desktop clients, IDE’s and languages

IBM Rational Team Concert
Software innovation through collaboration

- Collaborate in-context
  - Integrated release planning and reporting, source control, document collaboration, work item, build management, chat and process guidance
- Streamline agile development
  - Out-of-the-box agile process configurations
- Automate governance
  - Assess project status and trends in real-time with web-based dashboards, metrics and reporting
- Scale to the enterprise
  - Supports teams ranging from a few to thousands of developers and stakeholders
- Unify diverse teams
  - Supports J2EE, .NET, IBM i, System z, co-existence with popular toolsets

Open and extensible on Jazz
- Collaborate
- Automate
- Report

JAZZ SERVICES
- Collaboration
- Discovery
- Query
- Storage
- Administration: Users, projects, process
- Business Partner Extensions
- Your Extensions

IBM Rational Extensions

Eclipse Platform
- Business Partner Extensions
- Your Extensions

Jazz™ Client Extensions
- Eclipse Clients
- Web Clients
- Microsoft .NET Clients

Rational Desktop Clients
- Rational Software Architect
- Rational Systems Developer
- Rational Business Developer
- Rational Developer for z
- Rational Developer for i

IBM Rational Extensions

IBM Software Group

© 2009 IBM Corporation

Collaborative Software Development Using IBM Rational Team Concert
Pain points before Rational Team Concert

- joining a team
- get my environment configured to be productive
- what is happening in my team
- collecting progress status
- following the team’s process
- ad hoc collaboration/sharing of changes
- starting an ad hoc team

- is the fix in the build?
- what will be in the next build?
- tracking a broken build
- Avoid breaking a build/personal build
- why is this change in the build?
- reconstructing a context for a bug/build failure

- creating, tracking iteration plans
- interrupting development due to a high priority bug fix
- working on multiple releases concurrently
- tracking the code review of a fix
- referencing team artifacts in discussions
- how healthy is a component?
- collecting project data/metrics?

Team awareness
Build awareness
Project awareness

Boring and painful

Team Concert: An Overview

Agile Planning
- Integrated release/iteration planning
- Effort estimation & progress tracking taskboards
- Out of the box agile process templates

SCM
- Integrated stream management
- Component level baselines
- Server-based sandboxes
- Identifies component in streams and available baselines
- SVN, Git, CC bridge, connector

Work Items
- Defects, enhancements and conversations
- View and share query results
- Support for approvals and discussions
- Query editor interface
- ClearQuest® bridge, connector

Build
- Work item and change set traceability
- Build definitions for team and private builds
- Local or remote build servers
- Supports Ant and command line tools
- Integration with Build Forge®

Project Transparency
- Customizable web based dashboards
- Real time metrics and reports
- Project milestone tracking and status

Jazz Team Server
- Single structure for project related artifacts
- World-class team on-boarding / offboarding including team membership, sub-teams and project inheritance
- Role-based operational control for flexible definition of process and capabilities
- Team advisor for defining / refining “rules” and enabling continuous improvement
- Process enactment and enforcement
- In-context collaboration enables team members to communicate in context of their work
Leveraging Rational Team Concert independently

- Rational Team Concert Enterprise, Standard, Express, Express-C
  - A standalone development environment optimized for small and mid-sized teams
  - All the collaborative capabilities of the Jazz platform – plus integrated work items, SCM and build management
  - Dashboards and real-time reports
  - Team and Process-aware

Incremental Adoption by Subversion, ClearCase/ClearQuest and Git teams

- Manage planning and project status with work items and dashboards in Team Concert and develop with existing artifacts that reside in subversion, Git or ClearCase® (new bridges in RTC 2.0)
- Enables teams to reuse assets, process and investment in Subversion, ClearCase/ClearQuest or Git
- Third party connector to Jira
- Take advantage of new collaborative ALM in an evolutionary way with lower business risk
Envisioning a platform that can transform software delivery

Jazz is a project and platform for transforming how people work together to deliver greater value and performance from their software investments.

- Enable team transparency of “who, what, when, why”
- Build team cohesion and presence
- Automate hand-offs – so nothing falls through the cracks
- Robust, extensible and scaleable
- Globally distributed, fluid & dynamic
- Community-based & open at Jazz.net

Collaborate in Context

- Automate team workflow improving productivity
- Automate data collection eliminating administrative overhead
- Real time reporting and alerts reduces project risk

Right-size Governance

- Dynamic provisioning of projects and teams
- Real-time iteration planning and workload balancing
- Unify teams with tools choice

Day One Productivity

Dynamic integration of people, process and projects across the lifecycle

Open Commercial Development at jazz.net

Delivering greater openness and customer participation in the products they depend on for software delivery

- IBM is opening up the Rational Software Delivery Platform for greater ease of consumption, extensibility and integration to meet the unique usage needs of our customers
- IBM is providing transparent, collaborative customer participation in the development of new Rational technologies through an open commercial community

Open Commercial Community

Creating, maintaining, and enhancing innovative commercial software through an open transparent process that allows customers to participate directly in the development process

Open Source Community

Creating, maintaining, and enhancing software through open, collaborative communities driving evolution of standards and common components

Open commercial development

Open source contribution of selected Jazz technology
Introducing the first wave of Jazz offerings

- **Team Concert**
  - Innovation Through Collaboration
  - Unify by “thinking & working” in unison with real-time project health

- **Requirements Composer**
  - Business Expert Collaboration
  - Elicit, capture, elaborate, discuss and review requirements

- **Quality Manager**
  - Collaborative Business-driven Quality
  - Coordinate quality assurance plans, processes and resources

**JAZZ TEAM SERVER**

- **Open Lifecycle Service Integrations**
  - Best Practice Processes
  - Search and Query
  - Collaboration Team awareness
  - Events notification
  - Security
  - Dashboards

