Innovate2010

The Rational Software Conference

Let's build a smarter planet.

Jazz Source Control Best Practices

Shashikant Padur

RTC SCM Developer

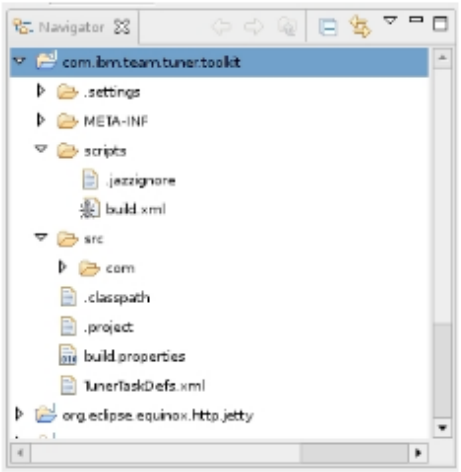
Jazz Source Control Mantra

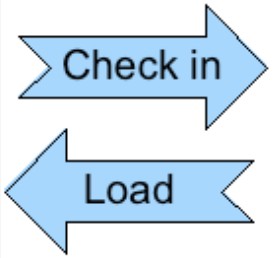
- The fine print
  - Fast, easy, and a few concepts to support many flexible workflows

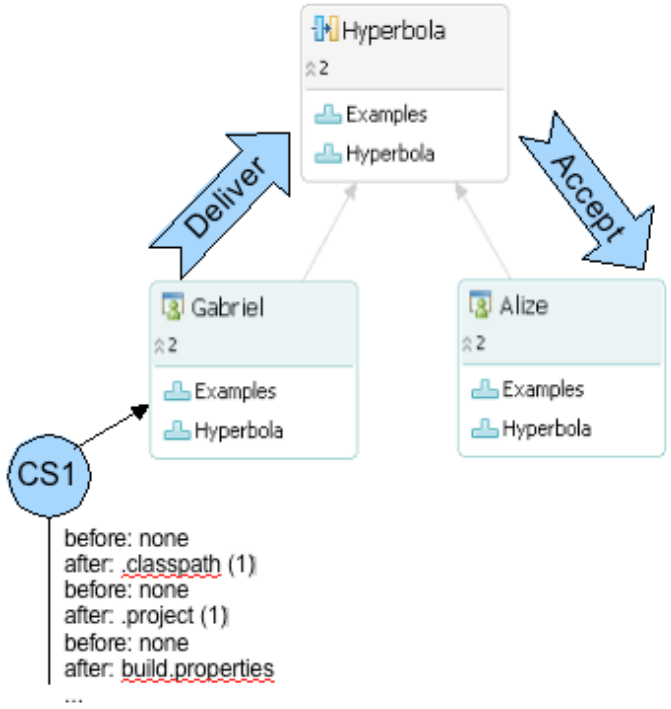
- Give all users access to source control
- Stop talking about it and just do it!
- It's never too simple
- Flowing changes to anyone

Give all users access to source control

- Decouple giving your changes to your team from using the SCM system as a tool.
- Check-in, discard, suspend, history, revert can be used before you make your changes available to others.







Jazz Source Control Mantra

- Stop talking about it and just do it!
  - Focus on making and collaborating on changes
  - Remove the word branch, but make parallel development just happen

Repository workspaces - Provides constant isolation. You don't have to make your changes visible to the team just to backup or use the repository features.

Suspend and Resume - Provides task level isolation for personal work.

