



## **An IBM Proof of Technology**

# **Collaborative software development using IBM Rational Team Concert**

Eclipse Client Version  
Lab Exercises



PoT.Rational.07.2.038.04

© Copyright International Business Machines Corporation, 2008, 2009. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

---

<b>LAB 1</b>	<b>SETTING UP THE TEAM.....</b>	<b>4</b>
	1.1 ACCEPT THE TEAM INVITATION .....	4
	1.2 CONFIGURE INSTANT MESSAGING.....	11
	1.3 MEET THE TEAM.....	14
<b>LAB 2</b>	<b>PLANNING YOUR WORK .....</b>	<b>16</b>
	2.1 INSTRUCTOR DEMO - CREATE NEW PLANS .....	16
	2.2 WORKING WITH WORK ITEMS.....	16
	2.3 TAKE A TOUR OF THE EXISTING SQUAWK PLANS .....	32
	2.4 USING TAGS IN TEAM CONCERT.....	37
	2.5 INSTRUCTOR DEMO - REVIEWS PLANS AND DISTRIBUTE .....	39
	2.6 OPEN THE REVIEWED PLAN .....	39
<b>LAB 3</b>	<b>KEEPING TRACK OF ALL OUR WORK.....</b>	<b>41</b>
	3.1 CREATE A SIMPLE QUERY WITH THE ECLIPSE CLIENT INTERFACE .....	41
	3.2 USE THE RATIONAL TEAM CONCERT WEB UI.....	48
	3.3 USING AND CONFIGURING REAL TIME PROJECT STATUS – TEAM CENTRAL VIEW.....	56
	3.4 TRACKING YOUR WORK – MY WORK VIEW .....	64
<b>LAB 4</b>	<b>PERFORMING AND SHARING YOUR WORK.....</b>	<b>69</b>
	4.1 CREATING YOUR REPOSITORY WORKSPACE FROM THE CORE LIBRARY STREAM.....	70
	4.2 EDITING YOUR SQUAWKER.....	82
	4.3 EDIT THE TEST CASE FOR YOUR SQUAWKER.....	85
	4.4 DELIVERING YOUR SQUAWKER CODE AND RESOLVING YOUR IMPLEMENTATION WORK ITEM .....	87
	4.5 ACCEPTING CHANGES FROM OTHER MEMBERS OF YOUR TEAM .....	93
<b>LAB 5</b>	<b>REMEMBERING WELL KNOWN SCM CONFIGURATIONS .....</b>	<b>95</b>
	5.1 INSTRUCTOR DEMO - CREATING BASELINES FOR THE CORE LIBRARY COMPONENTS.....	96
	5.2 INSTRUCTOR DEMO - TAKING A SNAPSHOT OF THE CORE LIBRARY REPOSITORY WORKSPACE.....	96
	5.3 INSTRUCTOR DEMO - PROMOTE THE SNAPSHOT TO THE APPROPRIATE STREAMS.....	96
	5.4 ACCEPT ALL INCOMING BASELINES AND CHANGE-SETS .....	97
	5.5 EXPLORE THE NEWLY CREATED SNAPSHOTS.....	99
<b>LAB 6</b>	<b>USER'S VIEW OF BUILD .....</b>	<b>103</b>
	6.1 INSTRUCTOR DEMO - START THE TEAM CONCERT BUILD ENGINE .....	104
	6.2 INSTRUCTOR DEMO - REQUESTING A BUILD.....	104
	6.3 EXPLORING AN EXISTING BUILD .....	104
	6.4 REQUESTING A PERSONAL BUILD.....	111
	6.5 BUILD INFORMATION IS AVAILABLE FROM THE WEB UI .....	115
<b>LAB 7</b>	<b>EXPLORING CHANGES AND TRACEABILITY .....</b>	<b>118</b>
	7.1 WHAT WORK ITEMS WENT INTO THE BUILD .....	118
	7.2 WHAT CHANGES WERE MADE FOR A WORK ITEM .....	120
	7.3 WHO CHANGED THIS FILE, WHEN AND WHY?.....	123
	7.4 WHAT ARE THE SPECIFIC CHANGES MADE ON THIS RESOURCE? .....	125
	7.5 HOW CAN WE VISUALIZE THE CHANGE HISTORY FOR A RESOURCE? .....	126
<b>LAB 8</b>	<b>ENDGAME AND A TIGHTENED PROCESS .....</b>	<b>130</b>
	8.1 INSTRUCTOR DEMO – REVIEW THE PROCESS FOR END GAME .....	130
	8.2 INSTRUCTOR DEMO – CHANGE THE PROCESS BEHAVIOR .....	130
	8.3 OBSERVE PROCESS ENACTMENT IN ACTION .....	131
	8.4 INSTRUCTOR DEMO – APPROVE THE NEW END GAME WORK ITEMS .....	144
	8.5 DELIVER APPROVED CHANGES.....	144
<b>LAB 9</b>	<b>TAKING CONTROL OF YOUR PROJECT.....</b>	<b>146</b>
	9.1 PERSONAL DASHBOARD CONFIGURATION.....	146
	9.2 EXPLORING REPORTS USING THE WEB UI .....	166

<b>LAB 12</b>	<b>BE AGILE WITH ADVANCED SCM .....</b>	<b>172</b>
12.1	START WORKING ON LOW PRIORITY LONG TASK.....	172
12.2	INSTRUCTOR DEMO - TEAM LEAD CREATES A NEW STORY FOR ENHANCEMENT.....	179
12.3	REVIEW WORK LOAD AVAILABILITY OF THE TEAM.....	179
12.4	SUSPEND CURRENT WORK, AND START WORKING ON THE NEW TASK.....	185
12.5	COMPLETE THE TASK, COLLABORATING WITH TEAMMATES BY 'CODE'.....	187
12.6	RESUME THE SUSPENDED WORK.....	198
<b>APPENDIX A: LAB 2</b>	<b>PLANNING YOUR WORK - DEMO .....</b>	<b>201</b>
A2.1	CREATE NEW PLANS.....	202
A2.5	REVIEW PLANS AND DISTRIBUTE.....	226
<b>APPENDIX B: LAB 5</b>	<b>REMEMBERING WELL KNOWN SCM CONFIGURATIONS – DEMO .....</b>	<b>230</b>
B5.1	CREATING BASELINES FOR THE CORE LIBRARY COMPONENT.....	230
B5.2	TAKING A SNAPSHOT OF THE CORE LIBRARY REPOSITORY WORKSPACE.....	239
B5.3	PROMOTE THE SNAPSHOTS TO THE APPROPRIATE STREAMS.....	241
<b>APPENDIX C: LAB 6</b>	<b>USER'S VIEW OF BUILD – DEMO .....</b>	<b>243</b>
C6.1	START THE TEAM CONCERT BUILD ENGINE.....	243
C6.2	REQUESTING A BUILD.....	246
<b>APPENDIX D: LAB 8</b>	<b>ENDGAME AND A TIGHTENED PROCESS – DEMO.....</b>	<b>254</b>
D8.1	REVIEW THE PROCESS FOR END GAME.....	254
D8.2	CHANGE THE PROCESS BEHAVIOR.....	260
D8.4	APPROVE THE END GAME WORK ITEMS.....	262
<b>APPENDIX F: LAB 10</b>	<b>INTEGRATING WITH OTHER SCM SYSTEMS – DEMO .....</b>	<b>266</b>
F10.1	PREPARING TO SYNCHRONIZE TEAM CONCERT AND CLEARCASE REPOSITORIES.....	268
F10.2	BROWSING AND COMPARING BASELINES.....	271
F10.3	SYNCHRONIZE (EXPORT) TEAM CONCERT CODE CHANGES TO CLEARCASE.....	273
F10.4	SHOW SYNCHRONIZATION RESULTS (FROM TEAM CONCERT TO CLEARCASE).....	279
F10.5	SYNCHRONIZE (IMPORT) CLEARCASE SOURCE CONTROL FILE CHANGES TO TEAM CONCERT.....	282
F10.6	SHOW SYNCHRONIZATION RESULTS (FROM TEAM CONCERT TO CLEARCASE).....	287
<b>APPENDIX G: LAB 12</b>	<b>BE AGILE WITH ADVANCED SCM - DEMO .....</b>	<b>290</b>
G12.2	INSTRUCTOR DEMO - TEAM LEAD CREATES A NEW STORY FOR ENHANCEMENT.....	291

THIS PAGE INTENTIONALLY LEFT BLANK

---

## Lab 1    Setting up the Team



### Lab Scenario

You arrive at work on day 1 and receive an email inviting you to join the project. You fire up Rational® Team Concert and get connected to the project right away. You use the built-in instant messaging support to connect to colleagues and discuss how things went so well for your favorite sports team last week.

Once you have completed this module, you will be ready to start planning the work for this iteration.



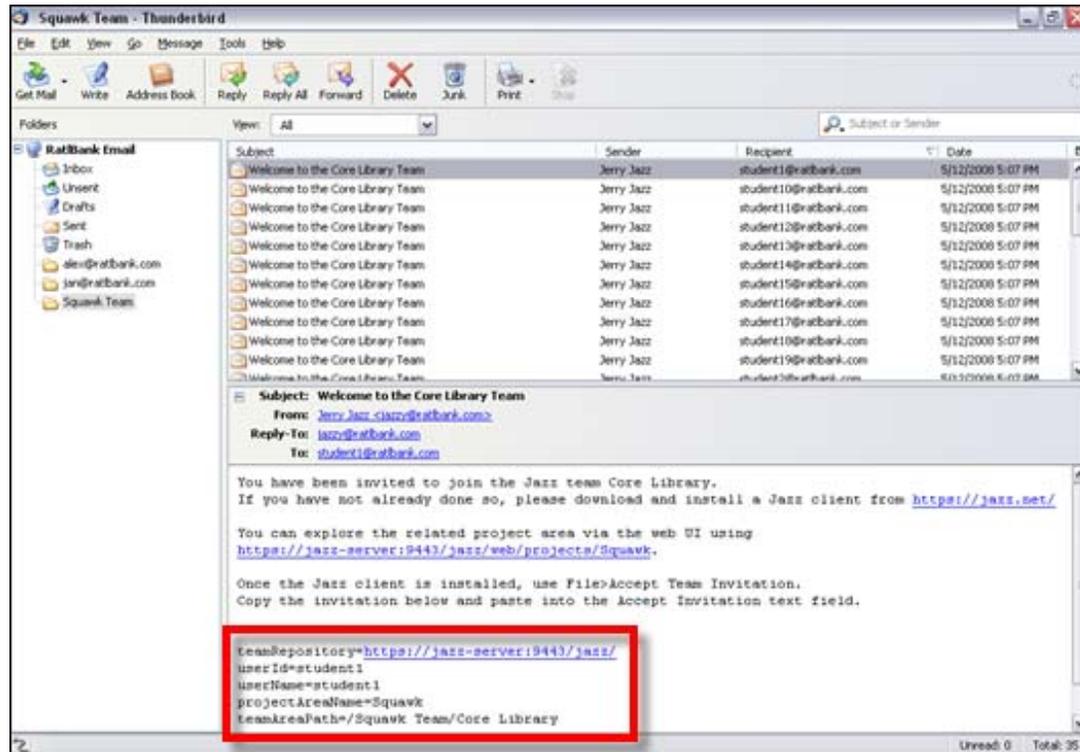
### Important!

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure what your student number is, please check with the instructor.

### 1.1    Accept the team invitation

- \_\_\_1.    Select  (Thunderbird Email Client) in the Quick Start menu to run the Mozilla Thunderbird email client

- \_\_2. Check for email from Jerry Jazz, team lead
- \_\_a. In the mail folders list, select **Squawk Team**.
- \_\_b. Scroll down the list of mail messages to find the **Welcome to the Core Library Team** message whose recipient matches your student id, e.g. student<N>@ratlbank.com.



Select and copy the invitation text from the email similar to the highlighted text shown above. The text will look similar to this:

```
teamRepository=https://jazz-server:9443/jazz/
userId=student1
userName=student1
projectAreaName=Squawk
teamAreaPath=/Squawk Team/Core Library
```

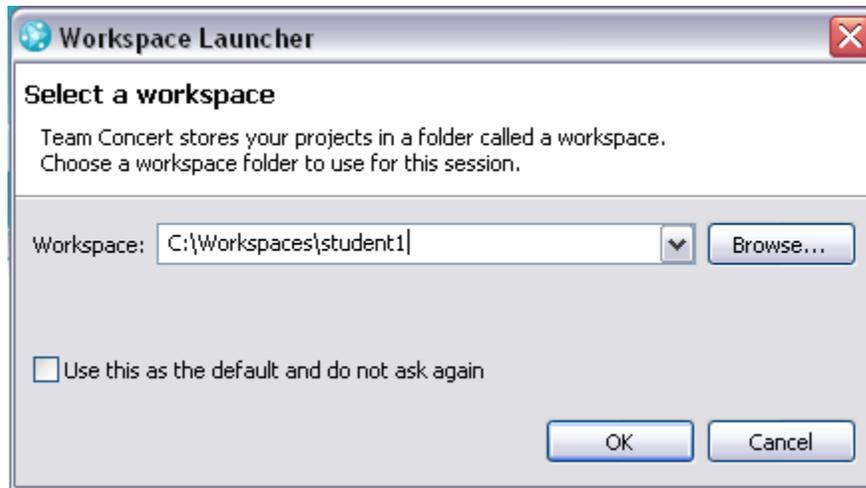
You will use this text to connect Rational Team Concert to the Squawk Project and the Team Areas you are a member of.

- \_\_c. Close the Thunderbird Email client.

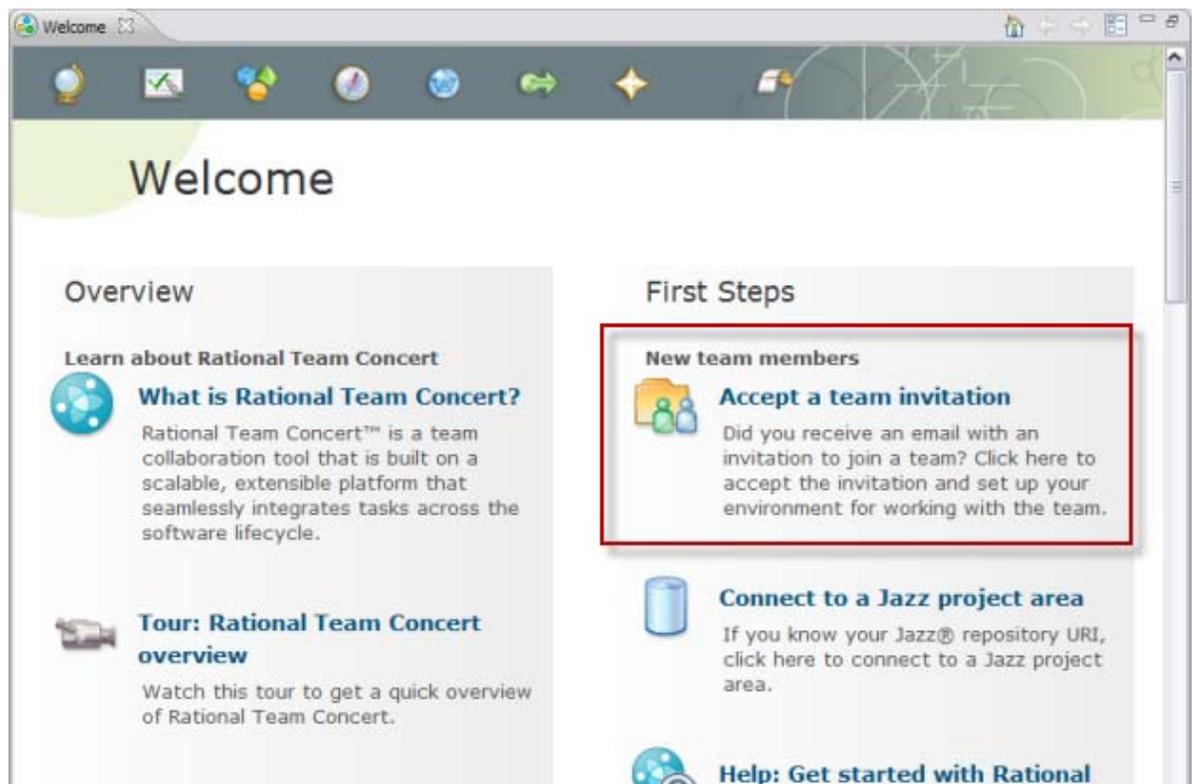
\_\_3. Start the Rational Team Concert client

\_\_a. Open Team Concert by double clicking the Team Concert shortcut  on the Windows® Desktop.

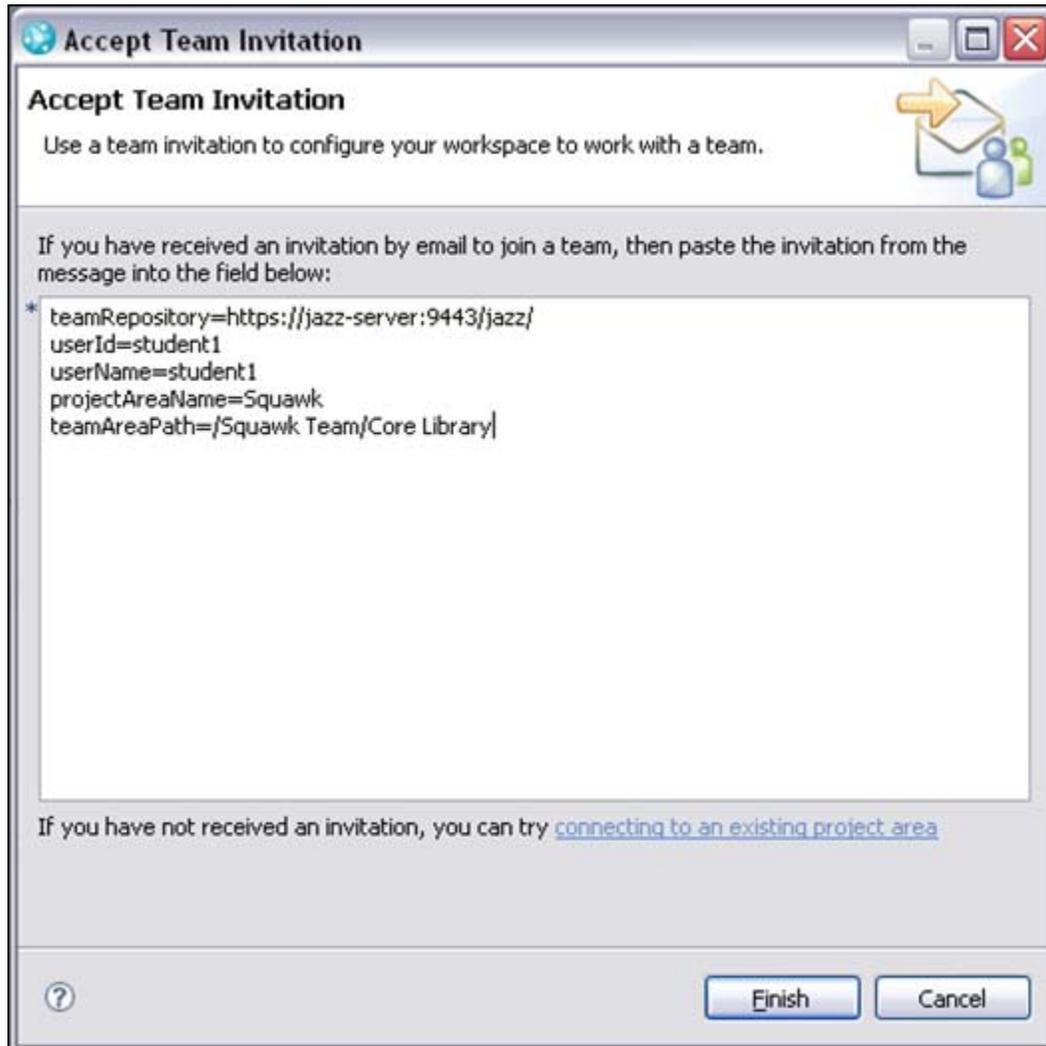
\_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` (replacing <N> with your id number). Optionally, you can check **Use this as the default and do not ask again** which will bypass this prompt if you need to restart Team Concert later in the workshop. Click **OK**.



- \_\_c. On the **Welcome** page, click **Accept an invitation** to join a Jazz™ team area.



- \_\_\_d. In the **Accept Team Invitation** window, paste in the invitation copied from the body of the email in step 2 above.

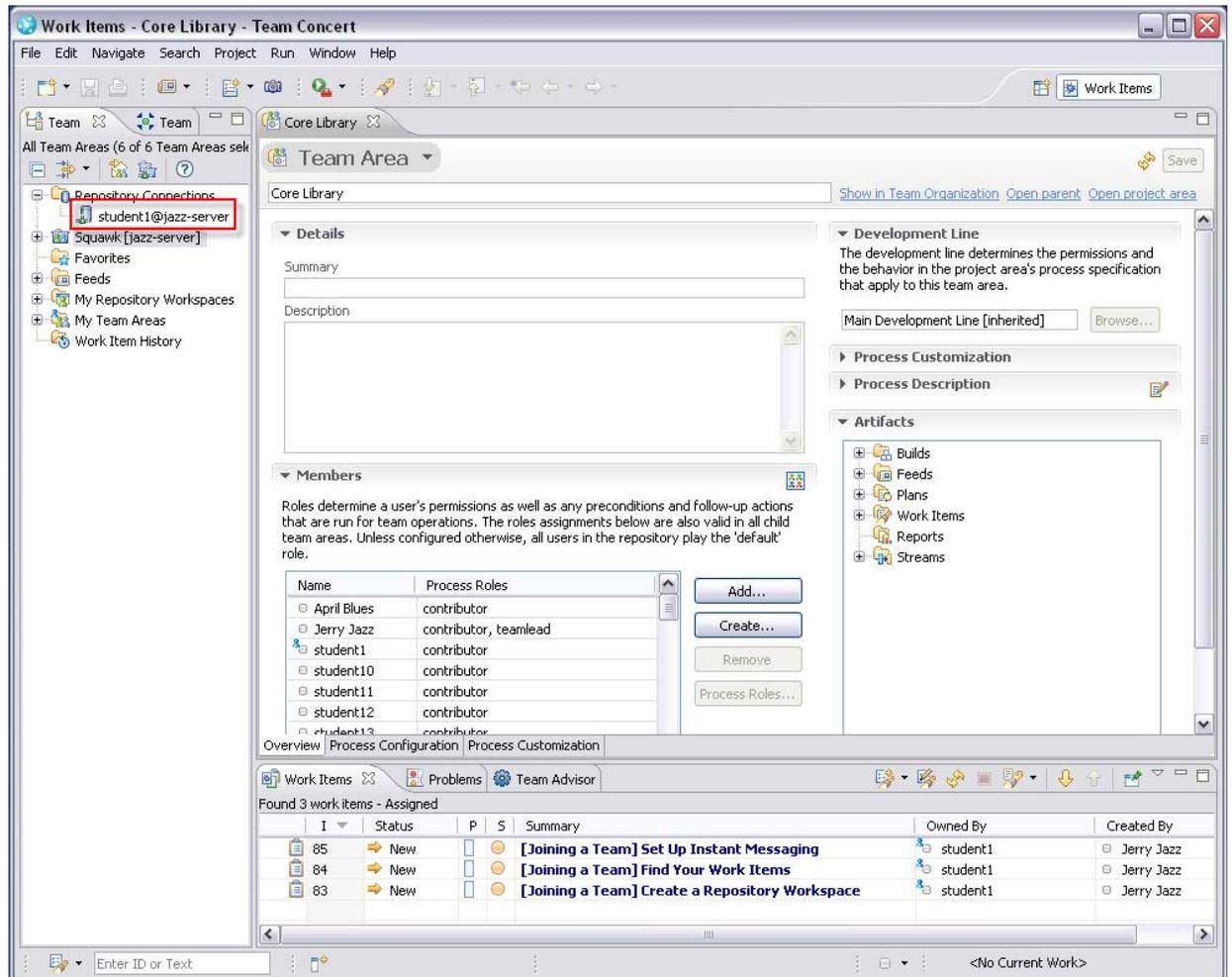


- \_\_\_e. Ensure that the values for **userId** and **username** match your assigned student id. Click **Finish**

- \_\_f. When prompted for a login and password:
- \_\_i. Type `student<N>` (replace `<N>` with your id number) for both **User ID** and **Password**,
  - \_\_ii. Select **Save password** and **Automatic login**, and
  - \_\_iii. Click **OK**.



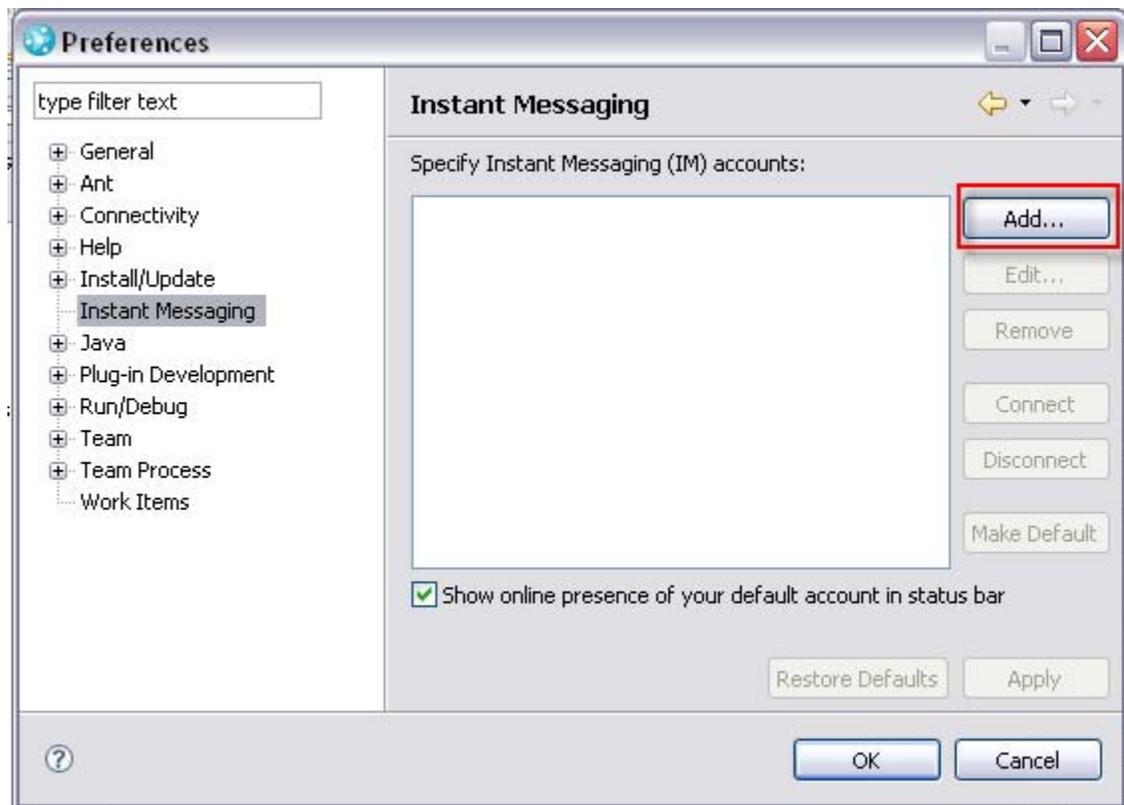
- g. You will be connected to your project and see a screen similar to the following. Ensure that you are connected to the Squawk Project area as *student<N>*.



You have now joined the project. You have access to the team area and artifacts for the teams to which you have been assigned. Note there are tasks in the **Work Items** view. These are automatically generated by your development process to help you get started.

## 1.2 Configure Instant Messaging

- \_\_1. Set up your instant messaging connection from within the Team Concert client
  - \_\_a. Click **Window** → **Preferences**
  - \_\_b. Click **Instant Messaging**
  - \_\_c. Click **Add**



- \_\_d. Select **Jabber XMPP Server** for **Provider**
- \_\_e. Type `jazz-server` for **Server**.
- \_\_f. Type your student login (in the form `student<N>`) for **User ID** and **Password**.

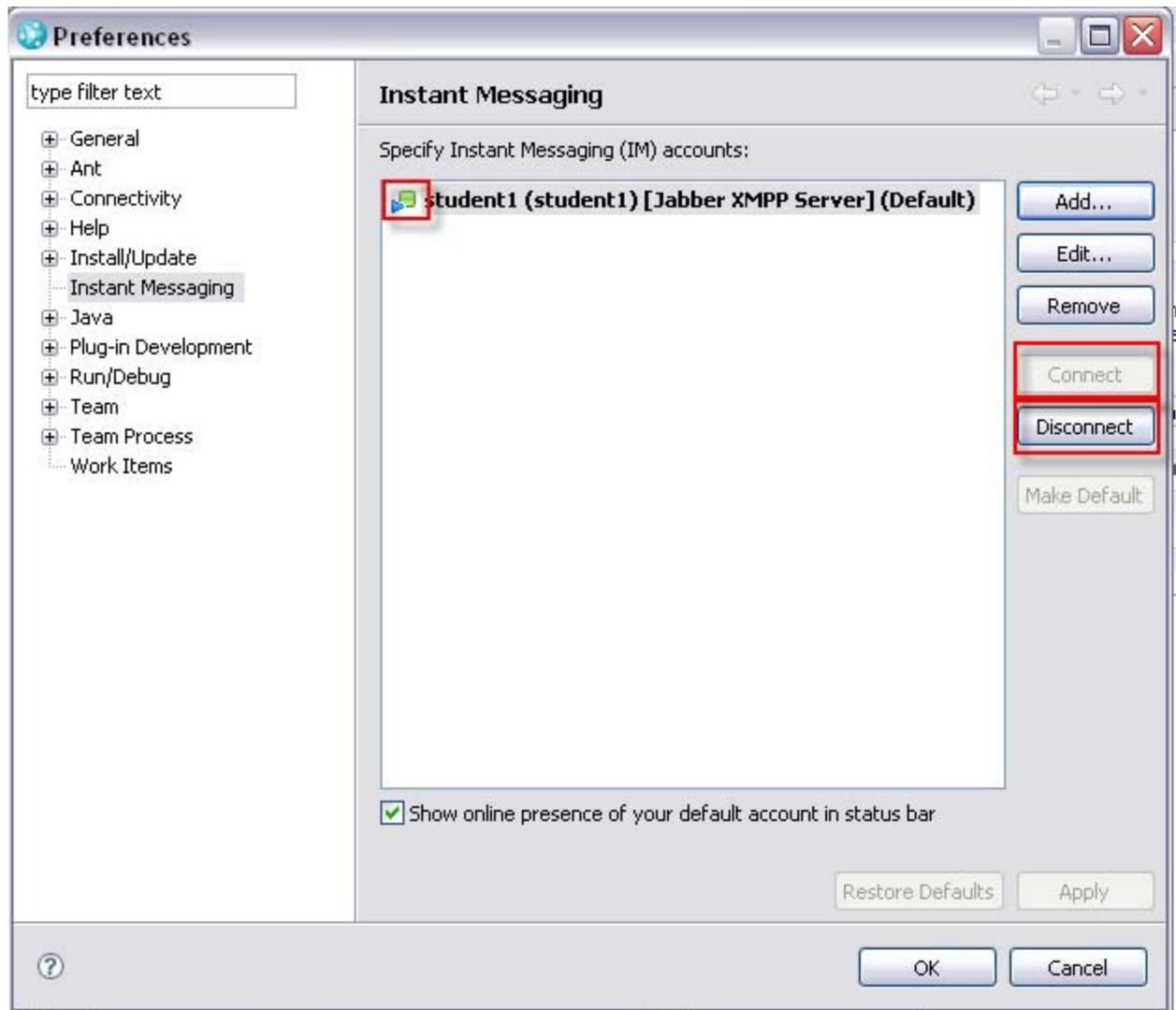
The screenshot shows a dialog box titled "Add IM Account". It has three main sections: "Service Provider", "Jabber Server", and "Authentication".

- Service Provider:** A dropdown menu with "Jabber XMPP Server" selected.
- Jabber Server:** A text field containing "jazz-server".
- Authentication:** Two text fields. The first is labeled "User ID:" and contains "student1". The second is labeled "Password:" and contains a series of dots. Below these is an optional field labeled "Resource:" which is empty.

At the bottom of the dialog, there is a question mark icon on the left, and "OK" and "Cancel" buttons on the right. A link "Click here for guidance to set up Instant Messaging." is located above the buttons.

- \_\_g. Click **OK**

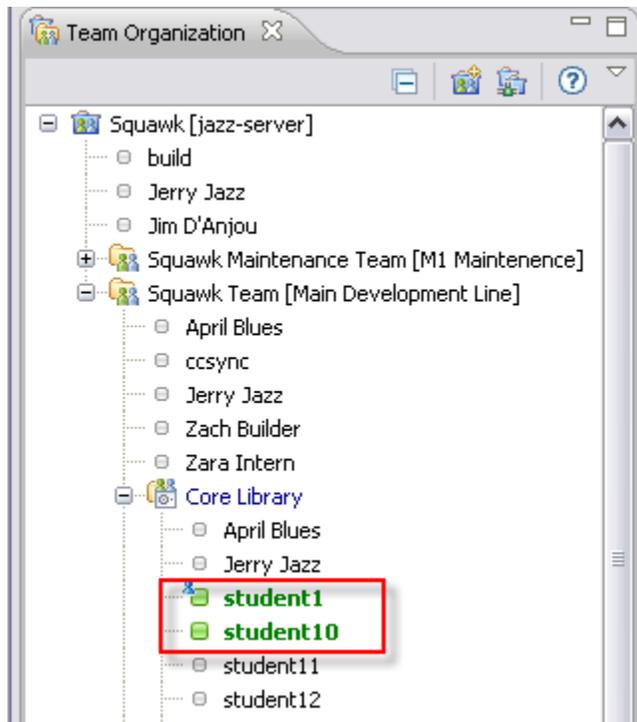
- \_\_h. In the **Preferences** window, press **Apply**. Note: if the newly created account is not marked “(Default)” select it and press **Make Default**, then click **Apply**. Click **Connect** and ensure that the status icon changes to green (you can also see that the connection is established because the **Connect** button will be disabled).



- \_\_i. Click **OK**.

## 1.3 Meet the team

- \_\_1. Open the **Team Organization** View
  - \_\_a. Click **Window → Show View → Team Organization**
  - \_\_b. There will be more details on this view later, but for the moment, expand **Squawk → Squawk Team → Core Library**. Here you will see all the team members for the Core Library team. Notice above the green icon next to the names that are currently connected to the Jazz Repository



- \_\_c. Select a teammate in the **Team Organization** view whose name is highlighted in green, right click on their name then select **Chat**. This will start up an instant messaging session with your fellow team member.

### Chatting with team members



The Jazz collaboration tools provide support for synchronous collaboration using IBM® Lotus® Sametime® Connect 7.5.1, IBM Lotus Notes® 8 and Sametime Connect 8.0, IBM Lotus Notes 8.0.1. You can start peer chats and n-way chats with several participants. Jazz also has built-in support for chat using the XMPP protocol. It supports peer chats, ad-hoc multi chats with several participants as well as persistent team chat in chat rooms. This chat capability is not limited to the Team Organization view but is available wherever team members are named.

- \_\_\_d. Once you have finished chatting, please close the active chat window for now. You can also close the **Team Organization** view.

**Conclusion**

This concludes the Lab for module 1. You are now able to easily connect to a Jazz repository and begin collaborating with your teammates.

## Lab 2 Planning Your Work



### Lab Scenario

You want to track all the work on your projects. All your work (plan items, stories, tasks and defects) is based around the concept of Work Items. You hear more about how Work Items are fundamental to Rational Team Concert and how you use these work items to track the work you do in each item. You prioritize your work so that you can do the right things at the right time in the plan.

The Squawker application needs some more squawkers. In Team Concert all the work to create these new squawkers will be planned using work items. You will create 1 new work item, set the priority, estimate how long the work will take and assign the work to the current iteration 1.0 M3. You will also get a chance to review the plan once everyone else has created their work items.



### Important!

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure which id number you should use, please check with the instructor.

## 2.1 Instructor Demo - Create new plans



### Instructor Demo

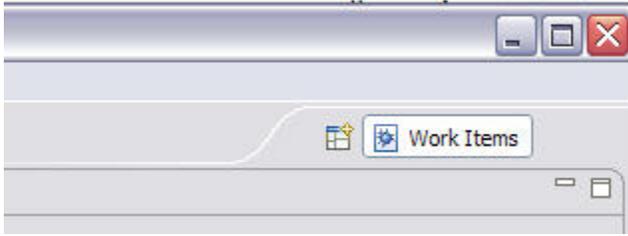
Section 2.1 is performed by the instructor as a demo. In this section, the instructor will create new stories that relate to the bigger story "Create a broader collection of Squawkers". These stories will be made part of the 1.0 M3 iteration backlog.

## 2.2 Working with Work Items

\_\_1. If you don't already have Rational Team Concert running, do the following:

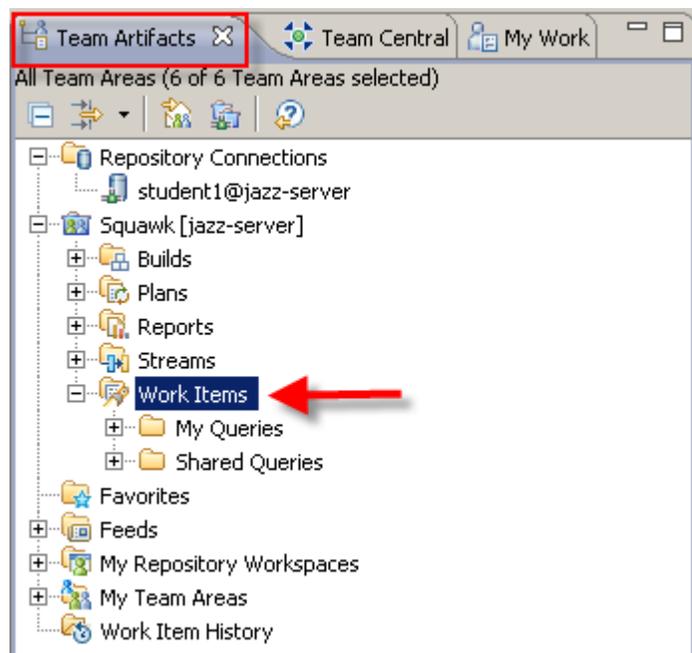
- \_\_a. Open Team Concert by double clicking the Team Concert shortcut  on the Windows Desktop.
- \_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` and click **OK**.

- \_\_2. Ensure that you are using the **Work Items** perspective (it should be open by default). It will be visible on the top right-hand corner of Team Concert



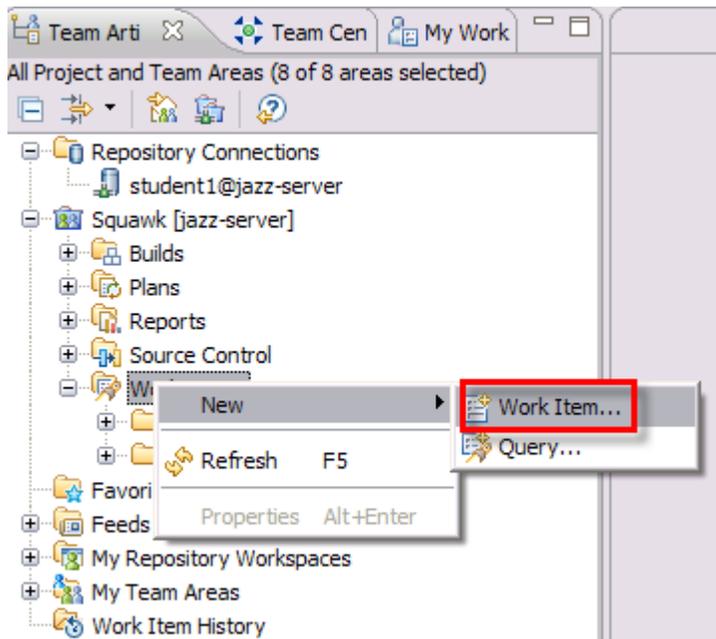
- \_\_3. Create a new Task work item to complete your Squawker

- \_\_a. Select the **Team Artifacts** view

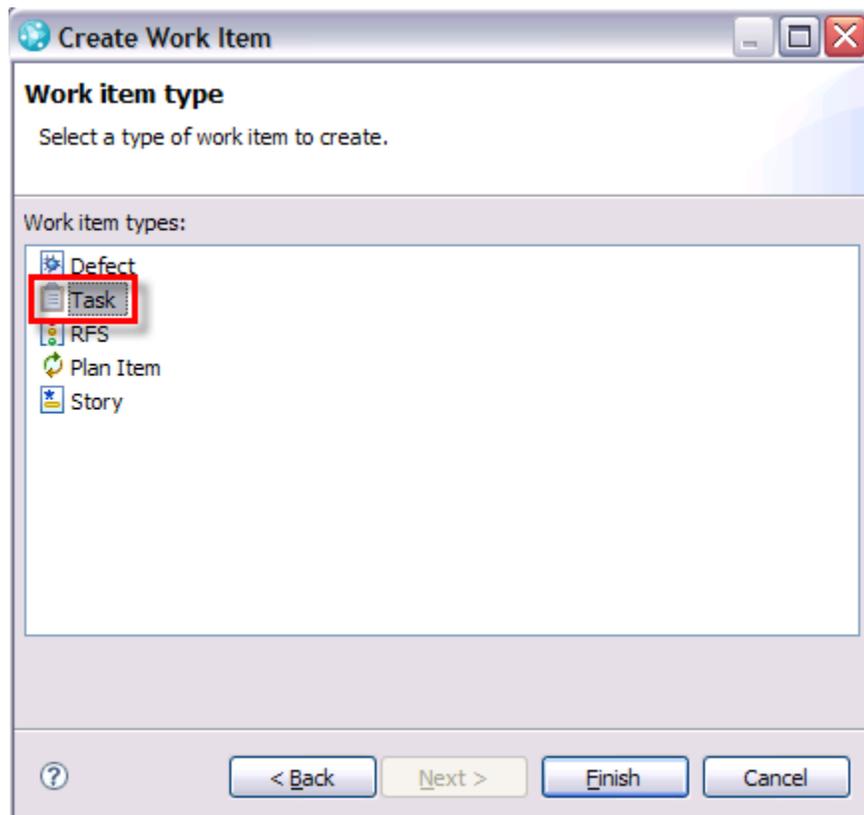


- \_\_b. Expand the **Squawk** project

\_\_c. Right-click the **Work items** folder then select **New→Work Item**.



\_\_d. Select Work item type **Task** and click **Finish**

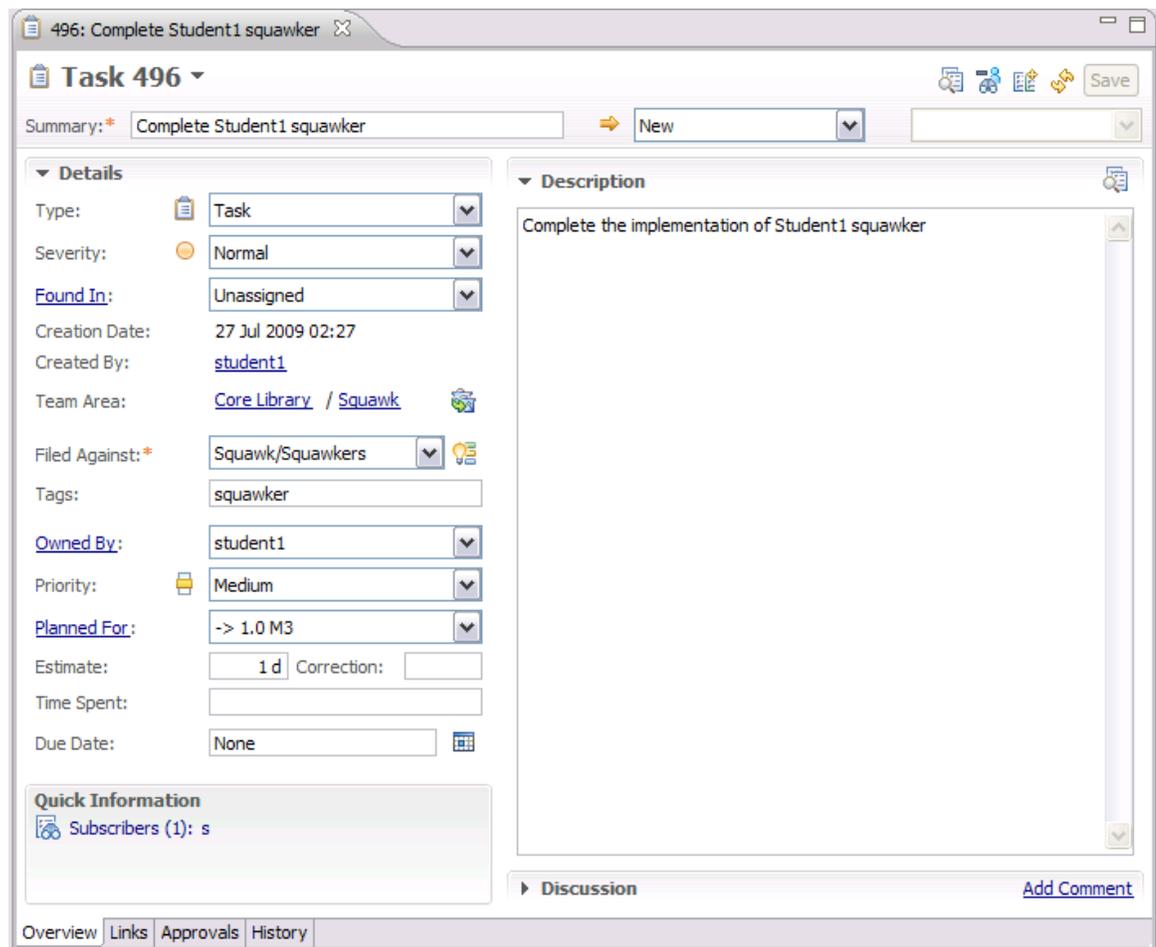


\_\_e. The Work Item editor contains the new Task:

\_\_f. For the details of the Task:

- \_\_i. Type Complete Student<N> squawker in the **Summary** (replace <N> with your student id).
- \_\_ii. Type Complete the implementation of the Student<N> squawker in the **Description** (replace <N> with your student id).
- \_\_iii. Set **Filed Against** to Squawk/Squawkers.
- \_\_iv. Set **Tags** to squawker
- \_\_v. Set **Owned By** to student<N>
- \_\_vi. Set **Priority** to Medium
- \_\_vii. Set **Planned For** to ->1.0 M3
- \_\_viii. In **Estimate**, type 1 day
- \_\_ix. Click **Save** in the right corner of **Work Item** editor view.

\_\_g. Your task should look similar to this:



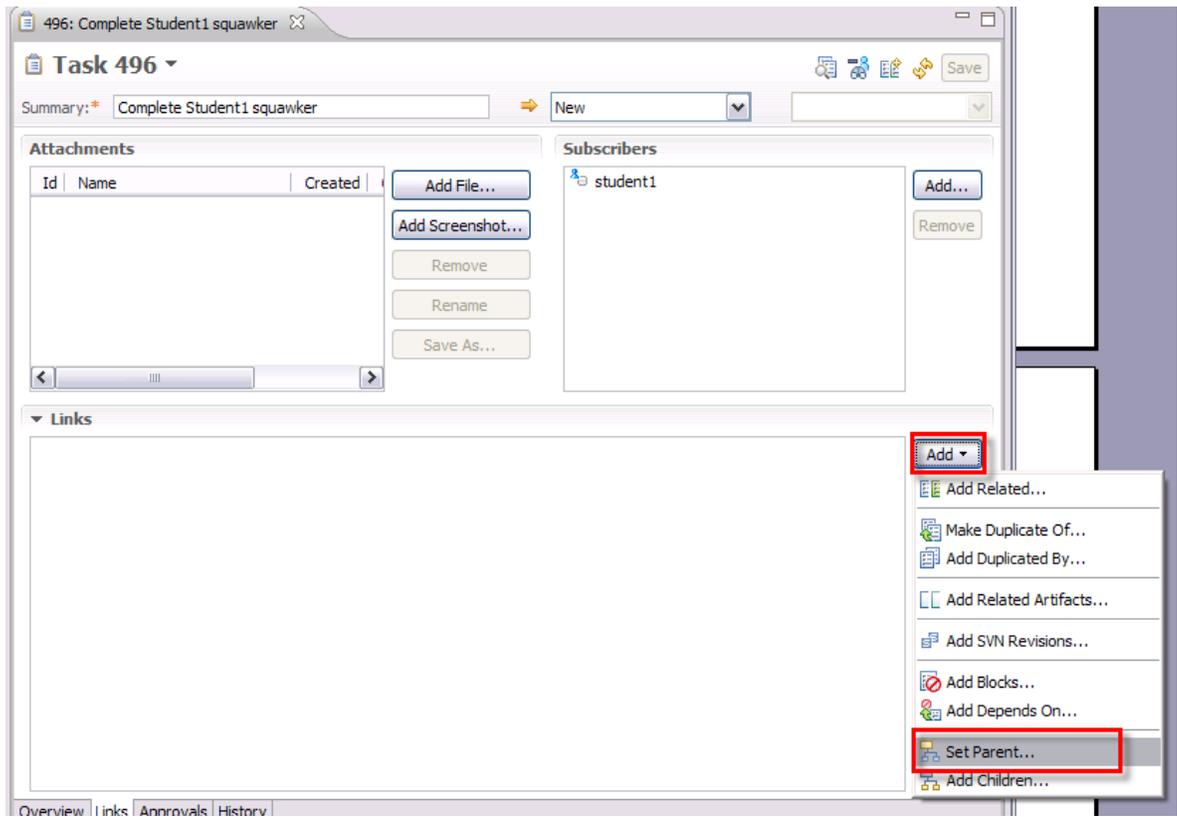
- \_\_4. Now associate your enhancement with the stories set by the Team Lead.
- \_\_a. Click on the **Links** tab of the Task

The screenshot displays the IBM Software interface for a task titled "Task 496". The task summary is "Complete Student1 squawker". The interface is divided into several sections:

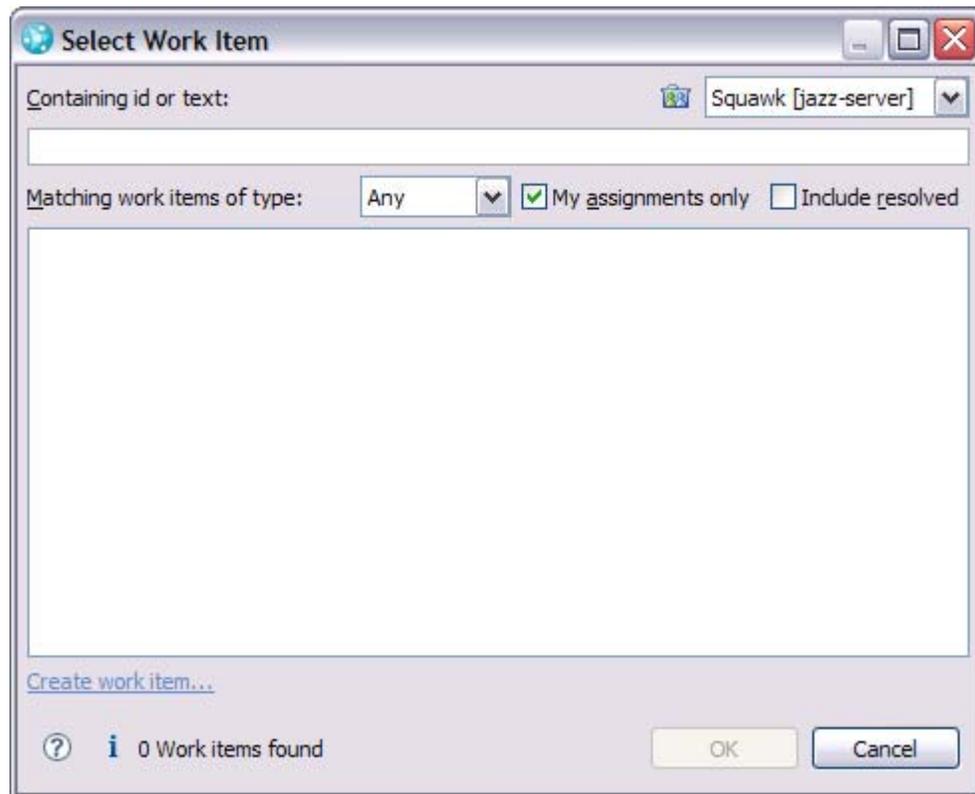
- Summary:** Complete Student1 squawker
- Details:**
  - Type: Task
  - Severity: Normal
  - Found In: Unassigned
  - Creation Date: 27 Jul 2009 02:27
  - Created By: student1
  - Team Area: Core Library / Squawk
  - Filed Against: Squawk/Squawkers
  - Tags: squawker
  - Owned By: student1
  - Priority: Medium
  - Planned For: -> 1.0 M3
  - Estimate: 1 d
  - Time Spent: (empty)
  - Due Date: None
- Description:** Complete the implementation of Student1 squawker
- Quick Information:** Subscribers (1): s
- Discussion:** Add Comment

At the bottom of the interface, there are four tabs: Overview, **Links** (highlighted in red), Approvals, and History.