**Rational**

- **ClearQuest**
- **ClearCase**
- **Build Forge**
- **Asset Manager**
- **RequisitePro®**

**The road ahead: What to expect from Rational Jazz based offerings that will be delivered throughout 2009**

- **Project Management**
  - Manage global projects and resources

- **Method Composer**
  - Dynamic process, enactment and management

- **ClearQuest, Build Forge & RequisitePro**
  - Native Jazz Implementations

- **ClearCase**
  - Native for UCM users

**Enterprise Reporting**

- Gain insight based on real-time and historical trend information

**Rational Integrations**

- Rational Software Architect
- Rational Application Developer
- Rational AppScan® & Tester
- portfolio
- Rational enterprise modernization including system z and i support

**More dynamic integration leveraging Jazz “Open Services for ALM” services**
IBM’s Internal Deployment of Team Concert

- In Rational, today we have 2,016 developers and testers
  - 80 development projects, two application servers (hosting 20 RTC server instances), one DB2 server
- In IBM we have 40,000 developers - By end of 2009 1/3 of these will be using RTC
  - And this is NOT being mandated, but rather through viral adoption and real productivity gains

Customer feedback

"By helping us to make project deliveries more repeatable and predictable, we anticipate that Rational Team Concert will reduce project overrun costs by 20%.”
--Matt Pomroy - Executive, Software Engineering, Ascendant Technology

"Its automated project management dashboards are transparent to everyone – not just managers. This immediate and automated feedback helps keeps teams on track and motivated to achieve project goals.”
--Han Jie - Senior Consultant, Siemens

"Where we previously used separate systems, with Rational Team Concert we now have well integrated functionality. Our developers are more efficient because they are better able to focus on important issues. Our project managers greatly value the ability to customize these dashboards and instantly provide status on their milestones!"
--Mika Koivuluoma - Production Manager, TietoEnator

"Having a unified and extensible environment is very compelling for us. Rational Team Concert provides the team transparency and visibility needed to keep work progressing so everyone knows what's going on without finger-pointing.”
--Carson Holmes - Unified ALM Services Manager, Noblestar

"With IBM Rational Team Concert we've seen a 30% productivity gain on our global projects. The ability to easily suspend and resume work, along with advanced SCM features, helps team members juggle multiple tasks and priorities without missing a beat."
--Alain Bergeron - VP Consulting Services, CGI
A Growing Ecosystem of Rational Team Concert Partners

- blackduck: Manage software licensing risk.
- WebLayers: Manage business and architectural policies.
- QSM: Track project actuals vs. plan.
- sourceIQ: Access to source code quality metrics.
- Comlogic: Integrate with Perforce SCM.
- XPRAXIS Engineering: Integrate with MKS ALM.
- ibis Group: Integrate with HP Quality Center.
- CAST: Access project and team health information.
- Legal Counsel
- Compliance Officer
- Project Manager
- QA Manager
- Development Manager
- Executive
- Project Coordinator
- Software Architect
- Functional Tester
- Application Developer
- Business Developer
- Lotus MashUps
- etc..

IBM Rational

IBM Software Group

Questions
### Enterprise Scalability for Team Concert

<table>
<thead>
<tr>
<th></th>
<th>Express-C</th>
<th>Express</th>
<th>Standard</th>
<th>Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Small Teams</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum developers/contributors</td>
<td>10 / unrestricted</td>
<td>50 / unrestricted</td>
<td>250 / unrestricted</td>
<td>Unrestricted / unrestricted</td>
</tr>
<tr>
<td>Database</td>
<td>Derby Only</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
</tr>
<tr>
<td>Application Server</td>
<td>Tomcat Only</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mid Sized Teams</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Corporate Teams</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum developers/contributors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
</tr>
<tr>
<td>Application Server</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Enterprise Teams</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum developers/contributors</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
<td>Derby, DB2, Oracle, SQLServer</td>
</tr>
<tr>
<td>Application Server</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
<td>Tomcat, WebSphere</td>
</tr>
</tbody>
</table>

- **New in 2.0 for Express-C/Express**
- **Customizable Dashboard Mashups**
  - 1 per project
  - 1 per project
  - unlimited project, team, and personal dashboards
  - unlimited project, team, and personal dashboards

- **Advanced Reports and Customization**
  - ✔
  - ✔

- **Customizable work item workflow**
  - ✔
  - ✔

- **Plan risk assessment**
  - ✔
  - ✔

- **CC/CQ Connectors and Bridges**
  - ✔
  - ✔

- **Floating Licenses Available**
  - ✔
  - ✔

- **LDAP import / synchronize**
  - ✔
  - ✔

- **HTTP proxy support**
  - ✔
  - ✔

- **High Availability**
  - ✔

*See Jazz.net for prerequisites.*
Lab Overview

An IBM Proof of Technology

Scenario for PoT Labs

- You are joining a new project called Squawk that has recently been started in your company.
- You will be using Rational Team Concert as the project’s collaborative development environment.
- You have joined the project at the start of Milestone 3. You and all your team mates will be contributing new content to the application.
Scenario for PoT Labs

● Squawk is a (simple) program that will print out different sounds depending on who “squawks”. The Dog squawker goes “bark”, the Cat squawker goes “meow”, etc. Your main task is to create a new squawker, along with tests and documentation.

● At the same time as creating new squawkers, you will get to participate in some planning activities, interact with your fellow team members, deliver your work to the project, trigger automated builds and various tasks typical for project teams everywhere.

● The project team structure mimics the four major components:
  ▶ Core Library
  ▶ Documentation
  ▶ User Interface
  ▶ Release Engineering (build)

● You are assigned to the Core Library and Documentation teams with a team leader (one of the instructors). Welcome to the team!

Eclipse Overview

Menu
Button bar
A View
Add new Views using the Window->Show View menu

Change Perspective
Current Perspective
This View has different areas accessed via the View tabs
View tabs
Jazz artifacts are stored in a repository.