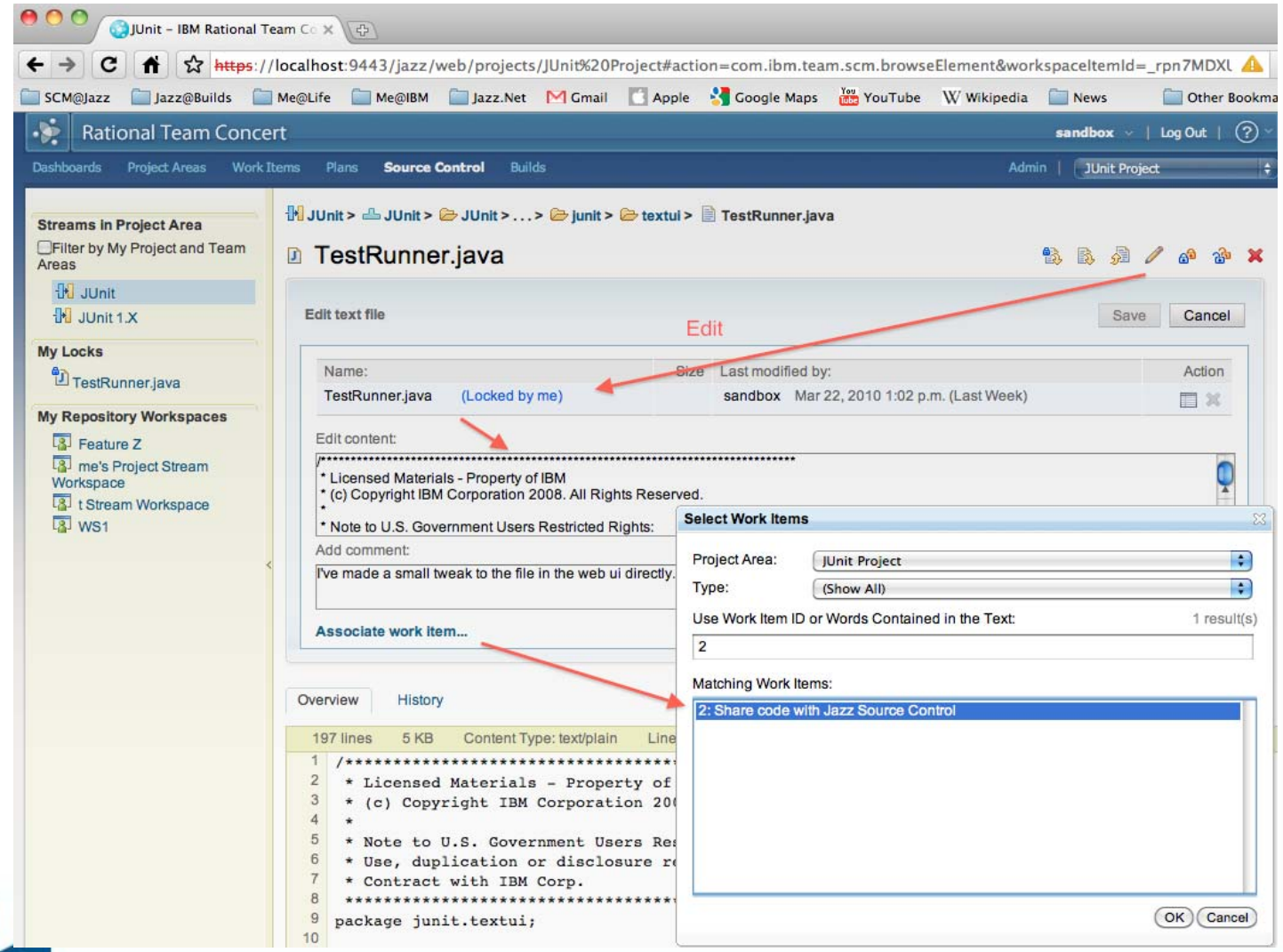
Work Item links - Provides light weight task level isolation for personal of team work. Work on a feature, attach to a work item and discard from your workspace. You or someone else continues the work by accepting the change sets back into their repository workspace.

Streams - Provides team or feature isolation.

Timelines - Provides process isolation.