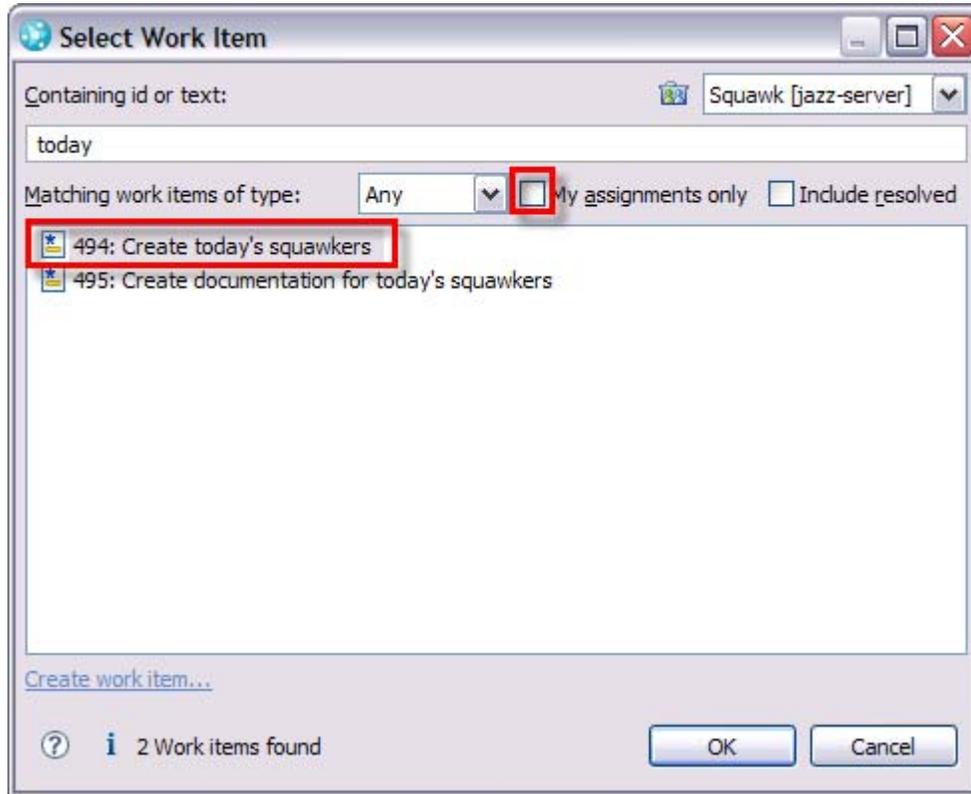
\_\_b. Click the **Add** button in the **Links** area of the Story work item and select **Set parent...**



- \_\_c. This will open up the **Select Work Item** dialog that will be used to create a link to a parent task.



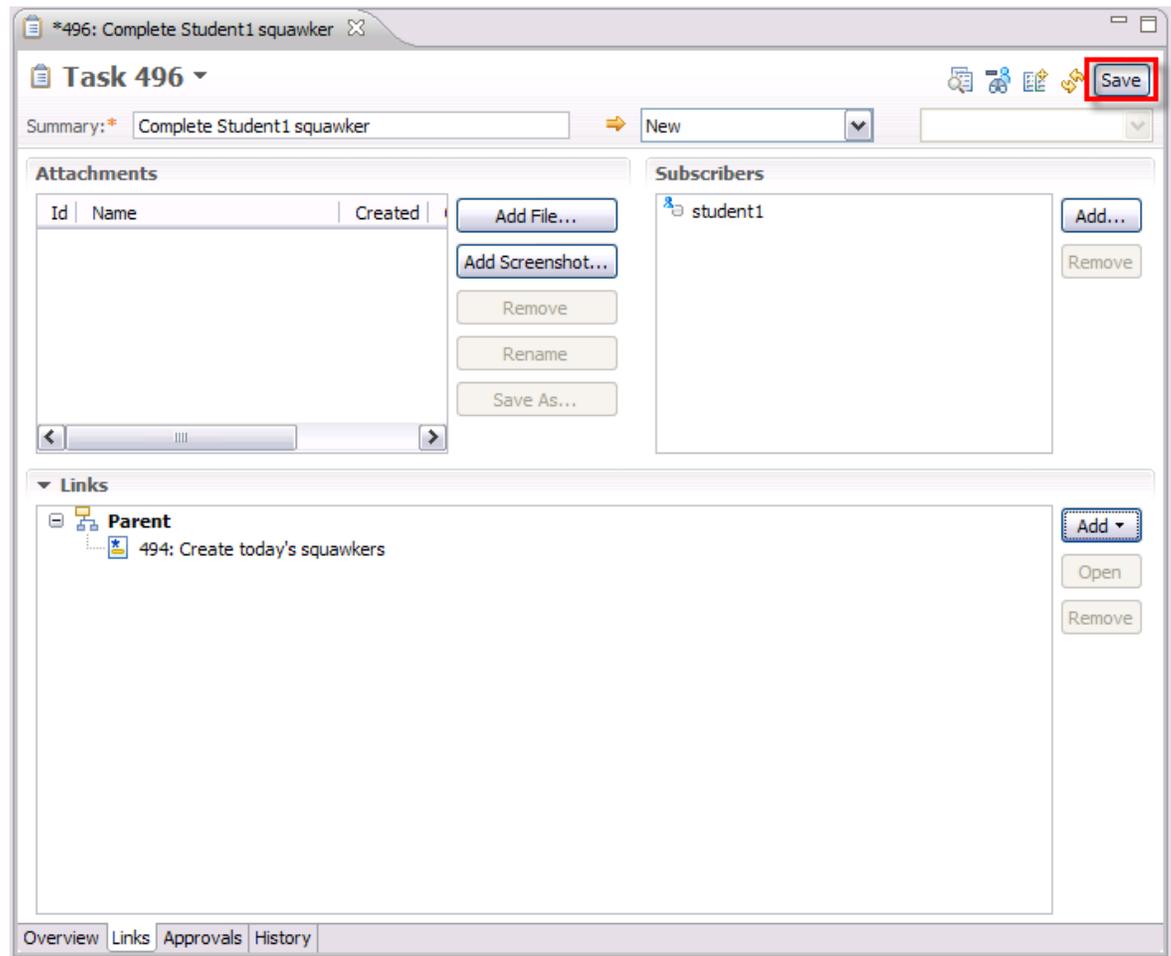
- d. In the **Select Work Item** dialog, deselect **My assignments** only. This allows you to search across all work items, not just your own. Enter `today` in the **Containing id or text:** field to find the parent work item the Team Lead created.



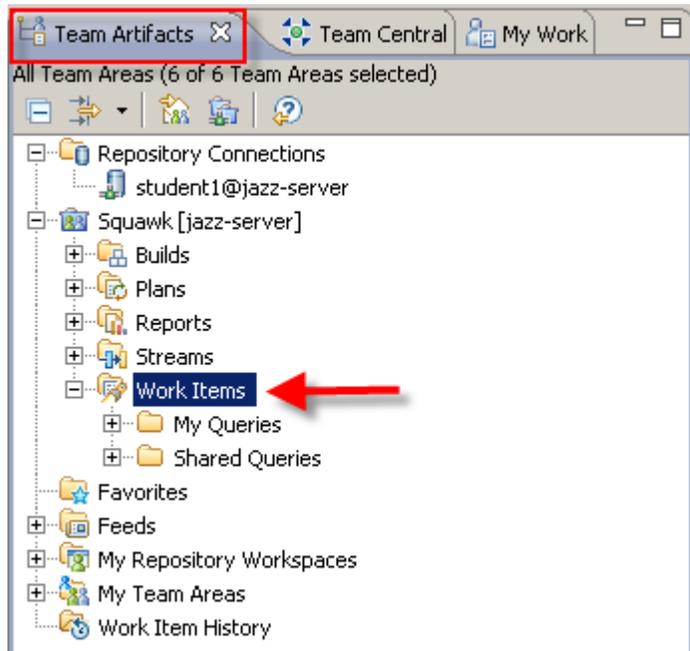
- e. The **Create today's squawker** story should be highlighted. Click **OK** to select this as the parent of your new Task.

The screenshot displays the configuration interface for 'Task 496'. At the top, the window title is '\*496: Complete Student1 squawker'. Below the title bar, the task name 'Task 496' is shown with a dropdown arrow. To the right of the task name are several icons and a 'Save' button. The 'Summary:' field contains 'Complete Student1 squawker' and a dropdown menu is set to 'New'. Below the summary field are two main sections: 'Attachments' and 'Subscribers'. The 'Attachments' section has a table with columns 'Id', 'Name', and 'Created', and buttons for 'Add File...', 'Add Screenshot...', 'Remove', 'Rename', and 'Save As...'. The 'Subscribers' section lists 'student1' with 'Add...' and 'Remove' buttons. Below these sections is the 'Links' section, which shows a 'Parent' link with '494: Create today's squawkers' highlighted. To the right of the links are 'Add', 'Open', and 'Remove' buttons. At the bottom of the interface is a navigation bar with tabs for 'Overview', 'Links', 'Approvals', and 'History'.

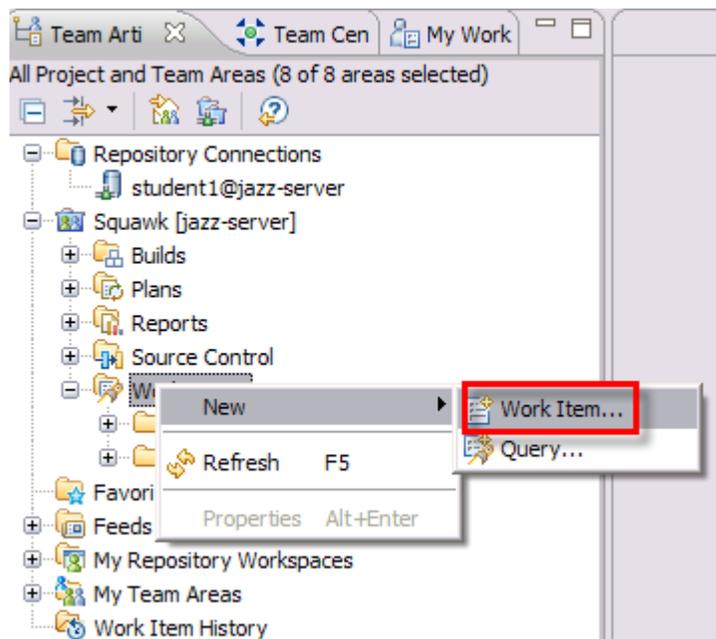
- \_\_f. Click **Save** to save your Task. **Make note of the work item number as you will use it to automatically create a hyperlink to it from the text of the next work item**



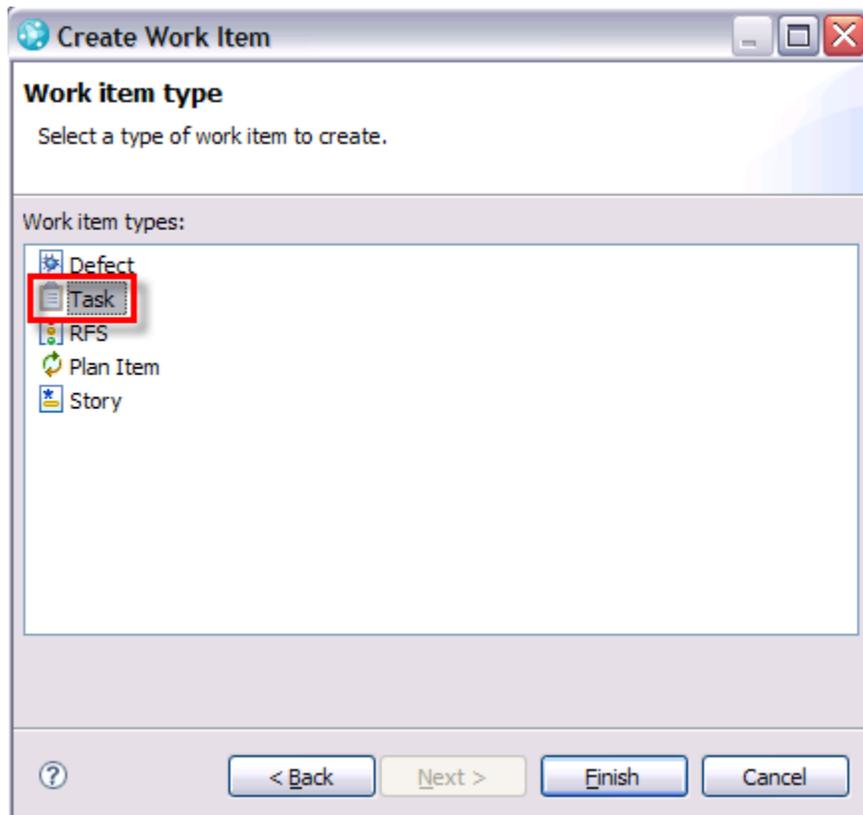
- \_\_5. Create a new Documentation Task work item to complete your Squawker documentation
- \_\_a. Select the **Team Artifacts** view



- \_\_b. Expand the **Squawk** project
- \_\_c. Right-click the **Work items** folder then select **New→Work Item**.



\_\_d. Select Work item type **Task** and click **Finish**



\_\_e. The new Task will appear as follows:

The screenshot displays a web-based task management interface. At the top, there are browser tabs for '496: Complete Student1 squawker' and '<02:42:10>: <untitled>'. The main header shows 'Task <02:42:10>' with a 'Save' button and a status dropdown set to 'Uninitialized'. Below the header is a 'Summary:' field. The interface is divided into two main sections: 'Details' and 'Description'. The 'Details' section contains various fields: 'Type' (Task), 'Severity' (Normal), 'Found In' (Unassigned), 'Creation Date' (None), 'Created By' (student1), 'Team Area' (Squawk Team / Squawk), 'Filed Against' (Unassigned), 'Tags' (empty), 'Owned By' (Unassigned), 'Priority' (Unassigned), 'Planned For' (Unassigned), 'Estimate' and 'Correction' (empty), 'Time Spent' (empty), and 'Due Date' (None). A 'Quick Information' box below the details shows '<No information>'. The 'Description' section is a large empty text area. At the bottom, there is a 'Discussion' section with an 'Add Comment' link. A navigation bar at the very bottom includes 'Overview', 'Links', 'Approvals', and 'History'.

496: Complete Student1 squawker <02:42:10>: <untitled>

Task <02:42:10> Uninitialized Save

Summary: \* Uninitialized

**Details**

Type: Task

Severity: Normal

Found In: Unassigned

Creation Date: None

Created By: student1

Team Area: Squawk Team / Squawk

Filed Against: \* Unassigned

Tags:

Owned By: Unassigned

Priority: Unassigned

Planned For: Unassigned

Estimate: Correction:

Time Spent:

Due Date: None

**Quick Information**

<No information>

**Description**

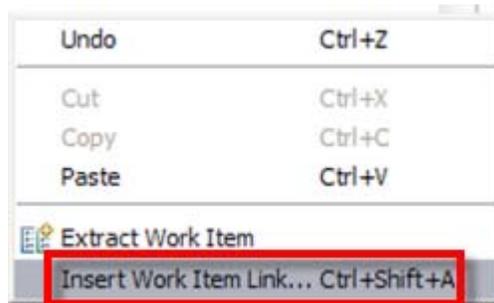
Discussion Add Comment

Overview Links Approvals History

\_\_f. For the details of the Task:

- \_\_i. Type Complete Student<N> squawker documentation in the **Summary**. (replace <N> with your student id).
- \_\_ii. Type Complete the documentation for the Student<N> squawker (task#<496>) in the **Description** (replace <N> with your student id and <496> with the number of your Task for your squawker).

The work item link can also be created by right-clicking within the **Description** text box and selecting Insert Work Item.



- \_\_iii. Set **Filed Against** to Squawk/Squawkers Documentation
- \_\_iv. Set **Tags** to squawker, documentation
- \_\_v. Set **Owned By** to student<N>
- \_\_vi. Set **Priority** to Medium
- \_\_vii. Set **Planned For** to ->1.0 M3
- \_\_viii. In **Estimate**, type 3 hours
- \_\_ix. Click **Save** in the right corner of **Work Item** editor view.

The screenshot displays the 'Task 497' editor in IBM Software. The 'Details' section on the left contains the following information:

- Type: Task
- Severity: Normal
- Found In: Unassigned
- Creation Date: 27 Jul 2009 02:47
- Created By: student1
- Team Area: Documentation / Squawk
- Filed Against: Squawk/Squawkers Docu
- Tags: squawker, documentation
- Owned By: student1
- Priority: Medium
- Planned For: -> 1.0 M3
- Estimate: 3 h
- Time Spent:
- Due Date: None

The 'Quick Information' section shows:

- Subscribers (1): s
- Mentions (1)

The 'Description' section contains the text: "Complete the documentation for the Student1 squawker ([task#496](#))".

The 'Discussion' section is empty, with an "Add Comment" link at the bottom right.

The bottom navigation bar includes: Overview | Links | Approvals | History

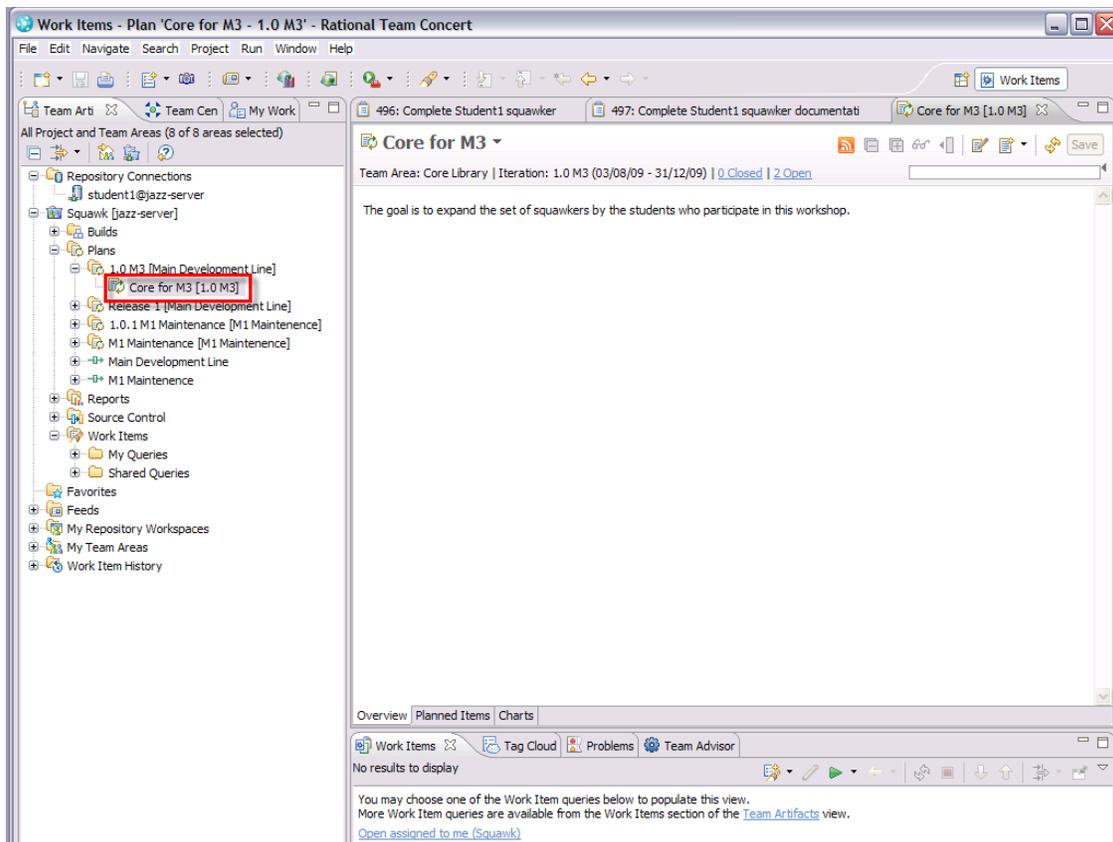
- \_\_g. Note that the mention of the earlier squawker task is highlighted as a hyperlink automatically – and the **Quick Information** section contains a **Mentions** entry for the same task.

## 2.3 Take a tour of the existing Squawk plans

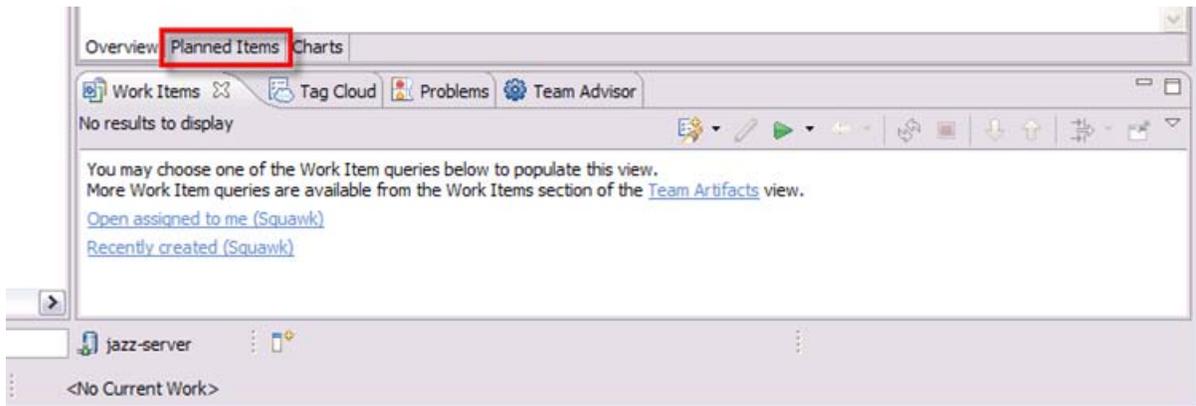


You will look at the plan for the Core Library team for the M3 iteration. In contrast to the **Backlog** plan you saw in the demonstration, this plan will display execution work items as well as plan work items.

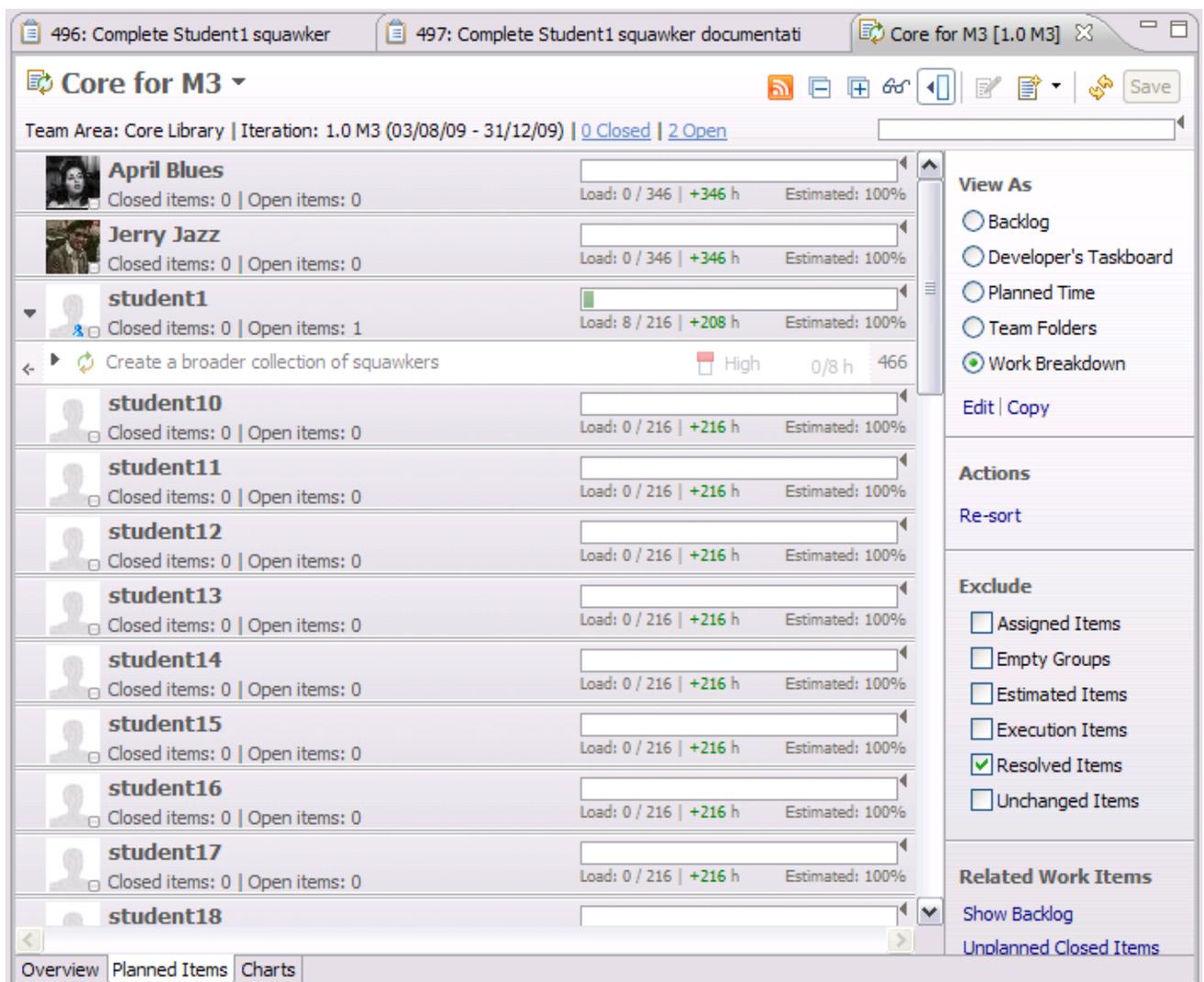
1. In the **Team Artifacts** view, select **Squawk**→**Plans**→**1.0 M3 [Main Development Line]** →**Core for M3 [1.0 M3]**. Double-click to open the **Core for M3** plan.



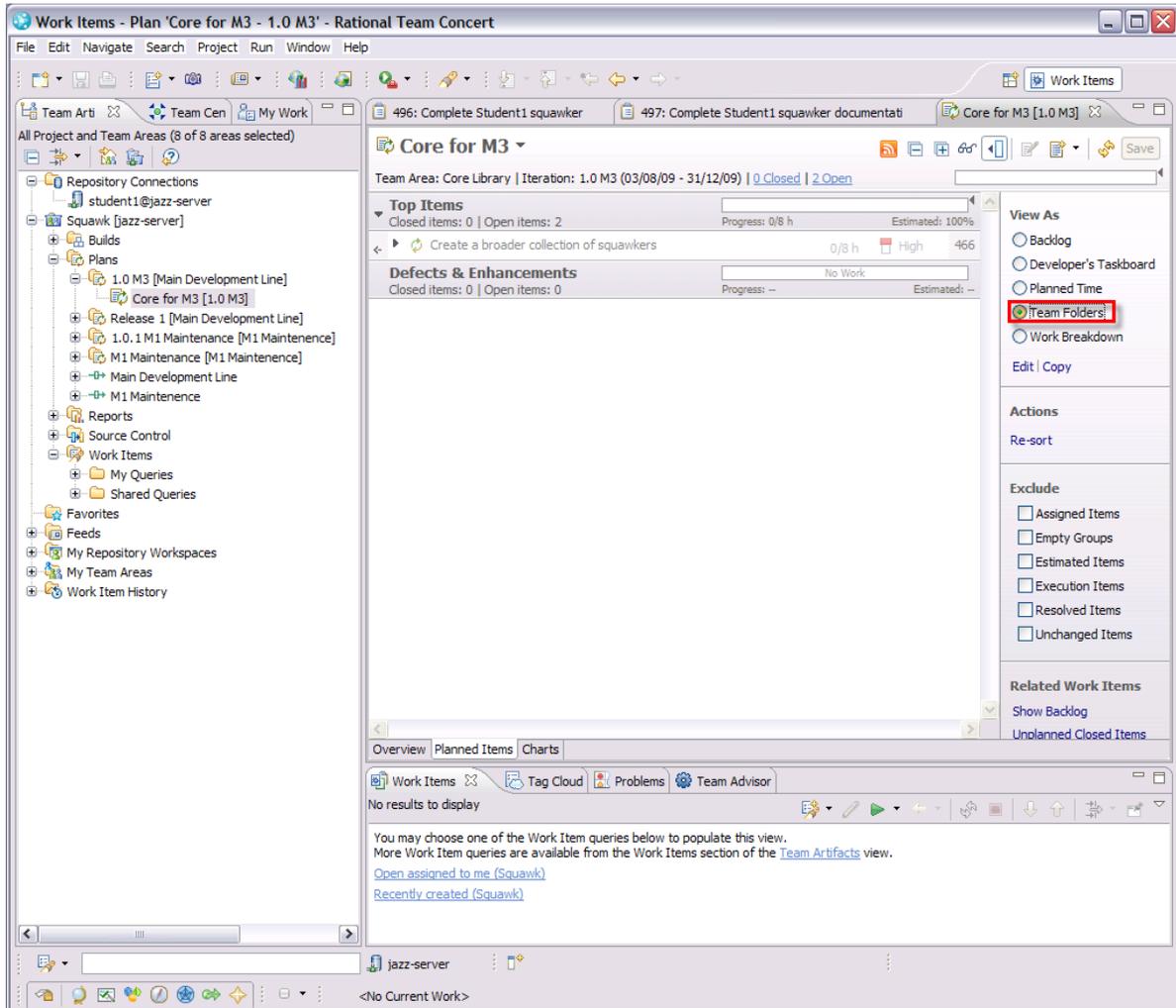
\_\_2. Select the **Planned Items** tab on the **Core for M3 [1.0 M3]** plan



\_\_3. The **Planned Items** appear as follows:



4. Click **Team Folders** to display the plan organized in terms of folders:



5. In the **Top Items** folder, expand the **Create a broader collection of squawkers** and **Create today's squawkers** work items until you see your Task (**Complete Student1 squawker** Task in the screenshot). You should also see other Tasks from the other students (if you don't see other Tasks here – you are the first person to finish creating your Tasks).

The screenshot displays the IBM Software interface for a project named "Core for M3". The main area shows a hierarchy of tasks under "Top Items". The "Complete Student1 squawker" task is highlighted with a red box. The interface also shows "Defects & Enhancements" and a right-hand sidebar with various view and action options.

Task Name	Progress	Estimated	Priority	Count
Create a broader collection of squawkers	0/8 h	100%	High	466
Create today's squawkers	0/8 h		High	494
Complete Student1 squawker	--		Medium	496

Defects & Enhancements: No Work, Progress: --, Estimated: --

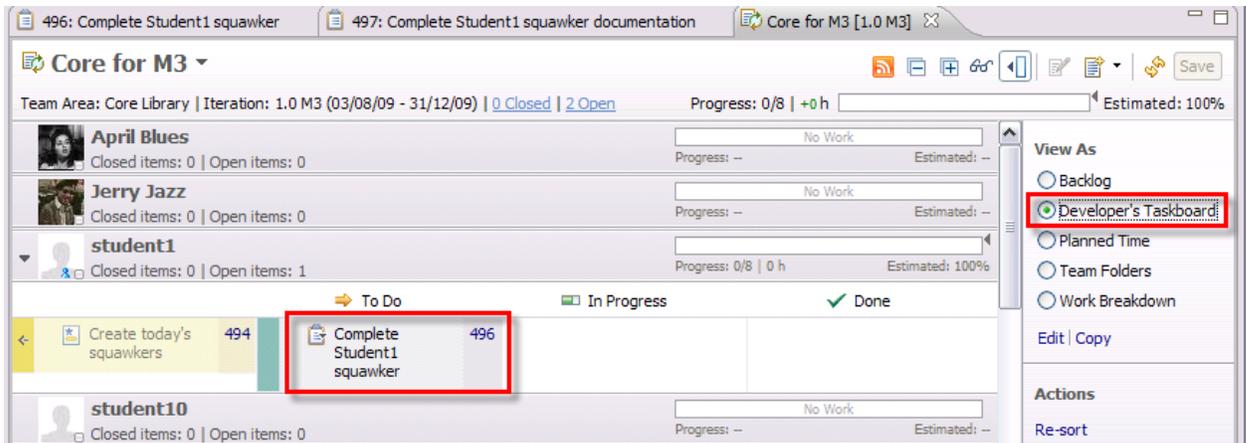
View As: Backlog, Developer's Taskboard, Planned Time, Team Folders, Work Breakdown

Actions: Re-sort

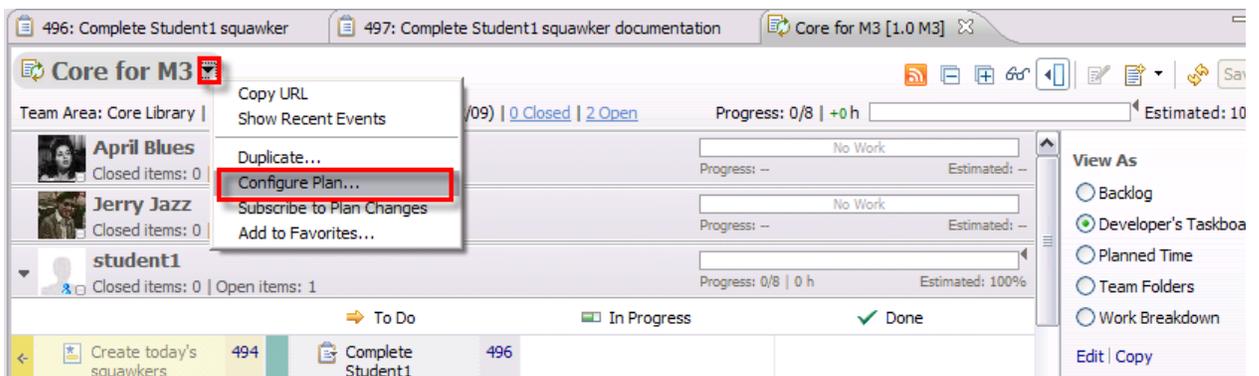
Exclude: Assigned Items, Empty Groups, Estimated Items, Execution Items, Resolved Items, Unchanged Items

Related Work Items: Show Backlog, Unplanned Closed Items

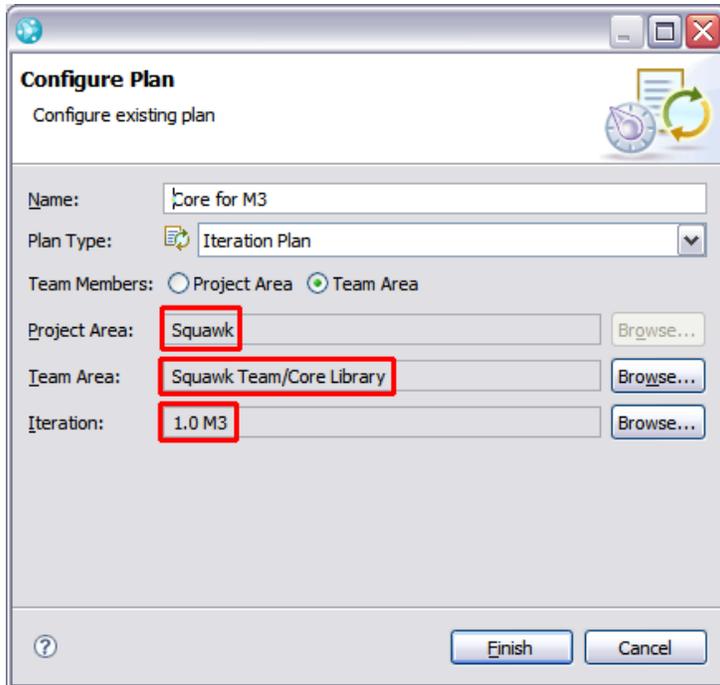
- \_\_\_6. Inspect another view of the plan by clicking **Developer's Taskboard** in the **View As** sidebar. Locate your student id and expand it to see your assignments. This view can be useful in team meetings to examine work status and make reassignments if individual workload needs to be adjusted.



- \_\_\_7. On the **Core for M3** title on the plan, select the menu icon and select **Configure Plan...** to open the selection criteria for the plan



- \_\_8. Note the selection criteria for the plan: the **Project Area**, the **Team Area** and the **Iteration**. This controls what work items appear in the plan, and why your work items appear in this particular plan. Selecting **Team Area** in **Team Members**: mean that this plan only shows items relevant to a specific team.



- \_\_9. Click **Cancel** to dismiss this dialog
- \_\_10. Close the **Core for M3 [1.0 M3]** Plan and the Tasks.

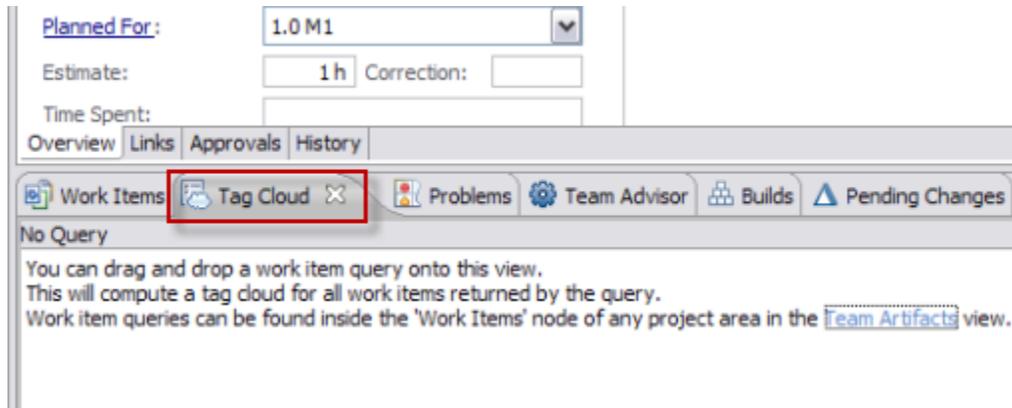
## 2.4 Using Tags in Team Concert

Tags can be used in Team Concert to make finding work items and other objects easier. Use tags to locate work items of special interest via a query.

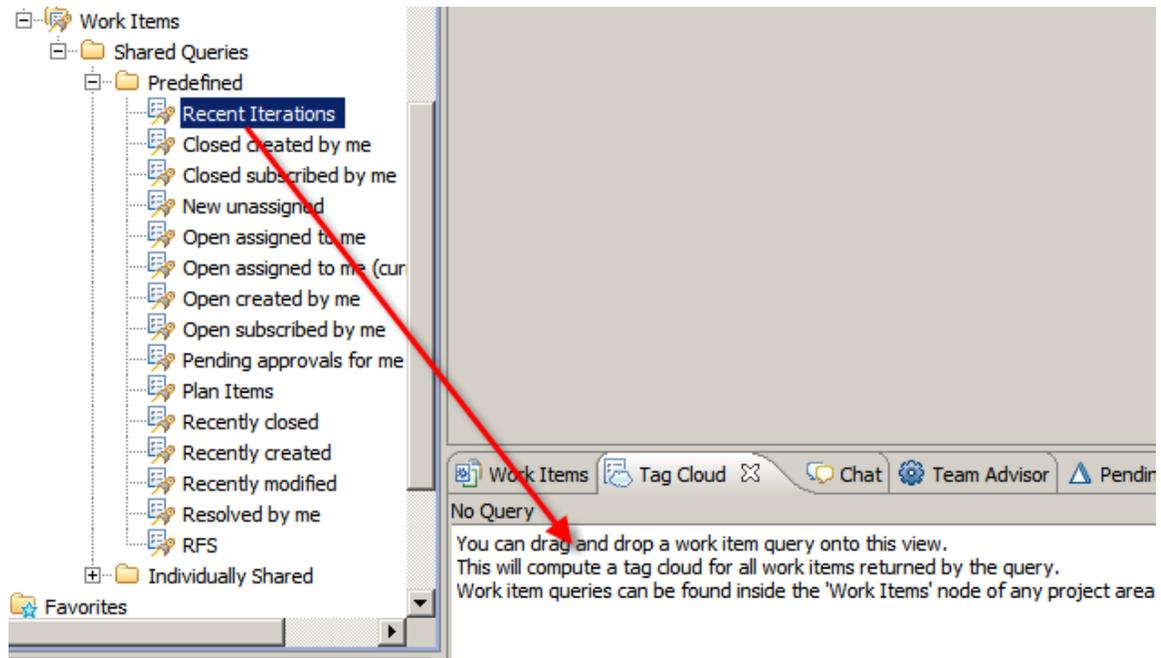
A tag cloud is a collection of words (tags) of various sizes, weights and colors presented in a group (cloud). The font size, weight and color of each tag combine to portray the priority of that tag in relation to the others. A tag that is larger and darker in color is more prominent than (for example) a smaller, lighter colored tag. This order of visual priority is usually based on the tags popularity however it can also be based on other metrics, like the volume of information related to that tag. Tag clouds can be used as a navigation mechanism and as a system for organizing content.

\_\_1. Using the Tag Cloud

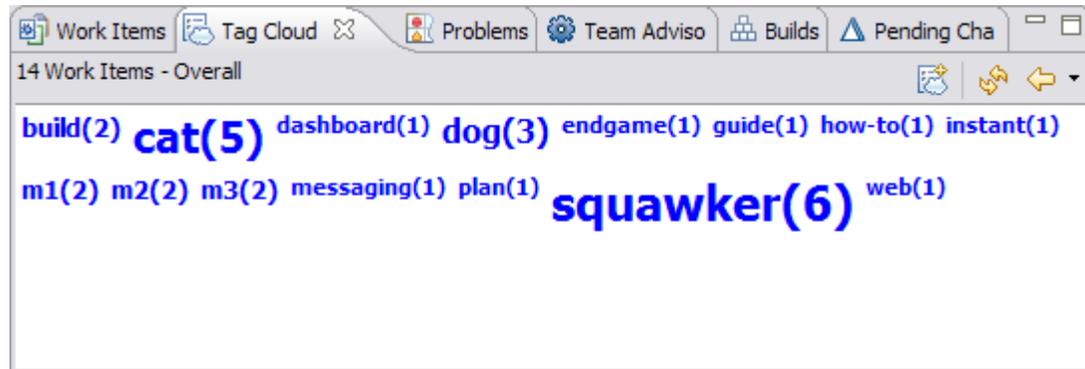
- \_\_a. Queries can be dragged to the **Tag Cloud** view, which will display results in the familiar and easy to use cloud display. To use this view, select the **Tag Cloud** tab at the bottom of your Team Concert window. If the **Tag Cloud** tab is not displayed, go to **Window → Show View → Tag Cloud**.



- \_\_b. To use the Tag Cloud view, drag and drop a query into it. In the **Team Advisor** view, navigate to **Squawk → Work Items → Shared Queries → Predefined**. Drag the Query named "Recent Iterations" to the Tag Cloud view.



- \_\_c. The Tag Cloud view will then show the tags of all work items matching that query. Depending on the tagging activity of team members, the Tag Cloud view could look something like this:



## 2.5 Instructor Demo - Reviews plans and distribute

### Instructor Demo

Section 2.5 is performed by the instructor as a demo. In this section, the instructor will show how a team lead would review the plans for Implement Additional Squawkers and Document Additional Squawker, and send out the plan to everyone.

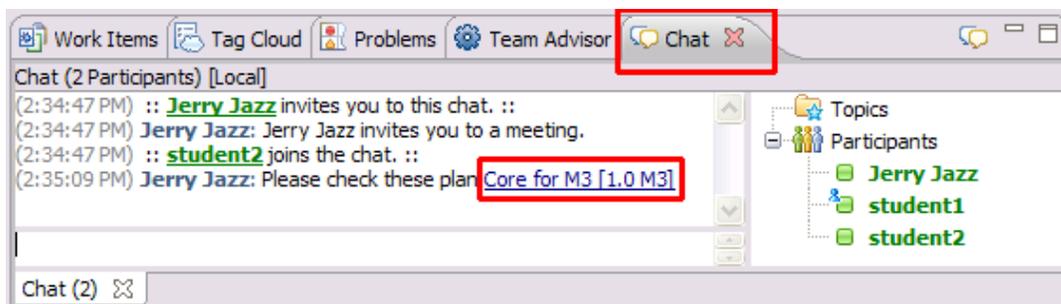


The team lead checks all the new Squawker tasks are sub-tasks on the plans. Any standalone tasks created by the students will be moved to the right parents.

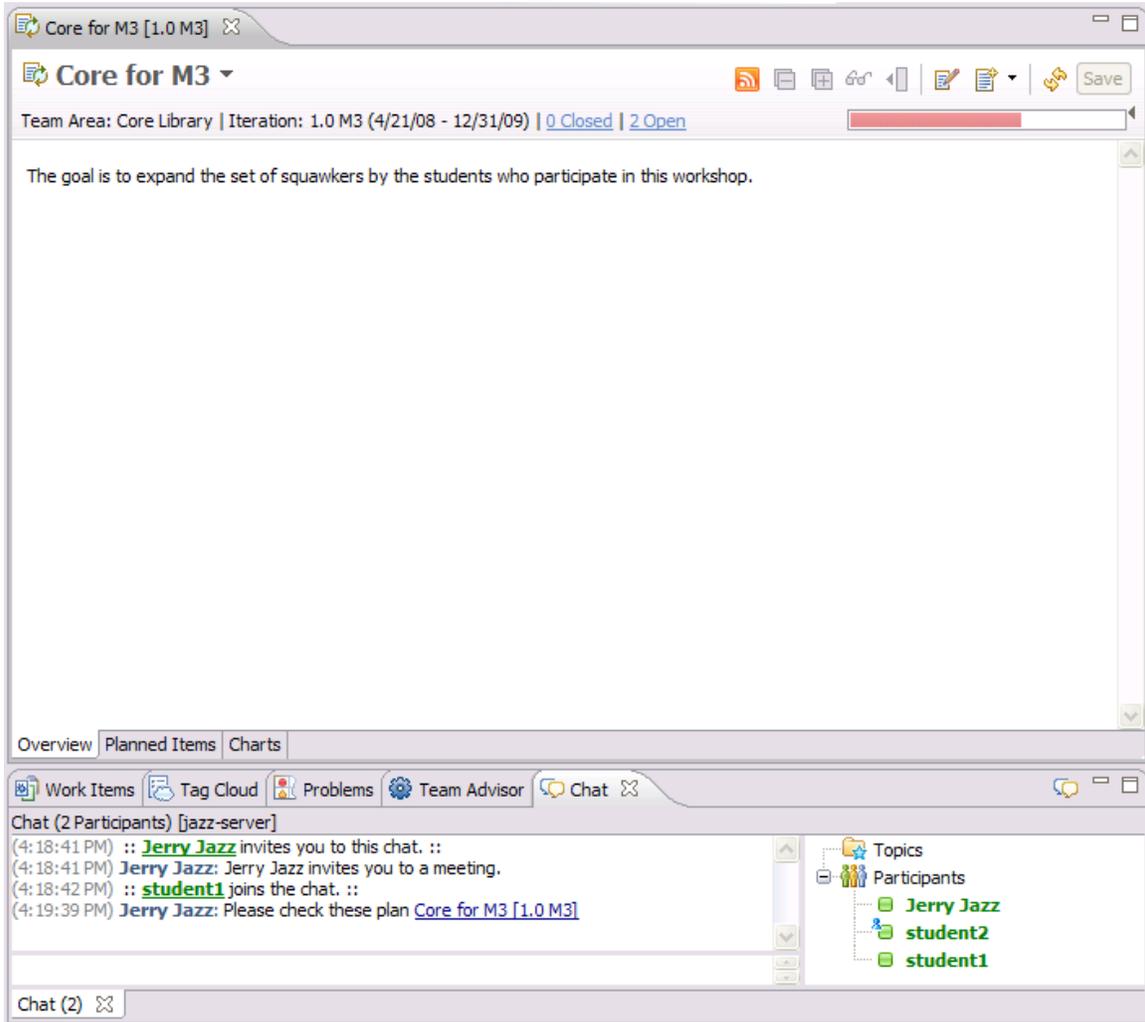
Once the team lead is happy with the plan, they send out the plan via chat. The students can click on the plan to see the details.

## 2.6 Open the reviewed plan

- \_\_1. Once the team lead has performed the review, they will ask you to look at the Chat view to see the plan for your team for this iteration (1.0 M3).



- \_\_2. Click the plan link in the Chat view.
- \_\_3. The plan will open in your student workspace.



### Conclusion



This concludes the Lab for module 2. You now know how plans can be created, how to create work items and understand how they fit in the context of iteration plans. Additionally, you explored how use of tags and the tag cloud view can make it easier to find work items.

## Lab 3 Keeping Track of All Our Work

In this module you will learn how to keep track of all your work and of the work that goes on in the rest of the project. To accomplish this task you will develop simple queries from the Eclipse Client and the Web UI. In addition you will use these and other queries to define the various views within Rational Team Concert. These views are dynamically refreshed and give you a real time status of your project and the workload within your project. By configuring the views properly, you have a powerful way of observing the health and the heartbeat of the project.

In this lab, you will cover the concepts of:

- writing and running work item queries in the Eclipse Client and use it in the Web UI
- examine plans in the web UI
- the capabilities of the Team Central view
- configuring the My Work view

Beside the query capabilities Rational Team Concert provides also powerful reporting capabilities out of the box. This will be covered in a later lab.



### Lab Scenario

You recently joined the development staff of the Squawk project. Your environment is properly setup 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, with the status of the project and with the work items assigned to you.



### Important!

Ensure that you carry out this lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.