The repository contains project areas, which are the system's representation of a software projects.

Each project area has an associated process, which governs how the project is run.

Project Areas are decomposed into a set of team areas, which describe the teams that work on the project.

Teams use a stream to store the master copy of project's files.

Team Members use a personal repository workspace to work on project files.
Sequence of events – Lab 1 & 2

Team Lead (instructor)

Accept Team Invite
Configure Instant Messaging*
Chat with the Team*

Create New Plans
Distribute Iteration Plan

Lab 1

Lab 2

Team Member (student)

Sequence of events – Lab 3 & 4

Team Lead (instructor)

Create Query
Explore Web UI
Project Status with Team Central View
Track your Work with My Work View

Create Workspaces
Create & Deliver Work

Lab 3

Lab 4

Team Member (student)
Sequence of events – Lab 4 & 5

Team Lead (instructor)

Team Member (student)

Create Component Baselines

Create & Promote Workspace Snapshots

Accept New Baselines

Explore Snapshots

Accept all Changes

Lab 4

Lab 5

Sequence of events – Lab 6 & 7

Team Lead (instructor)

Team Member (student)

Request Integration Build

Explore Build

Request Personal Build

Explore Changes for Build

Explore Changes for a Work Item

Explore Changes for a File

Lab 6

Lab 7
Sequence of events – Lab 8 & 9

- Change to the Endgame Iteration
- Experience Process Enactment
- Create a Personal Dashboard
- Explore Reports Using Web UI

Lab 8

Lab 9

Sequence of events – Lab 10

- Prepare to Sync with ClearCase
- Sync with ClearCase
- Update in ClearCase and Sync

Lab 10 (optional)
Lab Conventions

- The hostname used to connect to the Jazz Team Server is *jazz-server*.
- Each student is assigned a unique user id of the form *student<N>* based on their student number, e.g. *student1*.
  - Examples in the lab workbooks use student1, you will need to adjust per your assigned id.
- Every student creates their own unique Squawker.
  - Examples in the lab workbooks use *Lion*.
- Students can choose any squawker they want but should include your student id in the name.
  - `<squawker>_<student id>`, e.g. *Lion_student1*.
- Work items created should include the full squawker name in the summary title.
  - `<squawker name> Implementation` and `<squawker name> Documentation` e.g. *Lion_student1 Implementation*.
- Optionally, adjust the language settings in the VM for international keyboards. Go to *Control Panel -> Regional and Language Options*. Select the *Languages* tab and then click *Details* in the *Text Services and input languages* section. Add your local keyboard and make it the default input language.

Jazz.net Registration

- Not a Member yet?
  - If you have web access to your email server
    - You will receive a confirmation and password resetting instructions
  - Go to *www.jazz.net* and register now.

- Creating a Jazz.net account allows you:
  - to take part in the Jazz community.
  - download product trials, betas, and other previews of Jazz technology.
  - have access to articles, tech notes, tutorials.
  - interact directly with the development teams and other members of the Jazz community to ask questions, report bugs, provide feedback and help guide the evolution of Jazz technology.
Setting up the Team

An IBM Proof of Technology

Objectives

- In this lab you will learn how ramp up projects quickly and dramatically improve onboarding and offboarding of team members
- You will perform some initial setup of Rational Team Concert to enable your machine to communicate with the server
- You will enable instant messaging in Rational Team Concert
Joining a project

- For most environments, joining a project can be complicated
- Team Concert makes this as easy as possible
- Adding a new team member to a project generates a Team Invitation email
- Contents of the email can be used to set up the new team member’s access to the project resources in Team Concert

Communication

- Users of Team Concert can use a variety of tools to communicate with team members
  - E-mail
  - Instant Messaging/Chat *
  - RSS feeds
  - Web UI
  - Team Concert client
- Team members can use all the typical communication mechanisms to keep working together as a team, regardless of where they are physically located. This collaboration allows for a single view of project data
  - Integrated Instant Messaging/Chat for immediate feedback
  - RSS feeds to notify you of significant events on the project in real time
  - The Web UI used for anyone on the team, or who has an interest in the project

* Not currently available in Visual Studio
Lab #1 Scenario

- You arrive at work on Day 1 and receive an email inviting you to the Squawk project.
- You start Team Concert and get connected to the project right away.
- You use instant messaging to chat to your colleagues on the project.

Lab #1 Overview

- Use the team invitation received via email to get connected to the Squawk project.
- Configure your Team Concert workspace for Instant Messaging.
- Explore the organization of your new team and start up a friendly chat to introduce yourself.
Lab #1 Concepts Learned

- Team Invitations make it easier to get team member’s connected to your project resources managed in Rational Team Concert
- Rational Team Concert has built-in instant messaging support that makes it easy to connect and collaborate with your teammates

Questions
Planning Your Work

An IBM Proof of Technology

Objectives

- Understand Rational Team Concert’s agile project planning capabilities
- Learn about Work Items and how they are central to Rational Team Concert
- In the lab you will learn how to create and work with Work Items and Iteration Plans
Project Plans

● A plan revolves around the following elements
  ▪ Teams
  ▪ Time
  ▪ Work

● Planning levels
  ▪ Release
  ▪ Iteration/Sprint
  ▪ Your day-to-day work

Team

● The project team is divided into one or more teams
  ▪ A team will focus on one aspect of the project

● Each person can divide their time across many projects and between different teams
  ▪ Team Concert will take into account participation in different projects and teams
Time

- A project has one or more schedules or **timelines**.
- Each timeline is broken down into a series of iterations.
- Each iteration can be broken into smaller iterations as required.
- A milestone marks the end of an **iteration**.
- At the end of any iteration, you may produce a **release**.