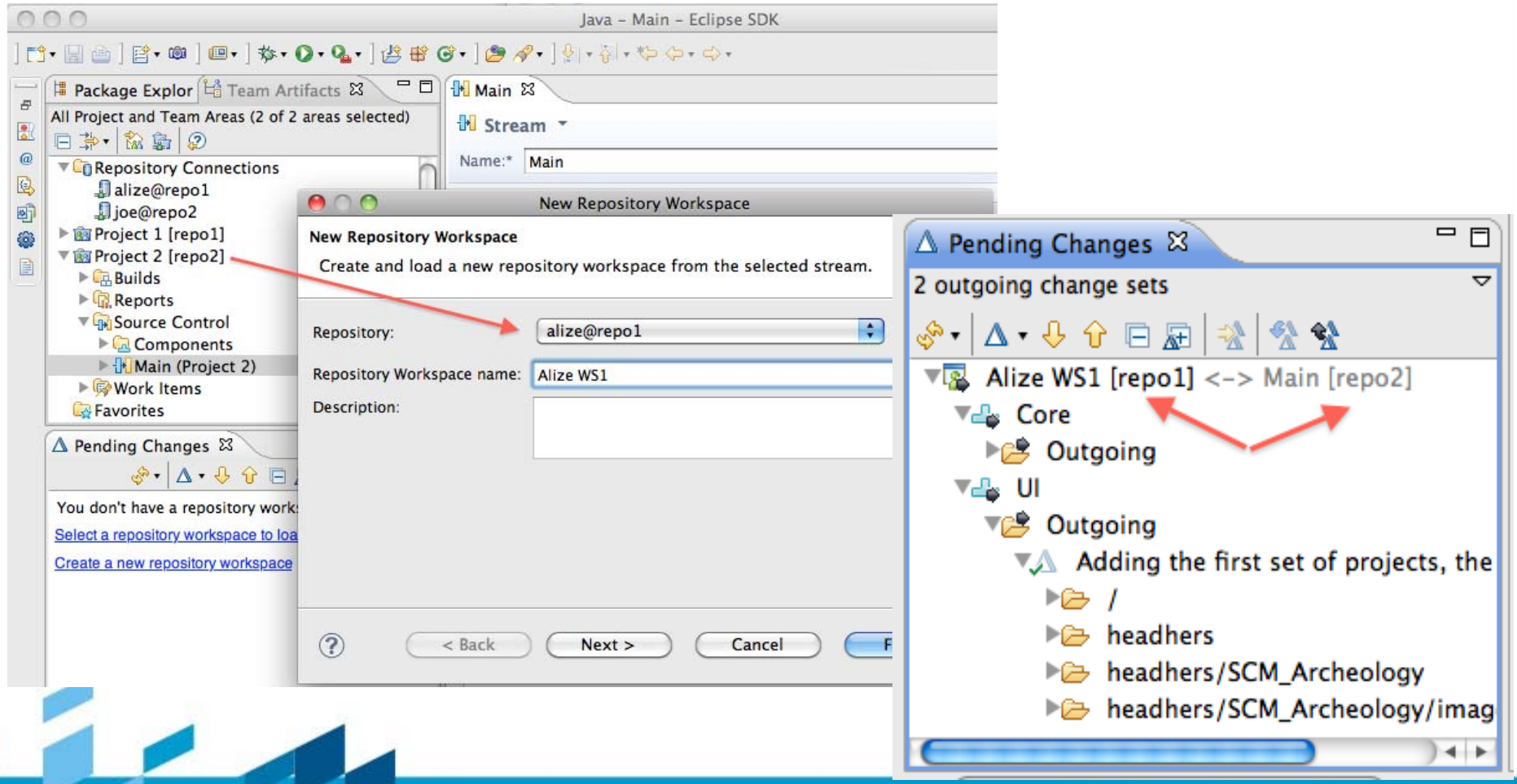
Jazz Source Control Mantra

- It's never too simple
  - Version control from the web



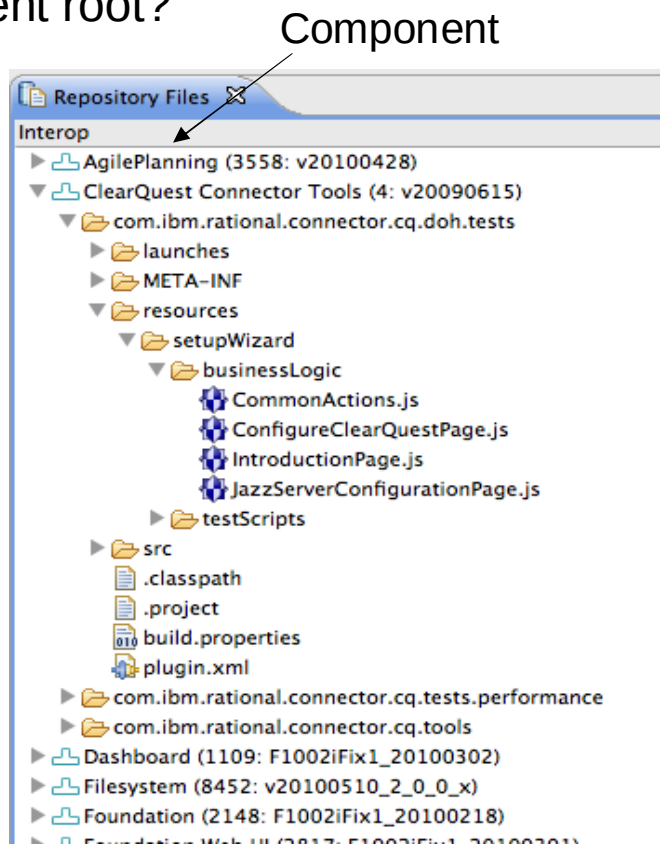
Jazz Source Control Mantra

- Flowing changes to everyone – interconnected change flows



Repository Layout

- File structure
  - Which file tree level should make as the component root?
- There isn't a clear rule but some considerations:
  - How do you plan on loading the files on disk?
  - Currently, loading at the component level is easiest to get started.
  - Depends on the IDE being used
    - Eclipse doesn't like files at the root of a component so mapping to sub-folders in the component can make it easier.