### 3.1 Create a simple query with the Eclipse Client interface

Rational Team Concert provides an out of the box project setup that includes queries for standard tasks. But you are not limited to using predefined queries. In addition it is easy to create queries with filters based on the different field types of the work items.

\_\_1. Create a simple query with the Eclipse Client Interface.

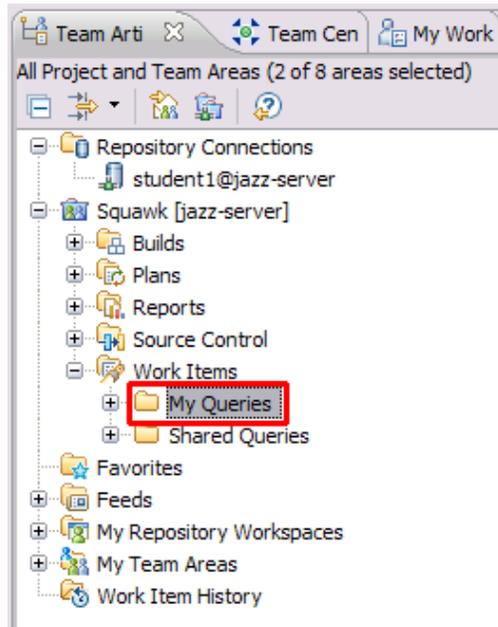
\_\_a. If not already started, open Team Concert by double-clicking the Team Concert shortcut



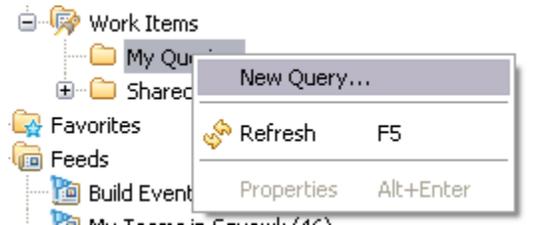
on the Windows Desktop.

\_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` and click **OK**.

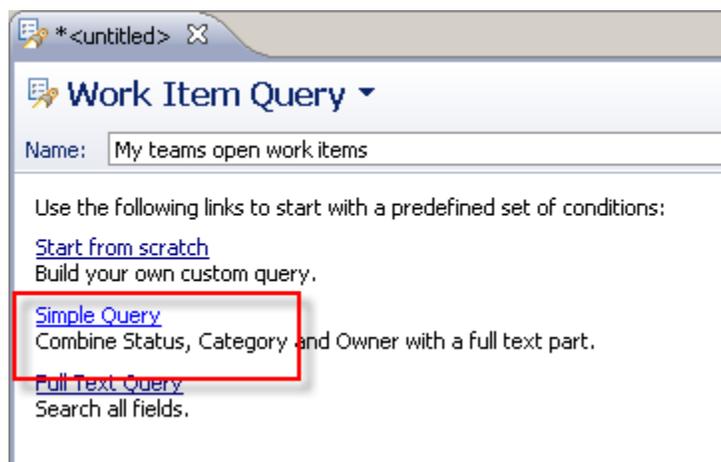
\_\_c. From the **Team Artifacts** view, select **Squawk**→**Work Items**→**My Queries**



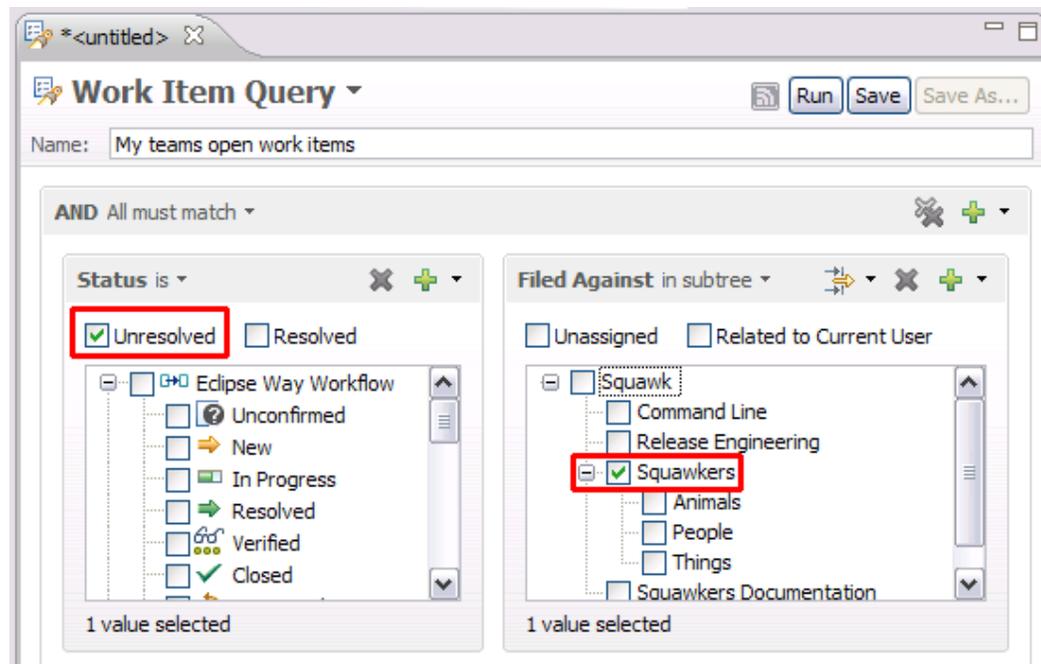
\_\_d. Right-click **My Queries** and select **New Query**. This opens the **Work Item Query** editor.



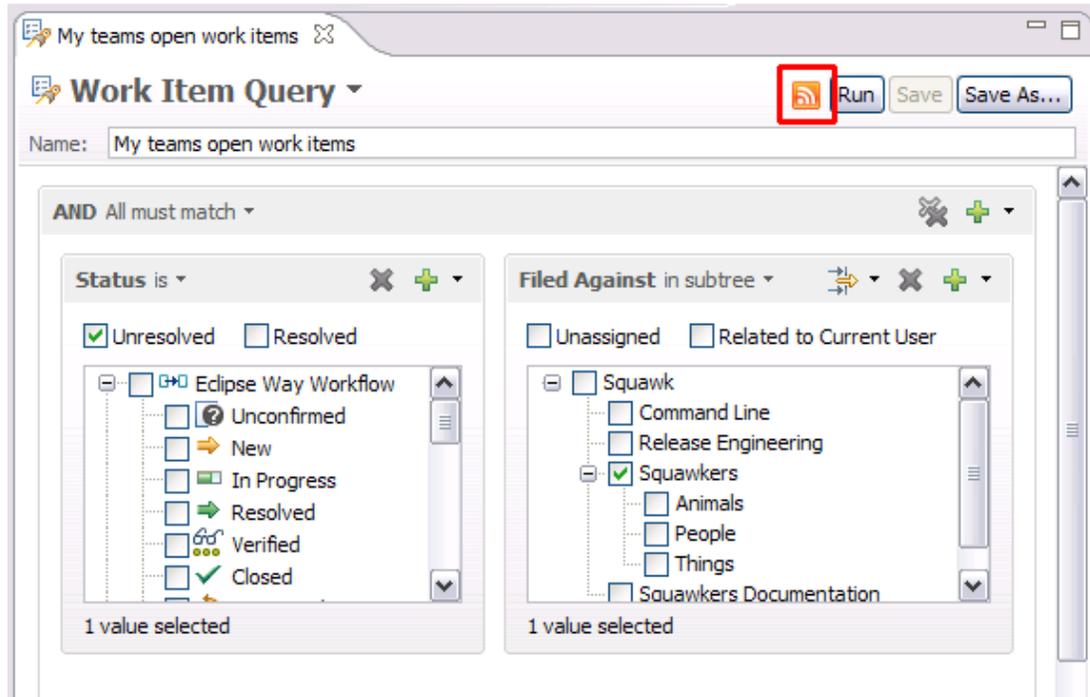
\_\_e. Type the name for the query in the name field: `My teams open work items`. Click **Simple Query**, which will provide you with a selection of default attributes to filter on, like Status, Category, Owner or Full text.



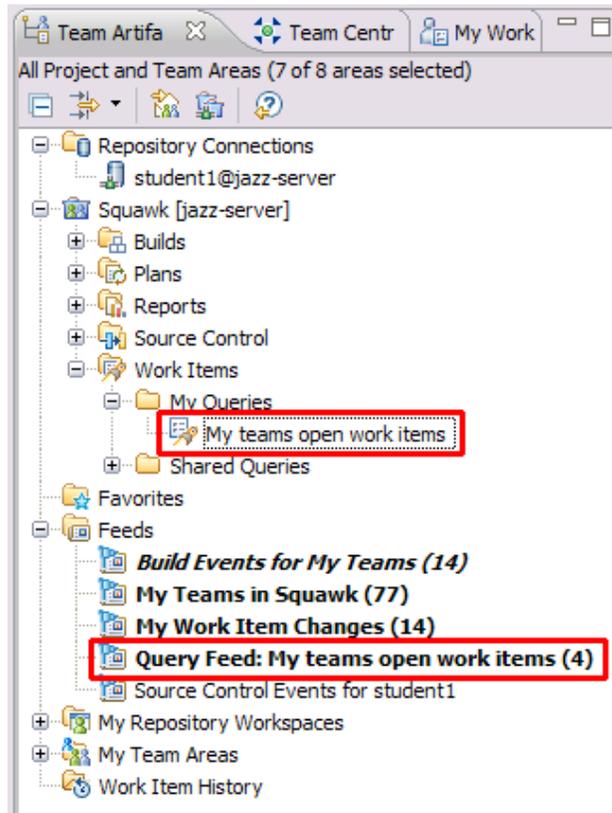
- \_\_f. Select the following filter options from the controls that are available:
- \_\_i. In the **Status** section, check **Unresolved**.
  - \_\_ii. In the **Filed Against** section, check **Squawkers**.
  - \_\_iii. In the **Owned By** and **Full Text** section, click the **X** to remove these filters.
  - \_\_iv. Click **Save**, to save the query and its settings to your workspace.



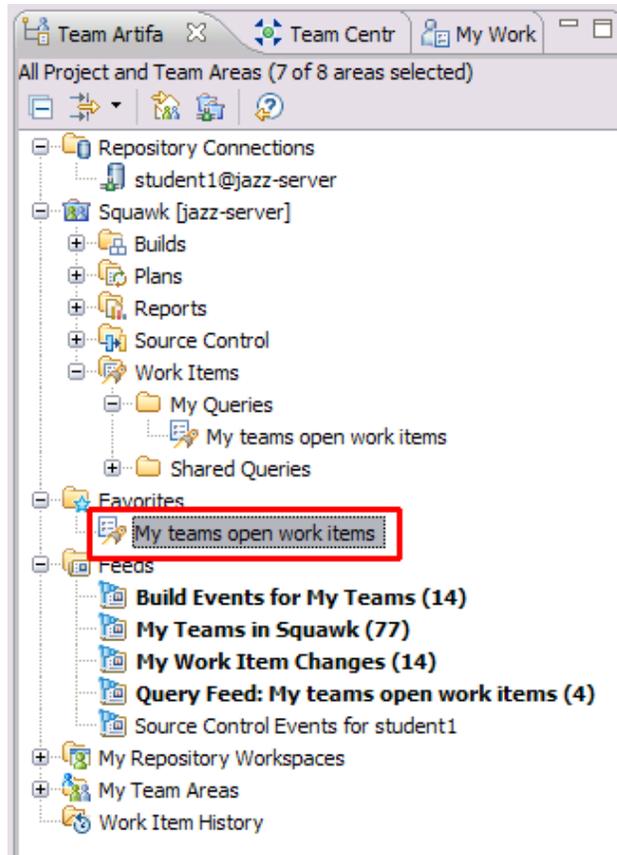
- g. Within Rational Team Concert you are able to subscribe updates on most anything, much like you subscribe to a news feed. This keeps you up to date automatically. Subscribe now to your newly created query. Click  next to the **Run** button on the **Work Item Query** editor. You will use this feed later in the module to automatically track your team's work items.



- \_\_h. After subscribing to the newly created query, it is visible not only in your **My Queries** folder, but also in the **Feeds** folder.



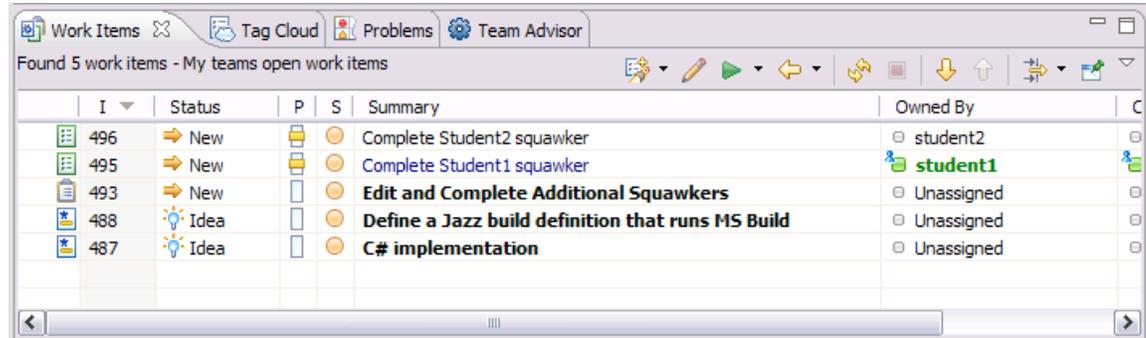
- i. To further keep track of your work, you can add the new query to your **Favorites** folder. The **Favorites** folder is used to organize queries, plans and other artifacts that are important for your work. Drag the **My teams open work items** query to the **Favorites** folder in the **Team Artifacts** view.



- j. It is not necessary to have the **Team Artifacts** view open to access the **Favorites** folder. At any time press **Ctrl-Alt-F** (press the three keys simultaneously) and a popup dialog will appear with your favorites.



- \_\_k. In the **Favorites** or **My Queries** folder, double-click your new query `My teams open work items` to run it. The resulting work items are displayed in the **Work Items** view. (That should be by default displayed at the bottom of the Eclipse client.) The list should be similar to the one shown in the following example picture.



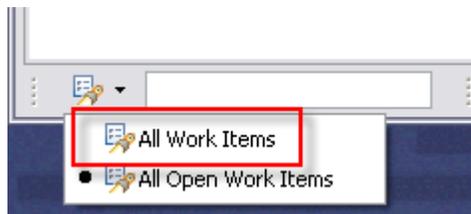
	I	Status	P	S	Summary	Owned By
	496	New			Complete Student2 squawker	student2
	495	New			Complete Student1 squawker	student1
	493	New			Edit and Complete Additional Squawkers	Unassigned
	488	Idea			Define a Jazz build definition that runs MS Build	Unassigned
	487	Idea			C# implementation	Unassigned

- \_\_l. Examine the results of your query.

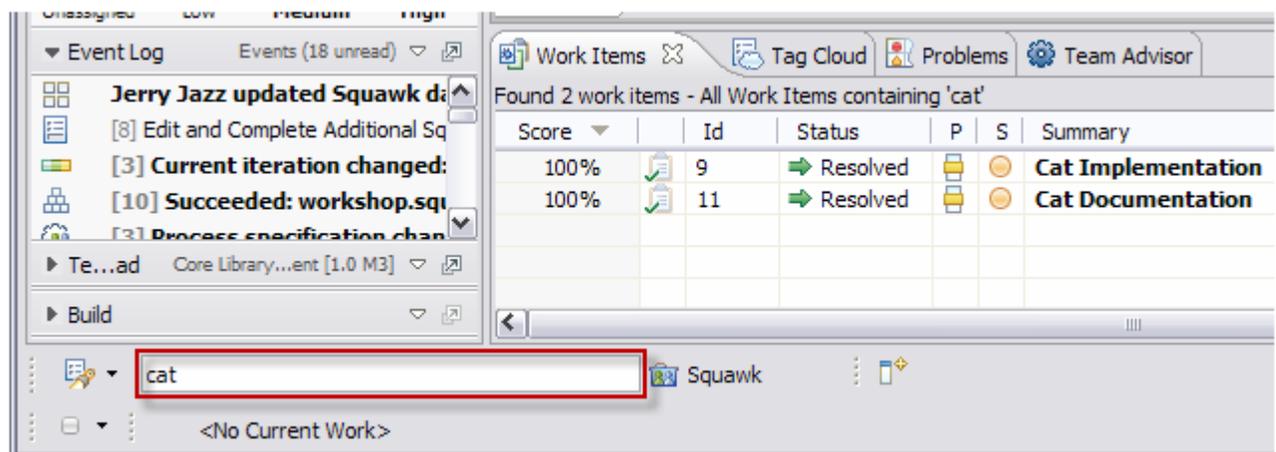
- \_\_m. Click the  on the **Work Item Query** editor to close it.

- \_\_2. Experiment with the quick search.

- \_\_a. In the bottom left corner of the Eclipse Client interface you find a drop down list, expand it and select **All Work Items**.



- \_\_b. In the text field next to the drop down list, type in a word that appears in the title of several work items, for example type `cat` and press **Enter**. The result of this query is displayed in the **Work Items** view.



Score	Id	Status	P	S	Summary
100%	9	Resolved			Cat Implementation
100%	11	Resolved			Cat Documentation



### Quick search

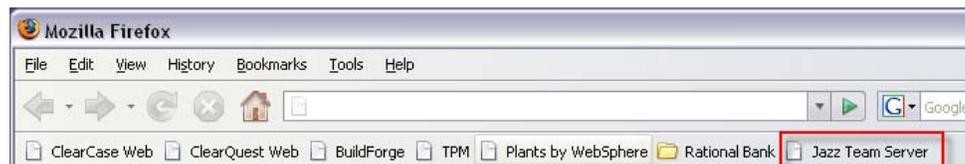
Try typing in a partial word (ca). Note that partial words do not match. However, if you include a wildcard, such as (ca?), Jazz will find it.

You can include wildcard characters as follows: the asterisk (\*) character to represent any number of characters in a string, and the question mark (?) to represent one character in a string. Enclose a phrase within quotes (").

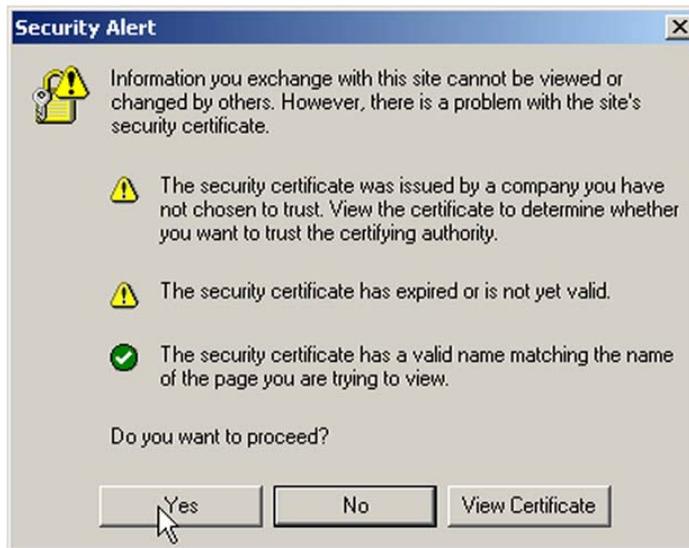
## 3.2 Use the Rational Team Concert Web UI

In addition to the Eclipse and Visual Studio Client interfaces, Rational Team Concert also provides a web user interface (UI). With this web UI you are able to keep track of your team's work too. You can browse through the project area artifacts like iteration plans and work items or create work items. Even if you are not onsite at your office and have only access to an internet browser, you have a real time view to your development project.

- \_\_1. Explore the Squawk project in the Web UI.
  - \_\_a. Open the Web UI
    - \_\_i. Open the **Firefox** internet browser by double-clicking the **Mozilla Firefox** shortcut  on the **Windows Desktop**.
    - \_\_ii. Click the **Jazz Team Server** shortcut on the bookmarks toolbar.



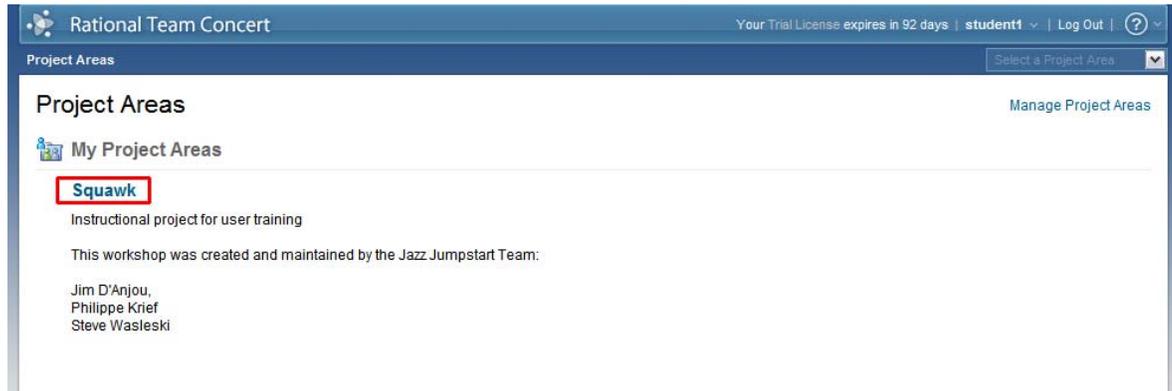
- \_\_b. If a **Security Alert** is displayed, click **Yes** to proceed.



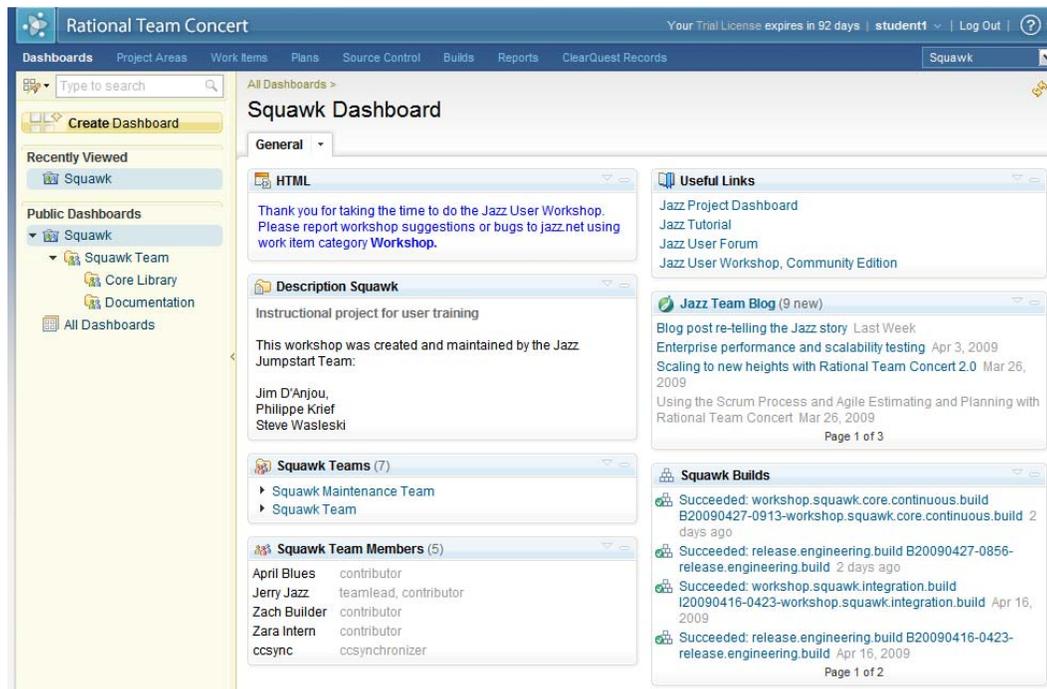
- \_\_c. The Web UI uses a secure connection. Enter your student id for both the **User ID** and **Password**. Click **Log In** to access the project areas.



- \_\_\_d. If you chose to login using your web browser, you will have to choose from one of the project areas. Click the **Squawk** project area.



- \_\_\_e. The appearance in the Web UI is different from the Eclipse or Visual Studio client. Once connected to the project area you will find the main links to project information at the top of the page.



With a dashboard similar to the one you see on this screenshot you have the most important project information directly available.

\_\_f. Click the **Plans** link to switch to that page.



You will see all the Iteration plans defined so far within your project area.



You directly see the progress information for each of your iteration plans, with the time your team members have planned for the iteration and the time they already have worked on.

\_\_g. Select the **Core for M3** plan to open it at the **Overview** page.



- h. Click the **Planned Items** tab to switch to the **Planned Items** page of the **Core for M3** iteration plan.

**Core for M3**

Team Area: Core Library | Iteration: 1.0 M3 (4/21/08 - 12/31/09) | 0 Closed | 2 Open

Progress: 0/16 | -9.75 h | Estimated: 100%

Overview | **Planned Items** | Charts

View As: Work Breakdown

	<b>April Blues</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --
	<b>Jerry Jazz</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --
	<b>student1</b> Closed Items: 0   Open Items: 1		Progress: 0/8 h	Estimated: 100%
<	▶  Create a broader collection of squawkers	High	0/16 h	466
	<b>student2</b> Closed Items: 0   Open Items: 1		Progress: 0/8 h	Estimated: 100%
<	▶  Create a broader collection of squawkers	High	0/16 h	466
	<b>Zara Intern</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --

- i. Select **Team Folders** in the **View as:** field to visualize the work items in the plan organized in folders

**Core for M3**

Team Area: Core Library | Iteration: 1.0 M3 (4/21/08 - 12/31/09) | 0 Closed | 2 Open

Progress: 0/16 | -9.75 h | Estimated: 100%

Overview | **Planned Items** | Charts

View As: Work Breakdown

- Work Breakdown
- Backlog
- Developer's Taskboard
- Planned Time
- Team Folders**
- Work Breakdown

	<b>April Blues</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --
	<b>Jerry Jazz</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --
	<b>student1</b> Closed Items: 0   Open Items: 1		Progress: 0/8 h	Estimated: 100%
<	▶  Create a broader collection of squawkers	High	0/16 h	466
	<b>student2</b> Closed Items: 0   Open Items: 1		Progress: 0/8 h	Estimated: 100%
<	▶  Create a broader collection of squawkers	High	0/16 h	466
	<b>Zara Intern</b> Closed Items: 0   Open Items: 0	No Work	Progress: --	Estimated: --

- j. Click the twisty icon  to expand the **Top Items** section. Expand the Story and Task to retrieve the squawker implementation tasks.

**Core for M3** 📄 🏷️

Team Area: Core Library | Iteration: 1.0 M3 (4/21/08 - 12/31/09) | 0 Closed | 2 Open

Progress: 0/16 | -9.75 h Estimated: 100%

Overview | **Planned Items** | Charts

View As: Team Folders

Top Items			
Closed Items: 0   Open Items: 2		Progress: 0/16 h	Estimated: 100%
<	▼ Create a broader collection of squawkers	0/16 h	High 466
<	▼ Edit and Complete Additional Squawkers	0/16 h	Unassigned 493
	📄 Complete Student2 squawker	--	Medium 496
	📄 Complete Student1 squawker	--	Medium 495
<b>Defects &amp; Enhancements</b>		No Work	
Closed Items: 0   Open Items: 0		Progress: --	Estimated: --

- k. Click the summary text of one of the work items in the plan to switch to the **Work Items** page. The selected work item opens.

Top Items			
Closed Items: 0   Open Items: 2		Progress: 0/16 h	Estimated: 100%
<	▼ Create a broader collection of squawkers	0/16 h	High 466
<	▼ Edit and Complete Additional Squawkers	0/16 h	Unassigned 493
	📄 Complete Student2 squawker	--	Medium 496
	📄 <b>Complete Student1 squawker</b>	--	Medium 495

**Task 496** 📄 🏷️ 📄 Save

Summary: Complete Student1 squawker ➔ New

Overview | Links | Approvals | History Loaded: Aug 12, 2009 10:27 A.M.

Details		Quick Information	
Type:	Task	Owned By:	student1
Severity:	Normal	Priority:	Medium
Found In:	Unassigned	Planned For:	1.0 M3
Creation Date:	Aug 12, 2009 9:44 A.M.	Estimate:	1 day
Created By:	student1	Time Spent:	
Project Area:	Squawk	Due Date:	
Team Area:	Core Library		
Filed Against:	Squawk/Squawkers		
Tags:	squawker		

**Description** Edit

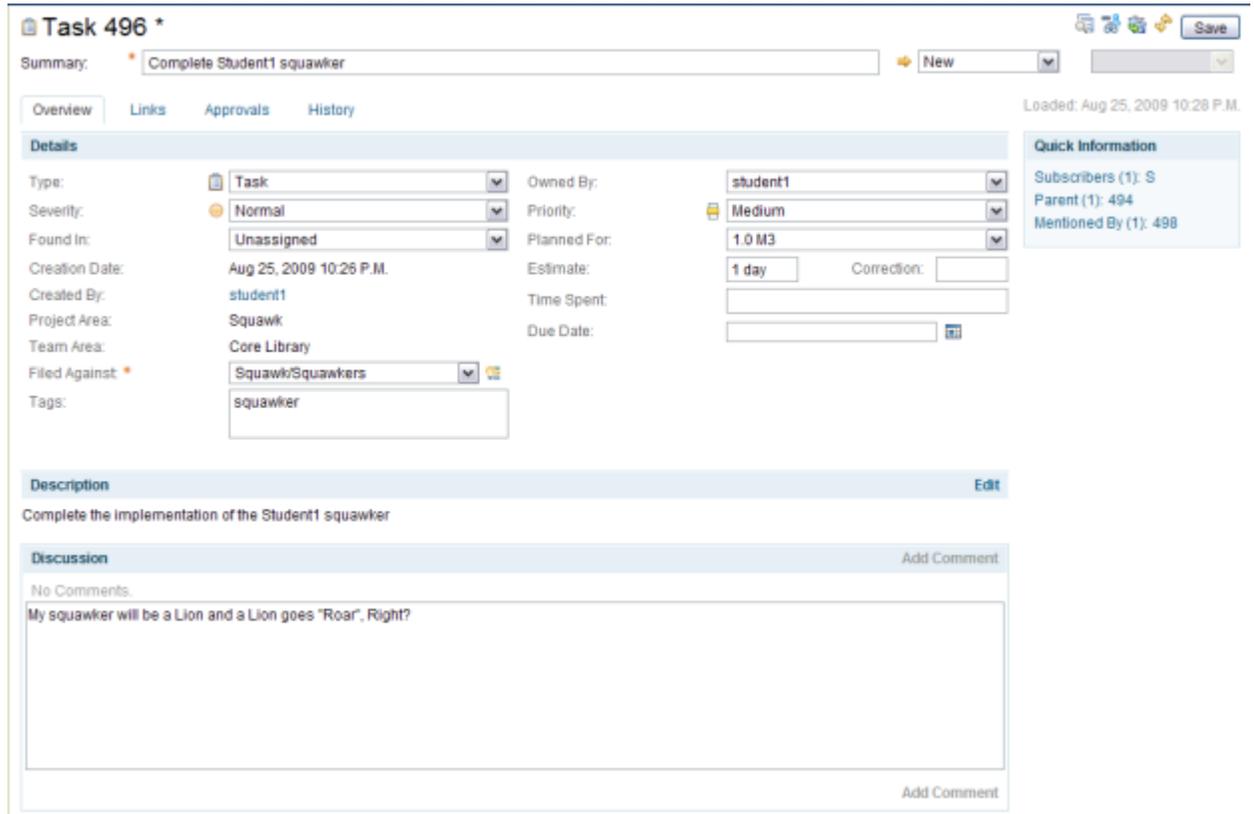
Complete the implementation of the Student1 squawker

**Discussion** Add Comment

No Comments.

Add Comment

\_\_l. Scroll down and click the **Add Comment** link and type a comment.



Click **Save** to commit your change to the repository.

\_\_m. The query created in the section before in the Rational Team Concert client is listed in the frame on the left.



Click the query **My teams open work items** to run it. Explore the result set of the query.

**Quick Search on the Web**

The Web UI has a powerful ad hoc search mechanism. At the top of the left panel, there is a search text box that you can enter a work item number or text to search for matches.

\_\_n. Close the browser.

**Web UI**

The Web UI has the same powerful capabilities as the Eclipse and Visual Studio clients for working on work items. You are able to create queries in the Web UI as well as you are able to use a quick search too.

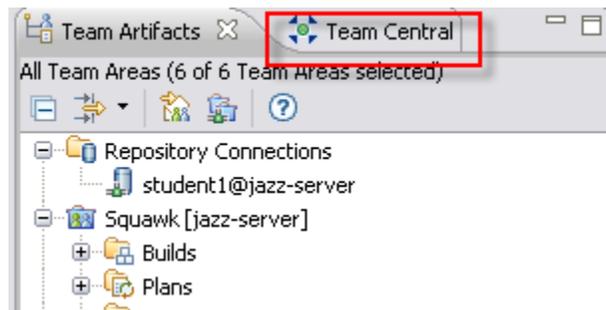
### 3.3 Using and configuring real time project status – Team Central View

Rational Team Concert has powerful capabilities to provide each role of the development project with a custom view to the project's status. The **Team Central** view is dynamically refreshed and therefore provides a real time view of the project's health. It is customizable to provide the user with all the necessary information.

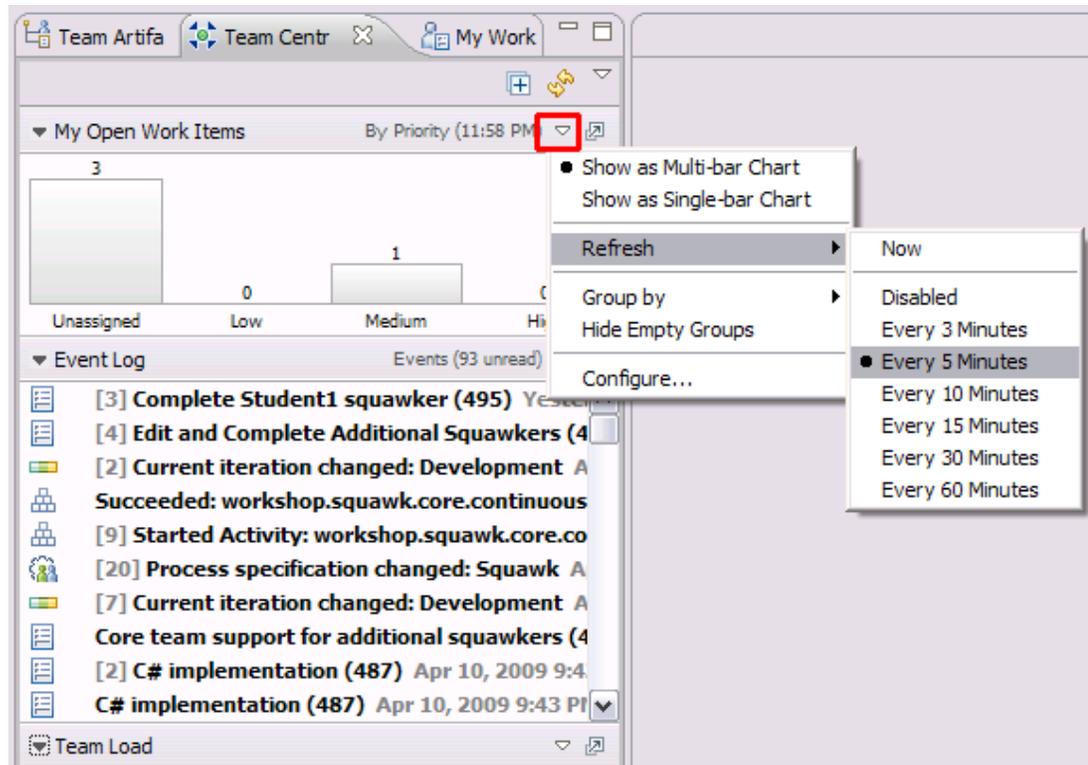
\_\_1. Explore the capabilities of the **Team Central** view

\_\_a. If not already selected, switch your application focus back to the Rational Team Concert Eclipse client.

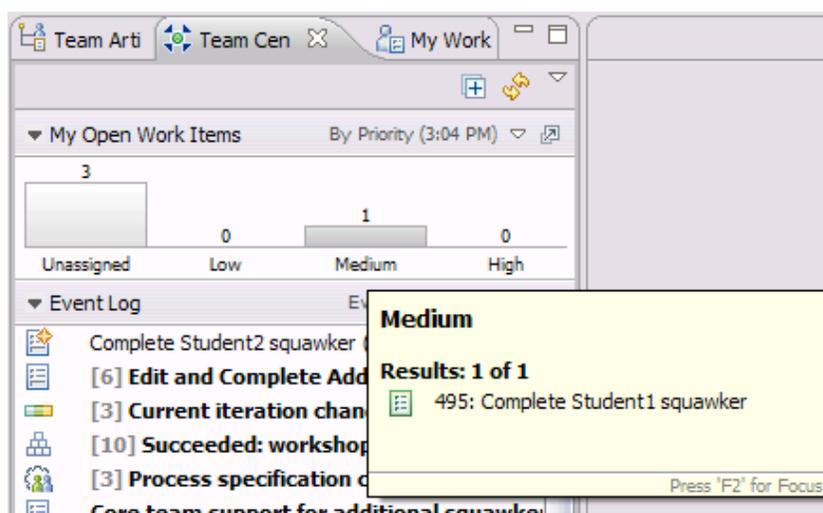
The **Team Central** view displays important information for the user; therefore it is by default available in the **Work Items** perspective. As you have started with a new Eclipse workspace, and the **Work Item** perspective is the default one in Rational Team Concert, the **Team Central** view is available next to the **Team Artifacts** view.



- \_\_b. The **Team Central** view is divided by default into the sections: **My Open Work Items**, **Event Log**, **Team Load**, and **Build**. Each of these sections can be individually configured with its own dynamic refresh rate, by clicking on the white triangle in the corner of the section ▾ to expand the configuration menu.



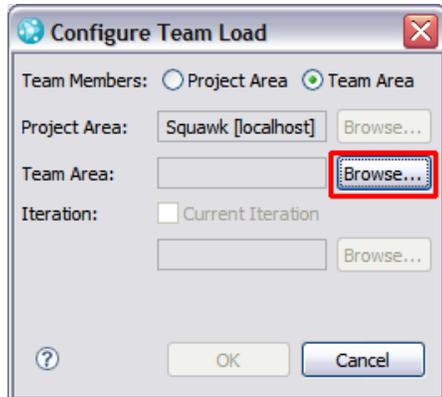
- \_\_c. The section **My Open Work Items** displays the result of the corresponding query as a bar chart. Hover over the **Medium** priority bar on the graph and see a summary of the work items in the pop up window.



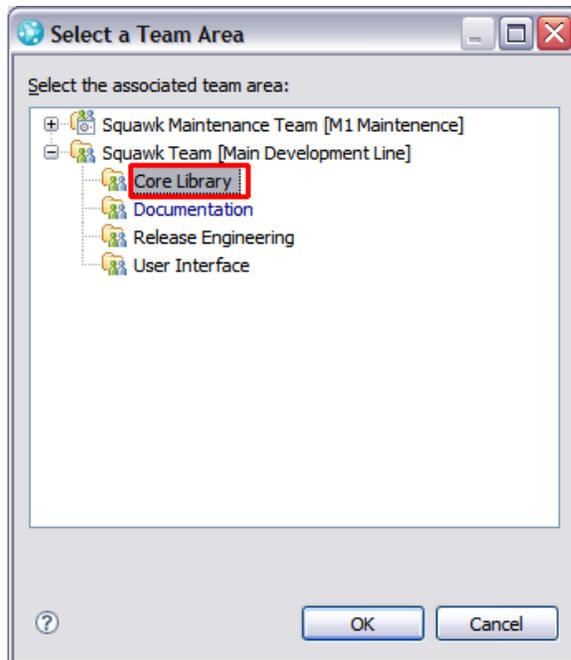
- \_\_d. Expand the **Team Load** section and select **Configure...** from the configuration section of the section.



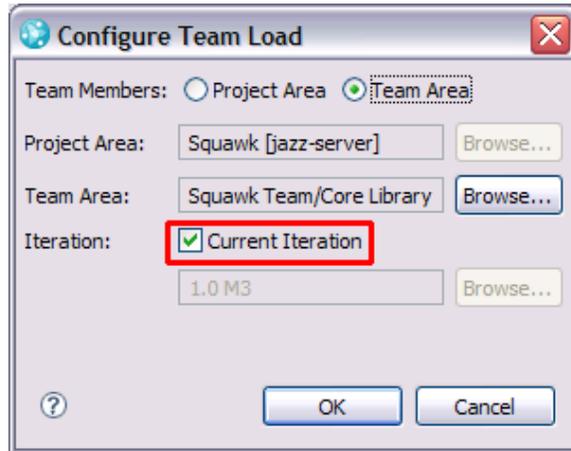
- \_\_e. Click on the Browse button to select the team you are working with



- \_\_f. From the proposed team tree, select the **Squawk Team/Core Library** team area and press **OK**.

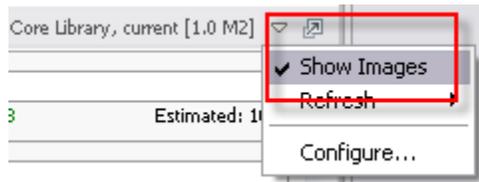


\_\_g. Check **Current Iteration**



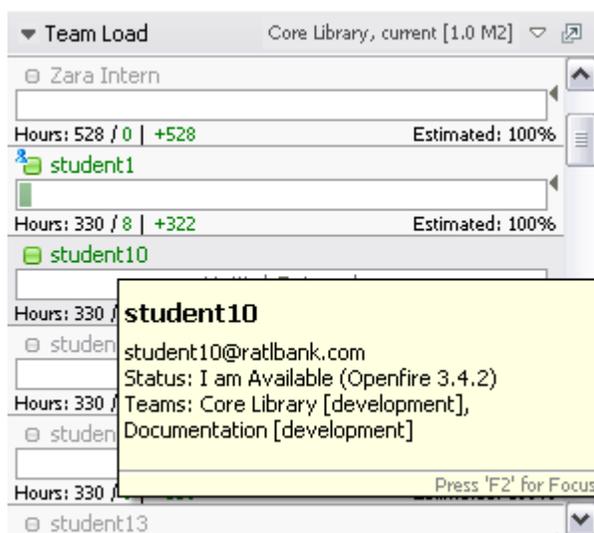
\_\_h. Close the **Configure Team Load** dialog by pressing OK.

- \_\_i. Then you should see your team members that are assigned to the **Core Library** team. By default Rational Team Concert displays the members' picture. No pictures are included for the student users in this PoT. To remove the pictures and display only the members name, click the  of this section to access the configuration menu.

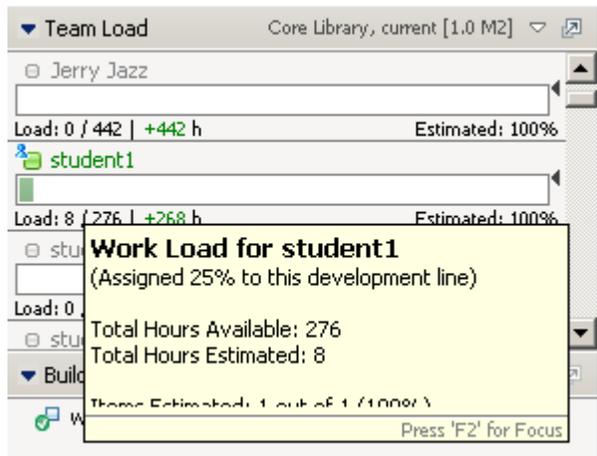


Deselect **Show Images**.

- \_\_j. In the **Team Load** section hover over a team member's name to see who it is.



- \_\_k. Hover over a bar displayed together with the team member's name to see a summary of the planned activity for a member of the team.



- \_\_l. The **Event Log** section shows the actual team events that happen. It displays the news of your team, like build events etc. You can regard this as a special kind of news feed.



**Configuring the Event log section**

Depending on the size of your team and the activity of your team members, the number of events that are displayed in your event log could be huge. Therefore the Event log section has powerful Filter and display options to handle great numbers of events. If you have time, explore the Configuration dialog that is available when you click the triangle ▾ of this section.

- \_\_m. If you click on the items in the event log, a summary box is displayed with detailed information and a link to the source for this event.

The screenshot displays the IBM Software interface. On the left, the 'Event Log' pane shows a list of events. One event, '[Joining a Team] Create a Repository Workspace (83)', is highlighted with a red box. A red arrow points from this event to the summary box on the right. The summary box, titled '\* [Joining a Team] Create a Repository Workspace (83)', contains the following information:

- On Apr 21, 2008 11:28:02 AM Jerry Jazz created:
- Summary: [Joining a Team] Create a Repository Workspace
- Status: New
- Description: Create a Repository Workspace

The description includes a detailed instruction: 'To create a new repository workspace to work with your new Jazz team:' followed by a numbered list of steps:

1. In the **Team Artifacts** view, expand the **My Team Areas** node.
2. Open the team area you have joined by double clicking on the team area.
3. In the **Team Artifacts** section of the opened **Team Area** editor, expand the **Streams** node.
4. Right click on the stream and select **New Repository Workspace...**
5. Specify the repository workspace name. Click **Next**.
6. Click **Finish**.

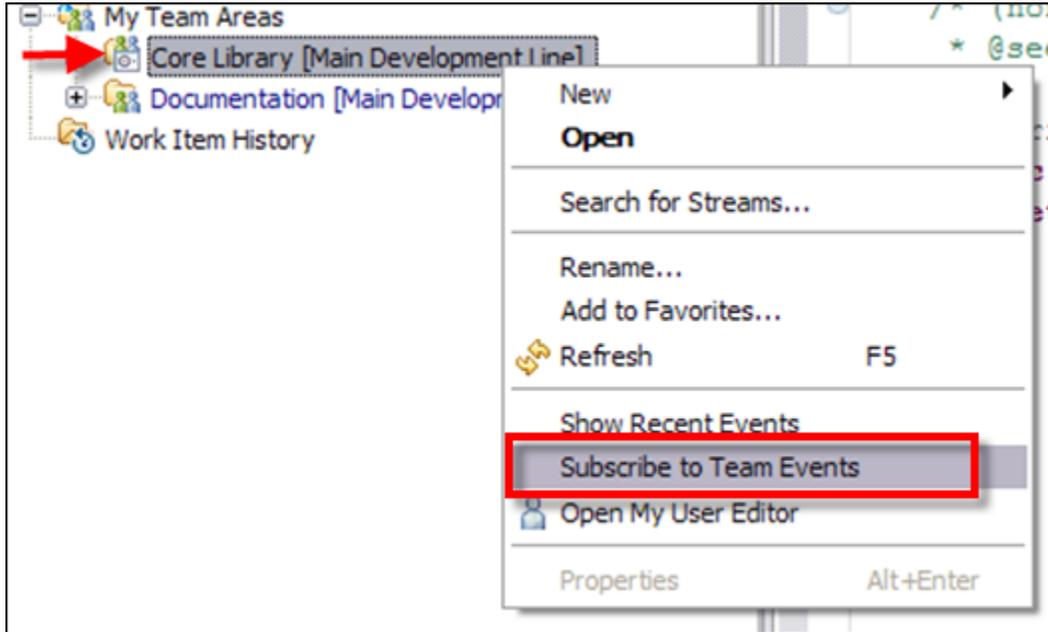
The interface also shows a 'Team Load' section with users like Jerry Jazz, student1, student10, and student11, and a 'Build' section with the entry 'workshop.squawk.core.continuous.build'.



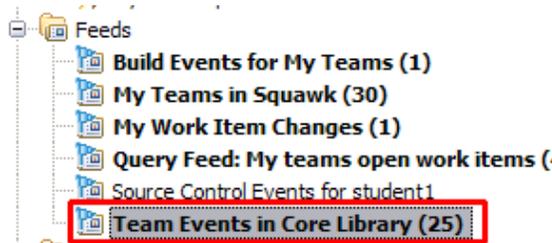
### Event Log Entries

Some event log entries have a number preceding its title. The number represents the number of changes to the entry since it was first logged. When you hover over that entry, you can scroll through the changes.

- \_\_2. To keep abreast of what your team is doing and further illustrate the transparency of the Jazz environment, you can configure a feed by subscribing to your team events.
  - \_\_a. In the **Team Artifacts** view, expand **My Team Areas** and right-click the **Core Library** team.
  - \_\_b. Select **Subscribe to Team Events**.



- \_\_c. You will now have a **Core Library Team Events** feed under the **Feeds** folder.



- \_\_d. You can filter the Team Concert event stream for two categories of Jazz source control events:

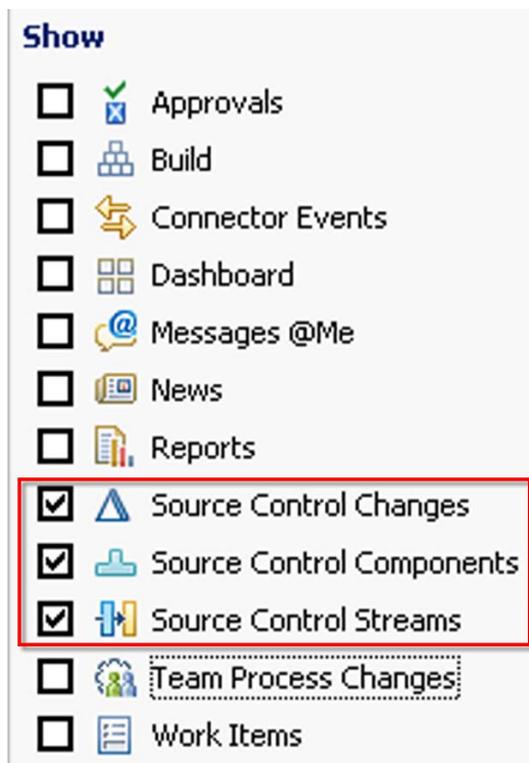
Source Control Changes events include the delivery of change sets and baselines to your team's streams.

Source Control Streams events include changes to a stream's name and changes to stream components (additions, removals, and replacements).

- \_\_e. Double-click the **Team Events in Core Library** feed.
- \_\_f. Under the Show section, click on the **Show All** button to visualize all the event filters:



- \_\_g. To filter event feeds for Jazz source control events, under the **Show** section, uncheck all options except **Source Control Changes**, **Source Control Components**, **Source Control Streams** and **Source Control User Events**. If interested in other event types, leave them checked as well.