---

Work

- All work in your project is tracked as one or more Work Items.
- Different kinds of work items are available including:
  - **Plan work item types**, for example:
    - Plan Item
    - Epic
    - Story
  - **Execution work item types**, for example:
    - Task
    - Defect
- Plan work item types are used to capture high-level plan elements.
- Execution work item types are used to capture the lower-level details and the work should be completed in a single iteration.
- Each kind of work item has its own lifecycle.
Team, Work Items and Plans

- Work items connect your team to the plan
- A project can have an overall plan
  - This shows all high-level plan work items for the project
- Each team can have a plan for the project
  - This shows all high-level plan work items for that team for the project
- Each team can have a plan for an iteration
  - This shows the work assigned to that team for that iteration
- Schedule Risk assessment plans can also be used to work out the level of uncertainty that work items will be completed on time
- Developers can use Developer Taskboards to more easily visualise their work
Schedule Risk Assessment Plan

More detailed developer estimation: low, nominal, high

Color codes high risk tasks for quick identification and action

Automatically calculates probability of task fitting into the schedule

Developer’s Taskboard

See the work currently in progress

Drag and drop work items to change their state.
Plan Modes

- Plans can be displayed in different modes
- **View As: Backlog** is ideal for managing SCRUM backlogs
  - Support coarse & fine grained prioritization
  - Ranking is reflected in all planning views, e.g. iteration plans and release plans

---

### JUnit Product Backlog

<table>
<thead>
<tr>
<th>Team Area: JUnit Team</th>
<th>Iteration: 4.4</th>
<th>Closed: 4 Open: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>Planned Items</td>
<td>Charts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View As: Backlog</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task Description</th>
<th>Points</th>
<th>Priority</th>
<th>Remaining</th>
<th>Estimated</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a user I want to re-run only failed tests</td>
<td>6</td>
<td>High</td>
<td>6</td>
<td>62</td>
</tr>
<tr>
<td>Provide improved Assertion syntax</td>
<td>0</td>
<td>High</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>Improve documentation for 4.4</td>
<td>3</td>
<td>Medium</td>
<td>3</td>
<td>56</td>
</tr>
<tr>
<td>No change to download link to latest version</td>
<td>12</td>
<td>Medium</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

---

Lab #2 Scenario

- You want to track all the work on your projects.
- All your work (for example: plan items, stories, tasks and defects) are based around the concept of Work Items.
- You see how Work Items are fundamental to Rational Team Concert and how you use these work items to track the work you do.
- You prioritize and link your work so that you can do the right things at the right time in the plan.
Lab #2 Overview

- The instructor will create new Stories to create additional squawkers.
- You will create Task work items for your squawkers
- You will assign your work to the right team member (you!)
- You will set the priority for your Task work items and estimate how long they will take to complete
- You will link your new Task work items to the Stories created by the instructor. The relevant plans will be updated automatically.
- You will explore how tagging can make it easier to find work items
- The instructor will send you the plans via Chat and you will examine the plan.

Lab #2 Concepts Learned

- Work Items in Rational Team Concert are a central team artifact in the development process
- Everything gets tracked using Work Items so nothing gets lost which provides project transparency and real time data access
- Work Items are used to create the Iteration Plan linking project data to the overall plan
- Plans are live, dynamic and visible to the entire team helping to create a collaborative project environment
- Video overview available from the online Help under Tours
Keep Track of All Our Work

An IBM Proof of Technology

Objectives

- Explore Rational Team Concert query capabilities
- Create and run queries
- Use and configure the Team Central * and My Work * views to get a real time view of project, team and individual status

* Not currently available in Visual Studio
Real time collaboration

• The modules before showed how the project team plans the work for an iteration.
  
  ▶ But how does the project keep track of all the planned work items?
  ▶ How do I see who may help me with my actual problem?
  ▶ How do I get the most recent status of the project?

• What if your tool knows the actual status of your team's work?
  
  ▶ Rational Team Concert stores all artifacts for the development project in one repository and provides powerful query capabilities to retrieve and display data.

Real time collaboration

• Utilize Rational Team Concert’s extensive collaboration capabilities
  
  ▶ Define queries on Work Items to find your work and the work of others.
  ▶ See who is online and ready to collaborate with you.
  ▶ See the event log for build or work item events that are interesting to you and follow RSS feeds for News.
  ▶ Generate, display and export reports on the status and health of the project.

• Rational Team Concert displays the information in automatically refreshed views that are configurable, so that you are up to date with the information you need in real time.
Work Item Queries

- Provides real-time project health information and transparency of status through automated data gathering.

- Rational Team Concert provides a query mechanism to find work items in a Project Area allowing for more project transparency.
  - The query scope for work items is the project area.

- The user interface includes
  - an editor for building structured work item queries
  - an end-user configurable work item view to browse the query results.

Lab #3 Scenario

- You recently joined the development staff of the Squawk project
- Your environment was properly set up by accepting the invitation for the Core project team
- Now it is your task to become familiar with the work and the tasks to do.
- You are using the real time collaboration capabilities of Rational Team Concert to be up to date with the
  - latest news feeds,
  - status of the project and
  - work items assigned to you
Lab #3 Overview

- As a user you will
  - Write and run work item queries in the Eclipse client and in the Web UI.
  - Use the capabilities of the Team Central View.
  - Configure the My Work View.