    - Visual Studio manages things differently we recommend good articles to help
      http://jazz.net/library/article/117 – Sharing solutions
      http://jazz.net/library/article/445 – Mapping source to components.



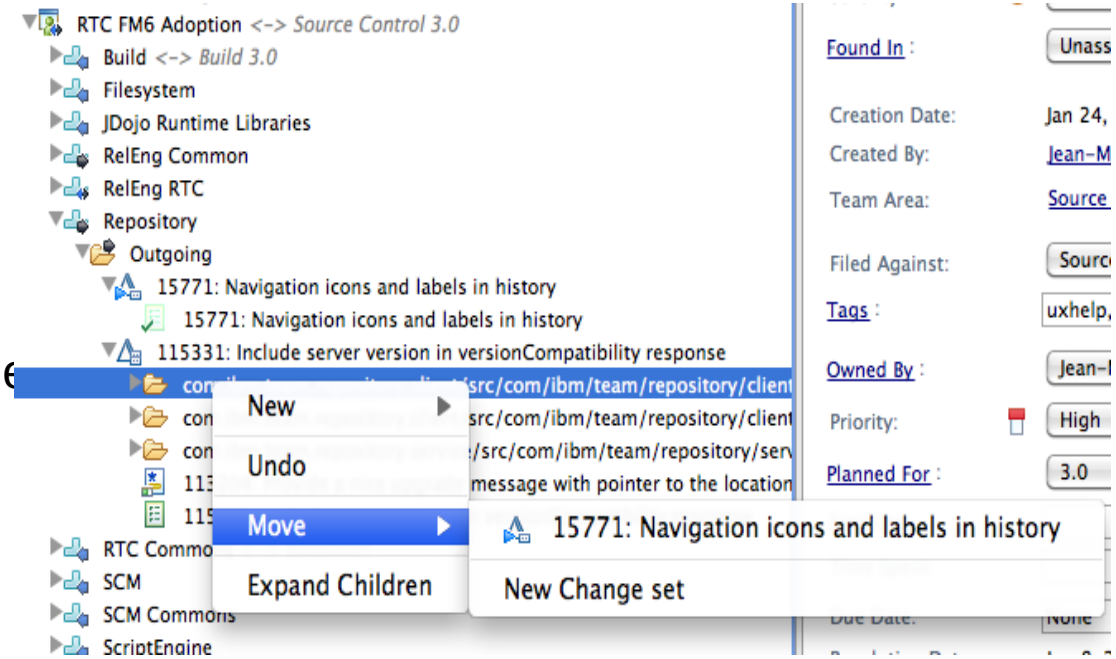
Repository Layout (cont...)

- Component structure
  - You can start with one component.
  - Recommendation is no more than 70K file per component (current performance limitation that will be removed in a future release)
  - You can easily move folders between components while retaining version history.