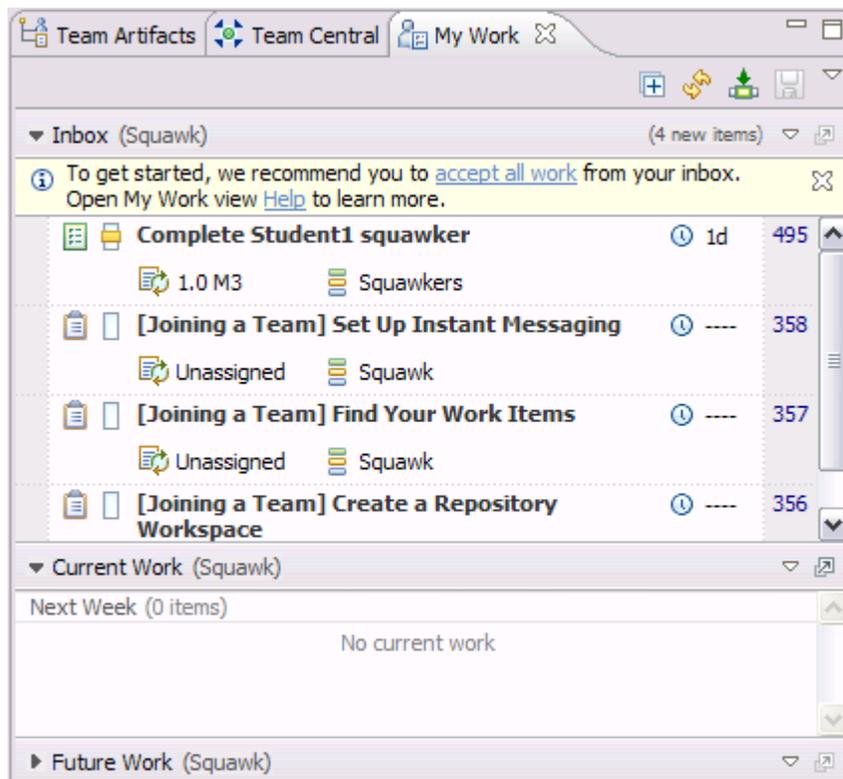
Imagine you are away for a week. When you return you can use the team feeds to catch up with the activities of your team without having to ask. Transparency in action for your benefit!

### 3.4 Tracking your work – My Work view

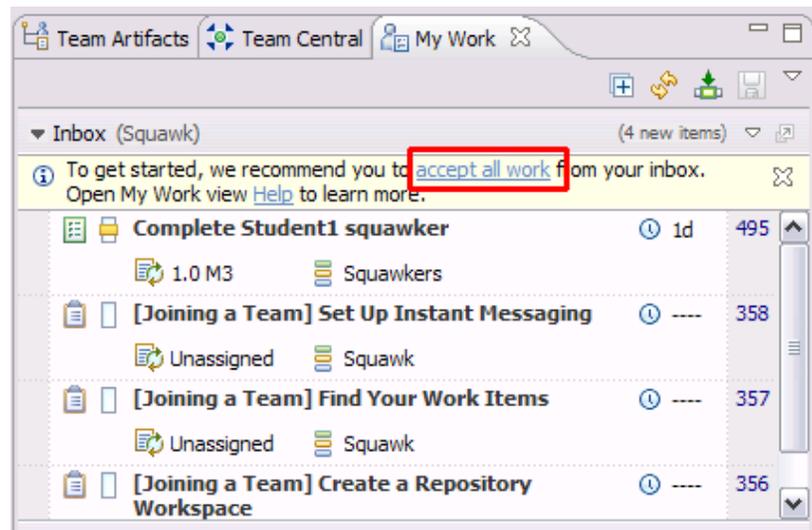
Whereas the **Team Central** view is intended to show you the real time status of the project and its health and work status taking all the team members and their work into account, the **My Work** view is meant to help you organize your own work. In addition to the sections that are available by default you can add your own queries and subscribe to your special interest news feeds.

\_\_1. Customize **My Work** view.

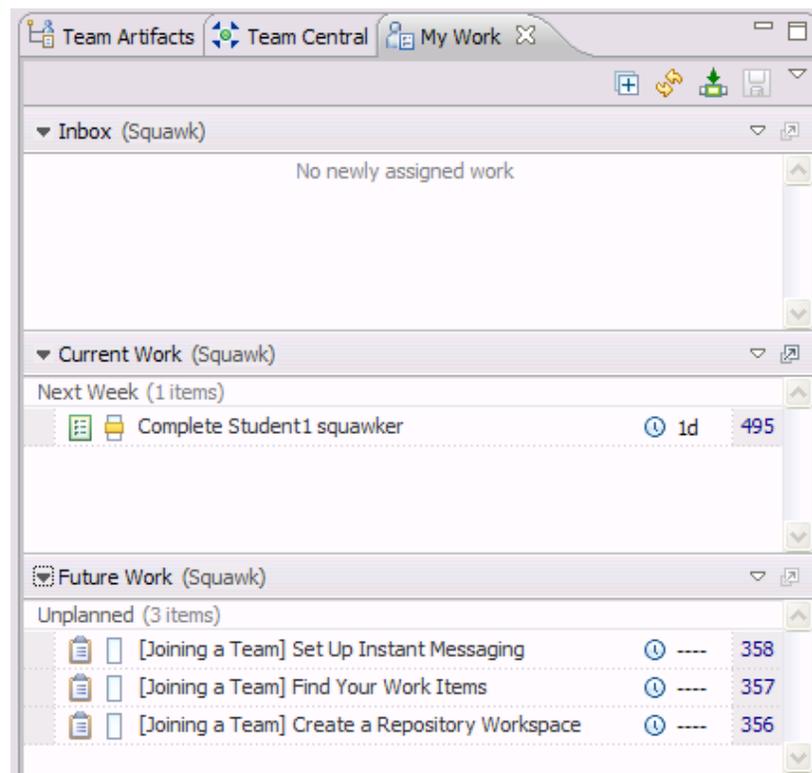
\_\_a. The **My Work** view is another view that is located in the same perspective as **Team Central**. With the new workspace you created the **My Work** view looks like this:



The Inbox shows new work Items that have been assigned to you.

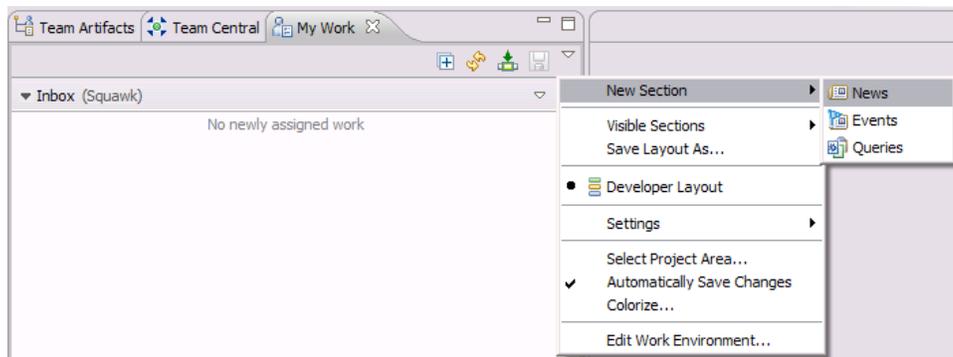


When you select the **accept all work** link from the help section you accept all the work items displayed in your Inbox, i.e. work items for the current iteration are shown in your **Current Work** section and those for future iterations are in **Future Work** and listed as **Unplanned**:

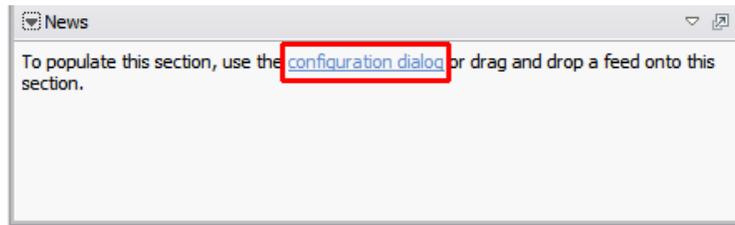


You are able to arrange the work items assigned to you in the current work section, simply by dragging and dropping them to the **Today** section. Note that you are only able to add work items to today's work if the maximum of your work hours is not reached. The **My work** view will add **Tomorrow** or **Later this week** sections for your work items that exceed your working hours for today.

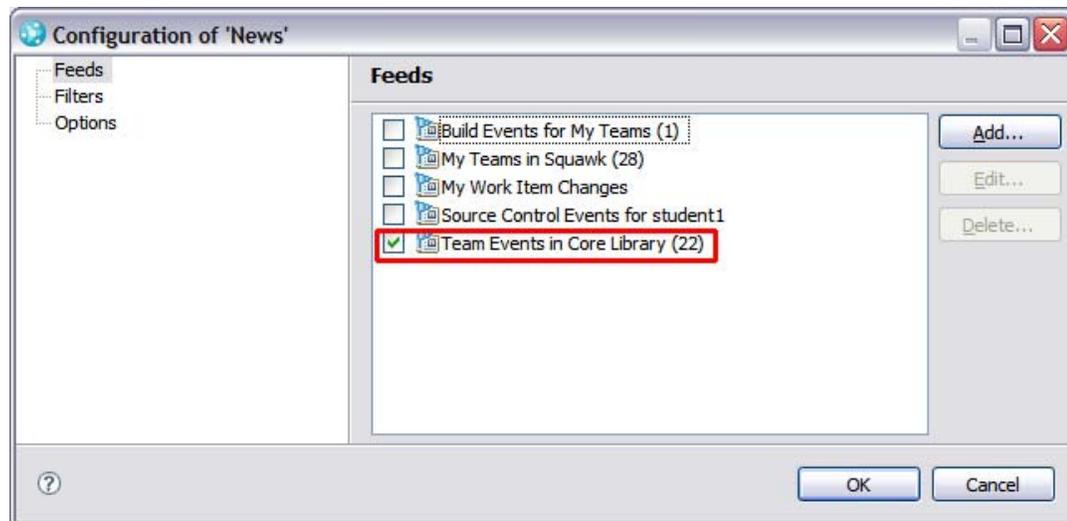
- \_\_\_b. During your work you will see new work items that are assigned to you in your **Inbox**. You are able to use this view to organize your work by dragging and dropping work items from **Inbox** to **Current Work**. Note, if you previously selected **accept all work**, there may not be any work items currently in your Inbox to move to **Current Work**.
- \_\_\_c. You can add new sections to your **My Work** view. Click the triangle icon  on the **My Work** view main menu bar and select **New Section** from the menu. Select **News** to create a new **News** section.



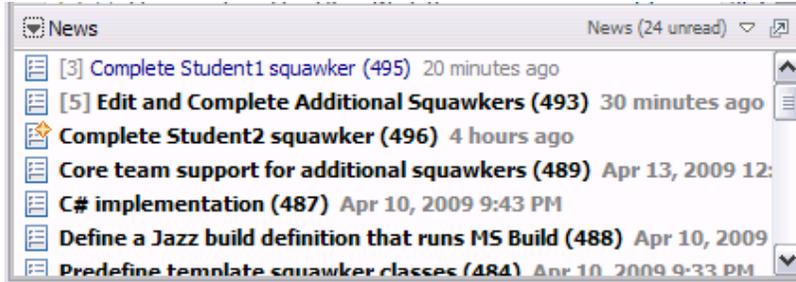
- \_\_\_d. The new section appears in your my **Work View** and requires configuration. Click the **configuration dialog** link to open the **Feed Section Configuration** dialog.



- \_\_\_e. In the **Feed Section Configuration** dialog you see a list of already available feeds. These are the same feeds that are available under the **Feeds** node in your **Team Artifacts** view. You see the feed **Team Events in Core Library** that you created earlier. Select this feed to configure your personal news section. Click **OK**.



- \_\_\_f. If you or any other team member submits a new work item or changes an existing one, you will see this change in your new **News** section. (By default the reload interval is set to 10 minutes for a news feed, i.e. you have to wait 10 minutes to see changes. You can reduce the interval to as low as 3 minutes in the properties dialog of the feed.)



### Conclusion



This concludes the Lab for module 3. You are now familiar with the most important capabilities on how to query for and retrieve work item and plan data in Rational Team Concert, either in the Eclipse or Web UI clients. Additionally you learned that Team Concert has specialized views to help you stay informed of what the team is doing and to manage your work.

## Lab 4 Performing and Sharing Your Work

In this lab you will learn how to setup your Rational Team Concert repository workspace from a stream to access the Squawk application. You will also explore the Rational Team Concert repository workspace's structure. You will then modify your own squawker class and a class to test it. After validating your work, you will associate the change set with a Work Item and deliver from your repository workspace to a stream and set the Work Item state as resolved. At the end, you will accept into your repository workspace all the squawkers developed by your peers.



### Lab Scenario

As the newest member of the Squawk project, you have been tasked with contributing your own squawker class. Creating and sharing your code involves working with a Software Configuration Management system. In Rational Team Concert, this is all built in. You will spend a little time understanding the key concepts of the SCM system including where your work gets done, how you share this with others and how code and other artifacts are typically structured in the SCM system.

Once you understand the basic concepts you get started writing code. You edit your own squawker its tests against the work items you created in Module 2 "Planning your Work".

When you have completed the code and tests, you will deliver the changes so that other people can use it.

Finally, you will bring in changes from other members of your team so your code and tests are up-to-date with everyone else. Optionally, this will also highlight how Rational Team Concert allows you to handle conflicting changes to software artifacts under Jazz Source Control.



### Important!

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will the use the id **student1**. If you are unsure please check with the instructor.

**Some Jazz Terms Defined:**Project Area

The project area is the system's representation of a software project. The project area provides definition for the project deliverables, team structure, process, and schedule.

Team Area

The structure of the project teams is defined by one or more team areas. A team area serves these functions:

- Defines the users (team members) on the team and specify their roles.
- Defines the development line in which the team is participating.
- Customizes the project process for the team.

Streams and Components

A stream is a repository object that includes one or more components. A component is a collection of related artifacts, such as an Eclipse plug-in or a group of documents that comprise Web site content.

Change Set

A change set is a repository object that collects a related group of changes so that they can be applied to a flow target (workspace or stream) in a single operation.

The change set is the fundamental unit of change in Jazz source control. The contents of any workspace, component, or stream can be expressed as a collection of change sets, beginning with the one created when the initial set of projects was checked in. A change set can include changes to the contents of individual files and changes to a component namespace (such as delete, rename, and move operations).

## 4.1 Creating your repository workspace from the Core Library stream

\_\_1. If you don't already have Rational Team Concert running, do the following:

- \_\_a. Open Rational Team Concert by double-clicking the Team Concert shortcut  on the Windows Desktop.

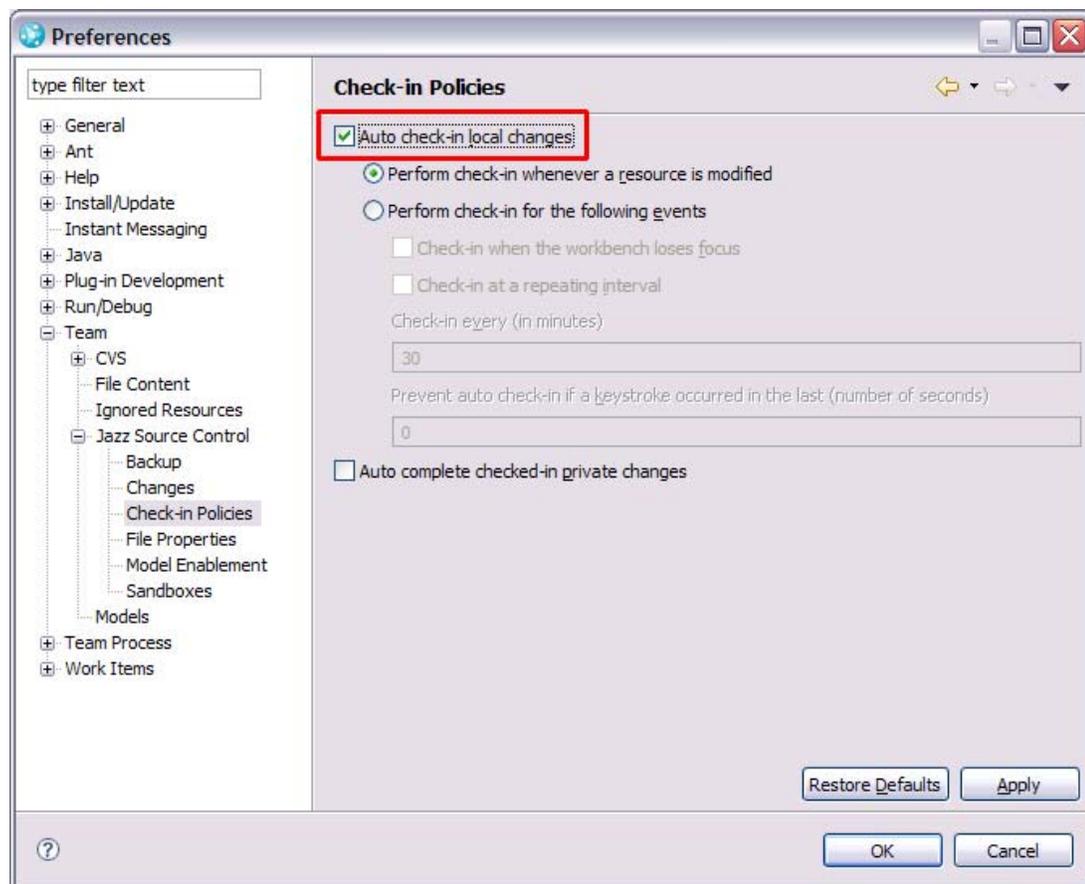
- \_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` (replacing `<N>` with your id number). Click **OK**.



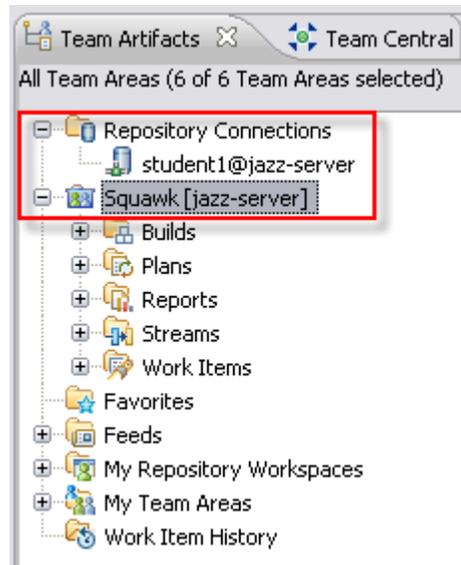
## \_\_2. Enable automatic check-ins

Once a change is made to a versioned file, its corresponding change set must be checked in to commit the change to your repository workspace. By default, check-ins must be initiated by the user. However, Team Concert provides the option for making check-ins occur automatically. This will be done now to simplify the steps in the workshop.

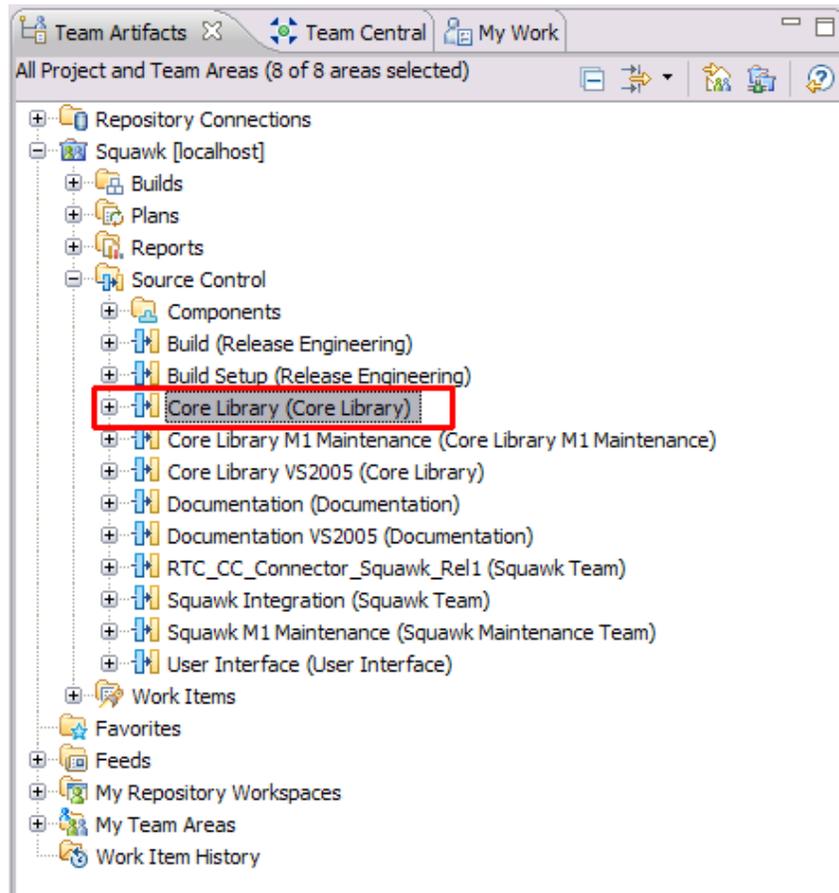
- \_\_a. Select **Window → Preferences** to open the Preferences settings.
- \_\_b. Expand **Team → Jazz Source Control** and select **Check-in Policies**
- \_\_c. Check **Auto check-in local changes**
- \_\_d. Click **OK**.



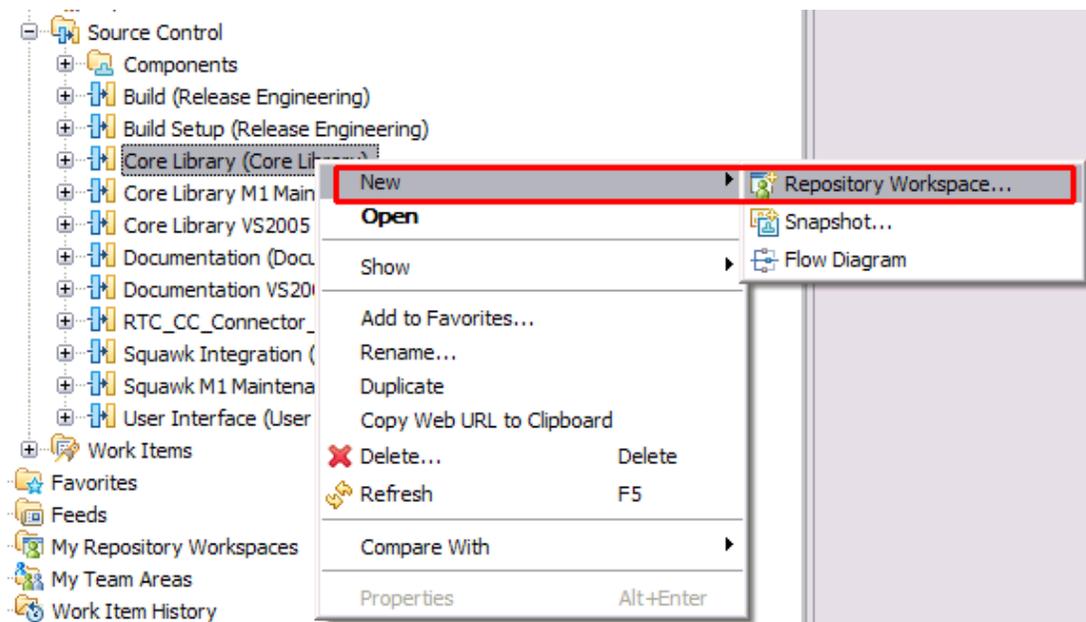
- \_\_3. Create your Repository Workspace from the **Core Library** stream
- \_\_a. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), ensure that you are connected to the **Squawk** project area as `student<N>`



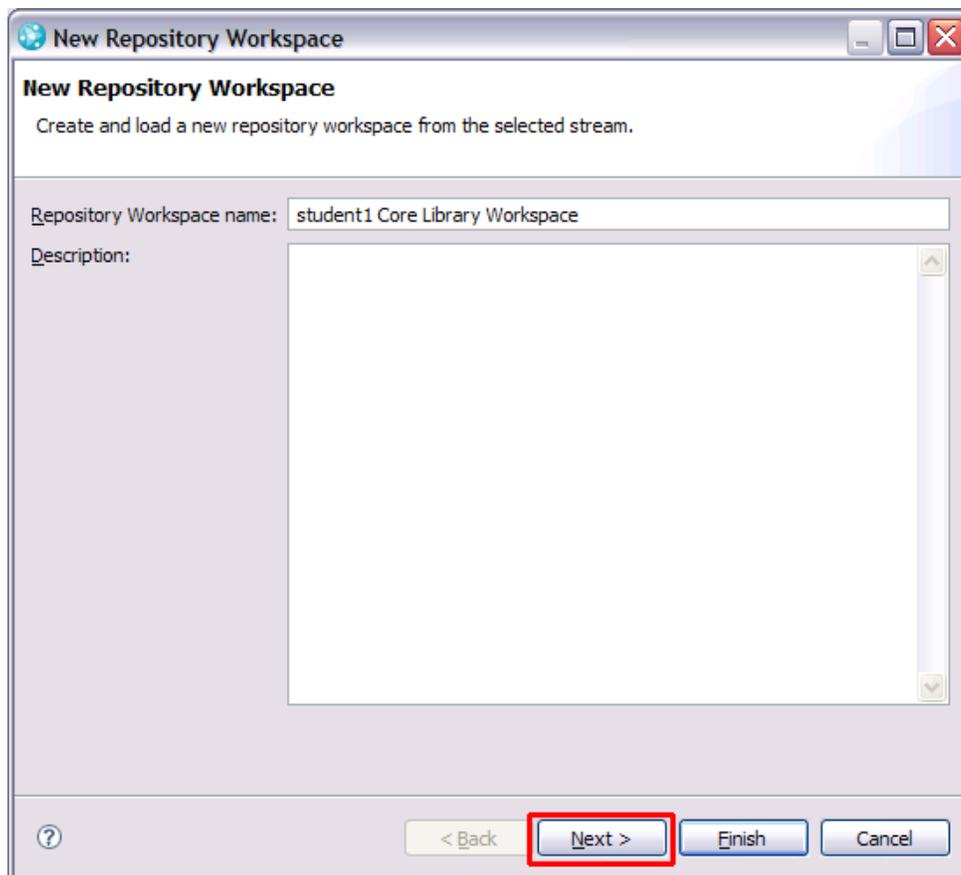
- b. Expand the **Squawk** project area and then expand **Source Control**, to show the **Core Library** stream.



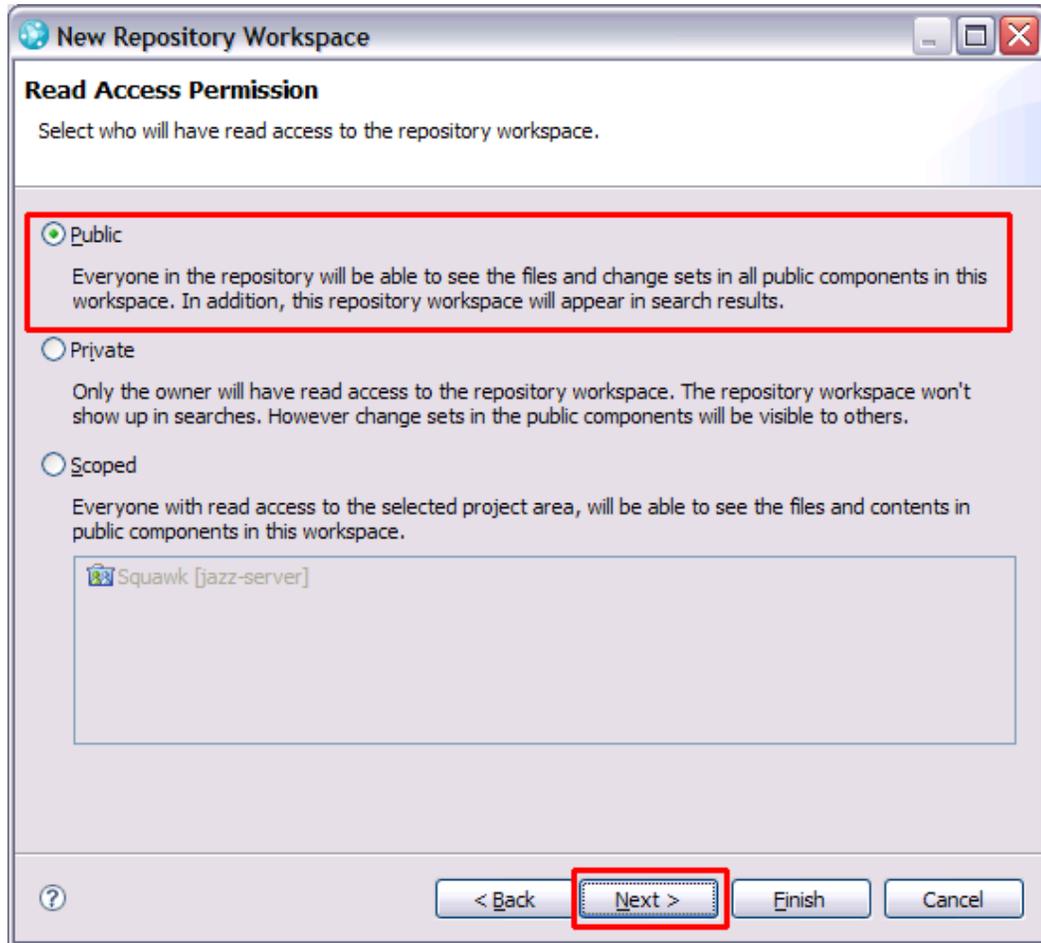
- c. Right-click the **Core Library** stream and select **New → Repository Workspace...**



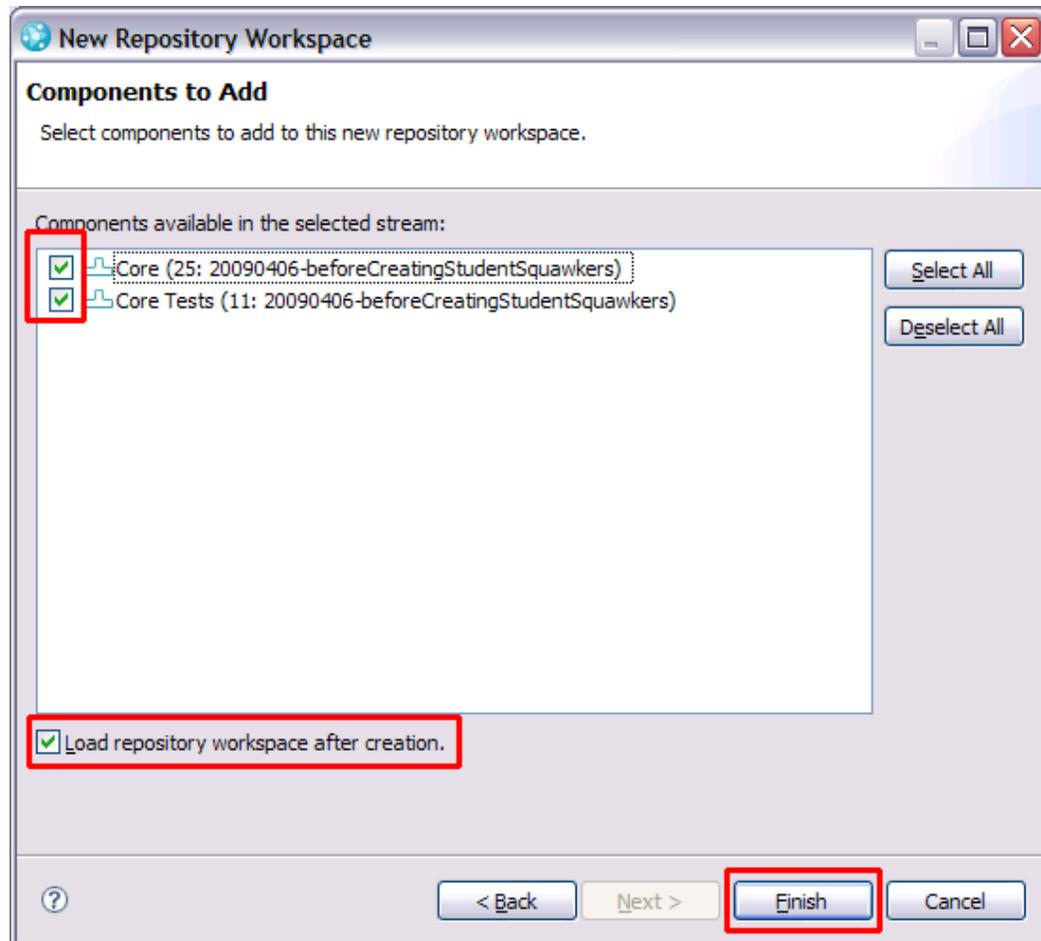
- \_\_\_d. In the **New Repository Workspace** window modify the **Repository Workspace** name to reflect your identity. For example type student<N> Core Library Workspace. Click **Next**.



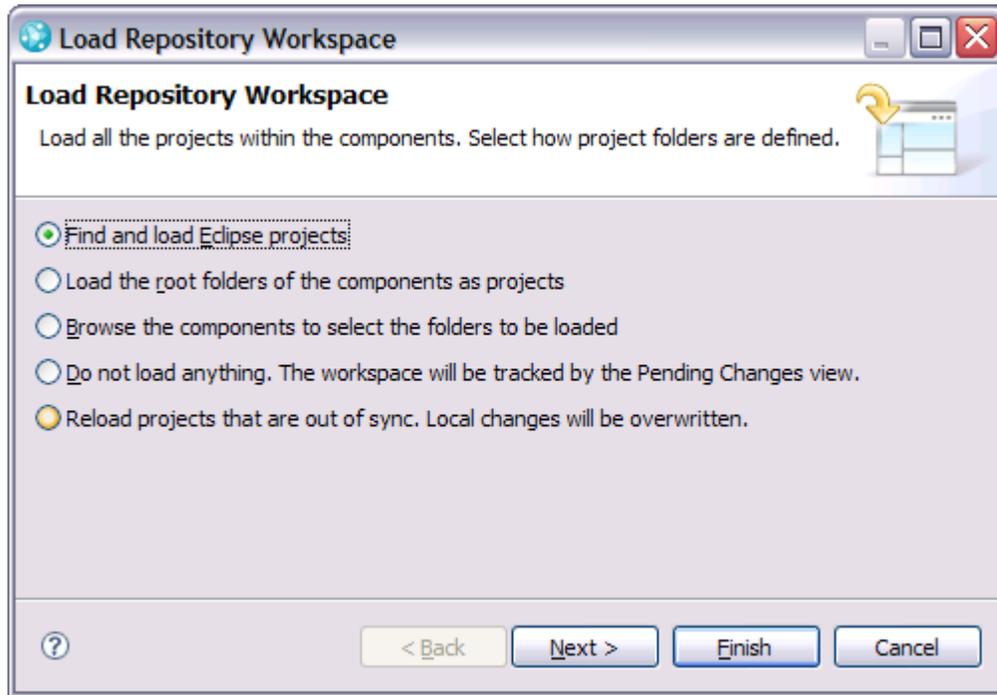
- e. In the **Read Access Permission** dialog select **Public** to make this new workspace visible by everyone who has access to the Jazz server. Click **Next**.



- \_\_\_f. In the **Components to Add** dialog, verify that all components included in the stream are selected and check **Load repository workspace after creation** if it is not already checked.  
Click **Finish**.



- g. On the **Load Repository Workspace** window, select **Find and load Eclipse projects** and click **Finish**.

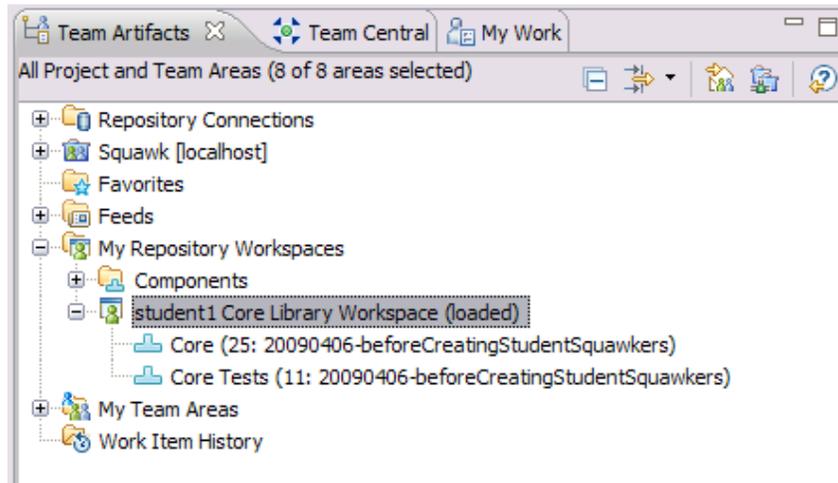


**Loaded Workspaces**

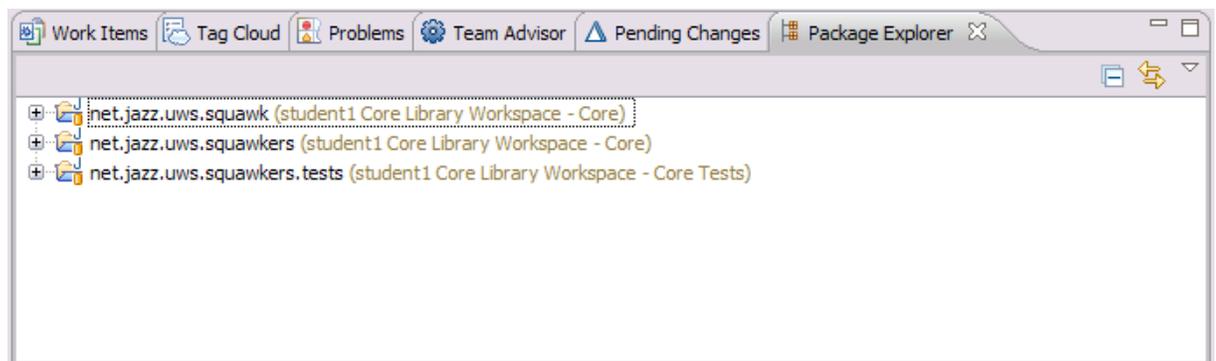
Loaded **repository workspaces** are special in that whenever you change a loaded file or folder in your Eclipse workspace, the changes are tracked and shown in the **Pending Changes** view. Here, you can manage your changes and perform common tasks such as:

- \* Check-in changes to your repository workspace.
- \* Organize changes into change sets.
- \* Undo changes you've made.
- \* Associate change sets with work items.

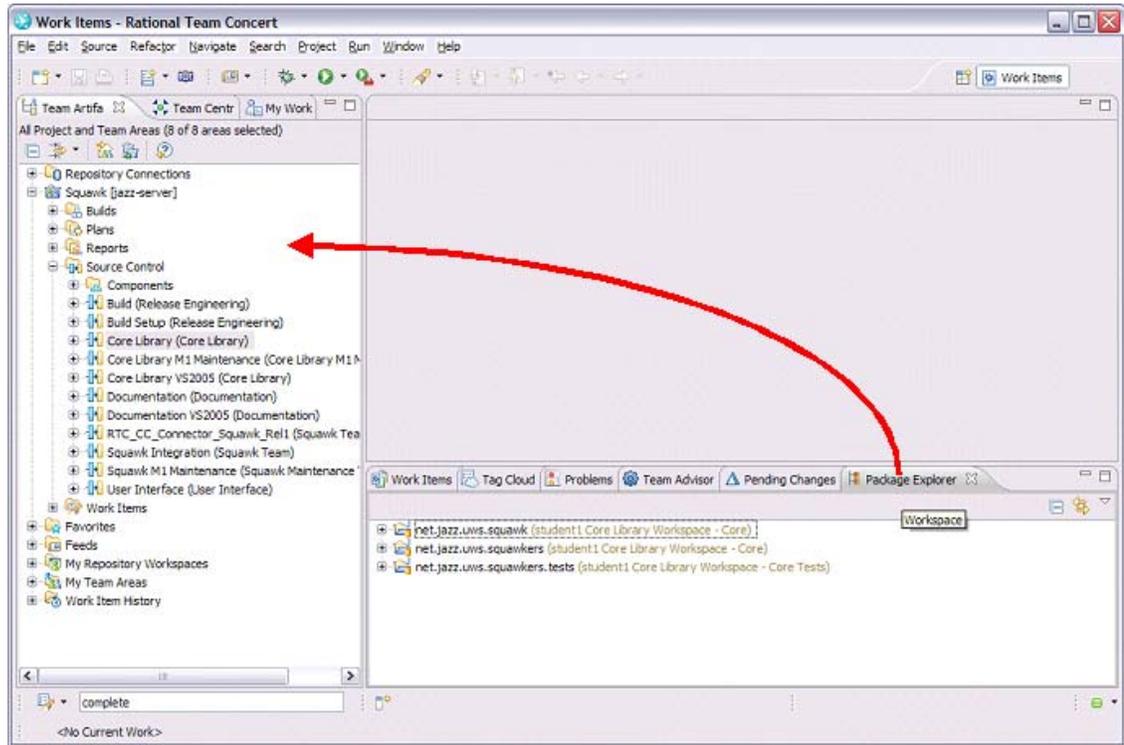
- \_\_\_h. Your repository workspace should now be visible under **My Repository Workspaces** in the **Team Artifacts** view.



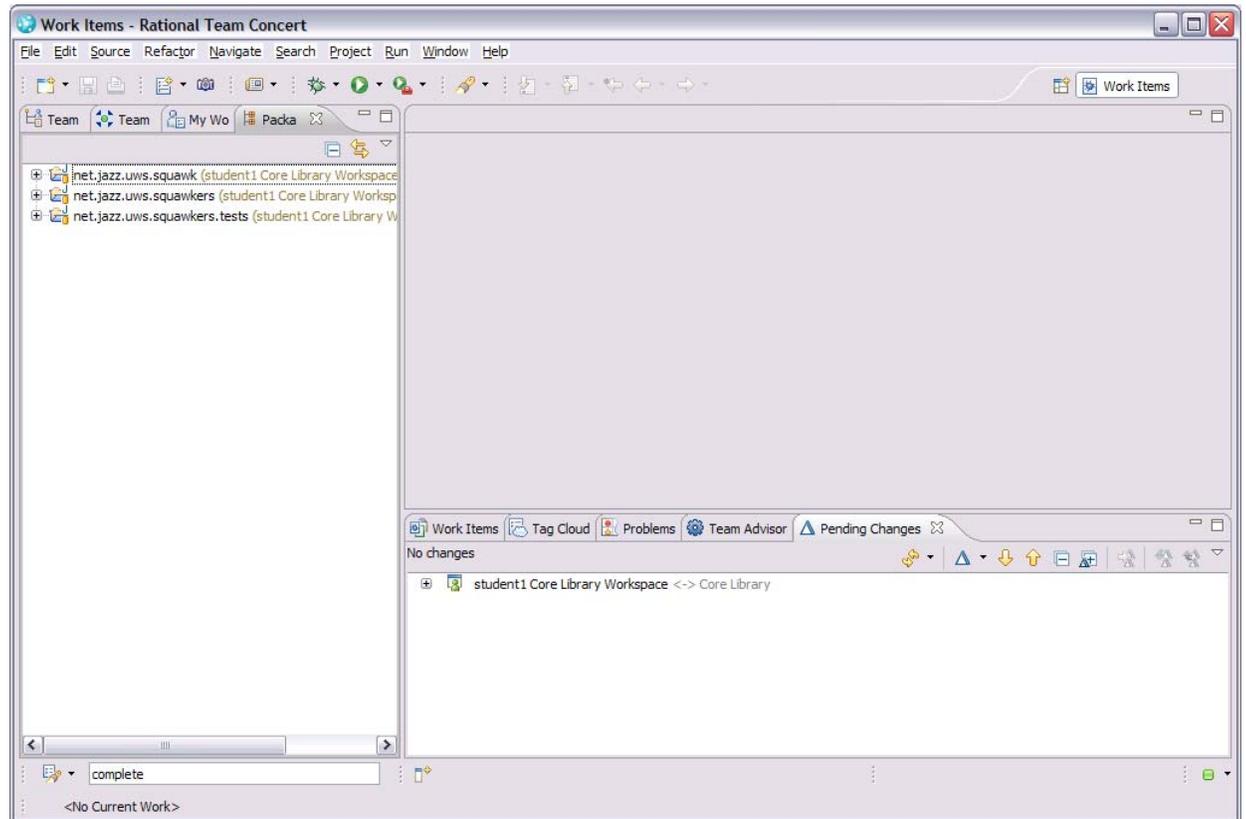
- \_\_\_i. The **Squawk** Java™ projects and their artifacts should be loaded in your local workspace and are visible in the **Package Explorer** view (**Window** → **Show View** → **Other** → **Java** → **Package Explorer**).



⌋. Drag and drop the **Package Explorer** view over the **Team Artifacts** view.



- \_\_k. You now have a loaded workspace and you are ready to begin work that has been assigned to you.



### Some Jazz terms reviewed



You use a personal **Repository Workspace** to work on project files under **Source Control**. You **Load** the repository workspace to copy the files and folders into your **local sandbox** (Eclipse workspace in this situation). Jazz tracks all changes made to source-controlled files with **Change sets**. Each change set itemizes the changes to one or more individual files or folders, carries a comment, and references the relevant work item motivating the changes. You **Check-in** your change sets to upload copies of the modified files from your **local workspace** to the **repository workspace**.

## 4.2 Editing your Squawker

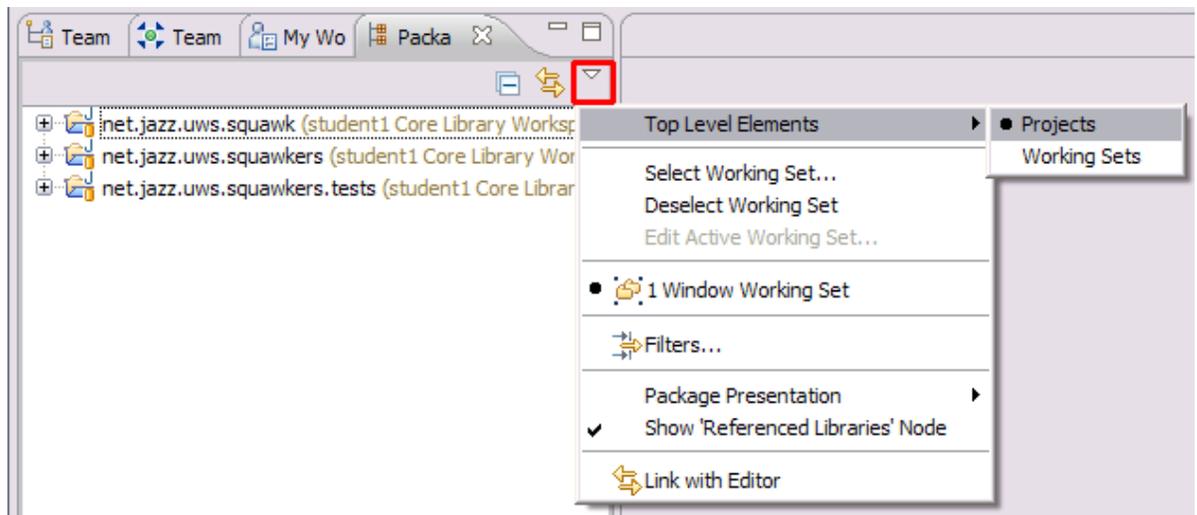


### Important!

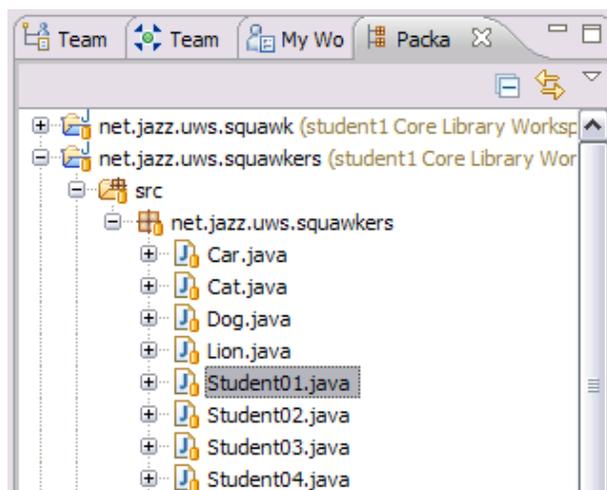
To ensure that the squawker class you edit does not conflict with a squawker that another member of the team, we have predefined a set of squawkers using a unique name based on the student ID.

\_\_1. Edit your squawker Java Class.

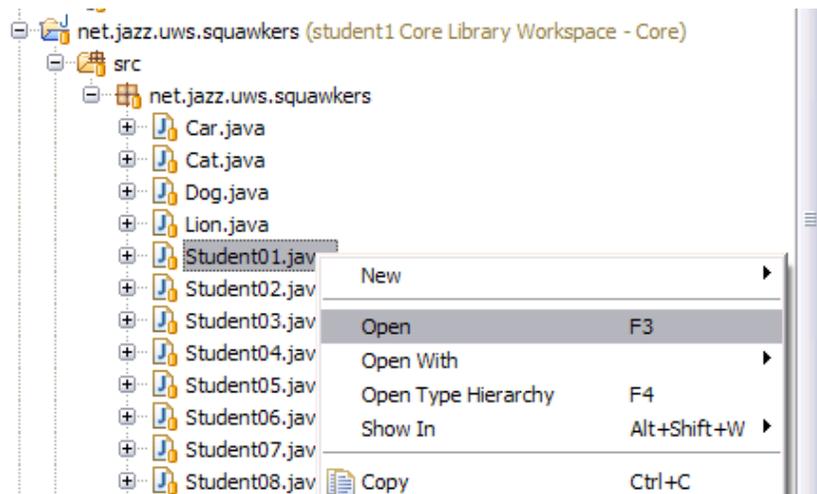
\_\_a. In the **Package Explorer** view, verify that the Package Explorer items are grouped by project by clicking the menu icon  and selecting **Top Level Elements** → **Projects**.



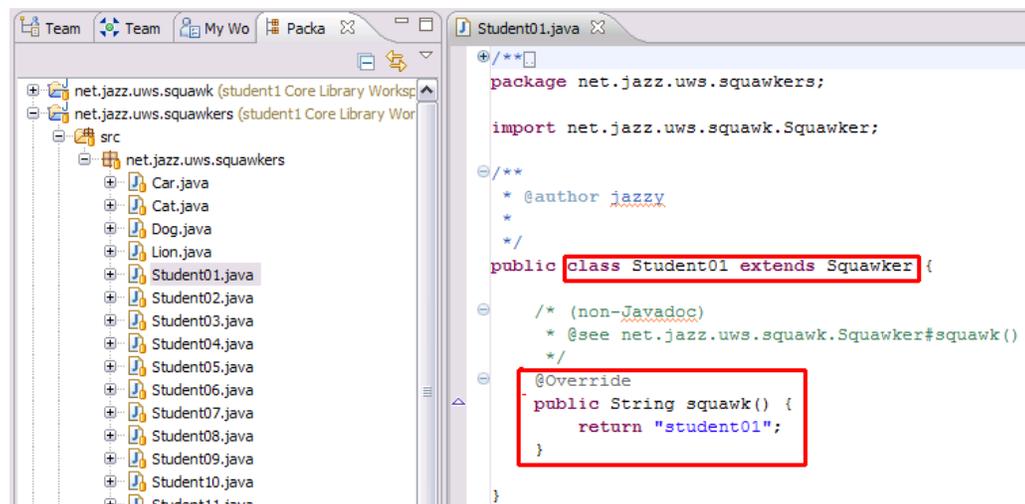
\_\_b. In the **Package Explorer** view, expand to the **net.jazz.uws.squawkers/src/net.jazz.uws.squawkers** Java package and select the **Student<N>.java** class file which you should replace <N> with your assigned id number.



c. Right-click the **Student<N>.java** class file and select **Open**



It will open the Java Editor.