Lab #3 Concepts Learned

- Rational Team Concert provides powerful query capabilities for work items creating real time access to detailed project data
- Create customized queries or use predefined queries to enable unique project views for a wide range of users
- Rational Team Concert helps teams collaborate by creating an environment where real time project status and data are available.
- Easily customized views to fit your needs
Performing and Sharing Your Work

An IBM Proof of Technology

Objectives

- Understand Software Configuration Management (SCM) concepts in Rational Team Concert
- Create and use a Repository Workspace for work assigned to you
- Create or make changes to artifacts under source control
- Associate changes with Work Items
- Deliver changes from Repository Workspaces to Streams
- Accept changes from other members of your team
- Understand conflict resolution
Basic Jazz SCM Anatomy

- Streams are for sharing
- Repository workspaces are your personal space
- The local workspace is a folder on your local file system where you develop and test
- Change-sets flow back and forth

Components

- Repository Workspaces
  - Partitioned into components
  - Jazz understands the structure of your components
  - Jazz directly supports component based development
Components (cont)

- Components Track Changes
  - Configuration of resources builds from the change set flow
  - Each change set builds on what came before

- Component’s Change History
  - A time-ordered sequence of change sets
  - Describes how the component’s content was built from nothing
Change set Details

- Composed from a collection of changes to one or more files and folders
  - A change set that affects multiple resources is committed as a single atomic unit
- Indicates the reason for the changes
  - Via a comment, and/or
  - By referencing the relevant work item
- Can be shared with another team member
  - Via a stream, or
  - From your repository workspace via a work item

Streams

- Stream
  - A place to share source with your team
Typical Journey For A Change set

1. Change

2a. Check-in

2b. Change set

3. Complete (usually combined with deliver)

4. Deliver

Change-set Delivery is Process Enabled

- The deliver operation is process-enabled, allowing the team's process to check and enforce delivery rules automatically.
Delivery Notifications

- An incoming change-set is
  - In the change history of the stream, but
  - Not in the change history of your repository workspace

- Accept adds the change-set to your repository workspace’s change history
Conflicts in Jazz SCM

- Component modified in multiple workspaces that have the same flow target, can lead to conflicts in the change sets that result
- Structural conflicts
  - incoming and outgoing change sets include changes to the same directory namespace, usually by moving, removing, or renaming files or folders
- Content conflicts
  - incoming and outgoing change sets include changes to the same file

Resolving Content Conflicts

- Manually Resolve
  - Resolve the conflicts in the Eclipse Compare Editor
- Auto Resolve
  - Attempts to automatically resolve by merging non-conflicting changes such as simple additions or removals
- Resolve with proposed
  - Replaces the file in your workspace with the one that contains the conflicting changes
- Resolve with mine
  - Replaces the file that contains the conflicting changes with the file that is currently in your workspace
- Discard
  - Resolve a conflict by discarding the change set in your workspace that conflicts with the one you have accepted
Resolving Structural Conflicts

- **Auto Resolve**
  - Attempts to resolve the conflict by automatically merging the content of the incoming change set with the content of your workspace

- **Resolve with proposed**
  - Applies all of the conflicting structural changes to your workspace

- **Move**
  - Apply a subset of the conflicting structural changes to your workspace by moving or renaming individual conflicted items

- **Resolve with Mine**
  - Remove the conflicting structural changes from the change set

---

Lab #4 Scenario

- You have been tasked with contributing your own squawker class along with its documentation and, optionally, its test case
Lab #4 Overview

- You will spend a little time understanding the key concepts of the SCM system in Jazz
- You will create your own squawker, basic documentation and optionally its test case against the work items you created in Module 2 Planning Your Work
- You will deliver this work so that other people can use it
- Finally, you will bring in changes from other members of your team so your code is up-to-date with everyone else

Lab #4 Concepts Learned

- Jazz Source Control provides private repository workspaces to track and back up your changes before you share them with the team using a stream for integration
- A change set is the fundamental unit of change and collaboration in your team environment
- A change set can be associated with a work item, which can then be delivered as a unit and provides traceability and transparency to the development lifecycle
- The Pending Changes view is central to these operations by enabling real time updates and efficiency
- Video overview available from the online Help under Tours
Remembering Well Known SCM Configurations

An IBM Proof of Technology

Objectives

- Understand how Component Baselines and Workspace Snapshots can be used
- Create new repository workspace from a snapshot for maintenance purposes
- Utilize the Pending Changes view to increase productivity
What About These Questions?

- How do I find a known good configuration of a component?
- How about a known good configuration of an entire stream?
- Hey, exactly what was in that milestone build a year ago?
- That is, what about fixed configurations that do not change anymore?
- Use baselines and snapshots…

A Baseline

- Is an immutable copy of a component’s configuration
  - At a particular point in time, and
  - There can be multiple baselines of a component
- Serves as a fixed point of reference
  - For initializing streams and repository workspaces
  - For sharing source with people or processes
- Can be easily compared
  - With the current state of a stream or repository workspace
  - With another baseline
A Snapshot

- Is a **collection of one baseline per component** in a repository workspace or stream
  - Captures an important repository workspace configuration for later re-creation
  - There can be multiple snapshots of a repository workspace or stream
  - Provides traceability to historical artifacts
- Like baselines, snapshots are used for sharing and collaborating with team members
  - Create a repository workspace or stream
  - Update the contents of a repository workspace
  - Re-create a prior build via a build created snapshot

Answers to those tough questions

- How do I find a known good configuration of a component?
  - Use a baseline!
- How about a known good configuration of an entire stream?
  - Use a snapshot!
- Hey, exactly what was in that milestone build a year ago?
  - Use a snapshot or baseline!
Lab #5 Scenario

- You have contributed your own squawker class along with documentation and delivered your work
- Your teammates have been creating and delivering their own squawkers and documentation, which you have accepted
- These changes need to be captured so that they can be used for further work or returned to at some point in the future if necessary

Lab #5 Overview

- The instructor will play the role of Team Lead, creating baselines and snapshots to capture all the work completed by the team
- You will then explore the new baselines and snapshots by querying their contents.
- You will revert a component in your workspace to a previous baseline version with the replace operation which provides a convenient way to reconfigure your workspace.
Lab #5 Concepts Learned

- **Baseline** and **snapshot** artifacts increase traceability and enable collaboration among teams and team members.

- **Baselines** are an efficient means to mark artifacts within a single component for later reference.

- **Snapshots** are an efficient means to mark artifacts across a set of related components for later reference.