- Create streams with components that developers can load as-is in most cases

Managing Changes

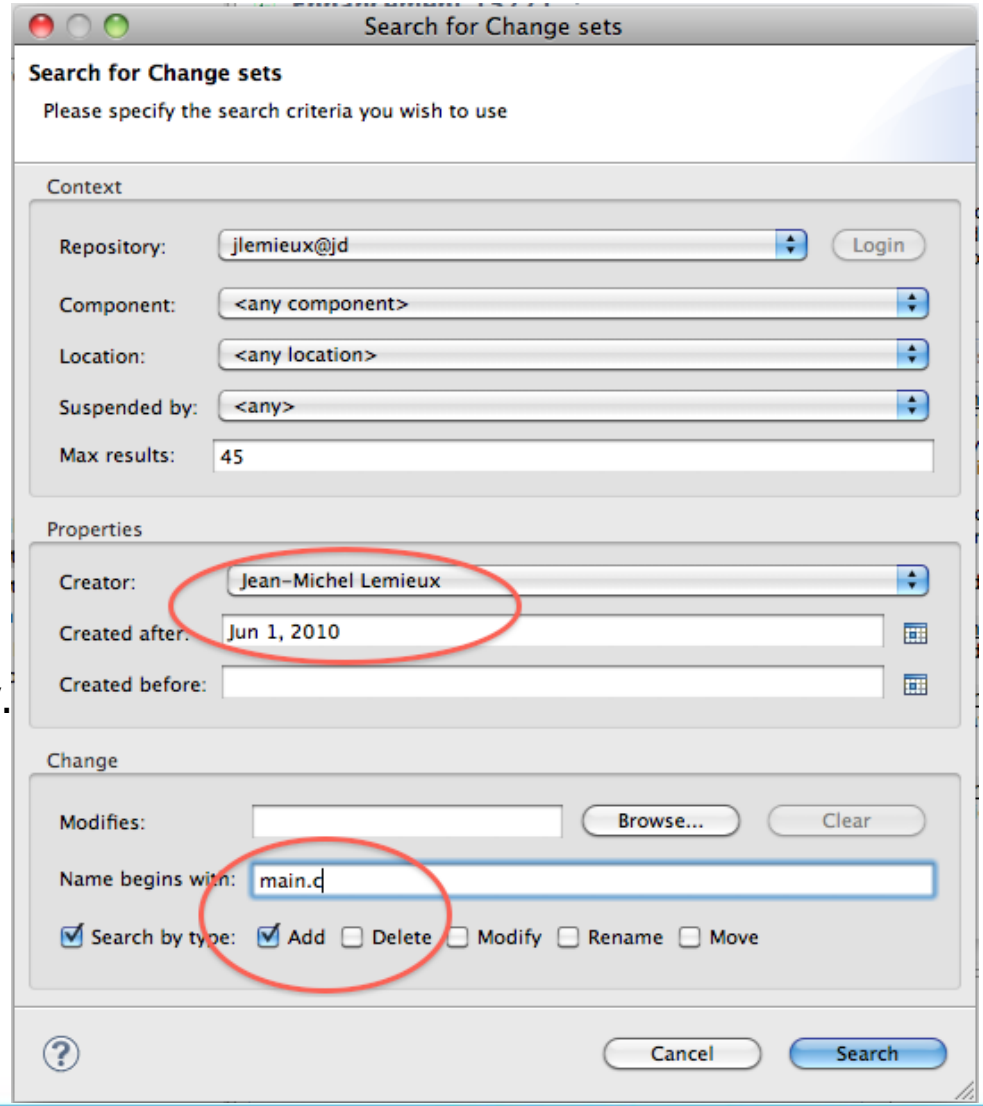
- Keeping logical sets of changes together
  - Move changes between change sets
  - Suspend to make sure changes are really isolated when working on tasks that should be isolated or when you aren't delivering all changes together.
  - Don't be afraid of closing change sets at stable points when making large changes.
- Using patches to merge changes
  - With large changes that take several days, it's possible to collapse a set of closed change sets together