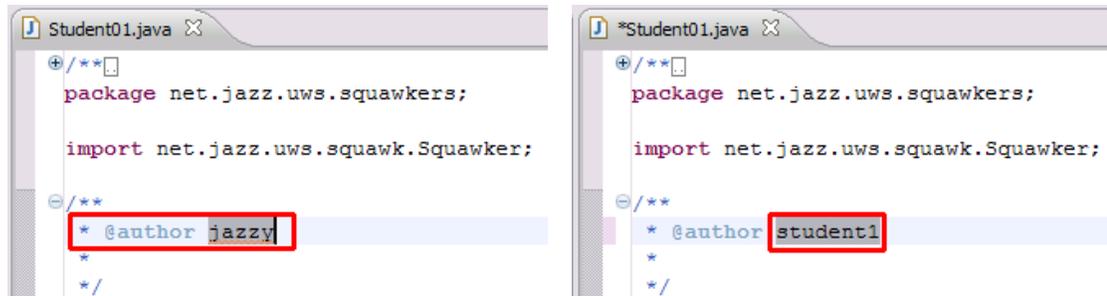


Observe that the class is a subclass of the `Squawker` class and returns a string by overriding the **squawk** method of the parent `Squawker` class.

- \_\_d. Replace the value in the return statement with a value of your choosing. Since this represents you, feel free to set it to something that might describe you (be creative!).

```
public class Student01 extends Squawker {
    /* (non-Javadoc)
     * @see net.jazz.uws.squawk.Squawker#squawk()
     */
    @Override
    public String squawk() {
        return "student01";
    }
}
```

- \_\_e. Optionally, change the Java class author from jazzy to **student<N>**.

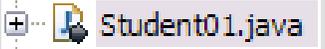


- \_\_f. Save the class. (Ctrl-S or )

**The Pending Changes view**

Rational Team Concert makes it easy for you to track what changes you need to commit by automatically showing them to you in the “**Pending Changes**” view which you can access through **Window → Show View → Other → Jazz Source Control → Pending Changes**. You will see in later exercises the usefulness of this view.

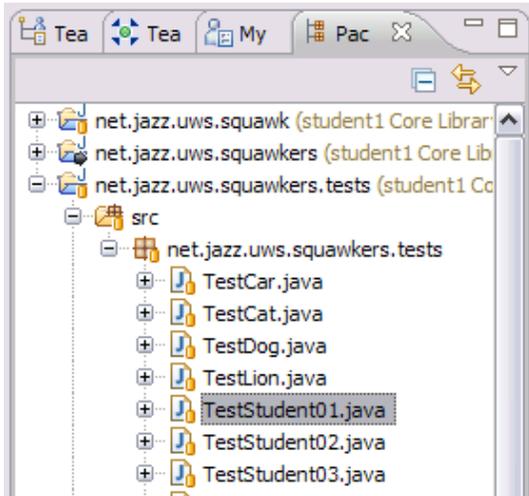
For now, notice the outgoing change icon () next to your new Java Class file.



The screenshot shows a list of files in the Pending Changes view. The file 'Student01.java' is listed with a small icon to its left that represents an outgoing change.

### 4.3 Edit the test case for your squawker

- \_\_1. In the Package Explorer view, expand to the **net.jazz.uws.squawkers.tests/src/net.jazz.uws.squawkers.tests** Java package and select the **TestStudent<N>.java** class file which you should replace **<N>** with your assigned id number.



- \_\_a. Double-click the **TestStudent<N>.java** file to open the Java Editor.

```

Student01.java  TestStudent01.java
+/**
package net.jazz.uws.squawkers.tests;

+import static org.junit.Assert.assertEquals;

-/**
 * @author jazzy
 *
 */
public class TestStudent01 {

-    /**
 * Test method for {@link net.jazz.uws.squawkers.Car#squawk()}.
 */
-    @Test
public void testSquawk() {
    Student01 squawker = new Student01();
    assertEquals("student01", squawker.squawk());
}

}

```

- \_\_b. Update the `testSquawk` method to verify that your squawk class returns the expected squawk text, by replacing lines 22-23 with the squawk value you set in your squawker class in Section 4.2, step 1 d. Also update the `@author` statement from jazzy to student<N>.

```

package net.jazz.uws.squawkers.tests;

import static org.junit.Assert.assertEquals;

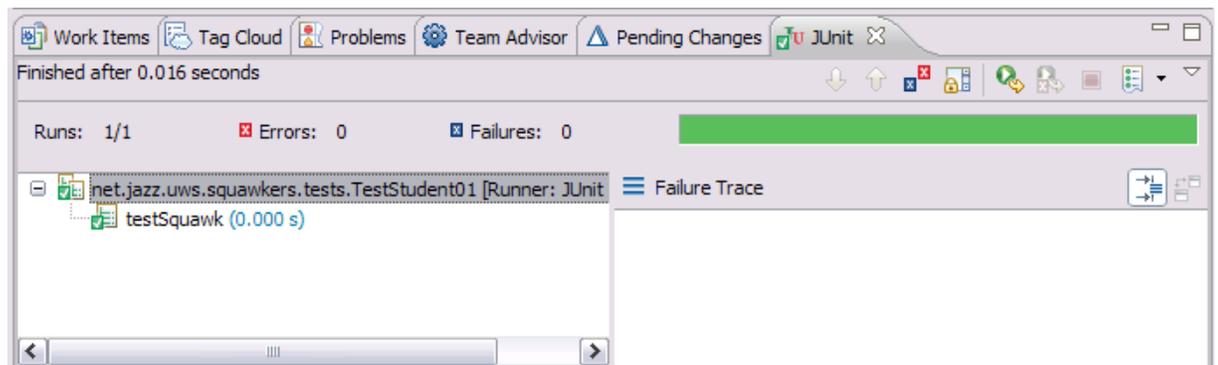
/**
 * @author student01
 */
public class TestStudent01 {

    /**
     * Test method for {@link net.jazz.uws.squawkers.Car#squawk()}.
     */
    @Test
    public void testSquawk() {
        Student01 squawker = new Student01();
        assertEquals("Roar", squawker.squawk());
    }
}
    
```

- \_\_c. Save the class (**Ctrl-S** or ).

\_\_2. Test your squawker class

- \_\_a. In the **Package Explorer** view, select the **TestStudent<N>.java** class, right-click and select **Run As → JUnit Test**.
- \_\_b. The **JUnit** view (**Window → Show View → Java → JUnit**) will show a green bar if the test is successful.



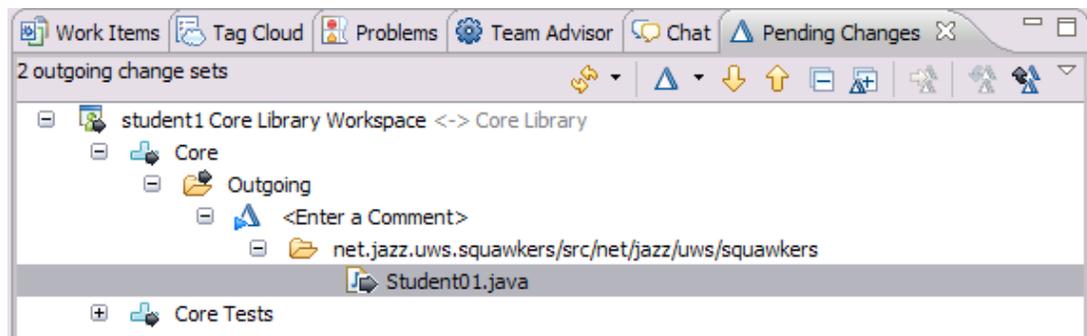


#### If your test fails

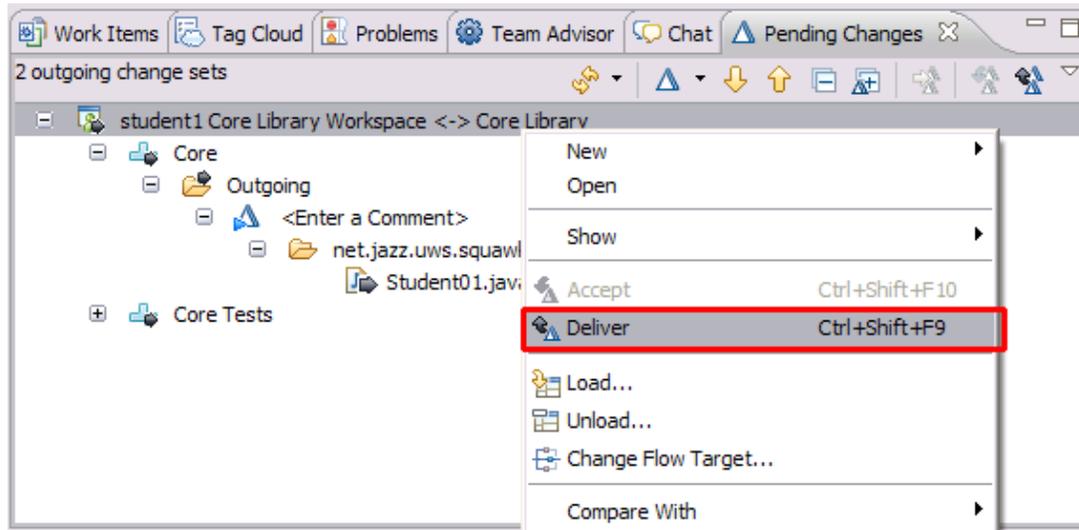
A primary cause for the JUnit test to fail is if the text specified in the `assertEquals` statement is not exactly the same as in the `squawk` method of the class under test.

## 4.4 Delivering your squawker code and resolving your implementation work item

- \_\_1. Deliver your squawker code to make it accessible to the rest of the team.
  - \_\_a. In the **Pending Changes** view (**Window** → **Show View** → **Other** → **Jazz Source Control** → **Pending Changes**), your repository workspace should show outgoing changes (indicated by the  icon) for the `Core` component (and the `Core Tests` component if you completed the previous section). Expand the tree under your repository workspace name and confirm that you have outgoing changes for your squawker class.

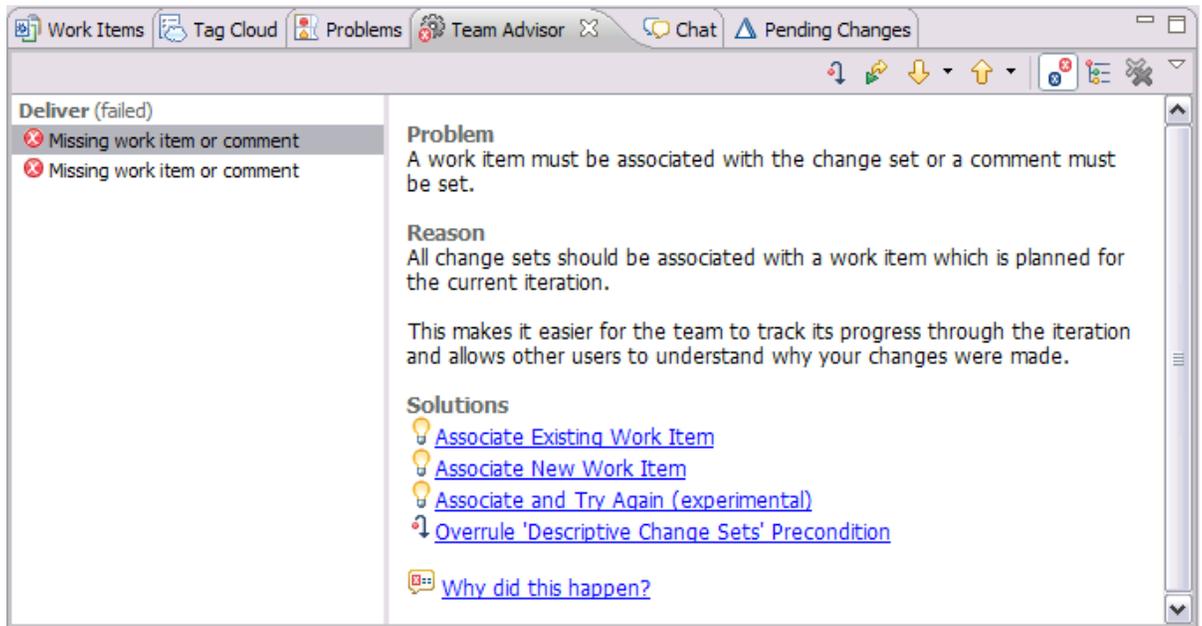


- \_\_b. Finally, right-click your Core Library repository workspace and select **Deliver** to send your changes to the Core Library stream.



- \_\_c. It didn't succeed, did it? What happened?

The **Team Advisor** view (**Window** → **Show View** → **Other** → **Team** → **Team Advisor**) displays the message: *Missing work item or comment*



- \_\_i. You can see that the **Team Advisor** view will help you associate an existing work item or a new work item with your delivery via one of the **Solutions** links.

### Process enactment and enforcement



Though it has not been discussed yet, when a project area is created a Jazz process template is required. When the **Squawk** project area was created, it was configured to follow the **Eclipse Way** process which requires a work item or a comment to be included during a deliver operation. It could be changed to require both or either one.

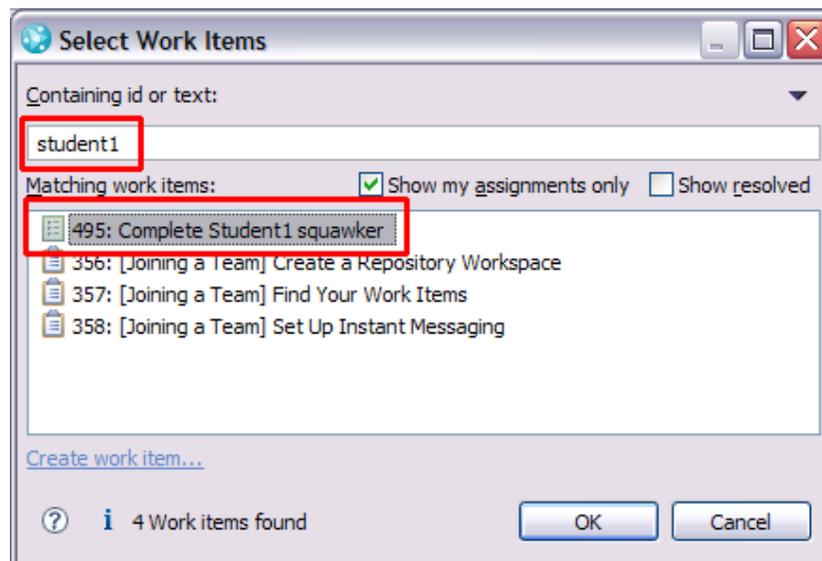
- \_\_ii. In the **Team Advisor** view, click the **Associate Existing Work Item** link.

### Finding work items



As you type characters into the **Containing id or text** field, the list of work items will change to display only those work items that have the typed characters in their **Summary**. Entering a single space into the **Containing id or text** field will list all the bugs that match the settings of the two checkboxes.

- \_\_iii. In the **Select Work Items** window:
- (1) Type your student id into the **Containing id or text** field.
  - (2) The work items assigned to you will appear in the list.
  - (3) Select your **Complete Student<N> squawker** work item and click **OK**.



- \_\_iv. Repeat the previous step for the second message in the **Team Advisor** view.

- \_\_d. In the **Pending Changes** view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), Right-click the **student1 Core Library** repository workspace and select **Deliver**. It should succeed this time. If it does fail for some reason, the **Team Advisor** view will appear again to remind you of the team’s rules.

i

**Other change set operations:**

**Suspending a change set** (  Suspend )

There might be times when you need to begin working on a new change set for a given set of items before you are finished with one that is in progress in your workspace. When in this situation, you can **suspend** the current change set, which removes it from your workspace but preserves it in the repository. The files in a suspended change set revert to the state they were in before the change set started, and the change set itself is moved to a special Suspended folder so that you can **resume** the work when you are ready.

**Discarding a change set** (  Discard... )

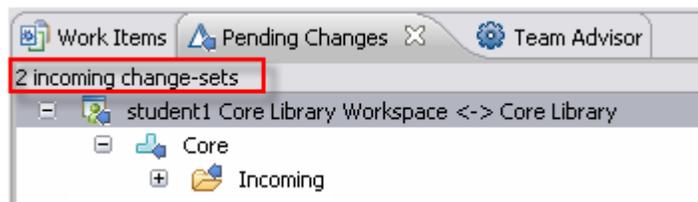
There are two basic scenarios for discarding a change set:

- If you have accepted an incoming change set but decide later that you don't want it in your workspace, you can discard it to undo the accept and return the change set to the component's Incoming folder.
- You can also discard a change set that you created but have not yet delivered. Discarded change sets of this type remain in the repository but are not placed in any special folder. To make it easier to retrieve a discarded change set that does not exist in any other stream or workspace, you can associate it with a work item before you discard it and then accept it from the work item later.

**Reversing a change set** (  Reverse )

If you want to undo the delivery of a change set, you can create a new change set that reverses all the changes in it and then deliver the reversed change set.

- \_\_e. The **Pending Changes** view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), should now show no outgoing changes or only incoming changes from other students.

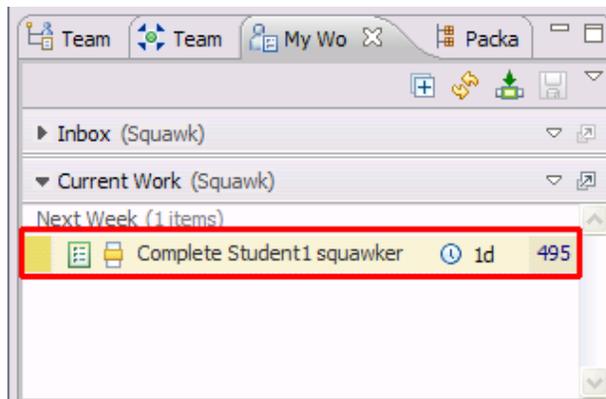


**Incoming change sets** (📁)

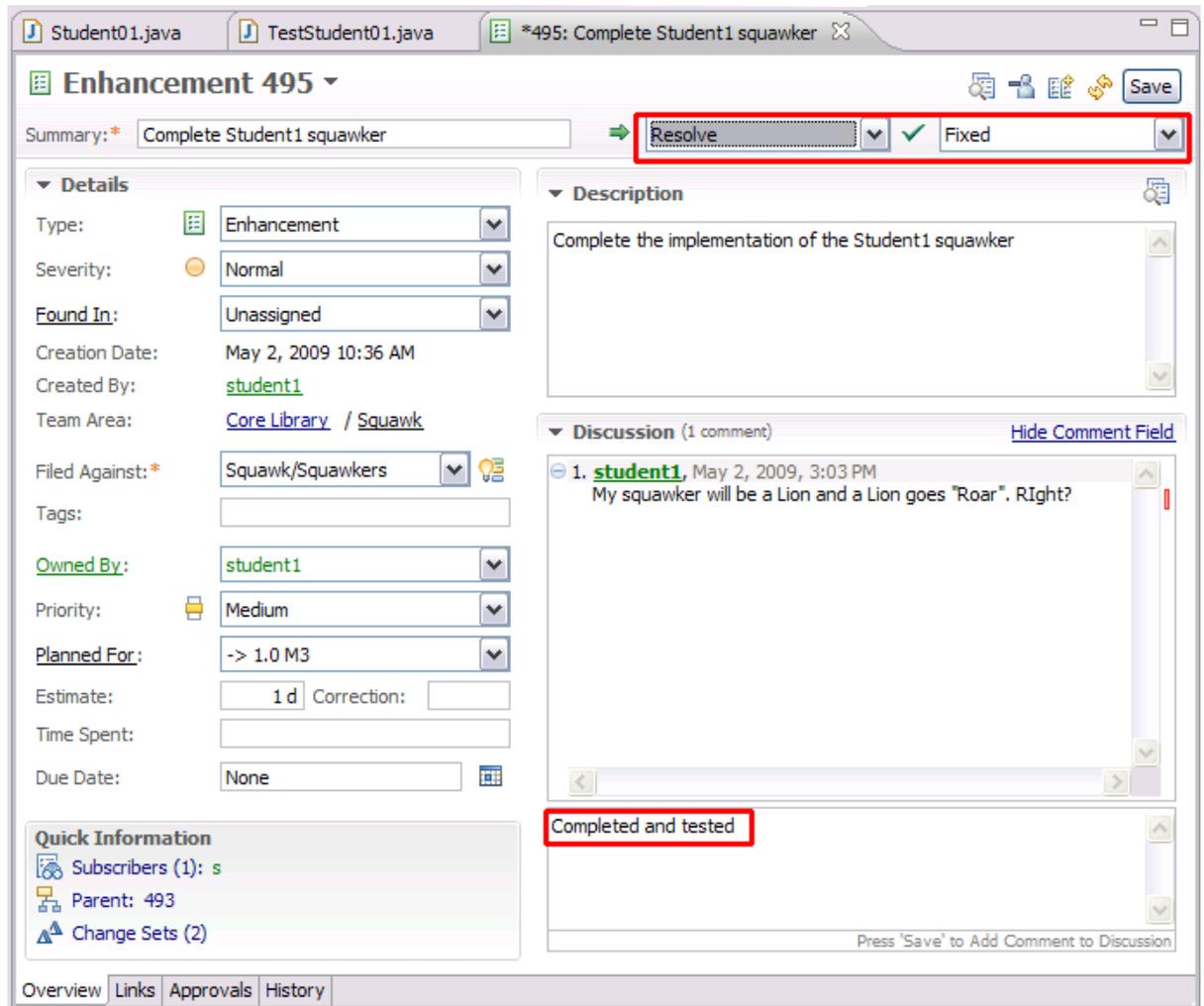
As other teammates deliver their new squawkers to the **Core Library** stream, these changes are made available to you as incoming changes. Later on in this lab you will accept all incoming changes.

\_\_2. Resolve your squawker implementation Work Item.

\_\_a. In the **My Work** view (**Window** → **Show View** → **Other** → **Work Items** → **My Work**), double-click your **Complete Student<N> squawker** work item.



- \_\_b. In the **Overview** tab change the Work Item state to **Resolved/Fixed**. Optionally, add a comment (you will need to click the **Add Comment/Hide Comment Field** link to add the comment).



- \_\_c. Save (  ) the Work Item.



**Congratulations!**

You have successfully implemented, and delivered your own squawker class.

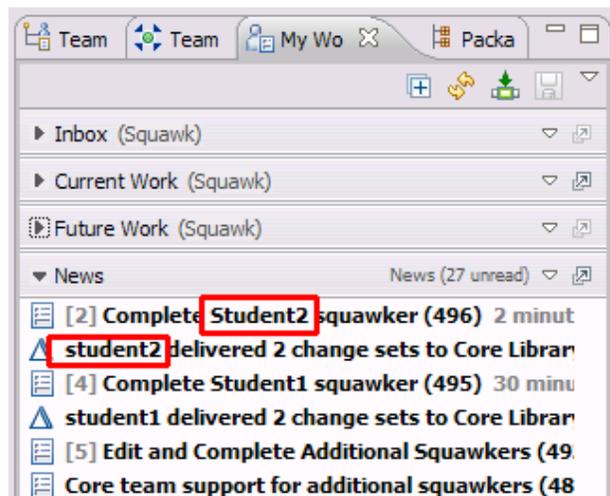
## 4.5 Accepting changes from other members of your team

### \_\_1. Accept your teammates' changes

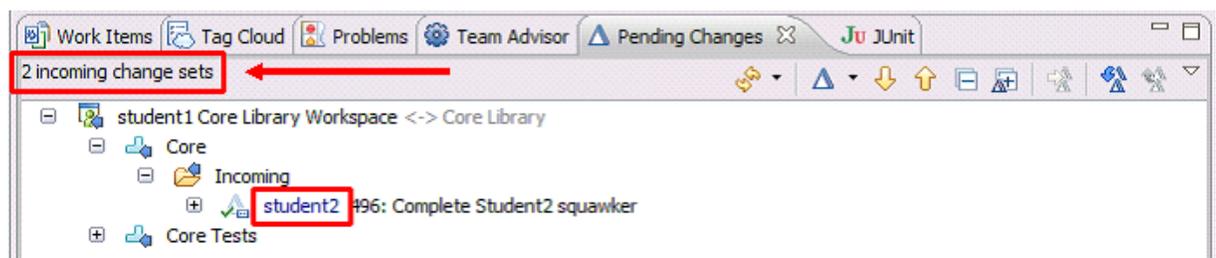
- \_\_a. You have not been the only one working on the Core Library components. Your teammates have been creating and delivering their squawkers to the streams also. These changes should be showing up as notifications in real time in the bottom right of the Team Concert window, similar to this:



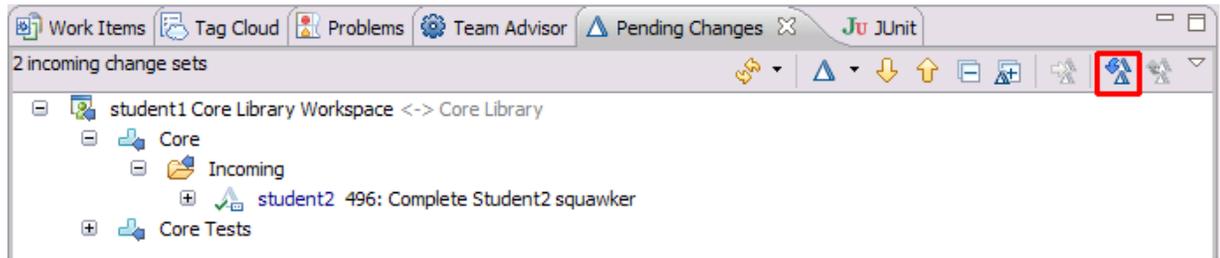
- \_\_b. You can also see these notifications in the **Event Log** section of the **Team Central** view (**Window → Show View → Other → Collaboration → Team Central**) as discussed in Module 3, *Keeping Track of All Our work*. Review Work Items from other students.



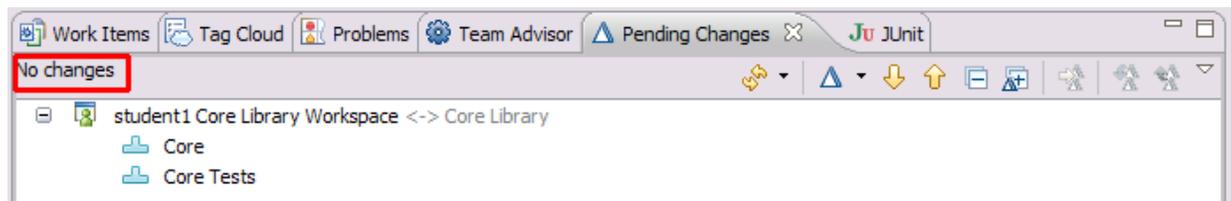
- \_\_c. Back to the **Pending Changes** view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), verify that there is a number of incoming changes. Expand the folders to locate the changes. In the following screen snapshot, at least one of the incoming changes was introduced by student2.



- \_\_\_d. In the **Pending Changes** view, click **Accept All Incoming Baselines and Change sets** (  ) icon to load changes from other teammates into your Repository Workspace(s).



- \_\_\_e. Your Repository Workspace(s) are now up to date with the Core Library stream.



- \_\_\_f. You may need to do this repeatedly as each of your teammates completes his/her work.

### Conclusion



Congratulations, you have just passed Jazz source control 101. You have made changes, delivered them to your teammates on the streams, and picked up their changes as well. This is the mainline path for performing source control in Jazz. Later on, you will experience other situations like code conflicts and work interruptions. Also, notice once again how the **Pending Changes** view is central to how you interact with Jazz Source Control

## Lab 5 Remembering Well Known SCM Configurations



### Demo Scenario

Thus far, you have contributed your own squawker class and delivered your work so that other people can use it.

You have not been working in isolation. Your teammates have been creating and delivering their own squawkers. As part of the last exercise, you accepted these changes from other members of your team bringing your code up-to-date with everyone else.

The instructor will now play the role of the Team Lead, creating baselines and snapshots, which can be retrieved and used at any point in the future.



### Lab Scenario

After the instructor creates baselines and snapshots, you will explore these new objects by querying their contents.



### Definition of a Component Baseline

A baseline is a repository object that provides an immutable record of the configuration of a component in a particular workspace or stream. Baselines are fixed points of reference, and are useful both for initializing streams and workspaces, and for sharing the changes to components at a granularity coarser than that of their individual change-sets. Baselines are also useful for auditing and reverting a component to an earlier configuration,

### Definition of a Snapshot

A snapshot provides a persistent record of the configuration of a workspace, expressed as a collection of component baselines. Snapshots are useful for capturing the state of a workspace, and are typically used to record important workspace configurations so that they can be recreated.

## 5.1 Instructor Demo - Creating baselines for the Core Library Components



### **Instructor Demo**

Section 5.1 is performed by the instructor as a demo. In this section, the instructor will show how a baseline can be created for a component in order to easily remember its current configuration.

Refer to the corresponding section in the workbook Appendix B.

## 5.2 Instructor Demo - Taking a snapshot of the Core Library Repository Workspace



### **Instructor Demo**

Section 5.2 is performed by the instructor as a demo. In this section, the instructor will show how a snapshot of a repository workspace can be created to easily remember the composite configuration of components in a workspace.

Refer to the corresponding section in the workbook Appendix B.

## 5.3 Instructor Demo - Promote the snapshot to the appropriate streams



### **Instructor Demo**

Section 5.3 is performed by the instructor as a demo. In this section, the instructor makes the previously created snapshot available to the rest of the team.

Refer to the corresponding section in the workbook Appendix B.

## 5.4 Accept all Incoming Baselines and Change-sets



### Important!

Sections 5.4 and 5.5 are to be performed by the student.

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote student<N>, which you should replace <N> with your assigned id number. Sample screenshots in all labs will use the id student1. If you are unsure please check with the instructor.



### Team role

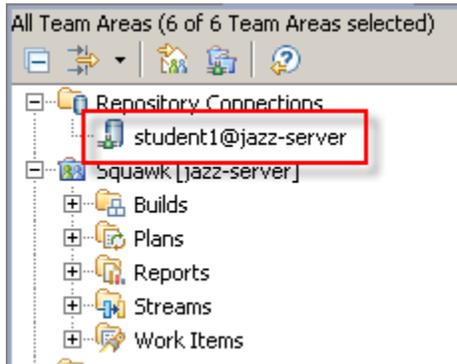
You are now performing the role of **student<N>**, a Developer.

\_\_1. If you don't already have Rational Team Concert running, do the following:

- \_\_a. Open Rational Team Concert by double clicking the Team Concert shortcut  on the Windows Desktop.
- \_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` (replacing <N> with your id number). Click **OK**.

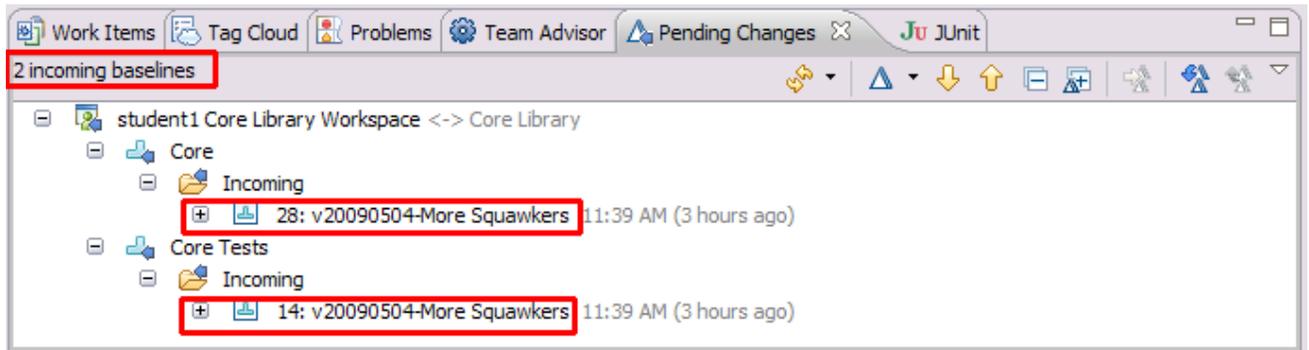


- \_\_2. In the Team Artifacts view (Window → Show View → Team Artifacts), ensure that you are connected to the Squawk Project area as student<N>.

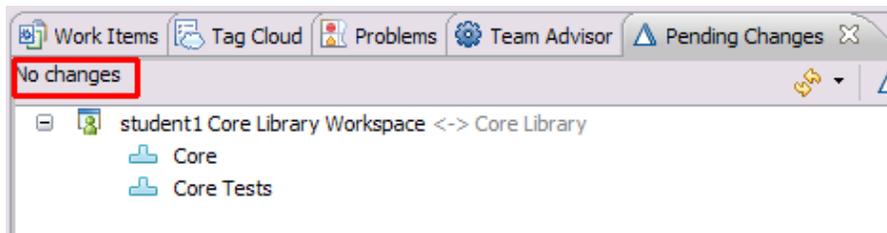


- \_\_3. Refresh the **Pending Changes** view (Window → Show View → Other → Jazz Source Control → Pending Changes) to synchronize with latest changes from the instructor. Use the  icon to **Refresh Remotes Changes**.

- \_\_4. The component baselines and snapshots created by the instructor during the demo are available as incoming changes to your Core Library Repository Workspace. In the **Pending Changes** view, use the  icon to **Accept all Incoming Baselines and Change-sets**.

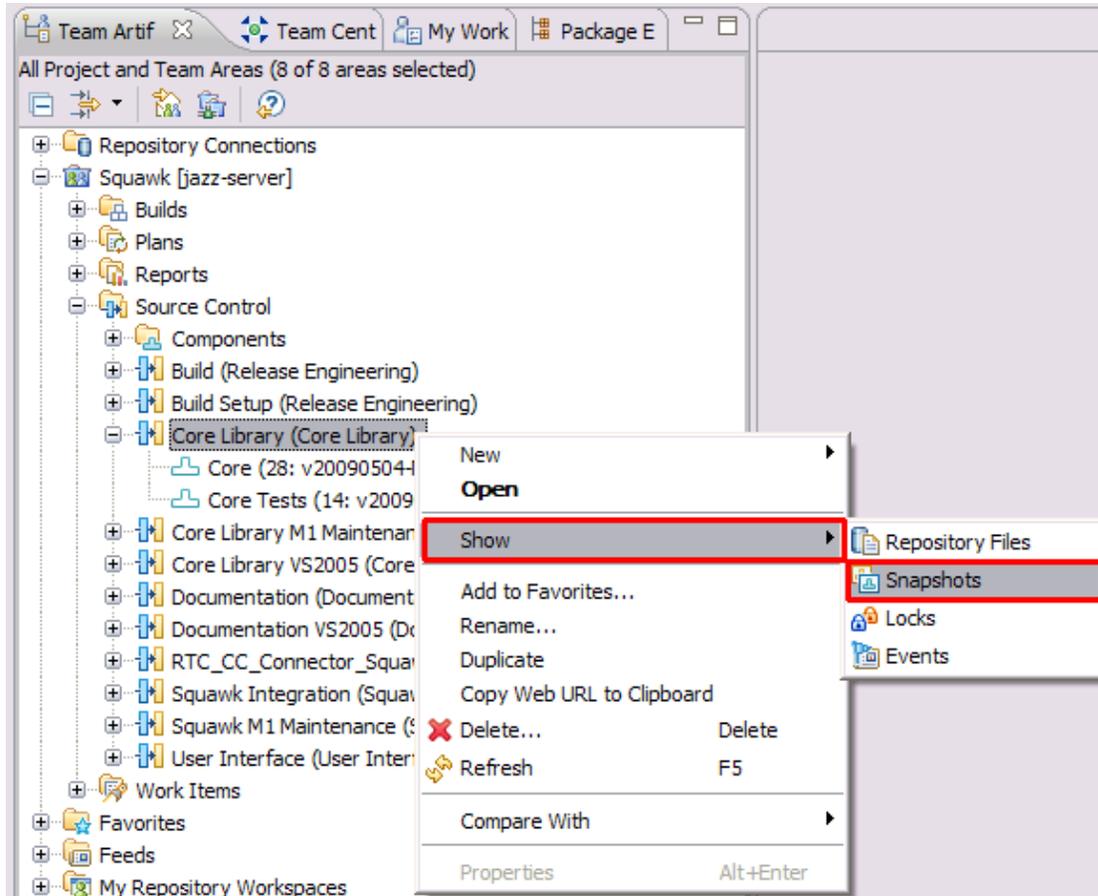


- \_\_5. Your Repository Workspaces and default flow Streams have a matching configuration again.

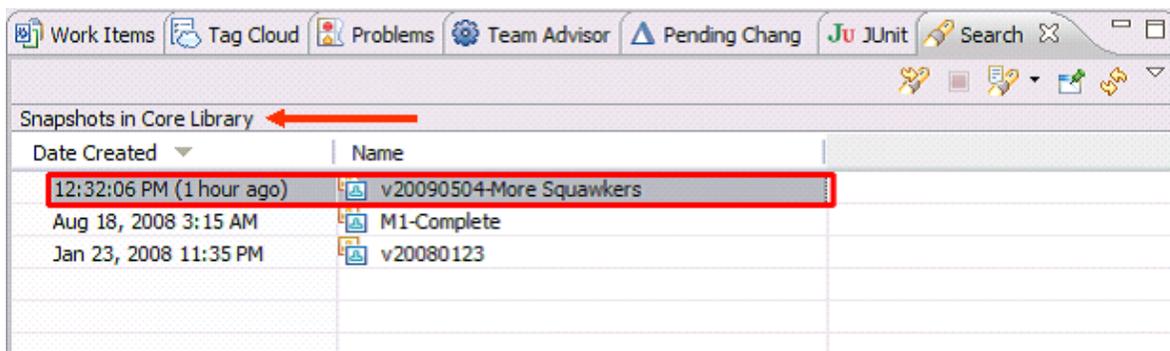


## 5.5 Explore the newly created snapshots

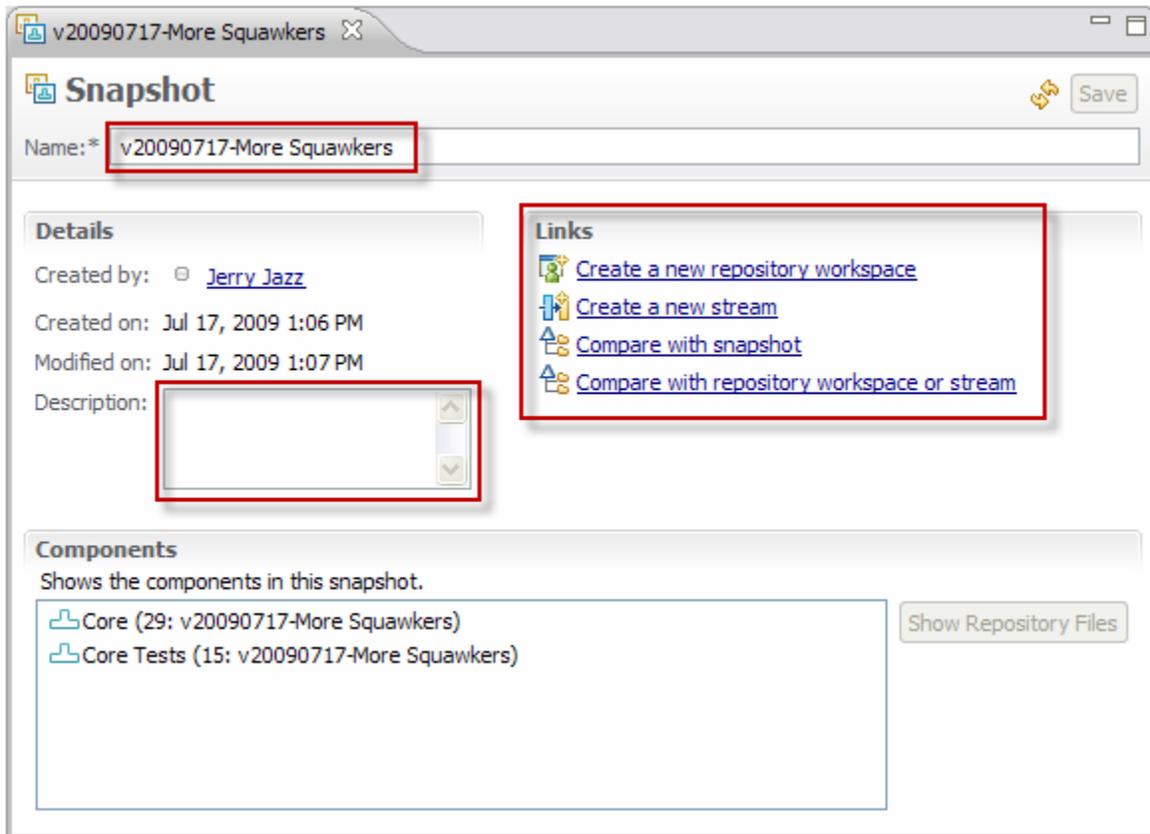
- \_\_1. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), expand the **Squawk** project area tree to find the **Core Library** stream, right-click it and select **Show Snapshots**.



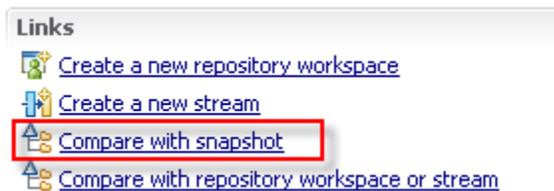
- \_\_2. The **Search** view opens. In the **Search** view, double click the most recently created snapshot.



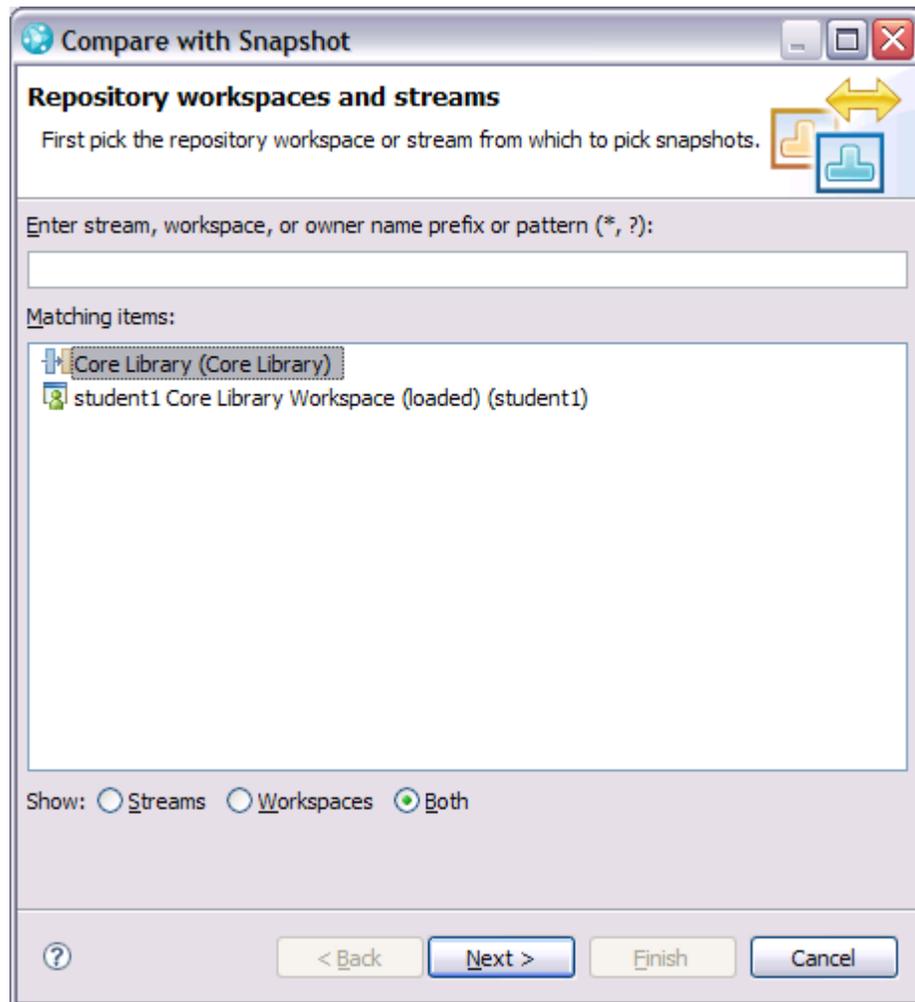
- \_\_\_3. In the editor that opens note that only the *name* and *description* of the snapshot can be changed at this point. Also take note of the links that are available to **create** a new stream or repository workspace from the snapshot and to **compare** the snapshot with another snapshot, stream or workspace.



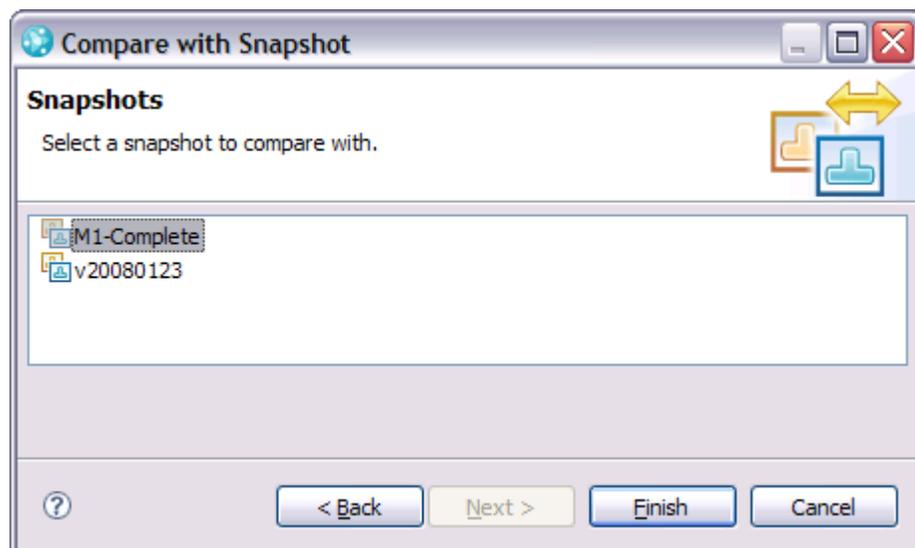
- \_\_\_4. Click the **Compare with snapshot** link to compare the snapshot just created by the instructor with the snapshot that represents the configuration that you started with to create your own Squawker.



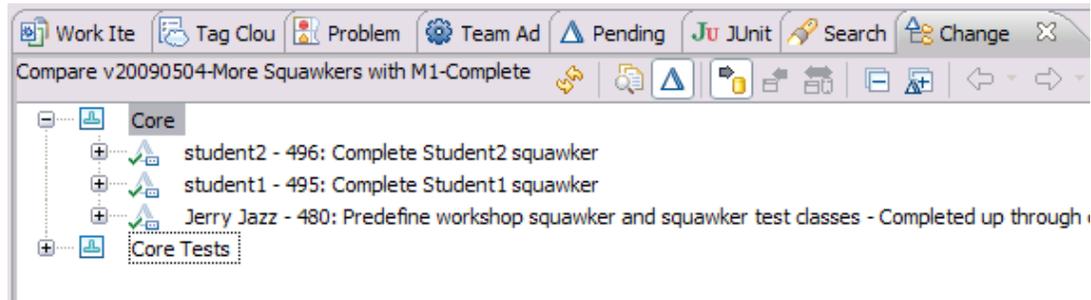
- \_\_a. Select the **Core Library** stream to filter the snapshot list to display. Click **Next**.



- \_\_b. Select the snapshot **M1-Complete**. Click **Finish**.



- c. The **Change Explorer** view opens. In the **Change Explorer** view, the differences between the two snapshots are listed as Work Items (  ) or change-sets (  ).



- 5. Close the snapshot editor.



### Conclusion



Baselines and snapshots are repository objects used to capture a set of related code/components for later reference. They are created in the context of repository workspace and then delivered (baselines) or promoted (snapshots) to streams where everyone else will be able to use them.

It is easy to create a new repository workspace or stream from a baseline or snapshot. These are useful for maintenance purposes, fixing builds or forking the code. Baselines and snapshots are also important for auditing and reporting purposes.

Finally, note how the Pending Changes view is central to these operations.

---

## Lab 6 User's View of Build

In this module you will observe and perform two types of Team Concert builds using the workshop Squawk application. Assembling any significant application requires the use of a good build process. Builds are very visible to you, and you can easily fire off builds as part of your everyday development activities, and see how build results are reported back to you.

We will cover the concepts of:

- starting the Team Concert Build Engine
- requesting a build
- exploring an existing build
- requesting a private build



### Lab Scenario

Now you have developed a squawker class, performed some unit tests and created baselines and snapshots. You are now ready to build your application with help of the Team Concert Build Engine.

Your project team will then have a built application you should all be happy with.

## 6.1 Instructor Demo - Start the Team Concert Build Engine



### Instructor Demo

Section 6.1 is performed by the instructor as a demo. In this section, the instructor will start the build engine on one or more workstations.

Refer to the corresponding section in the workbook Appendix C.

## 6.2 Instructor Demo - Requesting a Build



### Instructor Demo

Section 6.2 is performed by the instructor as a demo. In this section, the instructor will show how to initiate a build for the team.

Refer to the corresponding section in the workbook Appendix C.

## 6.3 Exploring an Existing Build



### Important!

Section 6.3 is to be performed by the student.

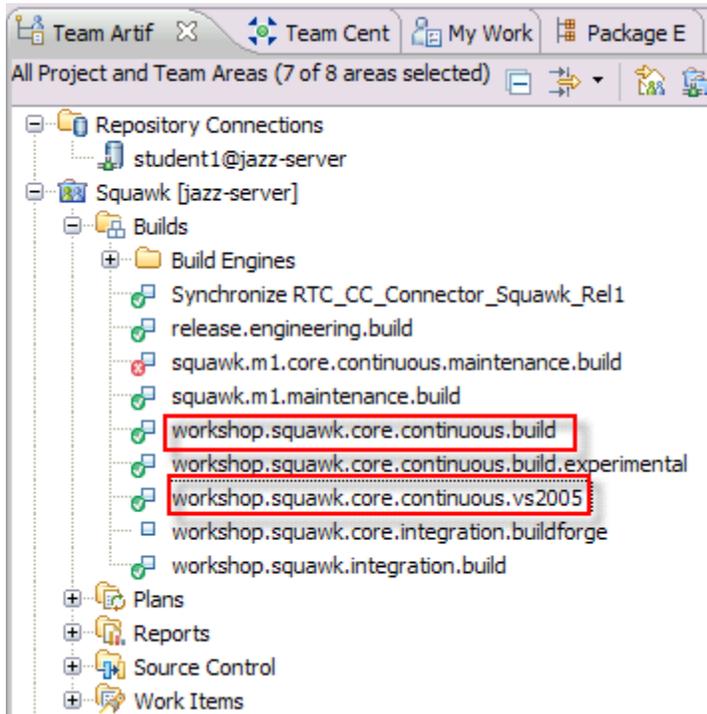
Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.

You joined the Squawker project at its milestone M3. That usually means that some builds have already been performed. You will list those builds and compare some of them.