- It is easy to create a new repository workspace or stream from a snapshot. This is useful for maintenance purposes, fixing builds or forking the code.

- The **Pending Changes** view is central to these operations by providing an easy to use interface to review changes and appropriately update your workspace.

Questions
User’s View of Build

An IBM Proof of Technology

Objectives

- Understand the build functionality of Rational Team Concert
- Understand the flexibility of the build process and how it enables collaboration and teaming
- Observe policies and processes that relate to consistency and repeatability
- Explore Build Results and observe traceability to artifacts
- Perform a build or a rebuild
Rational Team Concert Build

- Is an integral part of the project infrastructure
  - Consistent, repeatable process throughout the project
- Brings awareness of build progress and results to developers
  - Easy sharing of information
- Links build results to related Jazz artifacts
  - Integrated experience, traceability and tracking “baked in”
- Allow developers to have a private build area
  - Build and test code before delivering to the main branch
- Accommodates existing build technologies (Ant, CruiseControl, Build Forge, Maven, …)
  - Leverages technology that fits your project best
Build is very visible to the user

My builds
Published build
Recent builds
History

Personal builds

- Builds normally run from a dedicated repository workspace.
- Personal Builds
  - run from your repository workspace.
  - allow you to build your changes before delivering them to the stream.
  - provide you with some assurance that your changes will not disrupt the team builds when you deliver them.
Builds and Snapshots

- A build can request a snapshot
  - If there are any changes in a component since the last build
    - A new baseline is created with the same name as the snapshot name
  - Convenient for reproducing build problems

Web based Build Management

- From Rational Team Concert Web UI:
  - Request new build
  - Request rebuild of existing build
- Exposes build facilities to wider community
- Provides access to build function from any desktop
Lab #6 Scenario

- You have recently joined your company’s exciting new project called Squawk.
- By now you have
  - planned and tracked your work,
  - developed a new squawker,
  - and created baselines and snapshots.
- You are now ready to build your application with help of the Team Concert Build Engine.

Lab #6 Overview

- The instructor will then demonstrate how a build engineer, team lead or other appropriate role, can request a build for use by the project team
- You will explore the results of existing builds
- You will request a private build to ensure that your changes won’t break the build
Lab #6 Concepts Learned

- In this module you explored the build capabilities of Rational Team Concert. You have explored existing builds and learned how to request new builds or rebuilds.
- Treating the build as an integral part of the project infrastructure makes it easy to keep processes and policies consistent and repeatable.
- Every team member has access to build data which promotes communication and collaboration among the contributors – on local or remote sites.
- Linking build results directly to Jazz artifacts provides a high level of traceability.
- Using existing build technologies (Ant, CruiseControl, Build Forge, Maven, …) makes it easy to adapt to needs of different projects.

Questions
Exploring Changes and Traceability

An IBM Proof of Technology

Objectives

- This lab will demonstrate how information is linked within Rational Team Concert to establish traceability.
- Determine what work items and files are included in a build
- Determine change sets that are included in a build
- Determine who changes files, when and why
- Compare versions of a file
- Observe specific changes to files
Builds

**Keeps traceability with work items, change set, repository workspace, etc.**

- **Identify work items and change sets that went into the build**
- **Work Items**
  
  **Know what build it has been implemented, its change sets, etc.**

- **List associated work items**
- **Drill down into the details of a work item**
Change Sets
Easily allow users to understand to keep track of all related elements

Review change sets that make up the build
Change Explorer lists files that were modified for a given change set

Compare Changes
Quickly provide users the ability to identify differences in elements

Compare changes between versions of a file
Visualize Change History

Colors indicates when changes have been made, hover the mouse over the change and get more details

Use annotation to view specific changes

Lab #7 Scenario

- You have completed some builds for the Squawk project and are now ready to look at how Rational Team Concert links the software artifacts that make up the builds.
- You will investigate the build artifacts to see how Rational Team Concert automatically manages traceability.
- You will review the change sets (work items and associated changes under source control) that make up the build and explore the change history.
Lab #7 Overview

- You will experience how information is linked within Rational Team Concert.
- As a team member you will explore how traceability helps answer questions such as
  - What work items went into a build?
  - What changes were made for a work item?
  - What build did a work item get delivered in?
  - Who changed a file, and why?
  - What are the specific changes made on a resource?
  - How to visualize the change history for a resource?

Lab #7 Concepts Learned

- Rational Team Concert maintains full traceability for changes contained in a build
- Work items maintain a record of the changes made to resources maintaining consistency and transparency in the project
- Changes are collected and managed as Change Sets and available for reporting purposes and analysis
- Users can drill down into the detailed change history of every artifact, enhancing collaboration and quality
Endgame and a Tightened Process

An IBM Proof of Technology

Objectives

- Understand how process is defined in Jazz and implemented by Rational Team Concert
- Understand how roles can be used to control process workflow
Motivation for the Team Process Component

- Generally all software teams have some sort of process
  - May be formal, informal…
- Successful teams…
  - Believe their software process helps produce quality software
  - Own their process and accept accountability for it
  - Continually adapt their process to changing needs
- However, success depends on…
  - Common understanding by all team members
  - Consistent execution
- Many times…
  - Process relies on documents (or word of mouth) for understanding and human memory for execution and is otherwise very manual
  - Leads to inconsistent or erroneous execution

What if your tools understood how your team works?