Suspending and Resuming

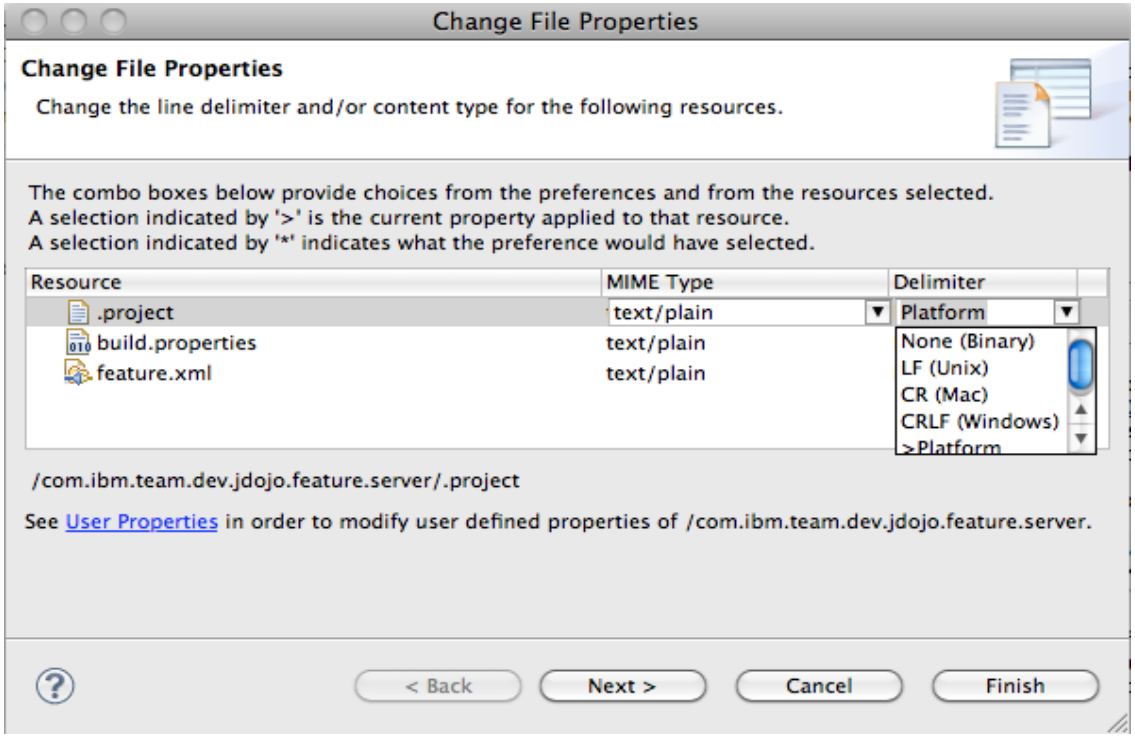
- Suspend before accepting when incoming causes conflicts

- Discarding isn't that scary
  - We don't delete changes and you can always find them
  - The difference between suspend/resume and discard is how active change sets are handled. Discard will close them automatically.
  - You can always search for changes.



Encoding and line endings

- Defaults are guessed and right in most cases, but...
  - If someone complains that on different platforms they are wrong, it's because the guess wasn't right.
  - It's easy to fix them all up.



Loading

- Basic Patterns
  - Loading Eclipse Projects
  - Loading from Visual Studio
- Advanced Load Scenarios
  - Loading Folders Outside of the Eclipse Workspace
  - Loading the Component Root Folder
  - Loading a Folder More Than Once
  - Loading a Single File

Loading using the Source Control Command Line Client

http://jazz.net/library/article/192 – Loading Content from Source Control

Undoing changes

- You've delivered a change set but it's bad, what are the options?
  - If there is a portion of the change that is bad, just undo it and check-in a new change.
  - If a large portion of the changes are bad, you can select the change set in the History view and run “Reverse” to create an inverse of the change set. You can then modify, pick and chose, and deliver a new change set.
  - At last resort, you can remove the changes from a stream. You can discard change sets from a repository workspace then replace the component in the stream with the exact contents of the component in your repository workspace. This will remove change sets from history. Don't worry, nothing is lost as a backup baseline is create in the stream that would allow you to go back.
- Recommendation is to deliver new change sets with reversals, as it makes it easier to track the audit trail and keep changes incoming.

Sharing repository workspaces

- Think of repository workspaces as backups of your work.
- Use them to work on a set of changes on two different machines.
- You'll have to reload as you move between machines re-using the same repository workspace.
  - Re-loading a workspace will only load what is new.
  - You will get a warning at the top of the pending changes view that reminds you to reload

Stream structure guidelines

- Start simple
- You don't require a stream per component
- Add them as needed
  - Add a stream per team who needs some isolation
  - Add a stream per team which needs to collaborate
  - Add a stream as a quality gate (eg, dev to GA)
- Setup of write permissions at the component level

Merging between streams

- Try not to deliver baselines between streams. If you do, they will be harmonised but don't provide
- Always merge everything into the latest so that you can keep track of backported changes.
  - We have en enhancement to allow marking change sets as merged, even when the changes aren't incorporated. Signal that merging occurred in other ways.
- Don't be afraid of overwriting on conflicts