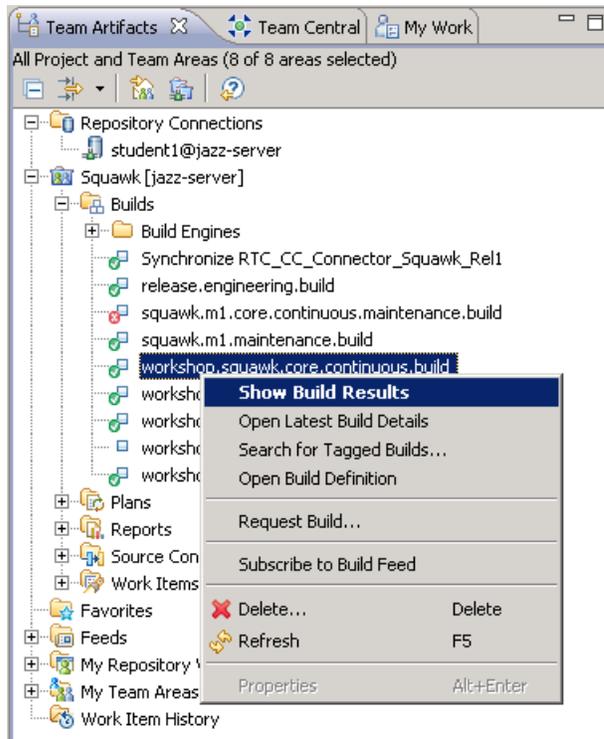
- \_\_\_1. If not already started, open Team Concert by double-clicking the Team Concert shortcut on the Windows Desktop.



- \_\_2. Open the **Team Artifacts** view and navigate to **Squawk → Builds**. The various build definitions are listed. The two that are highlighted will be used in the exercise depending on which Team Concert client you are using, Eclipse or Visual Studio.



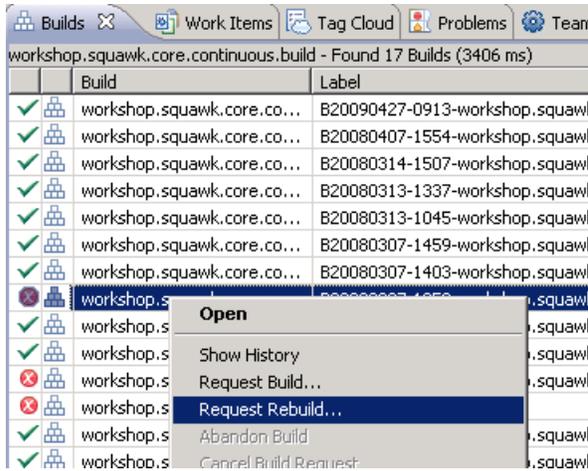
- 3. Expand **Builds** under the Squawk project and select build definition *workshop.squawk.core.continuous.build*. Right click and in the menu dialog **Show Build Results**.



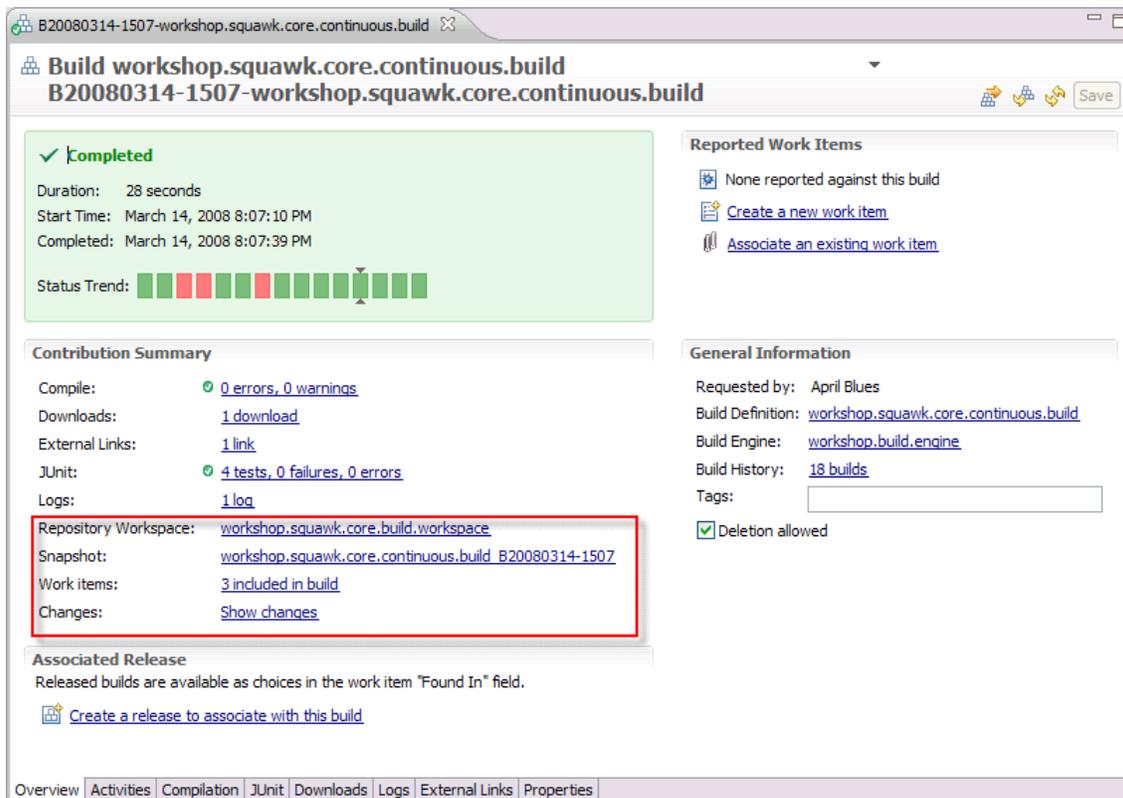
- 4. Examine some build results in the **Builds** view.

Build	Label	Progress
workshop.squawk.core.co...	B20090427-0913-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080407-1554-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080314-1507-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080313-1337-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080313-1045-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080307-1459-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080307-1403-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080307-1352-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080303-1057-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080303-1048-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080303-1042-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	20080303-1017	Completed
workshop.squawk.core.co...	B20080201-1707-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080128-1359-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080123-1623-workshop.squawk.core.continuous.build	Completed
workshop.squawk.core.co...	B20080123-1621-workshop.squawk.core.continuous.build	Completed

- \_\_\_5. Right-click any failed build result and note the option **Request Rebuild**. Besides the obvious value of requesting a rebuild of a failed build, you can also request a rebuild of a successful build. This can be useful to quickly submit a new build using the same options and configuration as the existing build.

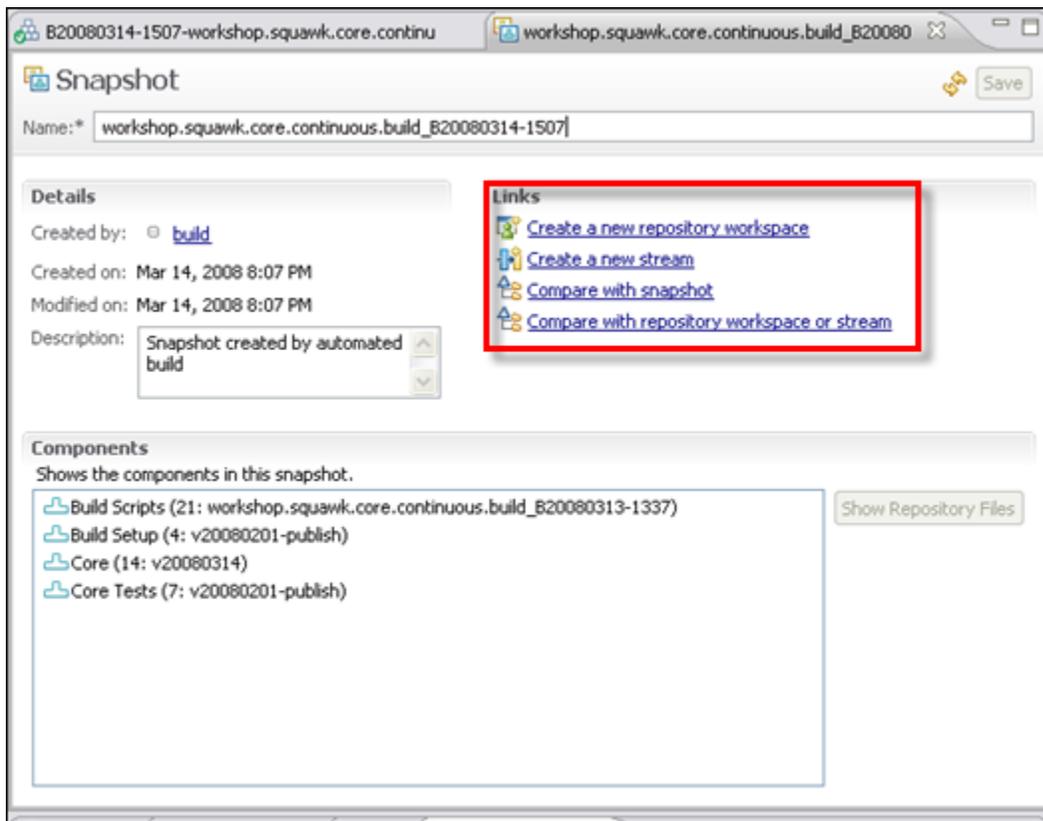


- \_\_\_6. Open a build result (double-click), for example *workshop.squawk.core.continuous.build* with label *B20080314-1507-workshop.squawk.core.continuous.build* (scroll down in the view as needed), and review the **Contribution Summary** section of the **Overview** page. Select and open the highlighted items like **Repository Workspace**, **Changes** and **Work Items**.



- \_\_\_7. One of the items in the **Overview** page is the Snapshot of the build repository workspace that was created after incoming changes were accepted (a snapshot is not available if there were no changes to the components since the previous build). Note that the snapshot is a composite set of baselines belonging to a repository workspace or stream and represents their state at a point in time.

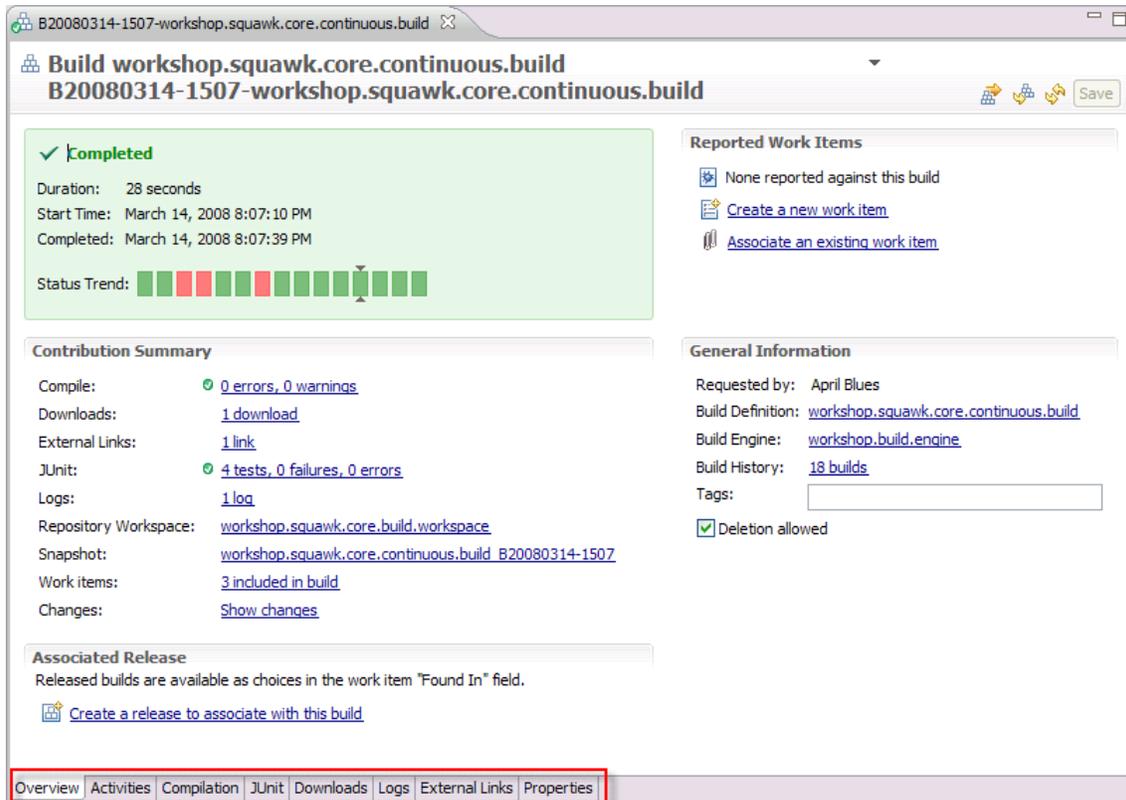
Open the snapshot of the build result by clicking **workshop.squawk.core.continuous.build\_B20080314-1507**. Examine the **Links** section.



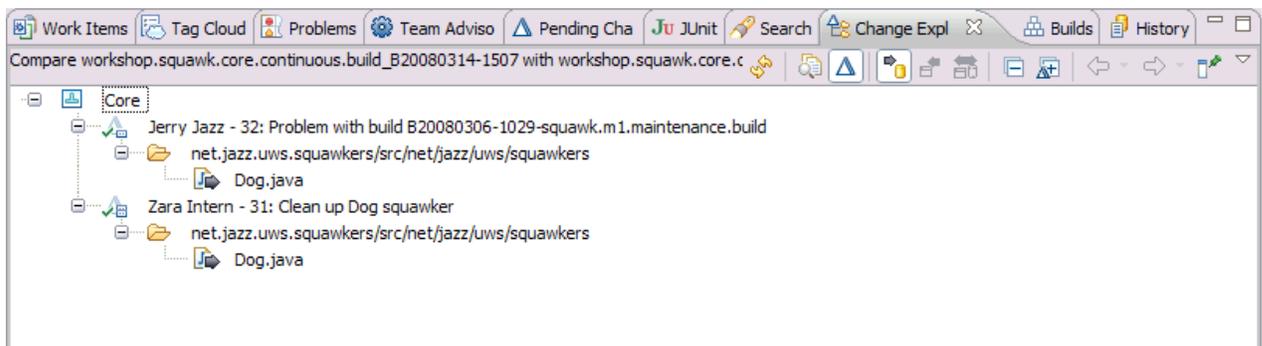
Here you can create a new repository workspace which is useful when a build fails and you want to recreate precisely the same content that the build used.

Creating a new stream from a successful build allows you to branch the code base at a known state for activities such as creating a maintenance stream or do some long range prototyping.

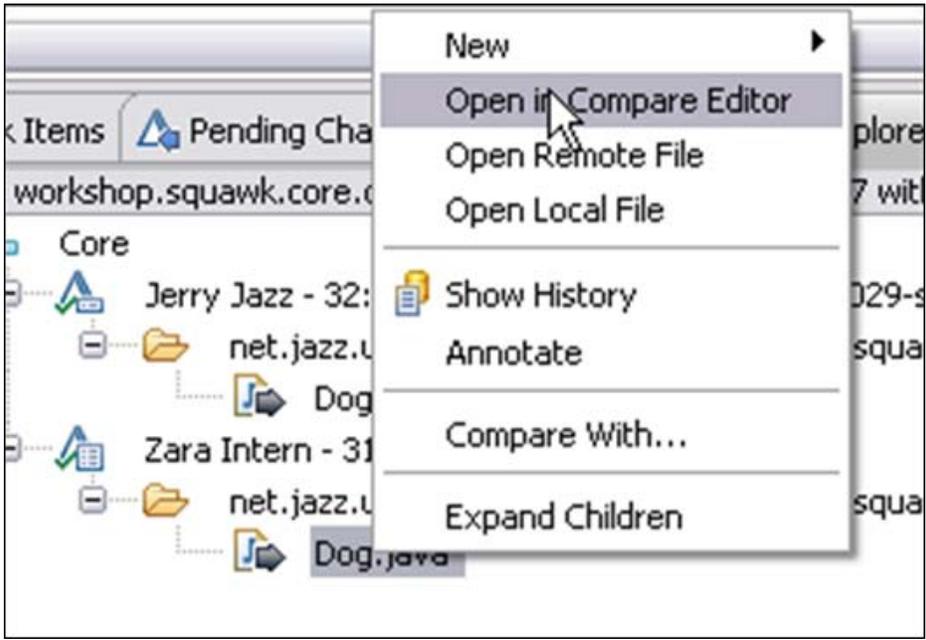
- \_\_8. Examine the other pages in the build result editor, like **Activities**, **Compilation**, **JUnit** etc. Additional pages can be defined to meet special requirements via an Eclipse extension on the server.



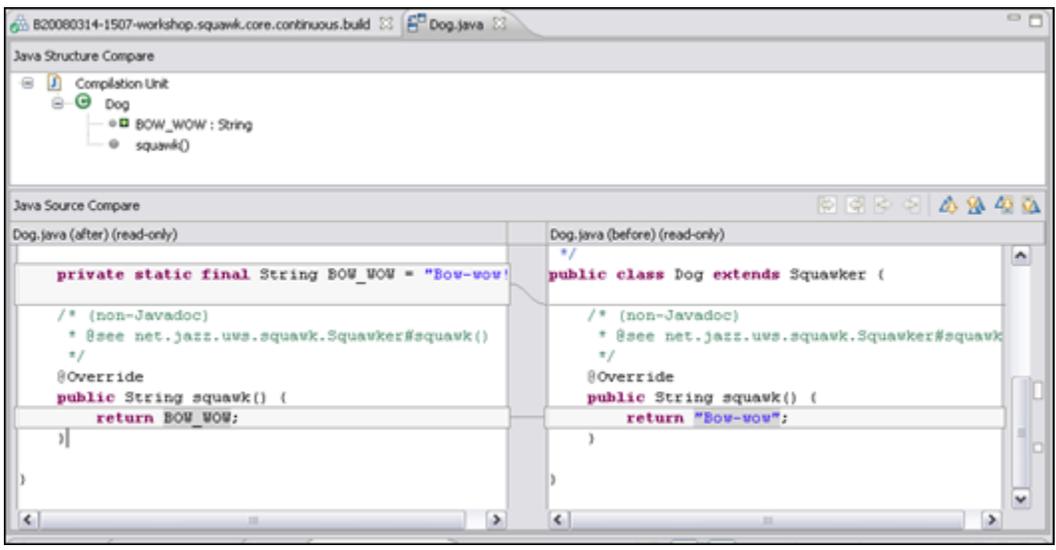
- \_\_9. In the **Builds** view, compare two adjacent builds. While pressing the **Ctrl** key, select the two continuous builds *B20080314-1507-workshop.squawk.core.continuous.build* and *B20080313-1337-workshop.squawk.core.continuous.build* from the **Builds** view. Right-click and compare them using the **Compare Builds** context menu action. The result is a list of change sets that differ between the two builds and appears in the **Change Explorer** view.



- \_\_10. Expand Zara Intern's change-set then right-click *Dog.java* and select **Open in Compare Editor**.



- \_\_11. Review the changes that were implemented by Zara Intern to fix Defect 31 - *Clean up Dog squawker*.



Note that you can also compare two non-adjacent builds (even between two milestones) and see all the changes between them. You can also compare unlike builds, for example continuous and integration builds.

## 6.4 Requesting a Personal Build

In this section, you will learn what a personal build is and how to request it.

### Important!

Section 6.4 is to be performed by the student.



Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.



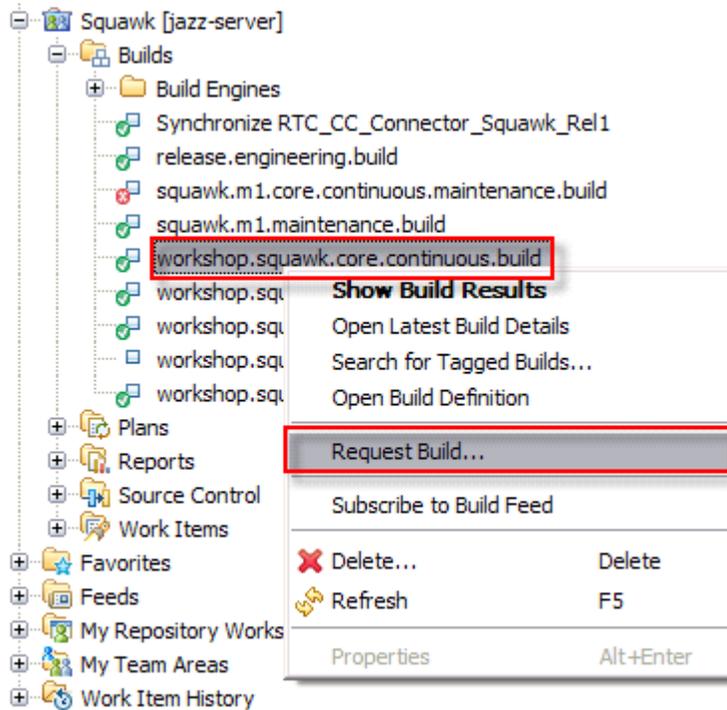
### Team role

You are now performing the role of **student<N>**, a Developer.

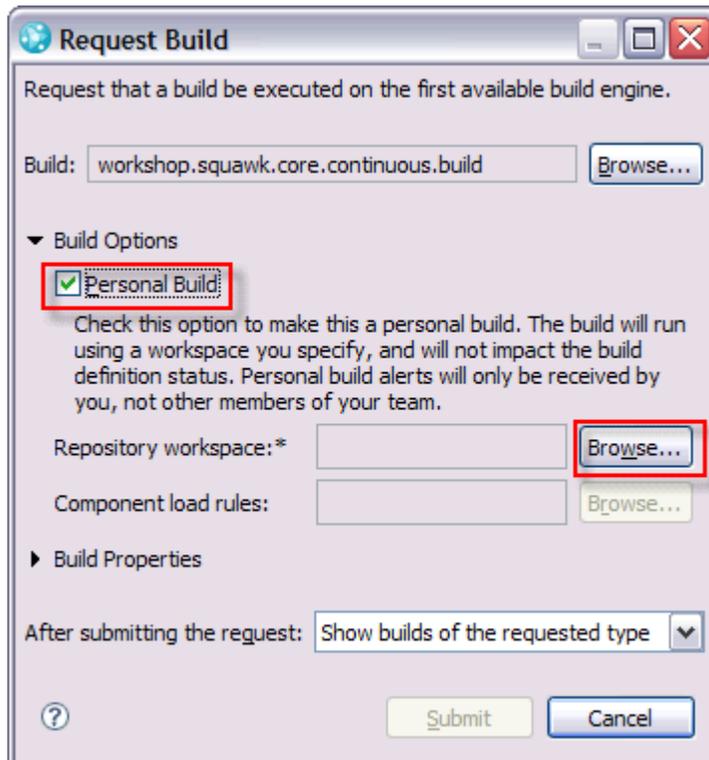
Private build allows you to build your changes before delivering them to the stream. This can provide you with some assurance that your changes will not disrupt the team when you deliver them. A private build request simply uses your repository workspace in place of the workspace assigned to the *build* user (defined with user id *build* in this repository).

- \_\_\_1. Request a private build as follows. Note: Since this is a private build you do not deliver your changes to the stream.

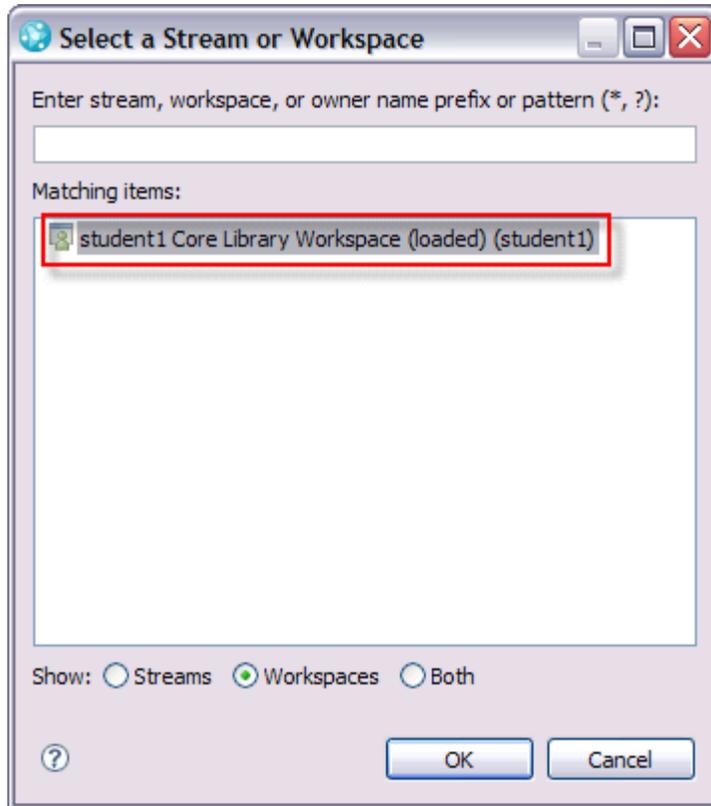
- a. In the **Team Artifacts** view, navigate to **Squawk -> Builds**. Right-click build definition **workshop.squawk.core.continuous.build** then select **Request build**.



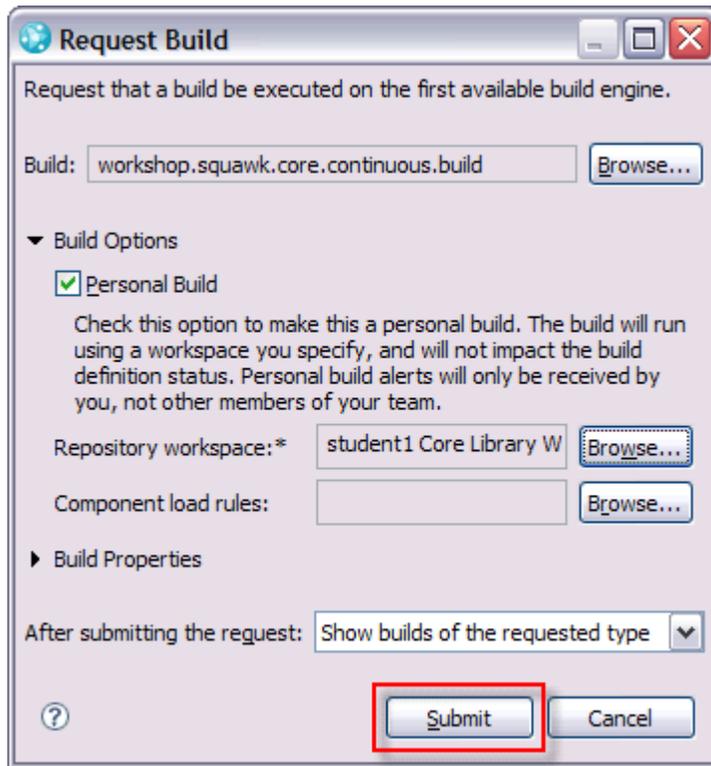
- b. When the **Request Build** dialog appears expand the **Build Options** and select **Personal Build**.



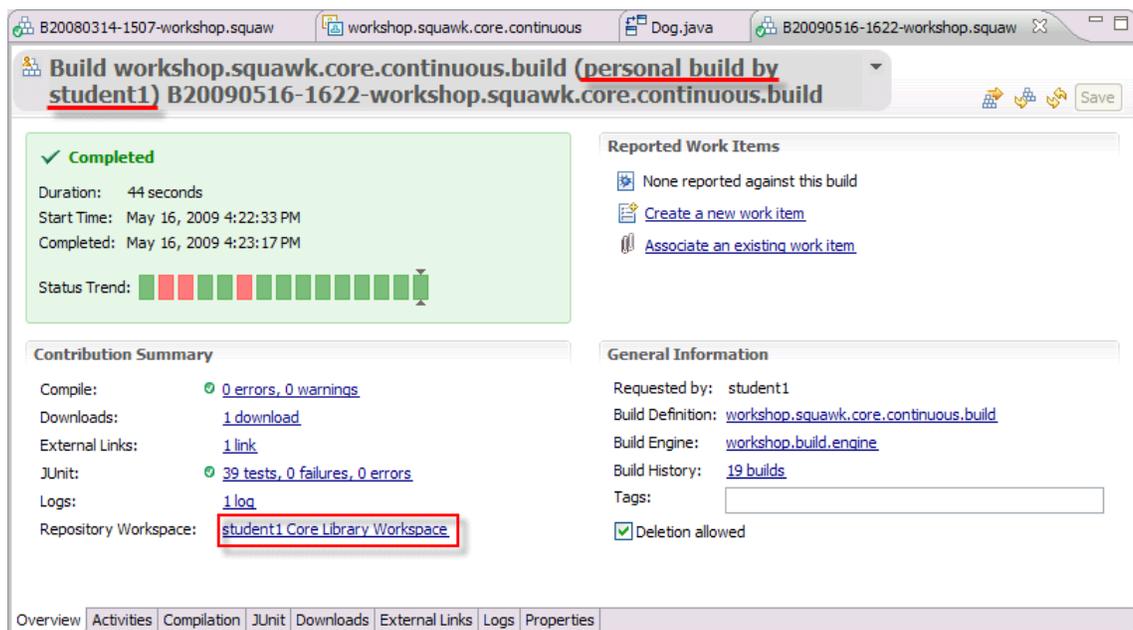
- \_\_c. Select **Browse** for the **Repository workspace**. In the **Select a Stream or Workspace** window type your student name then select your workspace (*student<N> Core Library Workspace*) and click **OK**.



- \_\_d. In the **Request Build** window press **Submit** and wait for completion.



- \_\_e. Open the completed build from the **Builds** view. The completed result should show your changes.



Notice that your workspace was used for this private build.

## 6.5 Build information is available from the web UI

Much of the same build information that you were just exposed is also available in the web UI.



Rather than repeat the last section in the web UI here is a sample set of screenshots of what is available there.

### \_\_1. Build definitions.



### \_\_2. Selecting a build definition displays build results.



\_\_3. Selecting a build result displays the details.

Build Definitions > workshop.squawk.core.continuous.build >

### Build workshop.squawk.core.continuous.build B20090427-0913-workshop.squawk.core.continuous.build



Status: **Successful**  
 State: **Completed**  
 Start time: Apr 27, 2009 9:13 a.m.  
 Completed: Apr 27, 2009 9:14 a.m.  
 Duration: 34 seconds

Overview Downloads External Links Logs Work Items

#### General Information

Requested by: Jerry Jazz  
 Build Definition: workshop.squawk.core.continuous.build  
 Build Engine: workshop.build.engine  
 Build History: [17 builds](#)  
 Tags: None

#### Contribution Summary

[1 download](#)  
[1 external link](#)  
[1 log](#)

\_\_4. Builds can be requested from a build results list or from a specific build result. The request is initiated from the icon in the upper right corner (  ) of either of these views.

Build Definitions > workshop.squawk.core.continuous.build > Submit

## Request Build

Build: [workshop.squawk.core.continuous.build](#)

#### Build Options

**Personal Build** Check this option to make this a personal build. The build will run using a workspace you specify, and will not impact the build definition status. Personal build alerts will only be received by you, not other members of your team. Browse

Repository Workspace:

#### Build Properties

Name:  Value:\*  Remove

Description:

Name:  Value:\*  Remove

Description:

**More about builds**

Hopefully, by now you should understand how build is our friend and helps keep your project from getting off track. If you are curious what a build script looks like they are in the Build Scripts component available from the Build stream. They are written as ANT scripts. You can learn more about build from the Jazz.net Learning section. See the document titled: [Getting Started with Setting up Jazz Builds](#).

**Conclusion**

This concludes the Lab for module 6. You now understand the difference between team/public builds and personal/private builds, how to request a build and then how to review its results.

---

## Lab 7 Exploring Changes and Traceability

In this lab, you will use the results of a build to demonstrate how information is linked within Jazz. There are some basic questions that this lab will answer:

- What work items went into a build?
- What changes were made for this work item?
- What build did this work item get delivered in?
- Who changed this file, and why?
- What are the specific changes made on this resource?
- How can we visualize the change history for this resource?



### Lab Scenario

With build operational and some interesting builds completed, this would be a good time to explore how Jazz links software artifacts in useful ways. You will now explore a part of the integration of Jazz Source Control Management with the rest of Jazz. You will also explore some of what can be done with change history.



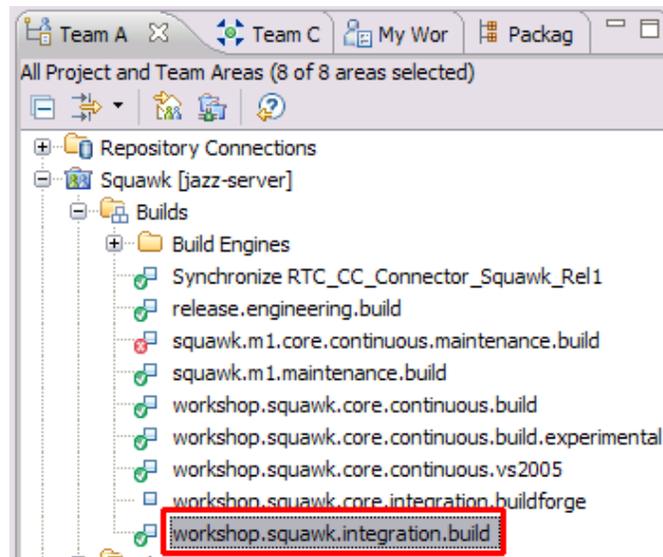
### Important!

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.

### 7.1 What work items went into the build

Starting with the build “**B20080123-1638-workshop.squawk.integration.build**”, you will explore and trace what contributed to this build. Knowing exactly what contributed to the build will allow you to prepare for what needs to be done after the build – what tests to run, as well as provide you an indication about what is left to complete for the current iteration plan.

- \_\_1. Find the build to examine.
- \_\_a. Starting from the **Team Artifacts** view, expand the **Squawk** project area, and then expand the **Builds** folder.



- \_\_b. Double-click *workshop.squawk.integration.build*.
- \_\_c. Notice the **Builds** view has opened at the bottom of the window.

The screenshot shows the 'Builds' view for 'workshop.squawk.integration.build'. It displays a table with 7 builds, all of which are 'Completed'. The build 'B20080123-1638-workshop.squawk.integration.build' is highlighted with a blue selection bar.

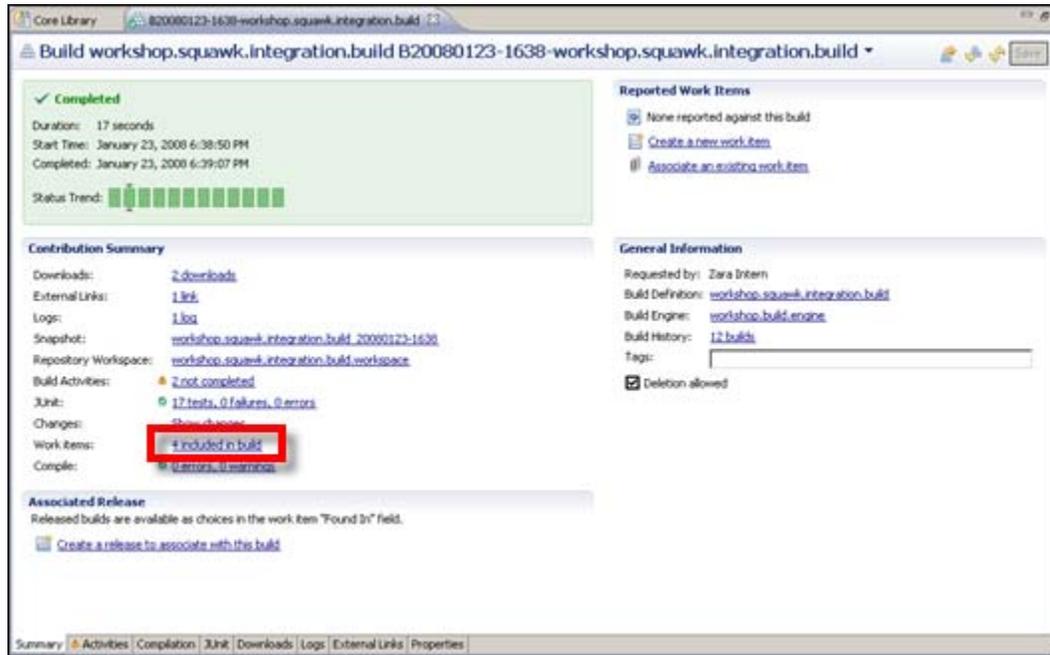
Build	Label	Progress	Estimated Completion	Start Time	Durat
workshop.squawk.integra...	B20080303-105...	Completed		March 3, 2008 12:54:...	26 sec
workshop.squawk.integra...	B20080201-170...	Completed		February 1, 2008 7:0...	14 sec
workshop.squawk.integra...	B20080130-094...	Completed		January 30, 2008 8:4...	22 sec
workshop.squawk.integra...	B20080130-094...	Completed		January 30, 2008 8:4...	28 sec
workshop.squawk.integra...	B20080128-140...	Completed		January 28, 2008 4:0...	25 sec
workshop.squawk.integra...	B20080123-163...	Completed		January 23, 2008 6:3...	17 sec

- \_\_d. Double-click build *B20080123-1638-workshop.squawk.integration.build* to open it.

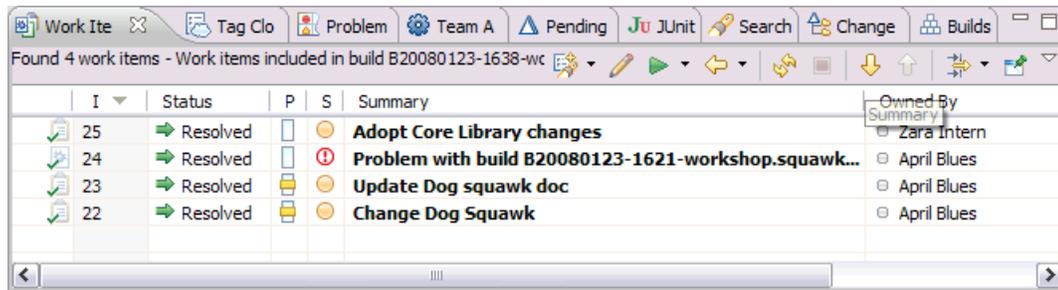
The screenshot shows the 'Builds' view for 'workshop.squawk.integration.build'. It displays a table with 13 builds, all of which are 'Completed'. The build 'B20080123-1638-workshop.squawk.integration.build' is highlighted with a blue selection bar.

Build	Label	Progress	Start Time	Dur
workshop...	I20080313-1336-workshop.squawk.integration.build	Completed	March 13, 2008 9:36:32 PM	26 s
workshop...	B20080307-1500-workshop.squawk.integration.build	Completed	March 8, 2008 12:00:07 AM	13 s
workshop...	B20080307-1454-workshop.squawk.integration.build	Completed	March 7, 2008 11:54:43 PM	25 s
workshop...	B20080303-1055-workshop.squawk.integration.build	Completed	March 3, 2008 7:54:48 PM	26 s
workshop...	B20080201-1709-workshop.squawk.integration.build	Completed	February 2, 2008 2:09:2...	14 s
workshop...	B20080130-0947-workshop.squawk.integration.build	Completed	January 30, 2008 3:47:0...	22 s
workshop...	B20080130-0944-workshop.squawk.integration.build	Completed	January 30, 2008 3:44:2...	28 s
workshop...	B20080128-1401-workshop.squawk.integration.build	Completed	January 28, 2008 11:01:...	25 s
workshop...	B20080123-1638-workshop.squawk.integration.build	Completed	January 24, 2008 1:38:5...	17 s
workshop...	B20080123-1607-workshop.squawk.integration.build	Completed	January 24, 2008 1:07:0...	13 s

- \_\_2. Find the work items.
  - \_\_a. Click the **Work Items** link in the build window.



- \_\_b. The **Work Items** view opens and shows only the Work Items that went into build *B20080123-1638-workshop.squawk.integration.build*.



## 7.2 What changes were made for a Work Item

Now that you have found what work items went into a build, you will explore the changes for a specific work item. This will allow you to see the specific files that changed. This makes the review process for the change easier. This also allows for an easier method to determine what work item could have broken a build.

- \_\_1. Open the work item.
  - \_\_a. From the **Work Items** view, double-click contributing work item **22: Change Dog Squawk**.

The screenshot shows the IBM Work Item Details page for 'Task 22: Change Dog Squawk'. The browser address bar shows 'B20080123-1638-workshop.squawk.integration.build' and the page title is '22: Change Dog Squawk'. The page is titled 'Task 22' and has a 'Resolved' status. The summary is 'Change Dog Squawk'. The 'Details' section includes the following information:

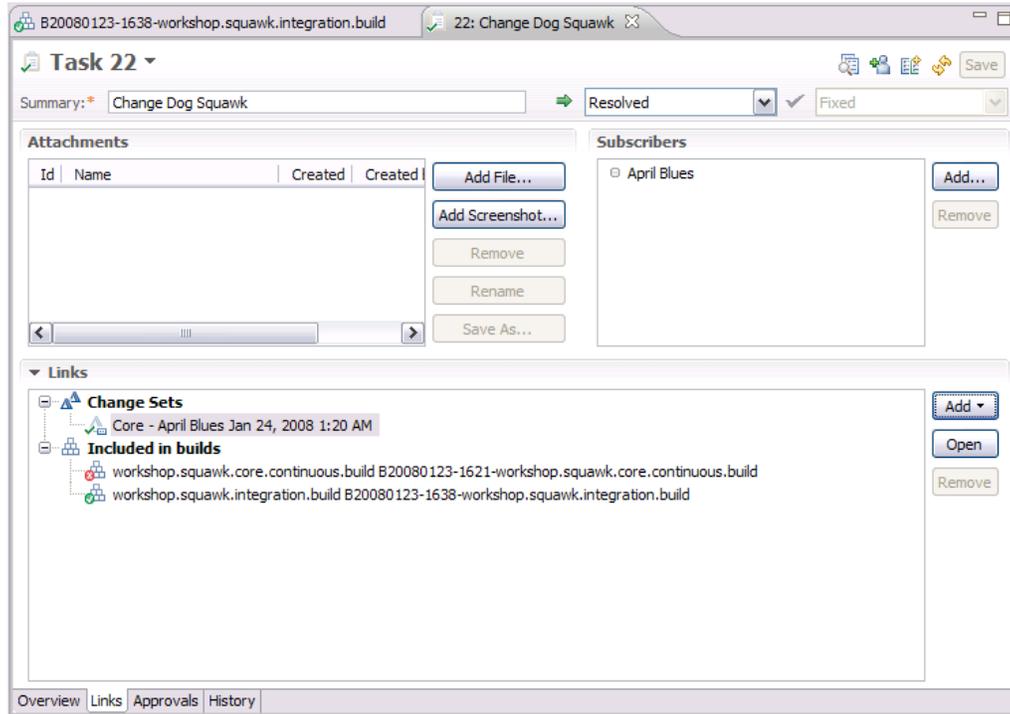
- Type: Task
- Severity: Normal
- Found In: Unassigned
- Creation Date: Jan 23, 2008 7:16 PM
- Created By: April Blues
- Team Area: Core Library / Squawk
- Filed Against: Squawk/Squawkers
- Tags: (empty)
- Owned By: April Blues
- Priority: Medium
- Planned For: 1.0 M2
- Estimate: 1 d
- Time Spent: (empty)
- Due Date: None
- Resolution Date: Jan 23, 2008 7:20 PM
- Resolved By: April Blues

The 'Description' section contains the text: 'Change squawk to "Bow-wow"'. The 'Quick Information' section shows:

- Subscribers (1): AB
- Change Sets (1)
- Included in Builds (2)

The 'Discussion' section is currently empty, with an 'Add Comment' link. The page has tabs for 'Overview', 'Links', 'Approvals', and 'History'.

- \_\_b. Note the **Change-sets** link in the **Quick Information** section of the **Overview** tab and the detail in the **Links** tab.

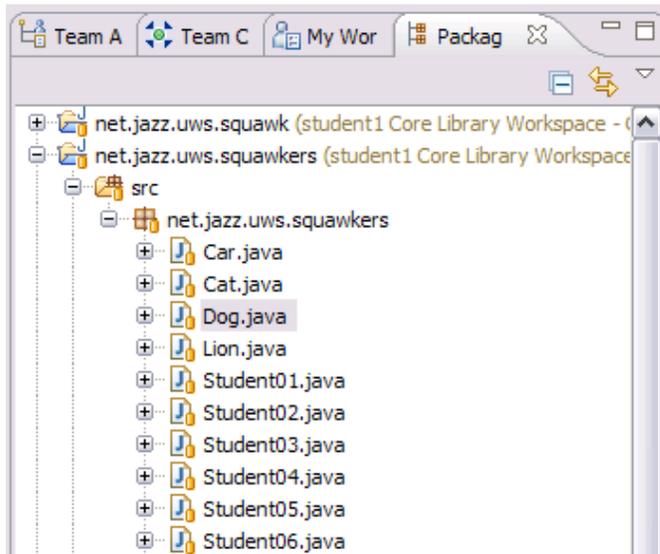


- \_\_2. Note the work item's **Included in builds** section in the **Links** tab above. Double click a build to display the build record. After we have looked at what was done to finish the work item, we start to look at where the work was introduced into the project and first built. The test community can now easily determine where to start testing the change this work item introduced.

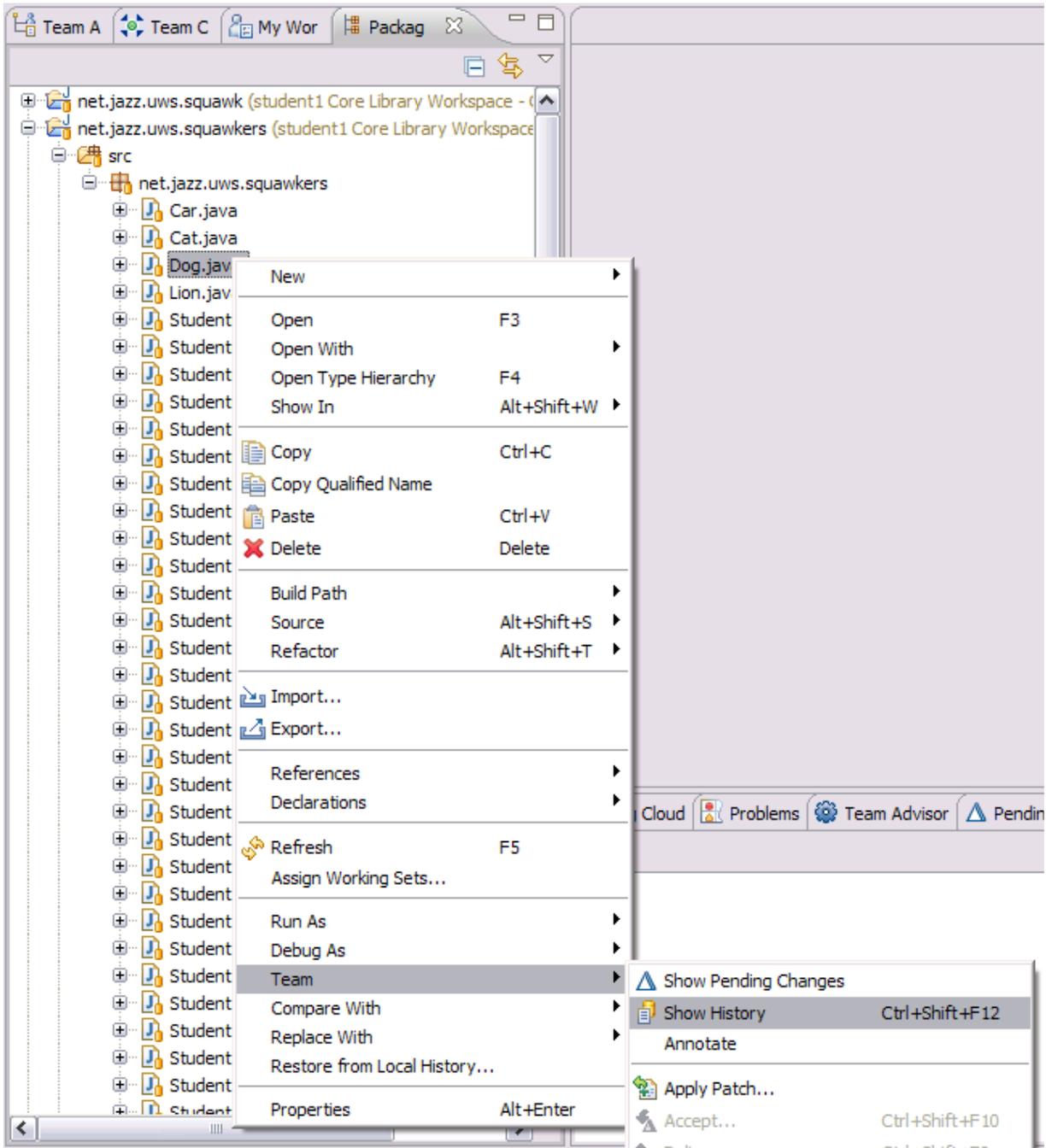
## 7.3 Who changed this file, when and why?

This section covers looking at the capabilities of finding out what has changes happened with a specific file.

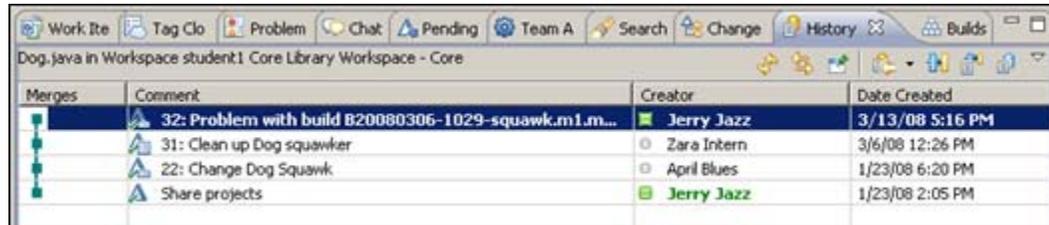
1. In the **Package Explorer** view, navigate to `net.jazz.uws.squawkers` → `src` → `net.jazz.uws.squawkers` package.



\_\_2. Right-click the **Dog.java** file and select **Team → Show History**.



- \_\_a. The **History** view opens and shows who has made changes to the file and when those changes were made. You can now start to understand how the file has evolved over time. This can also help with someone who is new to the project so they know who to talk to when they have questions over certain files. The **Merges** column gives a graphical view of the change-set merges that have occurred over time on this file.