In a Basic Process Model…

- Teams work on projects
- Each project follows a process
- Each team is unique and thus can work differently
- Work inside the scope of a team follows the team’s process
- Cross-team work follows the process of the broader team
- Team members play roles defined by the process
- Process manifests itself through artifacts types, operations manipulating the artifacts, and artifact change events.
Jazz Process Support

- Support different degrees of flexibility and formalism
- Allows for **predefined** processes
- Allows for **emerging** processes
- Allows for **variations**
- Allows for **exceptions**
- Allows for process **consolidation**
- Allows for process **evolution**
- Allows for **extensions**
- Put knowledgeable human in the center
- Comprises runtime, authoring, and inspection support

Project Area Iteration Structure and Terminology

**Timelines** are an element of a project area that own a set of deliverables and its production schedule (**maintenance**, **new release development**).
- Often represents parallel development
- A team area is associated with a timeline

**Iteration** represents some project work interval
- Any depth of nested iterations
- Process specification in any iteration
- May contain start and end dates

**Process state** is defined as the current iteration in a timeline
- Indicated by the blue arrow

Example:
*Main Development Line* process state: 2.0 M2
*Maintenance Development Line* process state: 1.0 Fix pack 1
Process is Defined in One or More Iterations

- Specified as a set of component operation rules
- Rules are assigned to user roles (default, contributor, team lead…)
- You can have the general process defined for the project
- Override/augment the general process in planned iterations

Active process defined by process state:
{1.0, M1, endgame} in this example

A team area can augment/override the process of any iteration

Project Delivery Plan in Jazz Terms

Iteration  | Q1 | Q2 | Q3 | Q4
--- | --- | --- | --- | ---
Timeline | Release 1 | Release 2 | Maintenance
Iteration | M1 | M2 | M3 | M4 | M5 | M6
Timeline | Release 2 | Release 3 | New Release Development
Lab #8 Scenario

- As you approach your final milestone, you have the chance to alter the process for the iteration so that your rules get stricter.
- For example, you might insist that all tests run to completion and without error before you are allowed to deliver any changes.

Lab #8 Overview

- In this lab, the team will move to the Endgame iteration of milestone 1.0 M2 and will experience a change in the process
  - In the M2 Endgame iteration the Core Library team has customized the process such that changes can be delivered only if the team lead has approved the work item associated with the delivery
- The instructor will move the project to the 1.0 M2 Endgame iteration
- As a user with contributor role on the Core Library teams, you will make a change to the squawker class
- During the delivery, user will notice the change to the squawker class for the Core Library will not complete because the Work Item associated with the delivery does not have the approval of the team lead
- The instructor (as team lead) will approve the Work Item
- The user will now be able to deliver the Work Item
- The instructor will move the project back to the 1.0 M2 Development iteration
Lab #8 Concepts Learned

- Jazz processes capture the idea and the notion of choreographies of collaboration
- With Jazz collaboration, rules are your friend not something you have to fight. Keep your processes as concrete as possible and as strict as necessary
- Process sandboxes allow ‘good things’ to happen on all levels
- Process support in Jazz is an ongoing endeavor
Taking Control of Your Project

An IBM Proof of Technology

Objectives

- Learn about the Jazz Dashboards and Reports and how these powerful capabilities can assist your team to track project status and make informed-based decisions to keep it on track.
Dashboards

- Dashboards are a Web UI component intended to provide information about the project status at a glance.
- It provides for easy drill down to get more complete information.
- Dashboards are available in the Rational Team Concert Standard and Enterprise editions.
Dashboards and Reporting for progress

Dashboards – personalized visibility

Display your choice of reports and queries in your own dashboard, e.g. to control the flow of work items.
Roll up information and drill down for details

Dashboards – project status with Burndown charts
Reports – Displaying the project status

For a detailed Report open the Reports page and choose from a variety of available reports.

- Reports are available in the Rational Team Concert Standard and Enterprise edition.
- Rational Team Concert uses the BIRT* reporting engine.
- A variety of out-of-the-box reports are available to display an actual overview of your projects:
  - Reports for the health of your builds
  - Reports for viewing the team load and the distribution of work items
  - Reports for your code
  - Etc.
- Reports can be arranged in the Web UI Dashboards
- Reports can be exported to: .pdf, .xls, .doc, .ppt formats

*BIRT is an open source Eclipse-based reporting system that integrates with your Java™/J2EE application to produce compelling reports.
Lab #9 Scenario

- As a member of the Squawk project, create and customize your own private dashboard and include content from another project area. You will also explore some out-of-box reports.

Lab #9 Overview

- The instructor will create a new Project Area which you will later include in a dashboard
- Customize your personal dashboard for the Squawk project
- Add content from the JUnit project on your Squawk project dashboard
- Explore reports using the Web UI
Lab #9 Concepts Learned

- Rational Team Concert provides **transparency and control** via customizable dashboards

- Rational Team Concert automated project management dashboards are transparent to everyone – not just managers. This immediate and **automated feedback helps keep teams on track and motivated** to achieve project goals

- Rational Team Concert comes with a variety of out-of-the-box report formats to display and export the actual real time, in context project status.

Questions
Integrating with Other SCM Systems

An IBM Proof of Technology

Objectives

- Explore how Rational Team Concert integrates with other SCM systems
- See how Rational Team Concert synchronizes its repository with Rational ClearCase
Interoperation with other CCM Systems

**Importer**

External CCM Repository

![Diagram](image1.png)

**Synchronizer (Interoperation)**

External CCM Repository

![Diagram](image2.png)

**Bridge (Interoperation)**

External CCM Repository

![Diagram](image3.png)

---

Interoperation with other Source Control systems

Synchronization between SCM systems and Rational Team Concert

**Source Control**

Uses:
- Manage code changes made to same code base by teams using Rational Team Concert and another SCM system
- Gradual and low-risk migration from other SCM system to Rational Team Concert

Example: ClearCase Synchronizer

---

**Bridge between SCM Systems and RTC Work Items**

Uses:
- Introduce Rational Team Concert when not willing or ready to change SCM systems. Allows project to use their current SCM system along with Rational Team Concert Work Items, Agile Planning, and Build

Example SCM Bridges: ClearCase, Subversion, Git (incubator)
Work Items – Synchronizers (Connectors) and Bridges