Ad-hoc collaboration options

- Jazz Source Control provides several options for collaborating on change sets, each with it's own pros and cons
  - You have changes that you want someone else to try out
    - Ask them to change their flow target to your repo workspace and accept them. They will have to be closed.
    - Attach them to a work item.
    - You can accept back and forth between repo workspaces or with work items.
    - Work items are good because they track the reason for the changes with the changes and allow a conversation to

Having shared files that shouldn't be checked-in

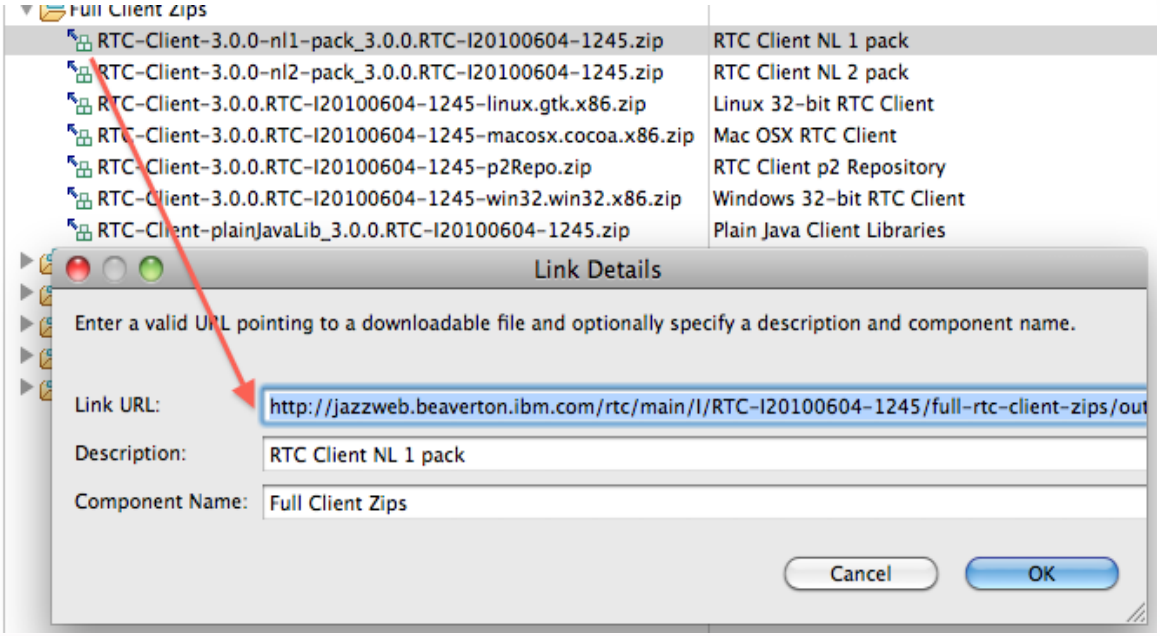
- There is a pattern in which files are checked-into the repository but shouldn't be modified and checked-in by accident.
- Check in the files and close the change set.
- Add a .cvsignore to ignore the files from now on, check it in and deliver all the changes.
- From now on changes to those files will be ignored, so they can't be accidentally checked-in.
- You can always un-ignore, check-in, and

Deleting repository workspaces and streams

- When you delete repository workspaces and streams we don't actually delete the contents. Baseline before you delete so that you can always go back to the components.
- Streams can't be deleted if they have snapshots associated with them. You can move the snapshots to another stream.
- Instead of deleting you can also repurpose streams by taking stable streams and replacing their contents with component other

Binary Files

- Check in considerations
  - No limit on binary file size
  - Binary files are delta compressed and similar content is shared
  - Currently can't delete source controlled content from the repository, is being considered in the future.
- Build output
  - Output of builds can be stored with the build result
  - Builds when deleted will clean-up their content
  - However, large build results (eg, over 1GB) can affect server performance on upload. Instead, you can keep the large build results on shared storage and keep links in the build result.



Thanks

- This was just a sneak peak
- Many great articles and videos at jazz.net

http://jazz.net/projects/rational-team-concert/learnmore/

- For an introduction to Jazz Source Control see http://jazz.net/library/presentation/20

Innovate2010
The Rational Software Conference

THANK YOU