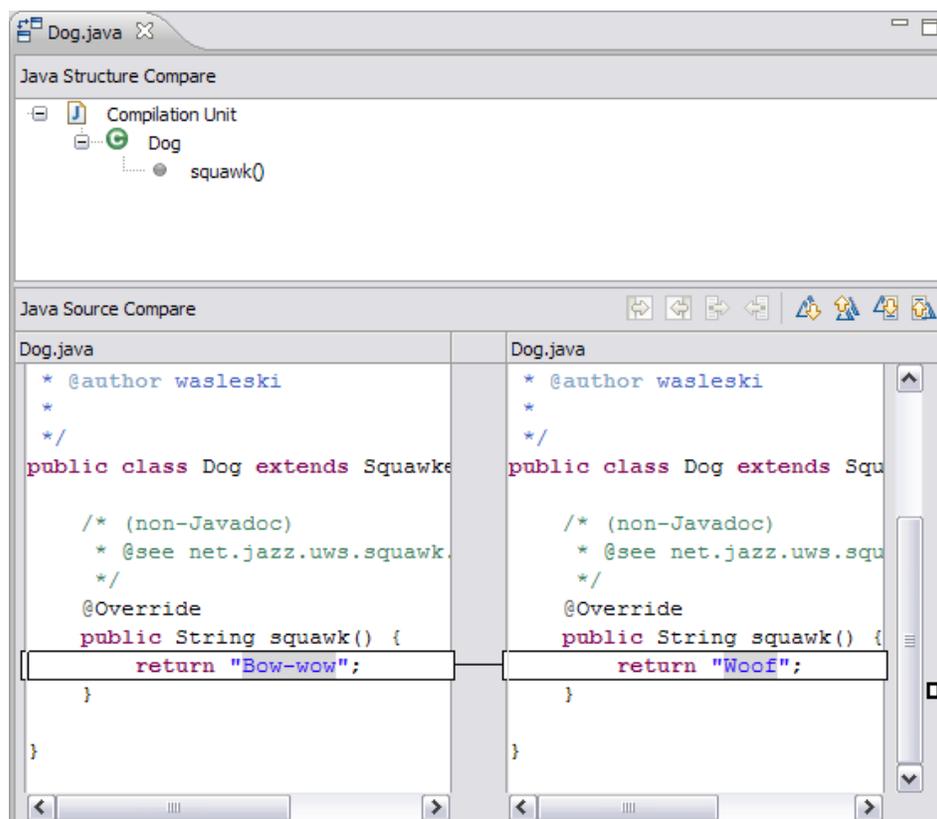


Merges	Comment	Creator	Date Created
	32: Problem with build B20080306-1029-squawk.m1.m...	Jerry Jazz	3/13/08 5:16 PM
	31: Clean up Dog squawker	Zara Intern	3/6/08 12:26 PM
	22: Change Dog Squawk	April Blues	1/23/08 6:20 PM
	Share projects	Jerry Jazz	1/23/08 2:05 PM

## 7.4 What are the specific changes made on this resource?

Once you know who has made changes to a file, you will next figure out what those developers did to the file. The more detail that is available allows for a greater ability to determine what others have done to the file. This allows for quicker resolution when there is an issue with the changes made to a file.

- \_\_1. Double click work item **22: Change Dog Squawk** in the **History** view to open the compare editor.



With the synchronized side-by-side view of the changes made use the navigational aids

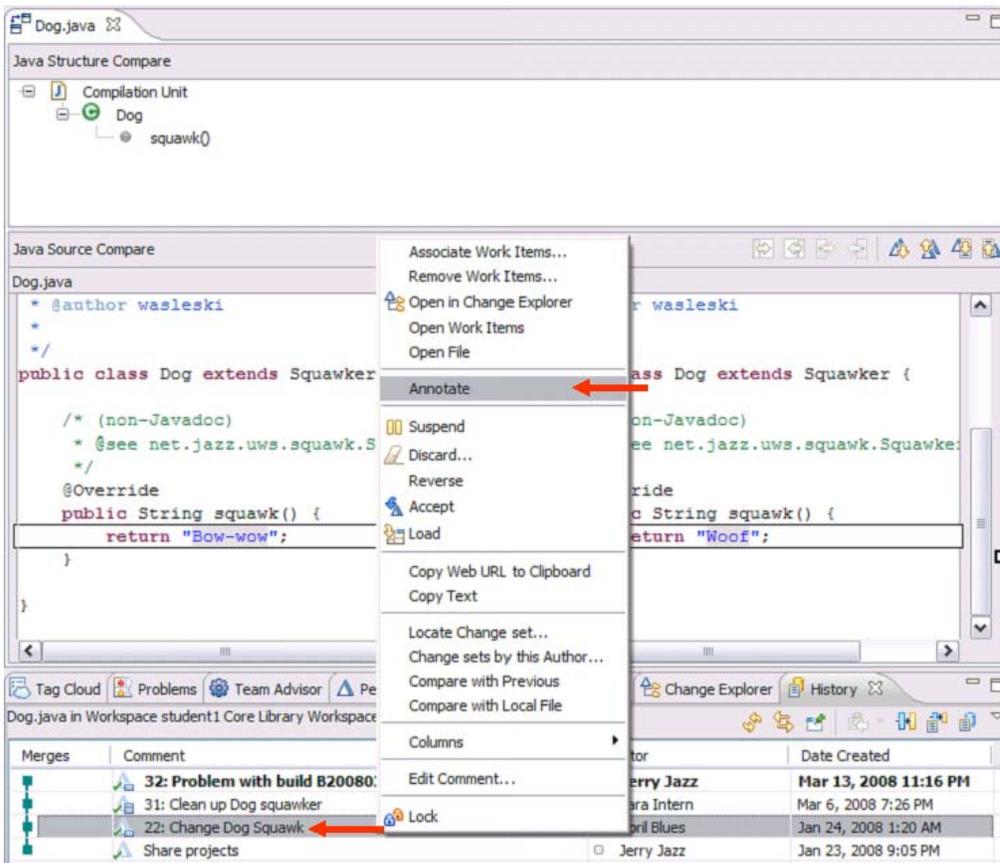


to go to next/previous changes/differences, it is easy to see what changes occurred as a result of a particular change.

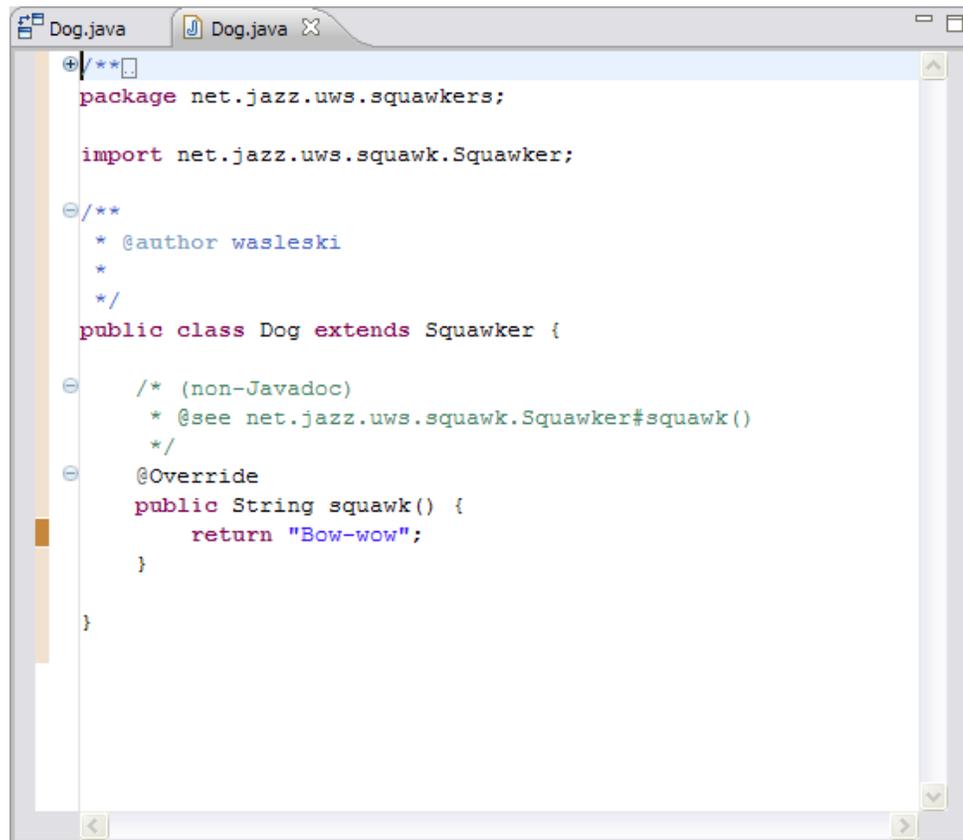
## 7.5 How can we visualize the change history for a resource?

One of the Rational Team Concert features is the ability to show who is responsible for writing specific lines in the class. If a file has caused a broken build, this will allow for the guilty party to be found, so that a solution can be developed.

- \_\_1. Return to the **History** view and right-click work item **22: Change Dog Squawk** again and select **Annotate**.



- \_\_a. Notice the bar on the left side of the source. The colors on the bar designate the age of the change – the darker the color the newer the change.



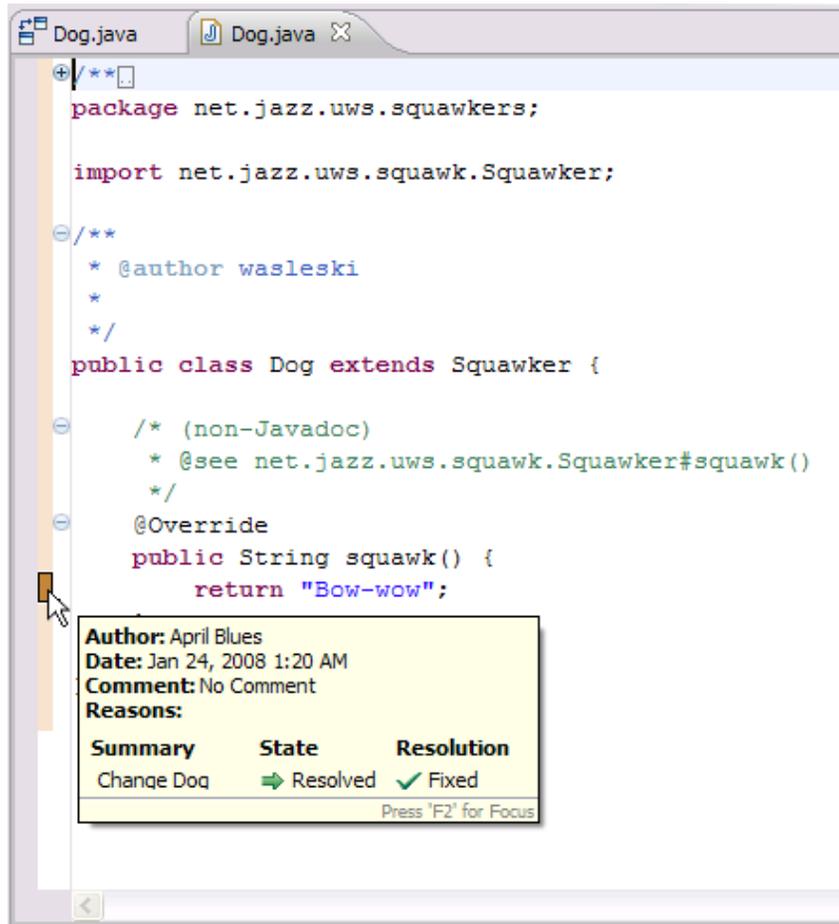
```
package net.jazz.uws.squawkers;

import net.jazz.uws.squawk.Squawker;

/**
 * @author wasleski
 *
 */
public class Dog extends Squawker {

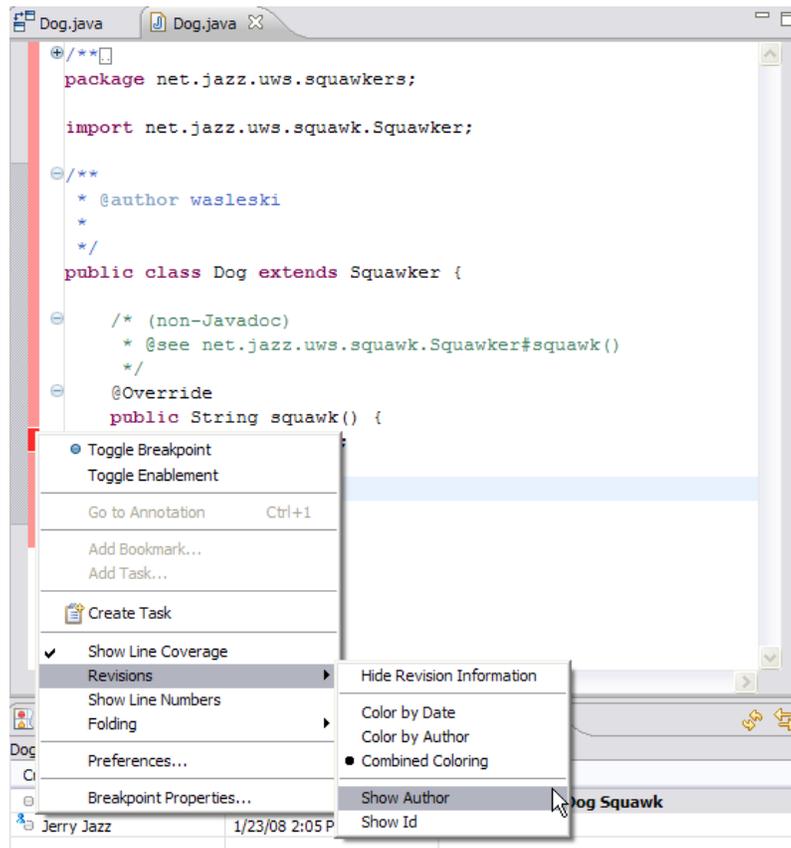
    /* (non-Javadoc)
     * @see net.jazz.uws.squawk.Squawker#squawk()
     */
    @Override
    public String squawk() {
        return "Bow-wow";
    }
}
```

\_\_b. Hover the cursor over the darkest color and notice the data for that particular change.



\_\_c. By pressing **F2**, you can switch focus to the popup window. This will allow you to follow the link to work item **22**.

- \_\_d. Right-click the color bar and see the options under **Revisions**. These options will change how the differences can be displayed.



### Conclusion



This concludes the Lab for module 7. You are now familiar with the different ways Team Concert allows you to explore and track changes you and your colleagues have made to the source code files, work items and builds (and more!). You have observed the tremendous transparency and traceability built in to the Team Concert environment which enables you to get significant insight into your project.

---

## Lab 8 Endgame and a Tightened Process

Process enactment in Rational Team Concert allows for the project and/or team processes to be adjusted as a project progresses through the lifecycle. In this lab exercise the project will move to the *End Game* iteration of Milestone *M3* and experience some differences in the team's process. In the *M3 End Game* iteration the *Core Library* team has customized their process so that changes can be delivered only if their team lead has approved the work item associated with the delivery.



### Lab 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.

### 8.1 Instructor Demo – Review the process for End Game



#### Instructor Demo

Section 8.1 is performed by the instructor as a demo. In this section, the instructor will review the state of the process for End Game iteration to ensure that process enforcements are in place.

Refer to the corresponding section in the workbook Appendix D.

### 8.2 Instructor Demo – Change the process behavior



#### Instructor Demo

Section 8.2 is performed by the instructor as a demo. In this section, the instructor will advance the state of the process by moving the current iteration to the End Game iteration.

Refer to the corresponding section in the workbook Appendix D.

### 8.3 Observe process enactment in action



#### Important!

This section is to be performed by the student.

Ensure that you carry out this Lab assuming the role and identity of the user you previously created.

The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number.

Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.



#### Team role

You are now performing the role of **student<N>**, a Developer.

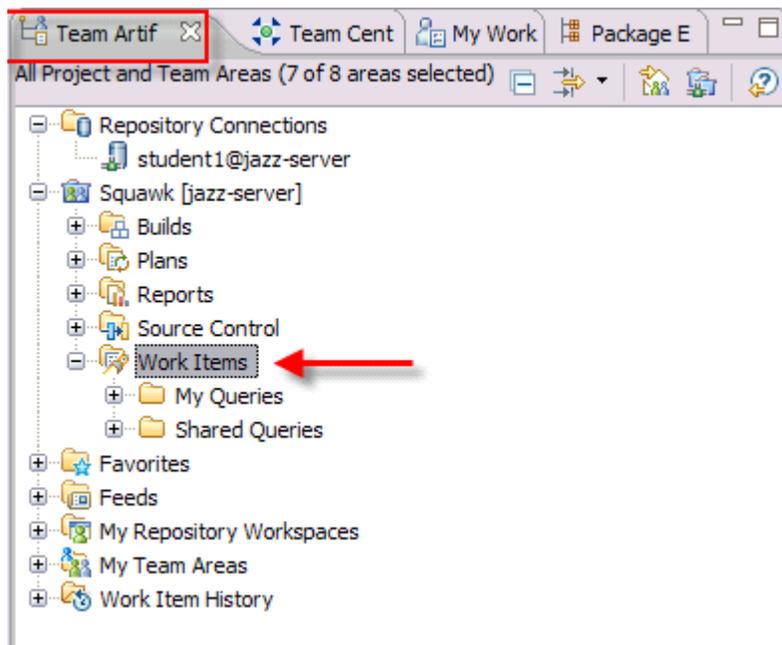
- \_\_\_1. Close then reopen Team Concert, to avoid a possible timing delay in the refresh of the process configuration change in your workspace.
  - \_\_\_a. If already started, close Team Concert.
  - \_\_\_b. Open Team Concert by double clicking the Team Concert shortcut  on the Windows Desktop.

- \_\_c. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` (replacing <N> with your id number). Click **OK**.



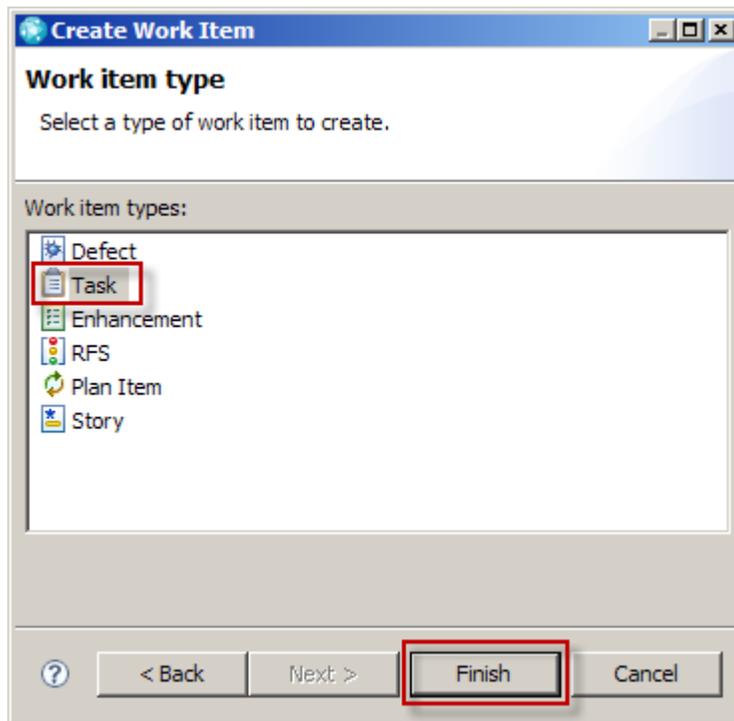
- \_\_2. Create a new Work Item to track a change to your Squawker class

- \_\_a. Select the **Team Artifacts** view



- \_\_b. Expand the **Squawk** project
- \_\_c. Right click the **Work items** folder then select **New → Work Item**.

\_\_d. Select Work item type **Task** and click **Finish**



\_\_e. The Work Item editor contains the new task:

The screenshot shows the 'Task <04:31:13>' editor window. At the top, there is a title bar with the task ID and a 'Save' button. Below the title bar, there is a 'Summary:' field and a status dropdown menu set to 'Uninitialized'. The main area is divided into two sections: 'Details' and 'Description'. The 'Details' section contains various fields for task metadata, including Type (Task), Severity (Normal), Found In (Unassigned), Creation Date (None), Created By (student1), Team Area (Squawk Team / Squawk), Filed Against (Unassigned), Tags, Owned By (Unassigned), Priority (Unassigned), Planned For (Unassigned), Estimate, Time Spent, and Due Date (None). The 'Description' section is a large text area for entering the task details. At the bottom of the 'Details' section, there is a 'Quick Information' box showing '<No information>'. To the right of the 'Description' section, there is a 'Discussion' section with an 'Add Comment' link. At the very bottom, there is a navigation bar with tabs for 'Overview', 'Links', 'Approvals', and 'History'.

- \_\_f. For the details of the Task:
- \_\_i. Type End Game change to my student<N> squawker in the **Summary** (replace student<N> with the name of your student id )
  - \_\_ii. Set **Filed Against** to Squawk/Squawkers.
  - \_\_iii. Set **Owned By** to student<N>
  - \_\_iv. Set **Priority** to Medium
  - \_\_v. Set **Planned For** to ->1.0 M3
  - \_\_vi. In **Estimate**, type 1 day

Task <05:22:29>

Summary: \* End Game change to my student1 squawker

Uninitialized

Save

Details

Type: Task

Severity: Normal

Found In: Unassigned

Creation Date: None

Created By: student1

Team Area: Core Library / Squawk

Filed Against: \* Squawk/Squawkers

Tags:

Owned By: student1

Priority: Medium

Planned For: -> 1.0 M3

Estimate: 1 d Correction: 0

Time Spent:

Due Date: None

Quick Information

<No information>

Description

Discussion

Add Comment

Overview Links Approvals History

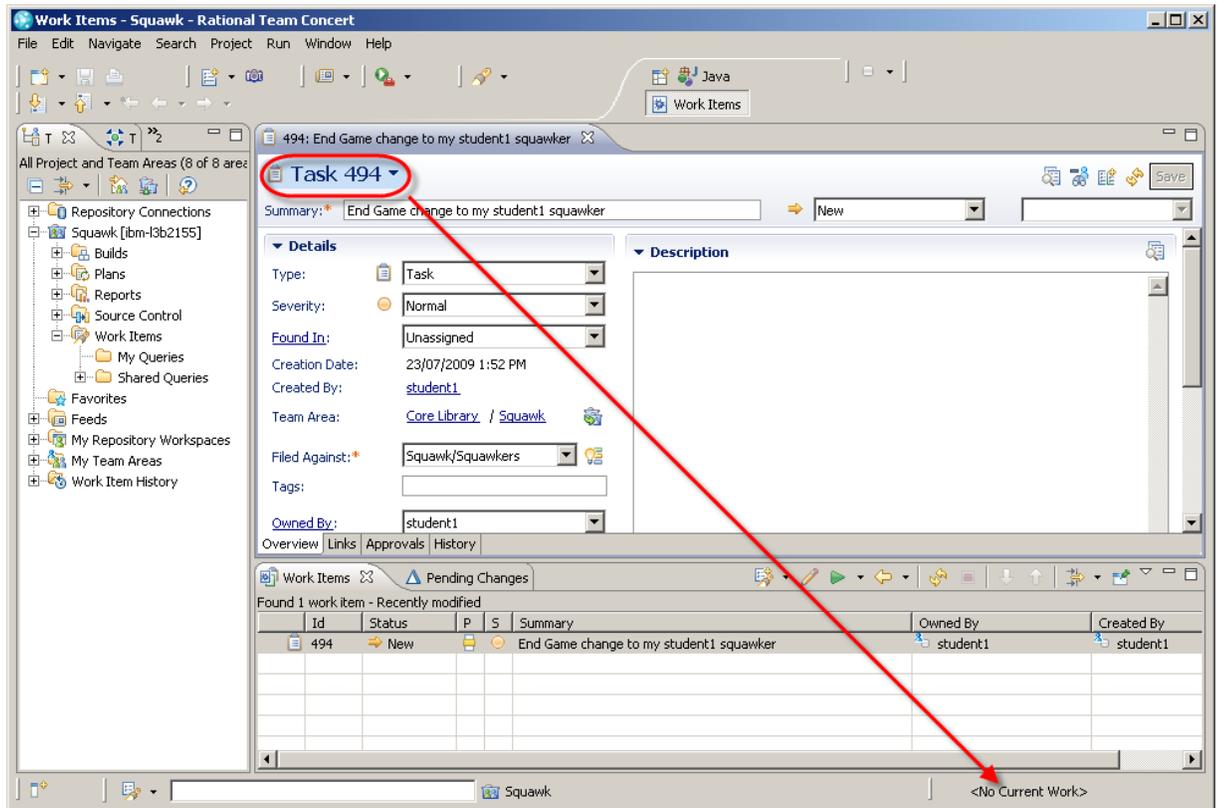
- \_\_g. Click **Save**.



### Working on a work item

You can specify any work item as the active work item for a workspace. By default, the current change set is associated with the active work item. The active work item number and part of its headline are displayed in lower right corner of the Eclipse main frame. If there is no active work item, this area displays the text <no current work>. Activating a work item allows other team members to see that you are working on it, and creates an automatic association between the work item and the current change set.

\_\_h. Click and hold the newly created WI and drag it to the Current Work area



- \_\_i. Notice that the new Work Item is displayed as the active Work Item and its state is automatically changed to **In Progress**.

The screenshot shows the 'Task 494' editor. The 'Summary' field contains 'End Game change to my student1 squawker' and the 'Status' dropdown is set to 'In Progress'. The 'Details' section shows the following information:

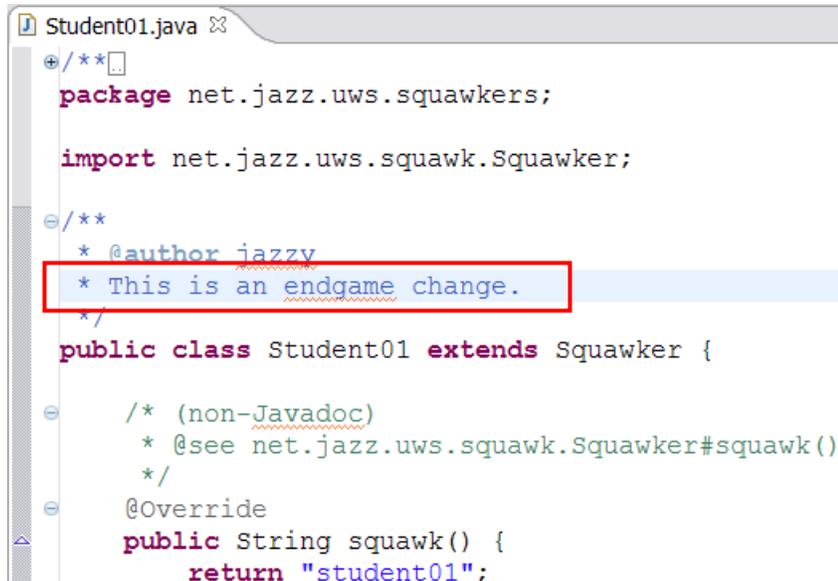
- Type: Task
- Severity: Normal
- Found In: Unassigned
- Creation Date: 23/07/2009 1:52 PM
- Created By: student1
- Team Area: Core Library / Squawk
- Filed Against: Squawk/Squawkers
- Tags: (empty)

The 'Description' field is empty. Below the editor, the 'Work Items' table shows one item:

Id	Status	P	S	Summary	Owned By
494	In Progress			End Game change to my student1 squawker	student1

- \_\_3. Close the Work Item editor view containing your new Task.

- \_\_4. Perform changes to your Squawker class
- \_\_a. Switch to the **Package Explorer** view (**Window** → **Show View** → **Other** → **Java** → **Package Explorer**).
  - \_\_b. Navigate to **net.jazz.uws.squawkers** → **src** → **net.jazz.uws.squawkers** and double-click the Squawker class created in a previous lab (e.g. Student<N>.java, where N is your student id).
  - \_\_c. Below the \* **@author student1** line, type a comment (use the image below as a reference).



```

Student01.java
/**
package net.jazz.uws.squawkers;

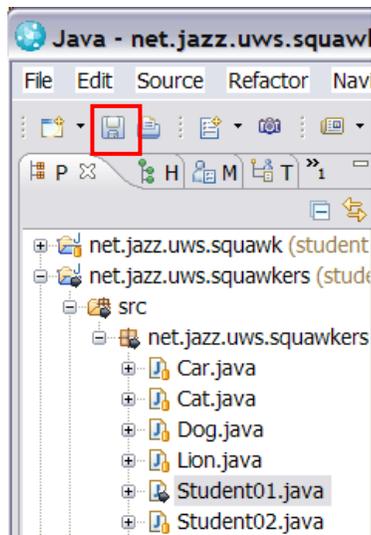
import net.jazz.uws.squawk.Squawker;

/**
 * @author jazzy
 * This is an endgame change.
 */
public class Student01 extends Squawker {

    /* (non-Javadoc)
     * @see net.jazz.uws.squawk.Squawker#squawk()
     */
    @Override
    public String squawk() {
        return "student01";
    }
}

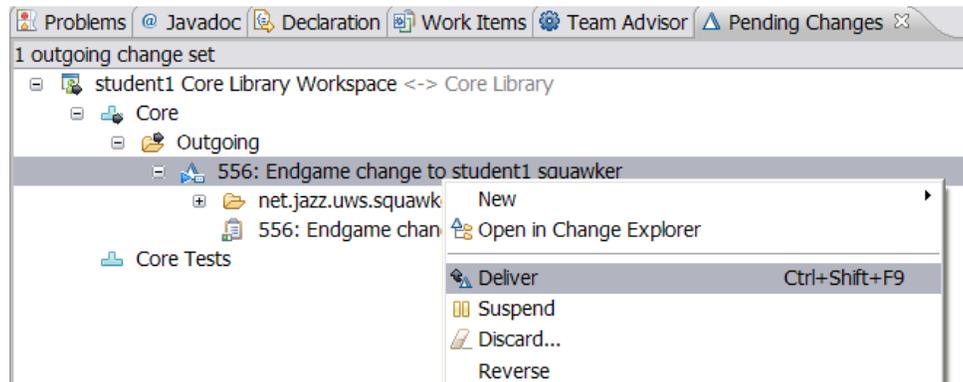
```

- \_\_d. Click **Save**.



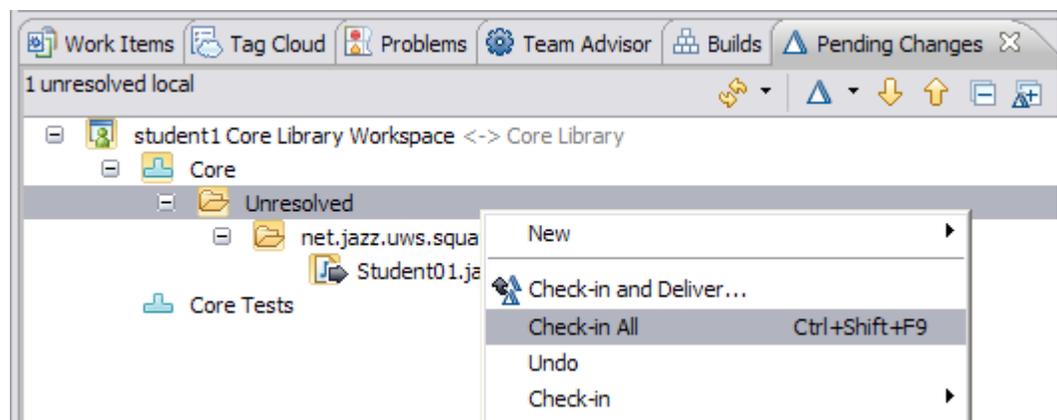
- \_\_5. Deliver changes to the Core component

- \_\_a. In the **Pending Changes** view, expand **student<n> Core Library Workspace**, right-click the **End game change to my student1 squawker** change set and click **Deliver**.

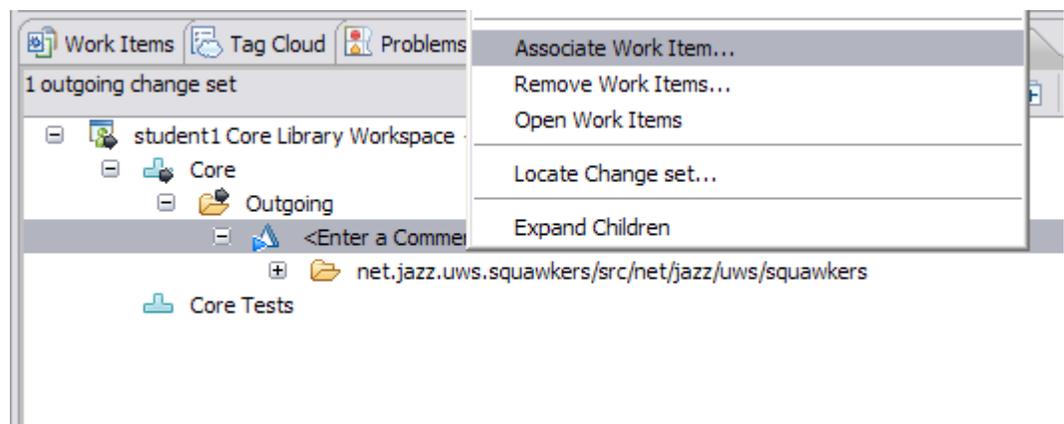


Note: If the change set is not associated with a work item (shown by the label *Unresolved* instead of *Outgoing*) follow these steps:

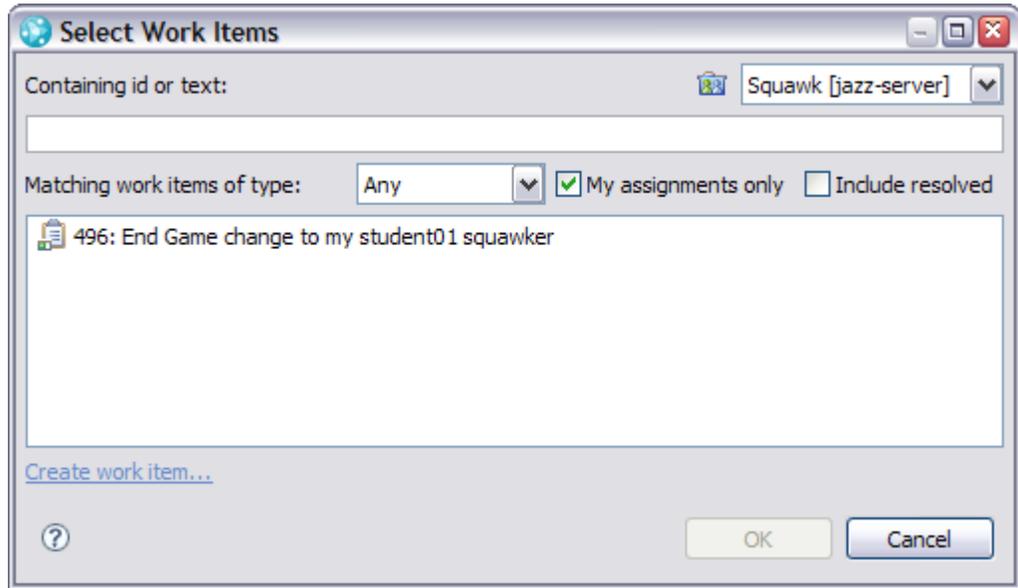
- \_\_i. Right-click on “Unresolved” and select “Check-in All”



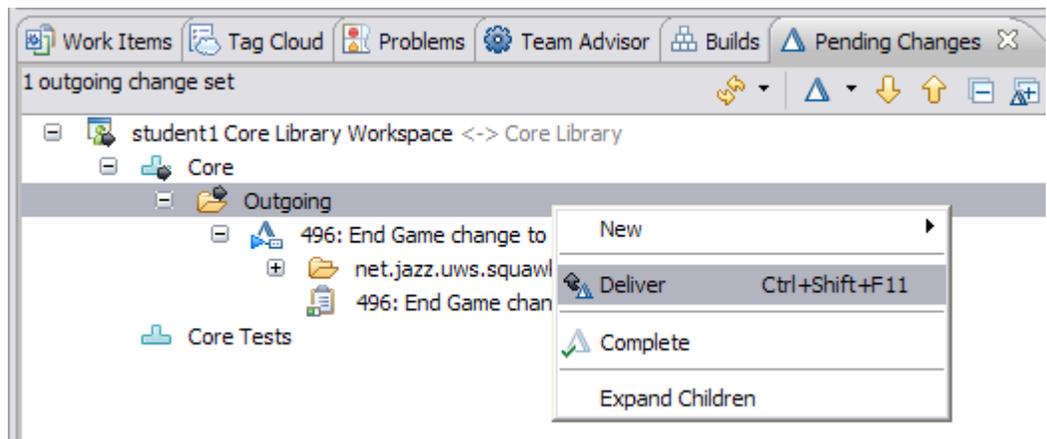
- \_\_ii. Then right-click and select “associate work item”



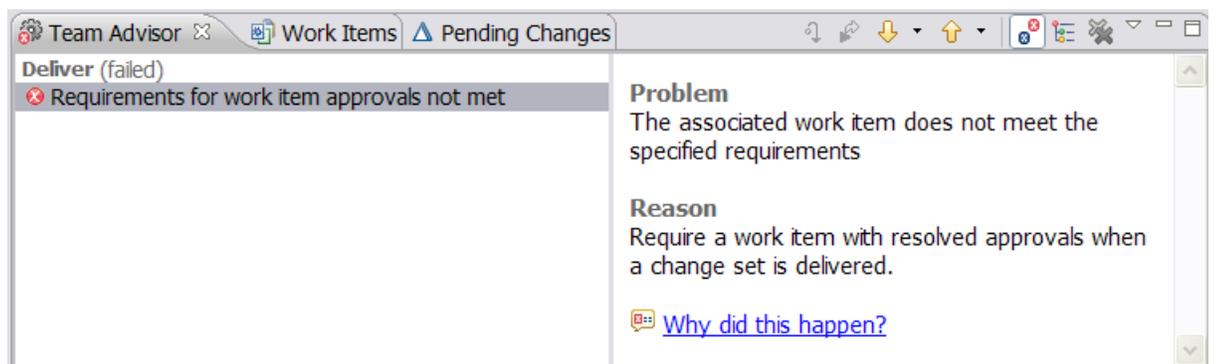
\_\_iii. Then pick the Endgame change to student<N> squawker



\_\_iv. Then right-click and select Deliver



\_\_b. The delivery will fail because the Work Item has not been approved by the team lead. Whereas in early iterations, you may have been able to successfully deliver changesets without approvals, in this iteration, an approved work item is required for deliveries.



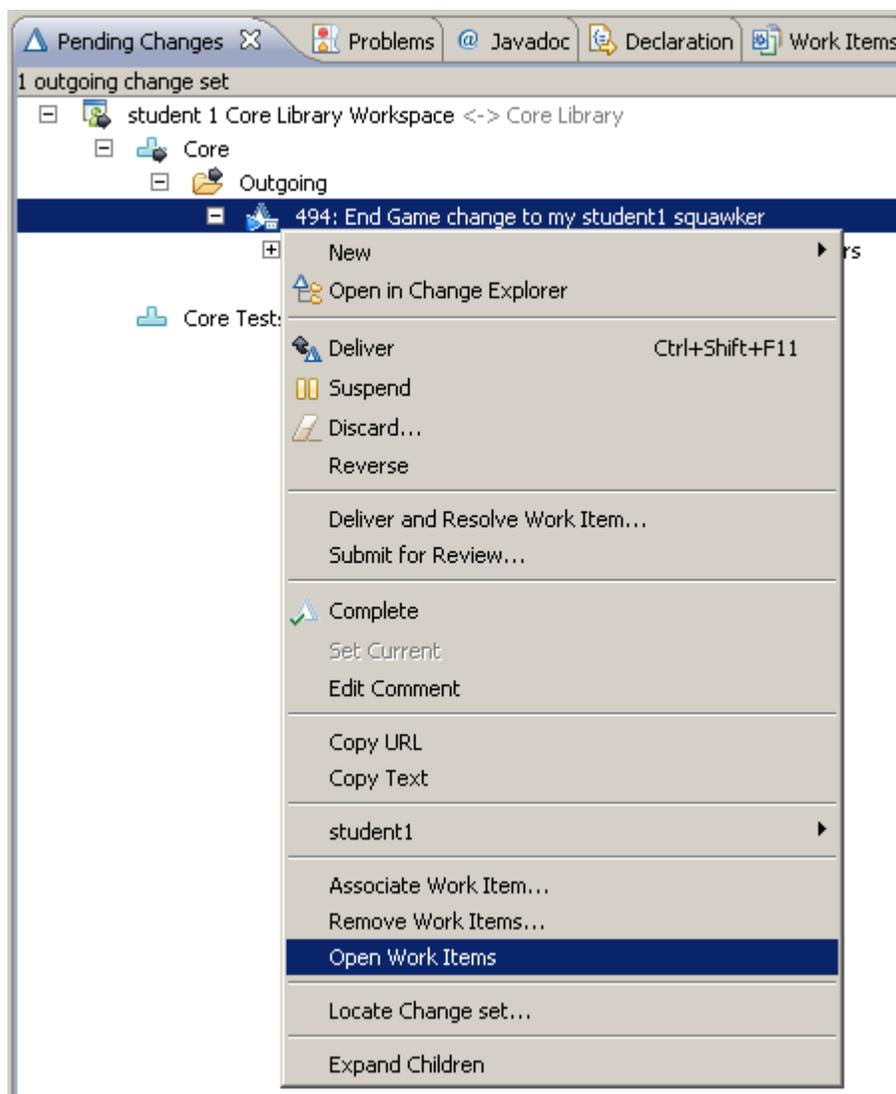


#### If the delivery succeeded.....

It is possible that the change by the instructor to the Squawk project process will not be propagated to your Eclipse workspace before you attempt the delivery. In that case, the delivery will succeed. If that happens, the easiest thing to do is exit Team Concert and open it again. Logging out of the project and then back in may also work too. You will then need to make another change to your squawker class and retry the delivery.

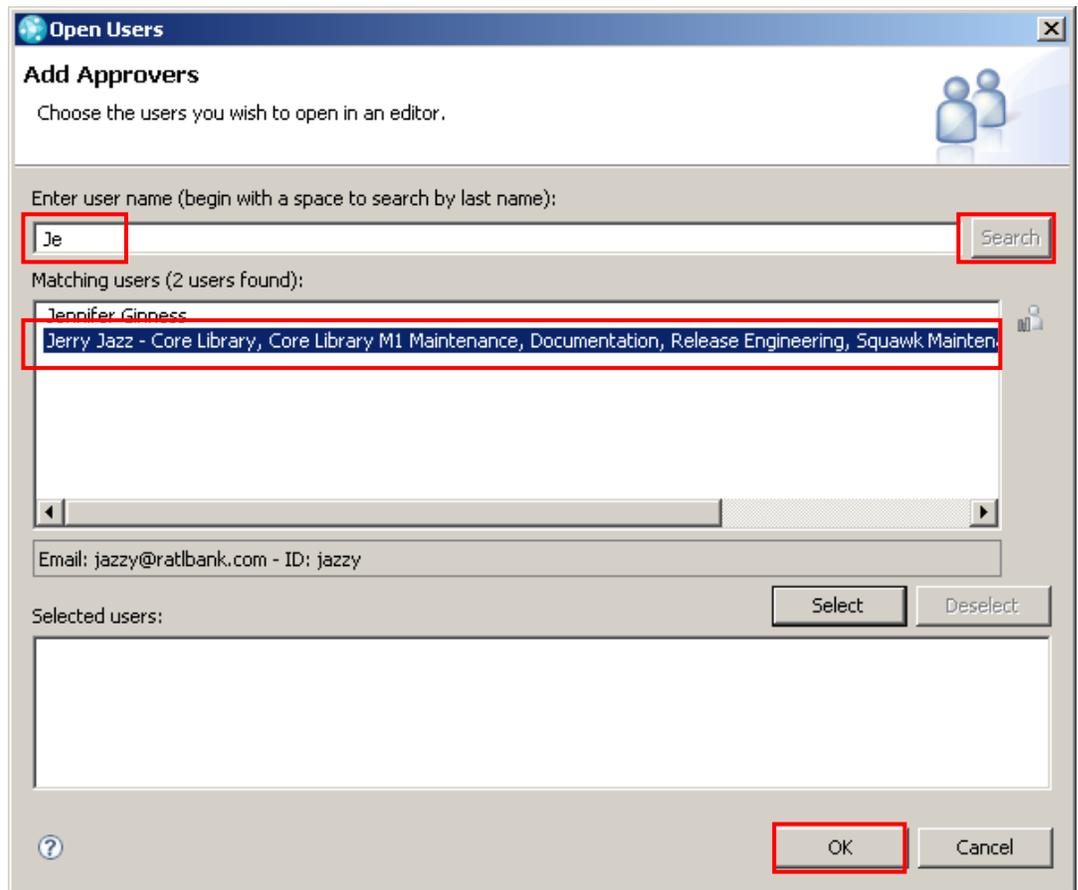
\_\_6. Add an approval for the Work Item

\_\_a. In the **Pending Changes** view, expand student<n> Core Library Workspace, right-click the End game change to my student<N> squawker change set and click **Open Work Items**.



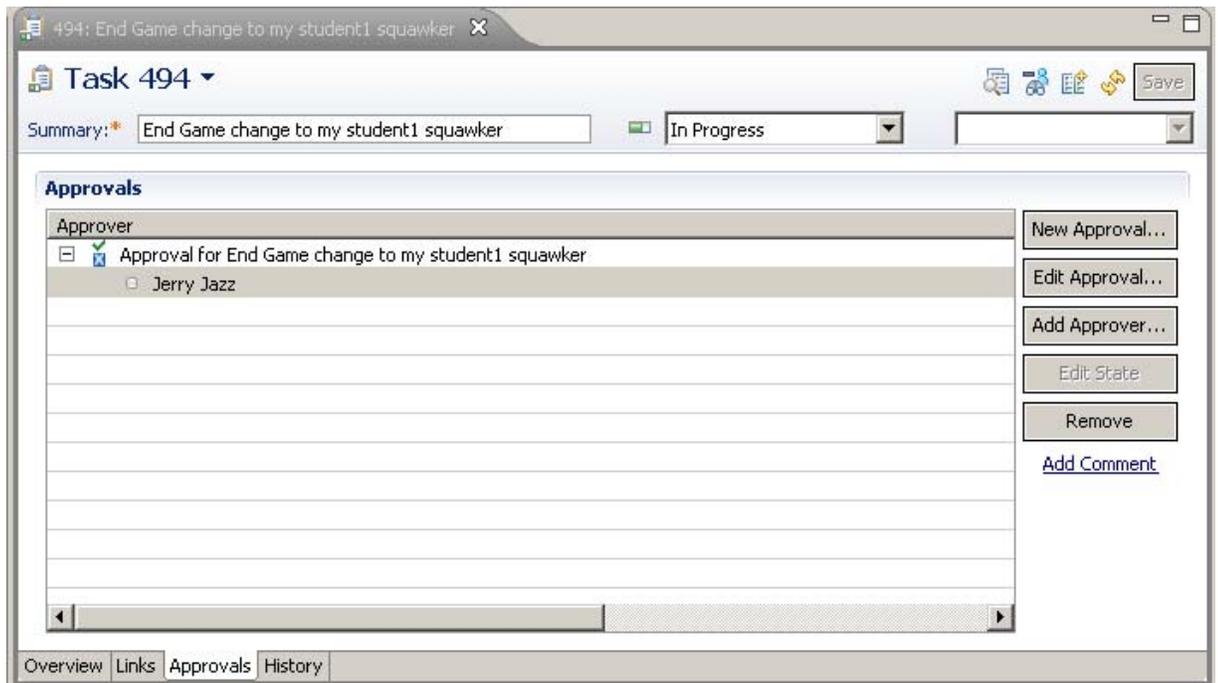


- \_\_iii. In the **Open Users** dialog, type `Je` for the username and click **Search** to bring up Jerry Jazz in the **Matching Users** section. Select **Jerry Jazz**, click **Select** and click **OK**.



- \_\_iv. Click **Save**.

- \_\_b. The **Approvals** tab of the new task should look similar to this:



- \_\_7. You will now wait for the Instructor playing the role of Jerry the Team Lead to Approve this Work item before you can deliver it.

## 8.4 Instructor Demo – Approve the new End Game Work items

 **Instructor Demo**  
Section 8.4 is performed by the instructor as a demo. In this section, the instructor will approve the End Game Work items, allowing the Work Items to be delivered and resolved.  
Refer to the corresponding section in the workbook Appendix D.

## 8.5 Deliver Approved changes

Once the instructor has approved your end game work item, repeat the delivery as in Step 5 of Section 8.3. This time the delivery should succeed.

**Conclusion**

In this exercise, you have experienced a small bit of what process definitions can do. If you want to understand this better see the document: Getting Started with Jazz Project Areas and Process on jazz.net at

<https://jazz.net/learn/LearnItem.jsp?href=content/docs/process/index.html>.

## Lab 9 Taking Control of Your Project

In this module, you will learn about Rational Team Concert Dashboards and Reports and how these powerful capabilities can assist your team monitor project status and make informed decisions to keep it on track. Dashboards are a Web UI component intended to provide information about the status of a project or projects at a glance and represents the integration point for the data provided by all Rational Team Concert components. The Reports component provides an awareness of the actions, behaviors and progress of a team or project.

You will learn that dashboards can be used in a variety of ways:

- The executive team, project managers, and team leads can track project health and trends at a glance
- Teams can discuss status using the dashboard as the data source
- Team leads can track team progress and balance workload
- Developers can track their workload

You will become familiar with the two main parts of the reports component: the data warehouse and the reports engine as you explore the many out-of-box report templates.



### Lab Scenario

First, you will see some examples of real dashboards that were extracted from Team Concert development on jazz.net. You are encouraged to register there so you can continue your journey with the Jazz platform and the Jazz product family. You will then, 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.

## 9.1 Personal Dashboard Configuration

### Important!

Section 9.2 is to be performed by the student.



Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote student<N>, which you should replace <N> with your assigned id number. Sample screenshots in all labs will use the id student1. If you are unsure please check with the instructor.



### Preparation

Please make sure that the steps related to the JUnit project in the Lab 9 Demo have been completed before starting this Lab

As previously discussed the dashboard capabilities are not just available at the project level, but also at the team/sub-team levels. Each team has the ability to create, customize their team dashboard. This is another great way to not only provide transparency to management and technical leadership but also to bring globally or organizationally distributed teams together to perform and operate more cohesively; truly starting to bring social networking to the workplace.

\_\_1. For the steps in this sub-section you will again connect to the Squawk Project Area.

**Rational Team Concert License Type and Permissions**

While [Rational Team Concert Standard edition](#) and [Enterprise edition](#) include the full dashboard functionality described in this topic, the [Express](#) and [Express-C](#) editions include limited dashboard support. In the Express and Express-C editions, each project area includes a single project dashboard with a single dashboard tab. Dashboards are not available for individual users or teams.



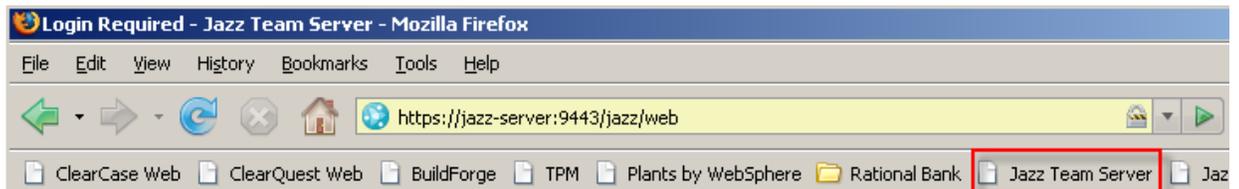
To create and/or modify a dashboard you must be assigned a Rational Team Concert – Developer License. In addition, your account must be assigned to a role that has permission to maintain Project and/or Personal Dashboards.

\_\_2. Connect to the Squawk Project's Jazz Team Server using the Web UI.

\_\_a. Open the **Firefox** internet browser by double-clicking the **Mozilla Firefox** shortcut on the **Windows Desktop**.



\_\_b. Click the **Jazz Team Server** shortcut on the bookmarks toolbar.



- c. The Web UI uses a secure connection. Enter your student id for both the **User name** and **Password**. Click **Log In** to access the project areas.



- \_\_d. After the login you must choose a project area. Note that there is another project area named JUnit Project. Ignore it for now. You will return to it later. Click the **Squawk** project area.