- Synchronizers (Connectors)
  - Synchronize data between systems
  - Synchronization Rule controls mapping
  - Changes to data values and process state are synchronized with the other system
  - Native user interface for either system can be used to make changes
    - Rational Team Concert IDE Client or Web UI
    - Native UI for external system

- Bridges
  - Support relationships between Work Items and a similar external domain object (CM)
  - Data is not synchronized; references provide a loose coupling of data in each system
  - Changes must be made using the appropriate user interface for each record
    - ClearQuest Bridge as an example:
      - Relationships must be defined using the ClearQuest UI as integrated in the Rational Team Concert Web UI

Integrating Other Repositories with Team Concert

- Importer – migrate to Rational Team Concert
  - Jazz SCM – CVS, Subversion, ClearCase
  - Jazz Work Items – JIRA, Bugzilla
- Bridge – Lifecycle integration
  - Jazz Work Items – ClearCase, Git, Subversion (SCM)
- Synchronizer – interoperation between repositories
  - Jazz SCM – ClearCase
  - Jazz Work Items - ClearQuest
ClearCase Bridge

- Now you can link Rational Team Concert work items with ClearCase UCM change sets
- If you use the ClearCase Remote Client and UCM you can associate a UCM change set with a Team Concert work item.
- Then you can use agile planning, taskboards and dashboards to show project status of work done in ClearCase!

References to ClearCase Activities will also show up on the Links tab of their associated work items.

Opening a ClearCase Activity link from a work item editor will show the activity properties in the Properties view, just as if the properties were requested from CCRC directly.

ClearCase Importer Wizard

- Imports from ClearCase base or UCM with history
- Choose all baselines or particular baselines or labels
- ClearCase 7.0, 7.0.1 and 7.1.x supported
ClearQuest Bridge
Lower Total Cost of Ownership

- Reduces Time/Network Traffic
  - No Waiting for Data Synchronization so Key Use Cases Will Be Faster
  - Quicker Access to Record Information
- Lower Administration
  - alternate mechanism for linking ClearQuest with Rational Team Concert; create & maintain sync rules not required
- More Manageable Repositories
  - No Data Replication - Helps Minimize Database Growth
- Integrated User Experience
  - Reduces Training Costs
    - ClearQuest Bridge UI Provides Integration On The Glass Between the Rational Team Concert and ClearQuest
    - Appears As If You Are Working In a Jazz-based Environment

ClearQuest Importer Wizard
Provide ability to move select ClearQuest records to a Rational Team Concert project

- Import ClearQuest records to an intermediate XML format
- Select scope of import using ClearQuest query
- Create a mapping file to map ClearQuest Record types to work item attributes
  - Sample provided
- Import into an Rational Team Concert project once mapping has been validated.
CSV Importer
*Streamlines migrations*

- Create new work items or update existing work items
- First row or custom XML mappings
- Allows for work items to be imported from other systems

Lab #9 Overview

- In this lab the instructor will demonstrate how to synchronize code changes between Rational Team Concert and Rational ClearCase
Project Growth and Multi-stream Development

An IBM Proof of Technology

Objectives

- Understand the support for parallel development in Rational Team Concert
Growth and Adding Teams

- Project growth leads to multiple inter-dependent teams
- Each team needs its own stream
  - Quickly collaborate and share changes with each other
  - Run builds on a scheduled basis, as well as ad hoc
- Enhance ability for cross team collaboration and communication
  - Manage cross team dependencies
  - Project build stability and transparency
- Need a stream for cross team sharing and project builds
  - Well known change adoption schedule and process
  - Consistent and repeatable successful full project builds

Typical Component Baseline Flow

- User Interface Stream
  - Core
  - UI
  - UI Tests
- Integration Stream
  - Core
  - Core Tests
  - Documentation
  - UI
  - UI Tests
- Core Library Stream
  - Core
  - Core Tests

April on User Interface
- Core
- UI
- UI Tests

Jerry on Core Library
- Core
- Core Tests
Other Scenarios

- Maintenance
  - New stream for maintenance
  - Created from final release snapshot
  - Isolated from daily development
  - Easy to move changes between streams

- Community exploration
  - Single person exploration can use a repository workspace
  - Community exploration will require sharing and collaboration
  - New stream created from a development stream snapshot

Concepts Learned

- In this module you explored the support for parallel development offered by Rational Team Concert.
- You have explored handling multiple streams and the sharing of component dependencies between them.
- Rational Team Concert enables
  - Increased team productivity by allowing parallel development
  - Enhances the delivery policies and process while improving baseline consistency
  - Supports seamless interaction for globally distributed teams
  - Establishes traceability and transparency among project artifacts
Session summary

An IBM Proof of Technology

We have described current collaboration challenges with distributed teams.

We have explored how Rational Team Concert can:

- Enable development teams to collaborate in real time in the context of the work they are doing, especially in globally diverse environments.
- Enable projects to be managed more effectively by providing visibility into accurate project health information drawn directly from actual work.
- Automate traceability and auditability by managing artifacts and their inter-relationships across the lifecycle empowering teams to deliver more value.
- Provide customizable process design and enactment through rule-based process guidance, automation and definable checkpoints.

We have provided a hands on experience using Rational Team Concert to automate the software delivery process.
Next steps

- Engage your local Rational team
  - Provide a customized demo for your team
  - Conduct a targeted proof of concept
- Register on jazz.net and explore learning tutorials and videos

Additional resources

- Learn more about and download free trials of Rational Team Concert at
  http://ibm.com/rational/rtc
- Explore Rational Team Concert tutorials, demos and other developer learning resources
  http://ibm.com/developerworks/spaces/jazz
- Participate in the open commercial development of Jazz by joining the community
  http://jazz.net
- Learn more about the Jazz technology and the future IBM Rational product roadmap
  http://ibm.com/rational/jazz/roadmap
The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.