**Rational Team Concert**

Project Areas

## Project Areas

### My Project Areas

**Squawk**

Instructional project for user training

This workshop was created and maintained by the Jazz Jumpstart Team:

Jim D'Anjou,  
Philippe Krief  
Steve Wasleski

### Other Project Areas

#### JUnit Project

A Team Concert example project area based on the JUnit project

This example project area illustrates the use of work items, SCM, builds and other Team Concert components in a project...

It exemplary shows project work on the next JUnit release 4.4.

- \_\_3. You are automatically directed to the Squawk Project **Dashboard** Page. On the left navigation, click the **Create Dashboard** link to create your private dashboard.

**Rational Team Concert**

Your Trial License expires in 49 days | student1 | Log Out | ?

Dashboards Project Areas Work Items Plans Source Control Builds Reports

Type to search

All Dashboards >

## Squawk Dashboard

General

**HTML**

Thank you for taking the time to do the Jazz User Workshop. Please report workshop suggestions or bugs to jazz.net using work item category **Workshop**.

**Useful Links**

**Jazz Team Blog** (10 new)

**Squawk Builds**

**Squawk Events** (100 new)

**Description Squawk**

**Squawk Teams** (7)

**Squawk Team Members** (5)

**Create Dashboard**

**Recently Viewed**

- Squawk
- student1's Dashboard

**My Dashboards**

- student1's Dashboard

**Public Dashboards**

- Squawk
  - Squawk Team
  - Core Library
  - Documentation
- All Dashboards

IBM. *Jazz*



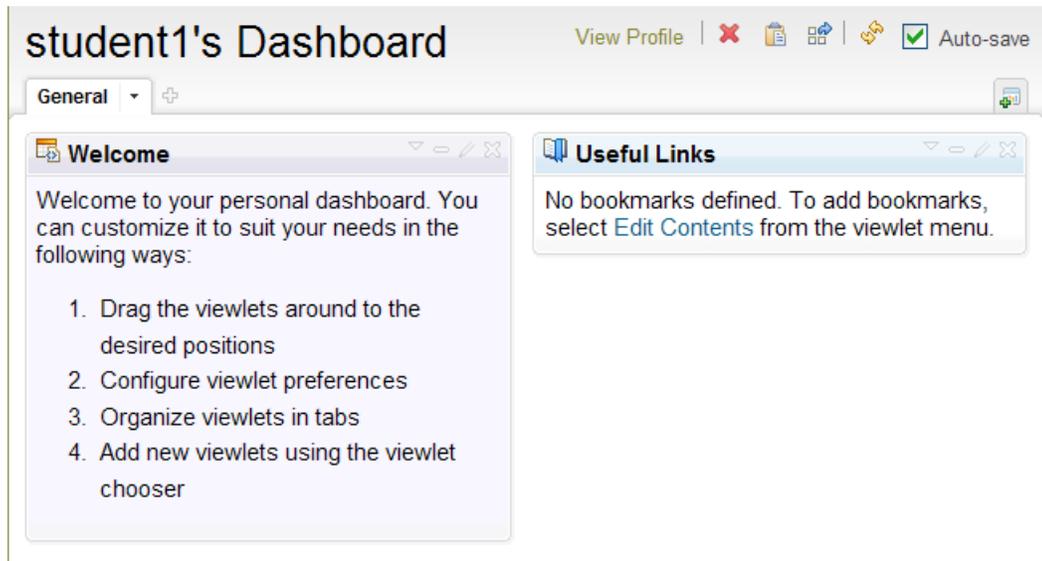
### Dashboard Template and Layout

When a new dashboard is created, a default template for the particular dashboard type is used. Dashboards:

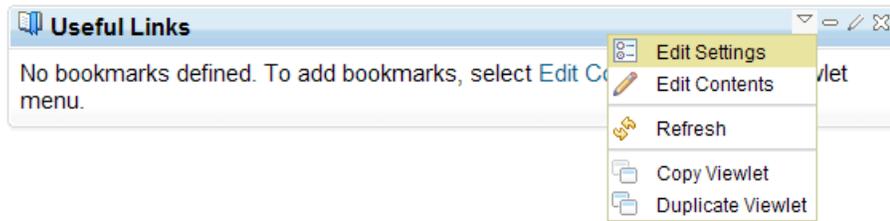
- Have a title and one or more page tabs.
- Page tabs have one to three columns.
- Columns contain basic building blocks – viewlets.
- Viewlets are rectangular widgets in Jazz dashboards that perform a useful function. The actual content shown in a viewlet depends on its type as well as the way the particular instance has been configured.
- Viewlets can be configured in a number of ways, showing dramatically different information as a result.

Dashboard Templates can be modified in the project's process specification.

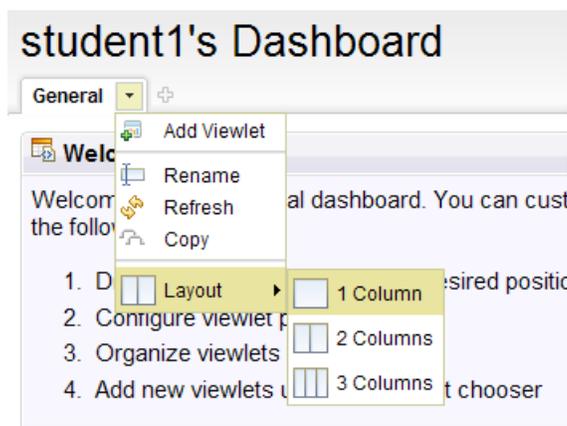
- \_\_4. As mentioned earlier, dashboard templates are available out-of-box. Your personal dashboard was pre-populated using the default template defined on the server for a private dashboard. Your private dashboard initial content is intended to provide you with a starting point and can be changed in a number of ways.
- \_\_a. The default dashboard displays the **Welcome** and **Useful Links** viewlets.



\_\_b. Also, notice the menu icon (  ) available on each viewlet. Depending on the type of the viewlet you will also see the edit content icon (  ).



\_\_c. Each tab also has a customization menu.

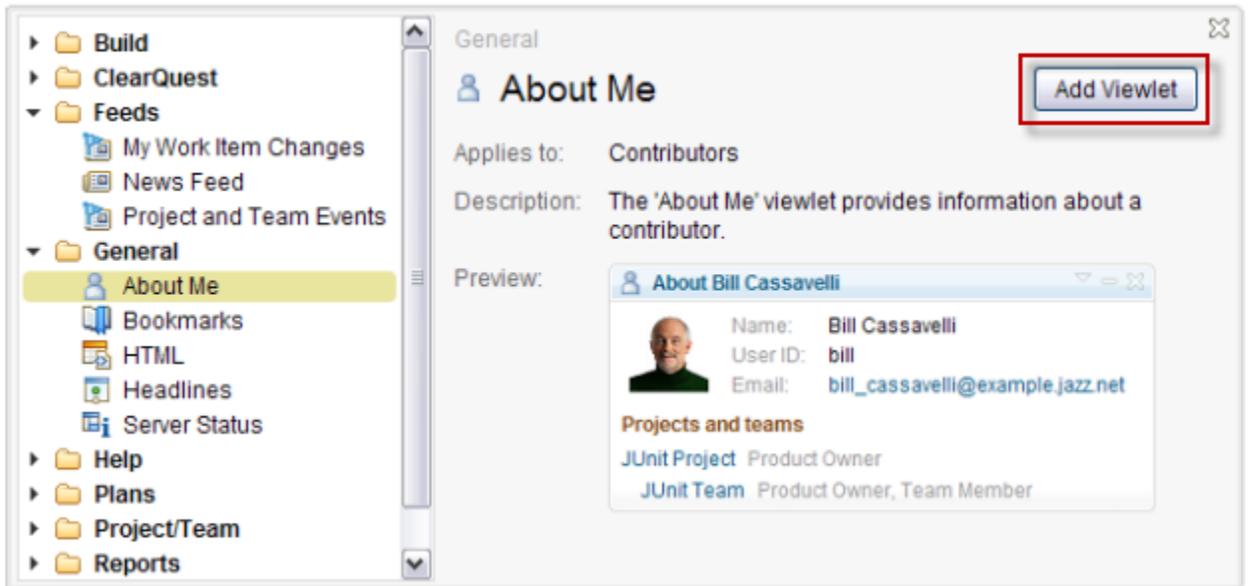


\_\_5. Add a viewlet to the **General** tab. Select the link **Add Viewlet** icon (  ) in the upper right corner of the dashboard

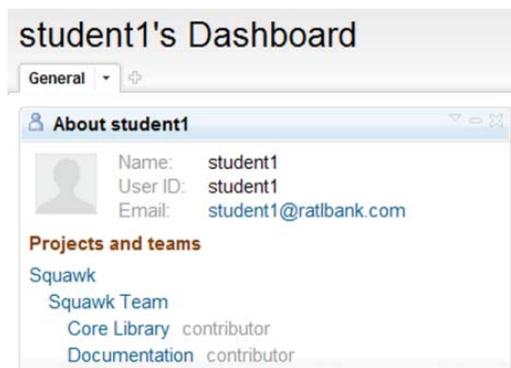


(or you could have chosen the **Add Viewlet** action (  **Add Viewlet** ) in the menu drop down shown in the image above). A **Viewlet Chooser** shows you the pre-authored customizable viewlets grouped by categories. When a viewlet is selected, a description and a preview image are shown.

\_\_a. Select the **About Me** viewlet. Click the **Add Viewlet** button on the right.



The viewlet appears in your dashboard.



- \_\_b. Select the **Work Item Statistics** viewlet. Click the **Add Viewlet** button on the right. The Work Item Statistics viewlet appears on your dashboard.

The screenshot shows a dashboard titled "student1's Dashboard" with a navigation menu on the left and several viewlets on the right. The "Work Item Statistics" viewlet is highlighted with a red box. It includes an "Add Viewlet" button, a description, and a preview of a pie chart. The pie chart is titled "Resolved by me (12) Type" and shows a distribution of work item types: Enhancement (5), Defect, Plan Item, Retrospective, Task, and Track Build Item. Below the viewlet, there are other viewlets like "Useful Links" and "Welcome".

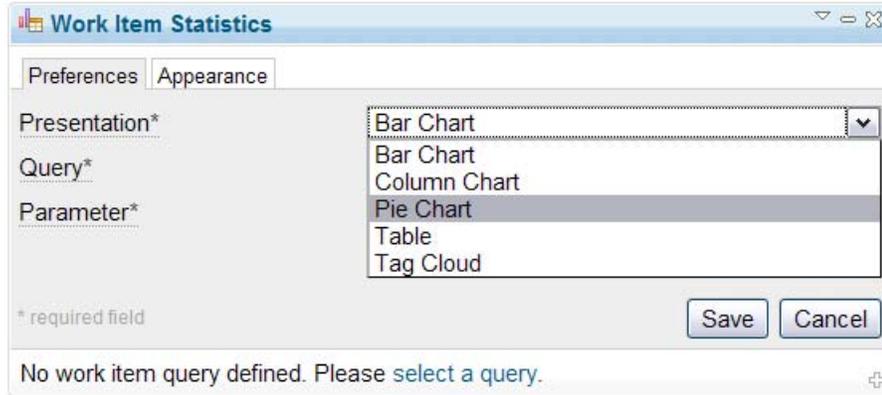
- \_\_c. Close the **Viewlet Chooser**.

The screenshot shows the "Viewlet Chooser" dialog box. On the left, there is a list of folders: Build, ClearQuest, Feeds, General, and Help. On the right, there is a message: "Select a viewlet from the chooser to add it to your dashboard." A red box highlights the close button in the top right corner.

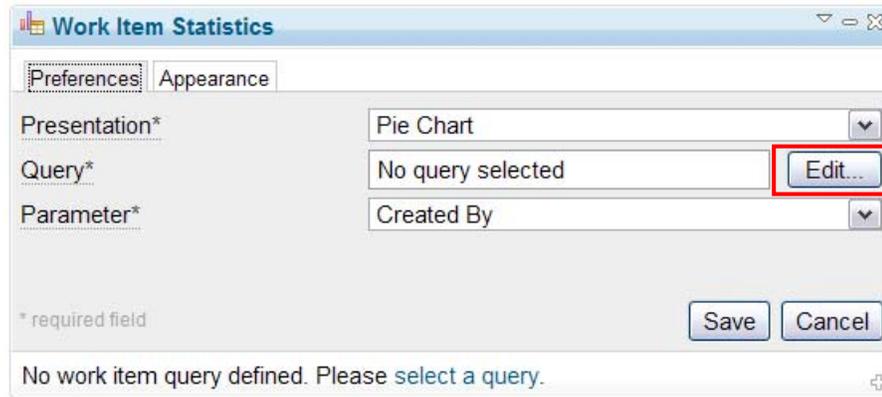
- \_\_d. The next step for this particular viewlet type is to customize it by selecting a query. You can select an out-of-box query or any query your team has authored. Click the **select a query** link.

The screenshot shows the "Work Item Statistics" viewlet. The text "No work item query defined. Please select a query." is displayed. The link "select a query" is highlighted with a red box.

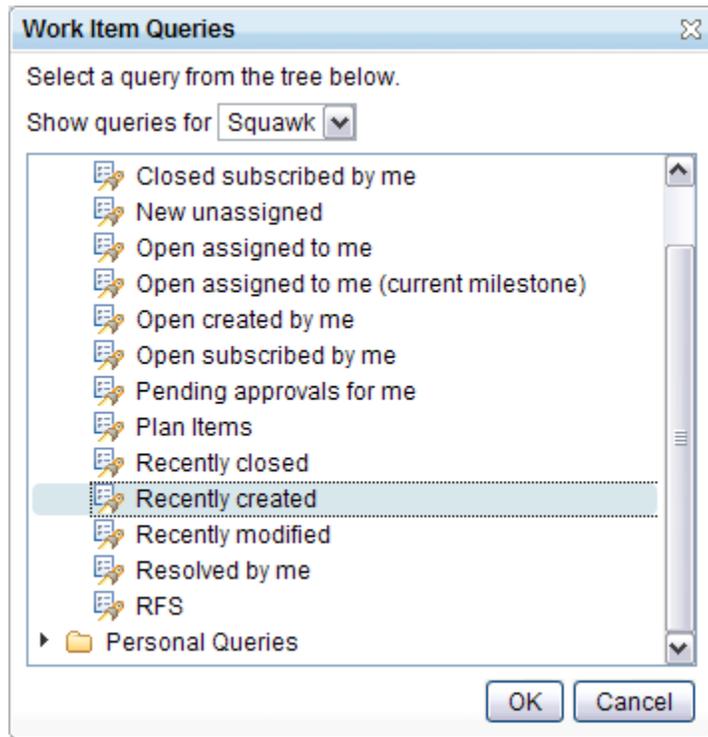
\_\_e. Select **Pie Chart** for **Presentation**.



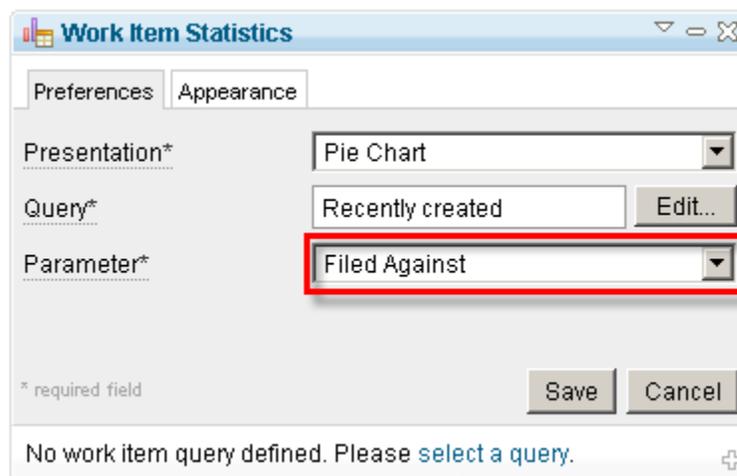
\_\_f. Click the **Edit** button in the **Query** field.



\_\_g. In the dialog select **Recently Created**. Click **OK**.

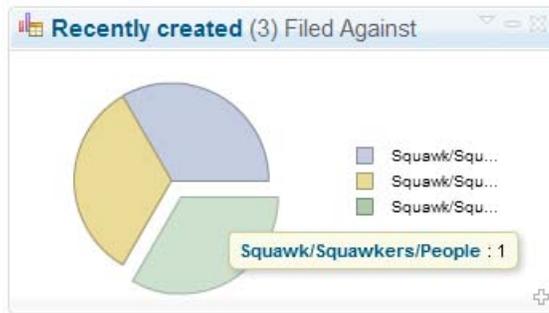


\_\_h. Select **Filed Against** for **Parameter**. Your viewlet parameters should look like this.



\_\_i. Click **Save**.

- \_\_j. Inspect the graph. This is a live query, so as new work items are created your dashboard will be updated.



- \_\_k. Click any portion of the pie chart to retrieve the detailed list of work items in a new browser page.

### Recently created

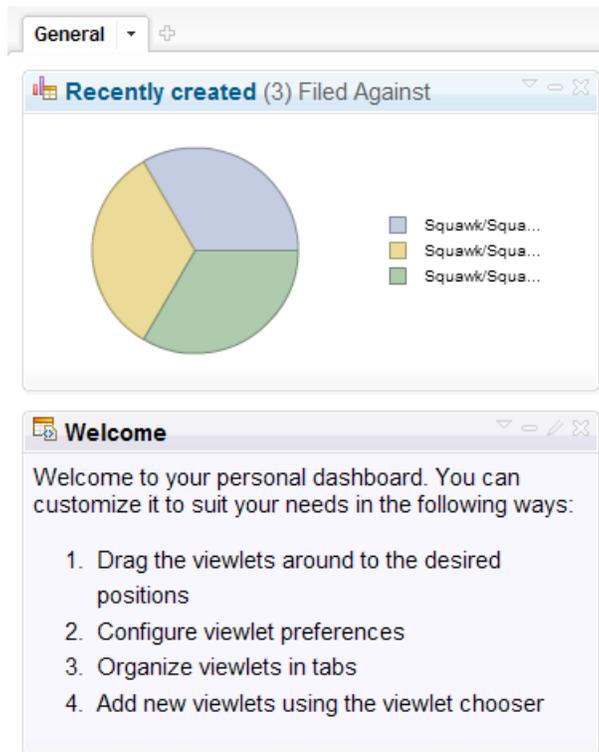
Show All  Items Per Page Previous | 1 of 1 | Next  Type Filter Text

Type	Id	Summary	Owned By	Status	Priority	Severity	Modified Date
	553	Student 1 Squawker Implementation	student1	New			6 minutes ago

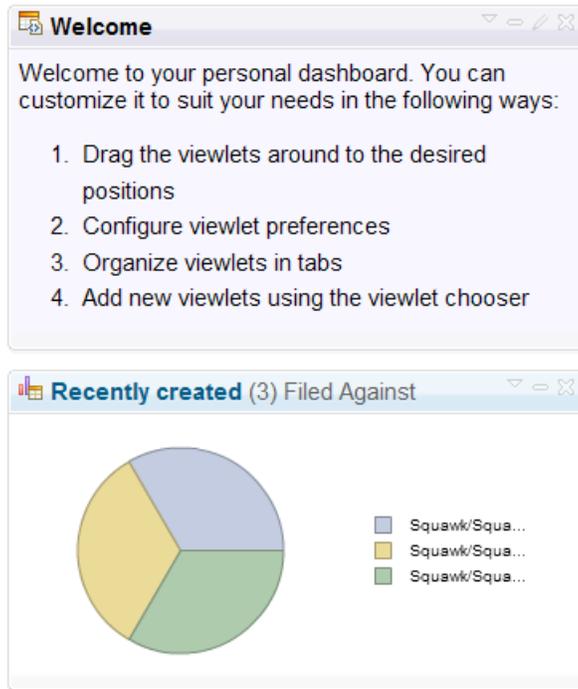
- \_\_l. Click the icon in your browser toolbar to go back to the Dashboard view.

- \_\_m. Click on the title bar of the new viewlet then drag and drop it under the **Welcome** viewlet switching their position in the page.

Before:



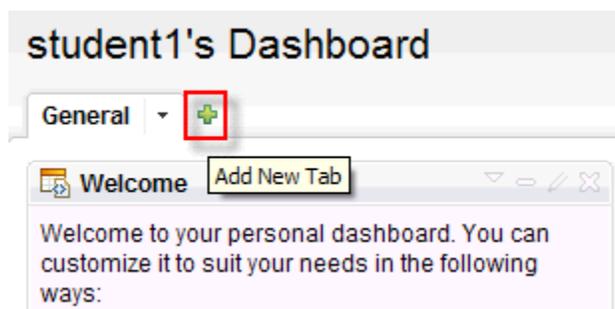
After:



 **Configuring viewlet chooser entries**  
A user that has permission to modify the process can add custom viewlet chooser entries in the project area process specification using the Eclipse IDE. This is particularly handy for RSS or Atom feeds that are only relevant within the context of a specific project.

\_\_6. Add a new tab. Select the link **Add New Tab** next to the General tab. If you do not have an external network connection you may skip this step and go directly to Step 7.

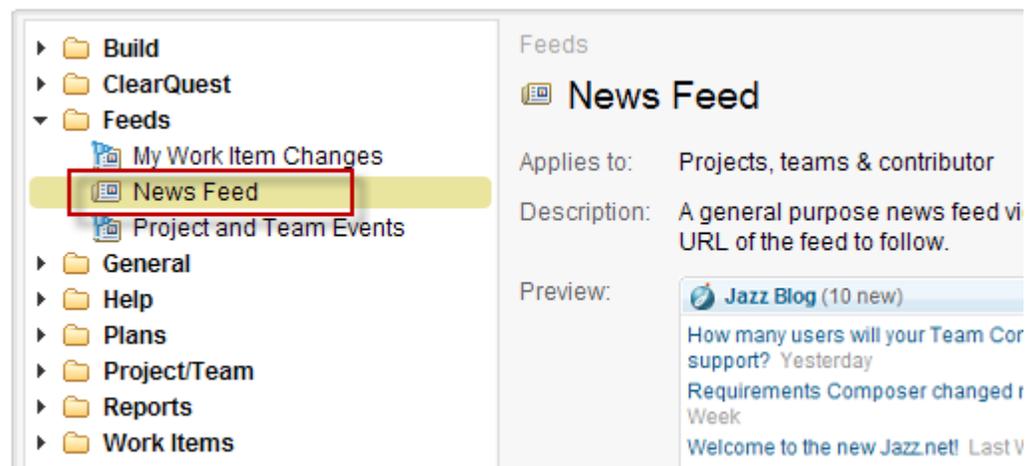
\_\_a. Click the **Add New Tab** button to add a new tab to your dashboard.



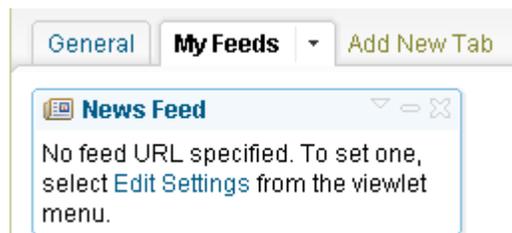
- \_\_b. Click on the **New Tab** title and rename it to *My Feeds*.



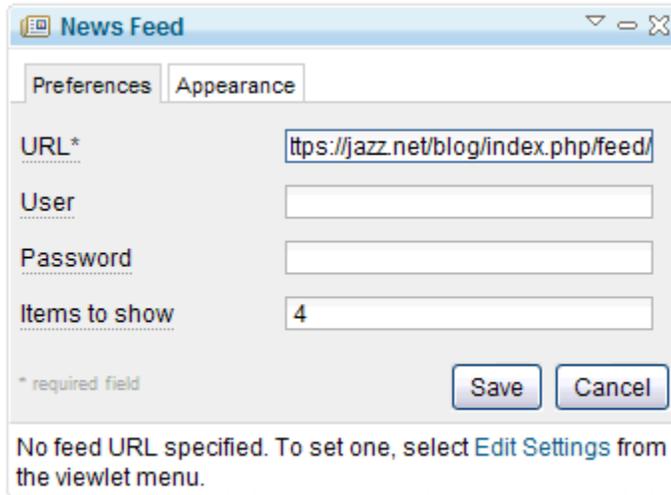
- \_\_c. Click the link **Add Viewlet** in the text of the new tab of the dashboard.
- \_\_d. Select the **News Feed** viewlet in the **Feeds** category. Click the **Add viewlet** button on the right.



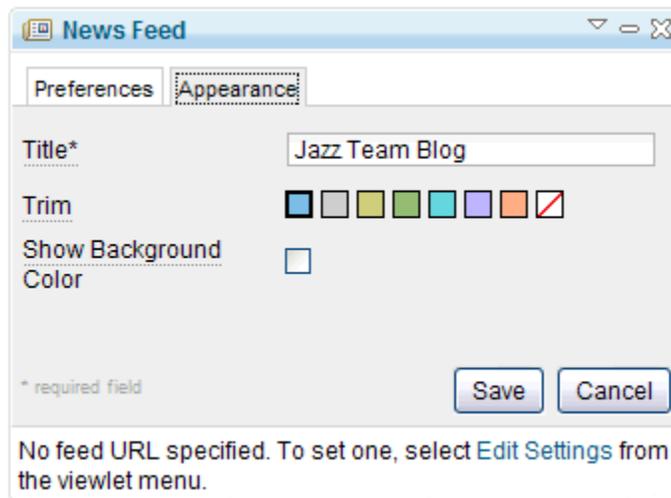
- \_\_e. Close the **Viewlet Chooser** view.
- \_\_f. Click the **Edit Settings** link to configure a **News Feed**. For example, let's try the **Jazz Team Blog** on Jazz.net.



\_\_g. Enter `https://jazz.net/blog/index.php/feed/` for **URL**.

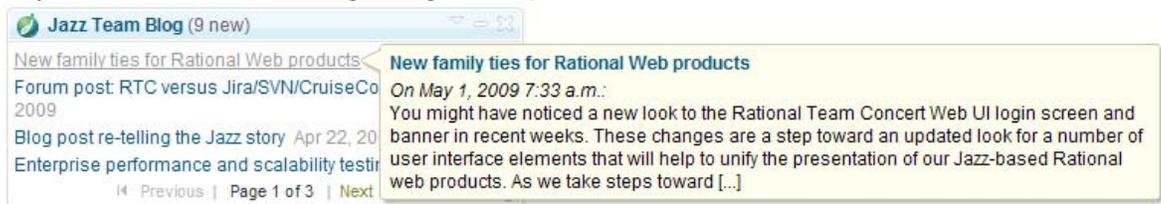


\_\_h. Click the **Appearance** tab. Enter `Jazz Team Blog` for the **Title**.



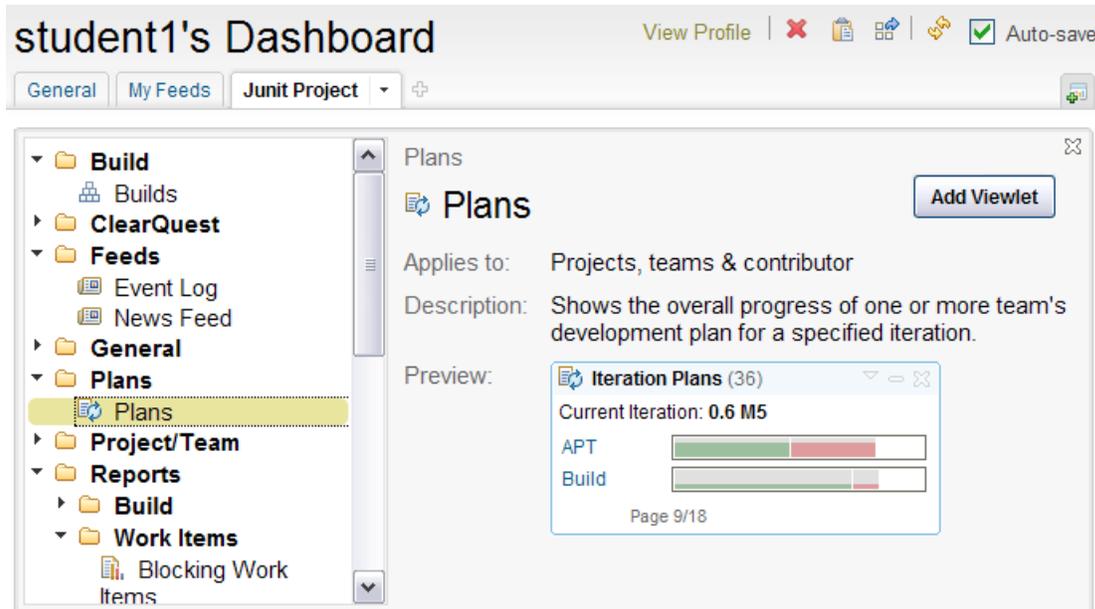
\_\_i. Click **Save**.

You now have access to the last 4 posts on the Jazz Team Blog. Place your mouse over any title to visualize the beginning of the post:



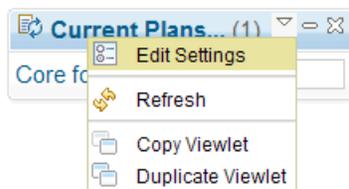
\_\_7. Dashboards can provide information that spans project areas. Now you will add a viewlet from the `JUnit Project` project area to your dashboard.

- \_\_8. As you did above create another tab named `JUnit Project`. Click the **Add New Tab** button (+) next to the `My Feeds` tab. Click on the **New Tab** title and rename it to `JUnit Project`.
- \_\_9. As before click the **Add viewlet** button on the right. Select **Plans** as seen here and click **Add Viewlet**.

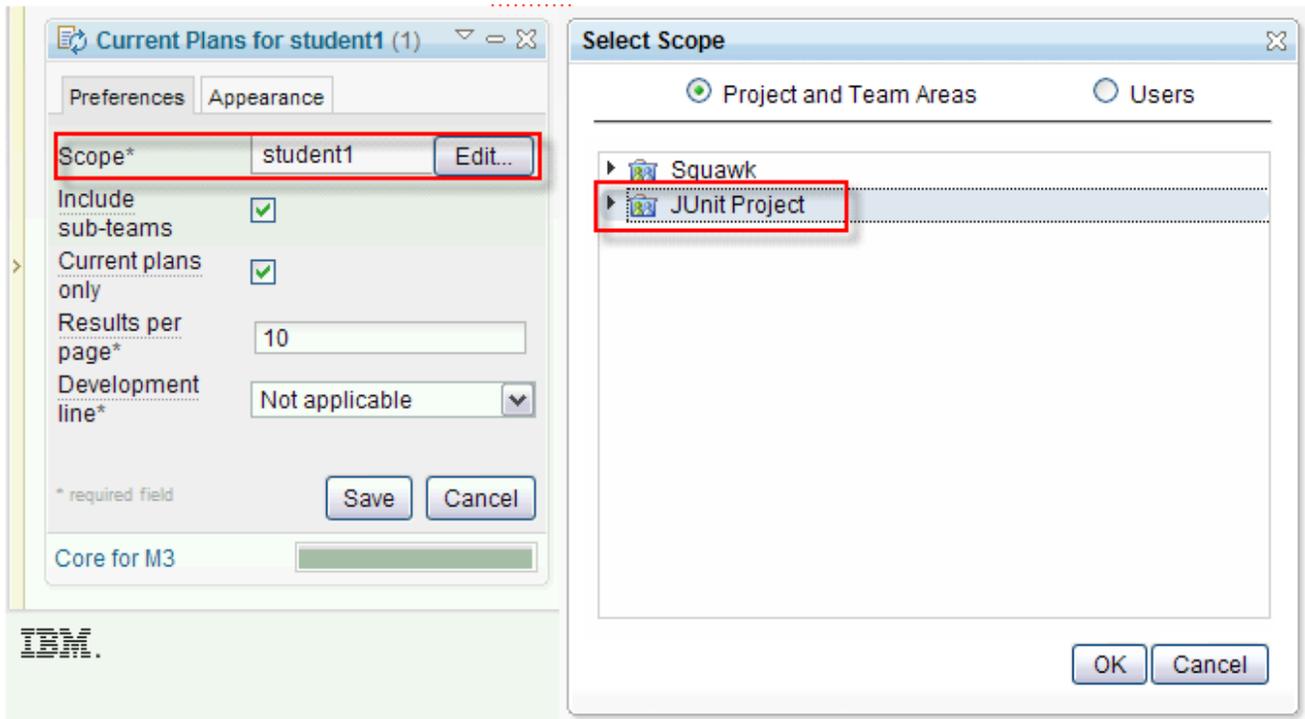


Leave the viewlet chooser open.

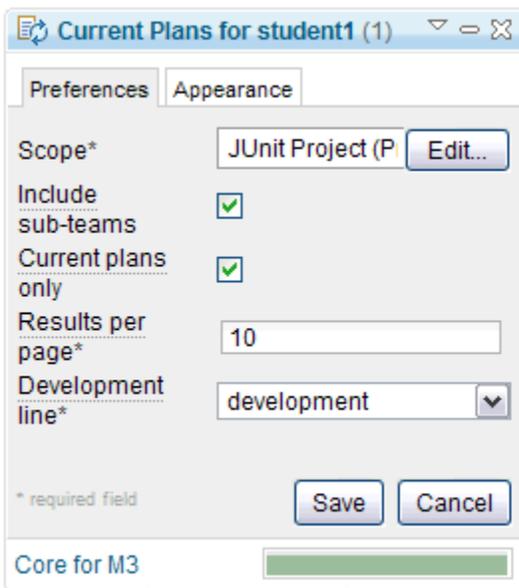
- \_\_10. In the drop down list click on **Edit Settings**.



- \_\_11. Click on the **Edit** button for the **Scope** attribute. In the resulting **Select Scope** dialog click on **Projects and Team Areas** and select **JUnit Project**. Click **OK**.



- \_\_12. Click **Save** in the viewlet.



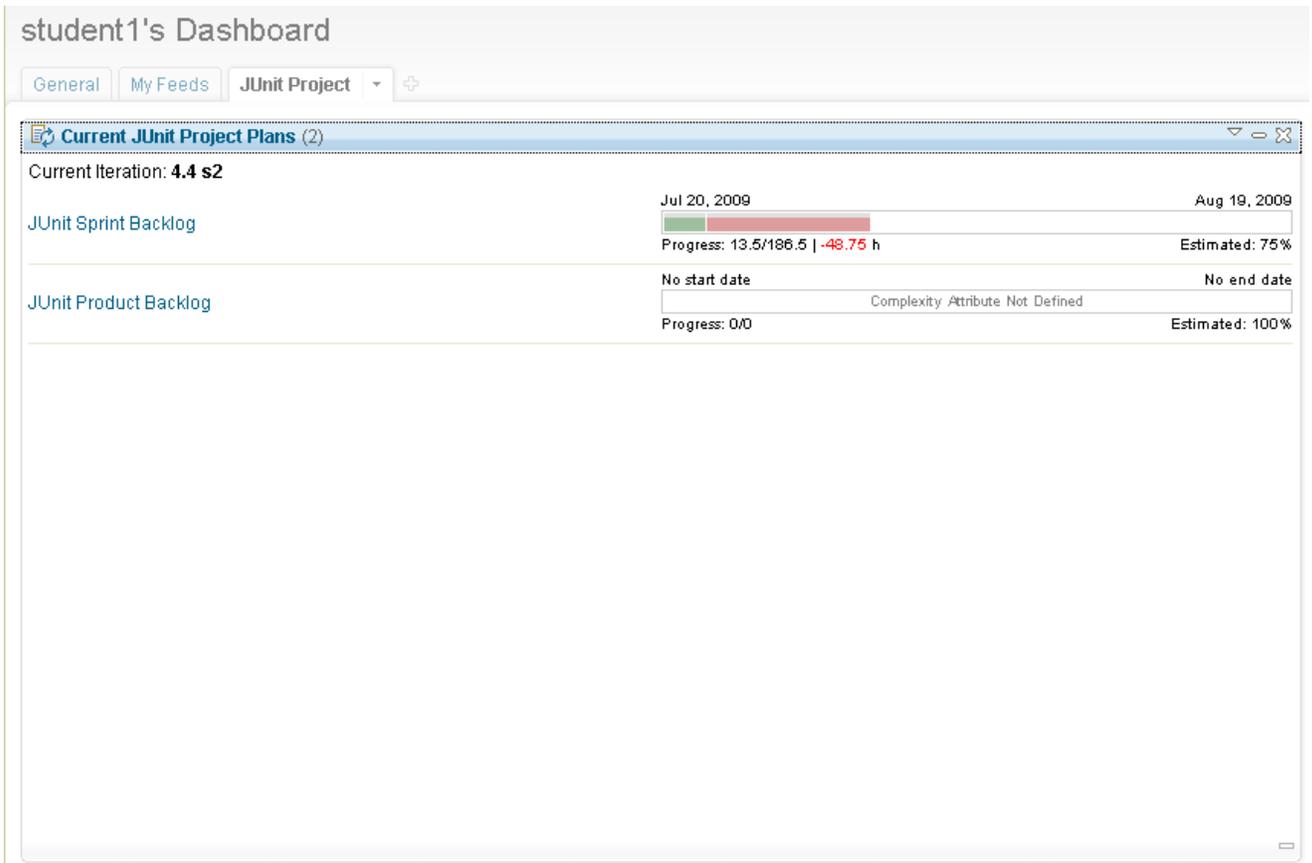
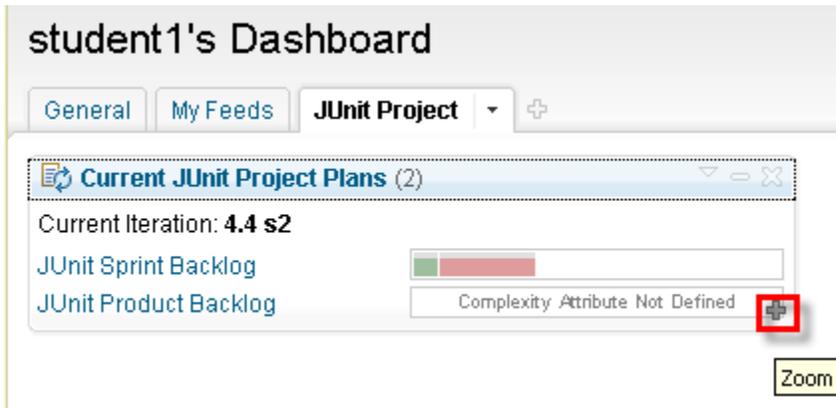
\_\_13. A summary of the status of the JUnit Project plans is displayed.



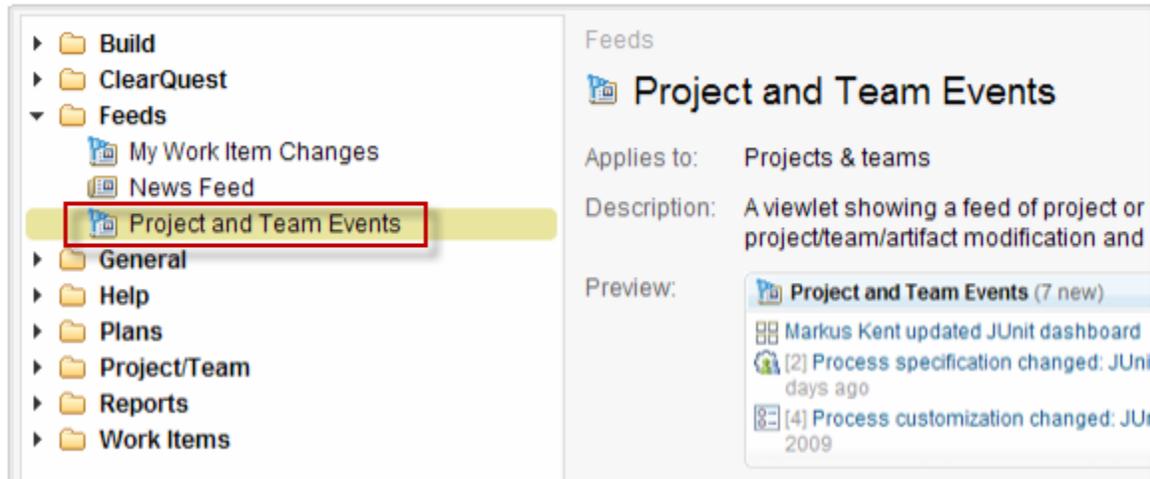
\_\_14. Note that hovering the cursor over the progress bar will show more details on the team's progress against the plan.

**Progress Report**  
 Work Hours Done: 13.5 of 186.5  
 Expected Work Hours: 62.25 (Behind by -48.75 hrs)  
 Items Estimated: 75%  
 Items Completed: 10 out of 22 (45%)

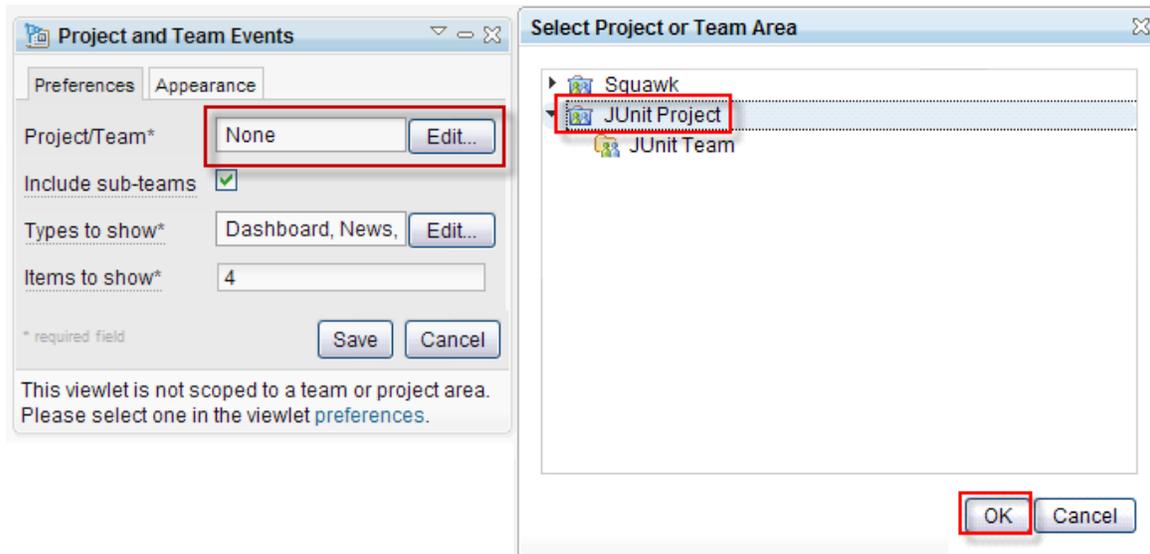
- \_\_\_15. In addition you can “zoom” in on the progress by clicking the  in the bottom right corner of the viewlet:



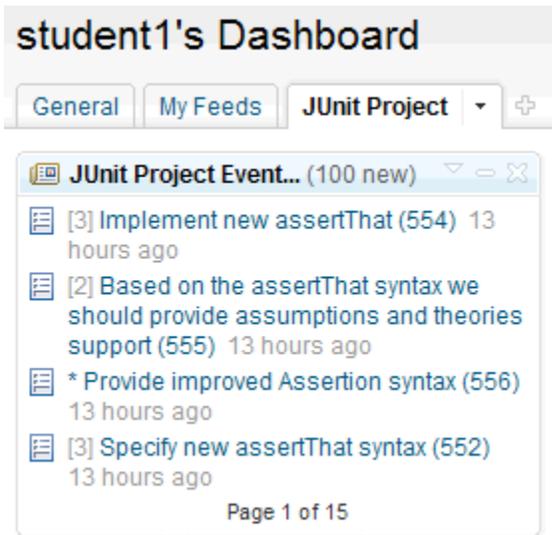
- \_\_16. Now add an **Events** viewlet to this page as follows. Under **Feeds** select **Project and Team Events** and click **Add Viewlet**. Close the viewlet chooser.



- \_\_17. In the viewlet select **Preferences**. Click on the **Project/Team** field click the **Edit** button. In the **Select Project or Team Area** dialog select **JUnit Project**. Click **OK** to close that dialog then click **Save** on the **Preferences** tab.



\_\_18. You should see an events list that looks something like this.



\_\_19. This concludes the Dashboards portion of the lab. Before moving on note that the **Auto-save** dashboard option is enabled. If you log-out or move to a different page, your changes will be preserved.



## 9.2 Exploring Reports using the Web UI

If you are a leader of a project or team, you can create, organize, and share reports to help track the status, progress, and health of the project. Individual developers or team members can also create reports for their personal use or to share with the team.

Note that this section examines reports from the web UI. Similar functionality is available in the RTC Eclipse based IDE.

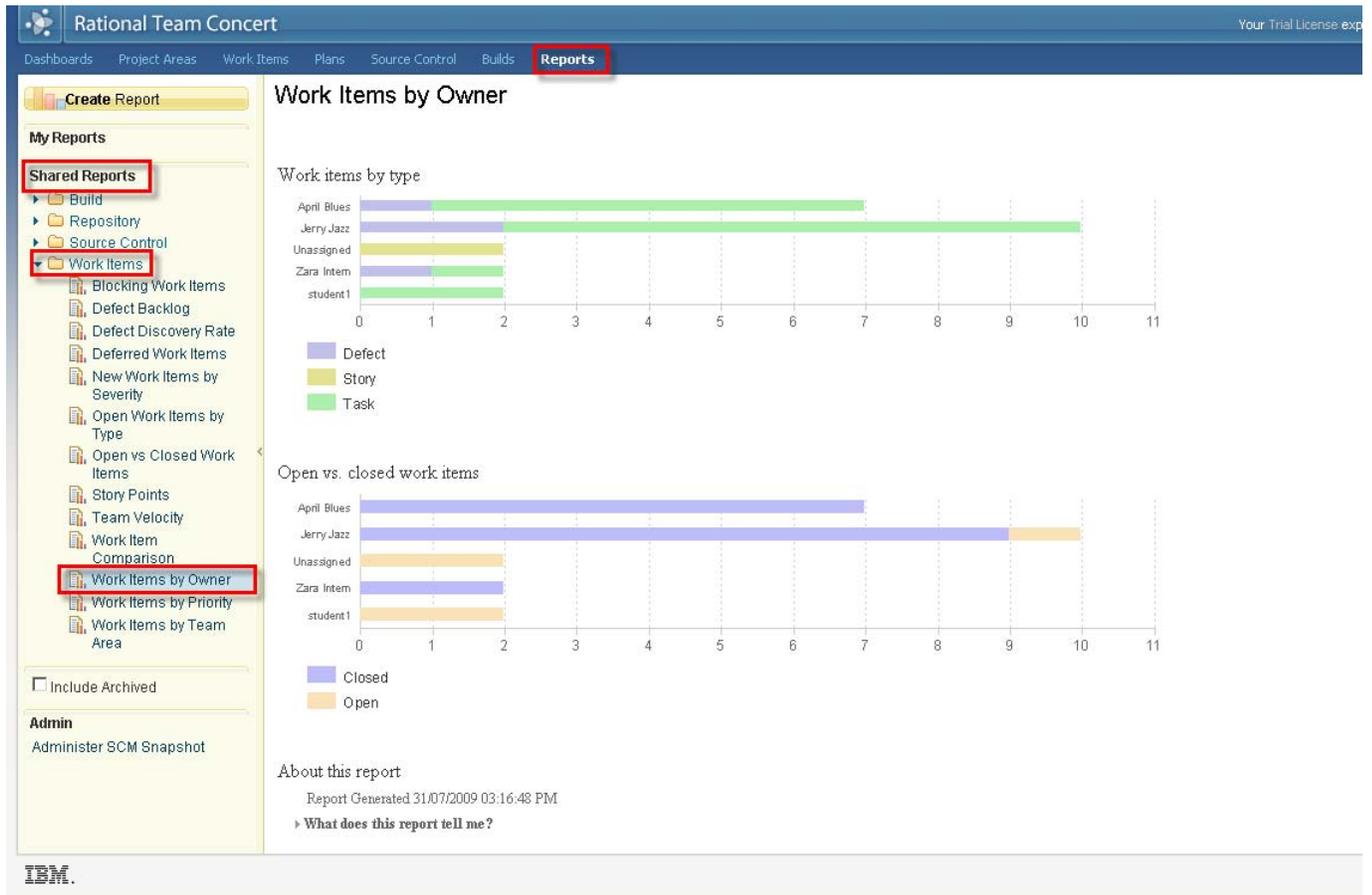
**Overview of the Team Reports Component**

The **data warehouse** portion of the data repository is a storage facility for read-only, historical, and aggregated data optimized for efficient queries and quick response times.

 The **reports engine** is based on the Eclipse BIRT (Business Intelligence and Reporting Tools) project. BIRT reads a **report template** (BIRT report design file), gathers the needed data from the data warehouse and generates a **report** which can be viewed in the Jazz web UI and rich client.

Reports are also made visible on the web UI dashboard using the Trend Reports viewlets.

1. You should still be in the web UI **Dashboards** section. Select **Reports** on the menu bar. Then under **Shared Reports** click on **Work Items**, then **Work Items by Owner**.



2. Expand **What does this report tell me?** to see a description of this report.

**About this report**

Report Generated May 1, 2009 5:11:36 PM PDT

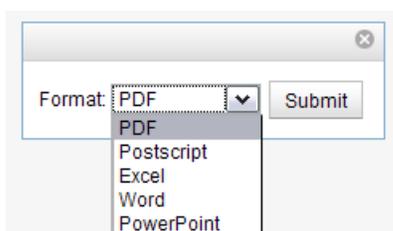
» **What does this report tell me?!**

This report plots the current state of work items in the repository, broken down by owner. This report uses live data from the repository, rather than historical data from the data warehouse.

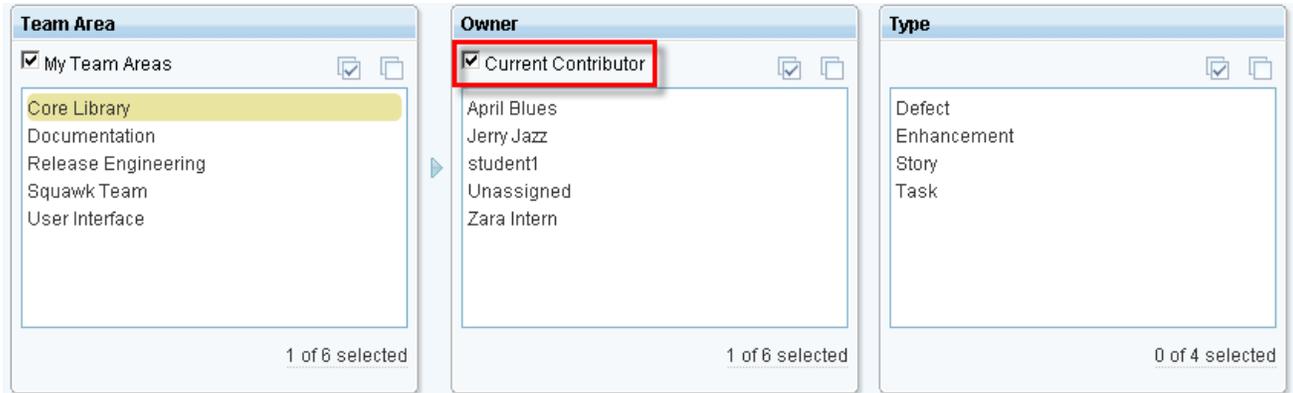
The first chart plots work items broken down by type for each owner. The second chart plots work items broken down by open vs. closed for each owner.

3. Scroll back to the top and inspect the actions in the upper right corner: 

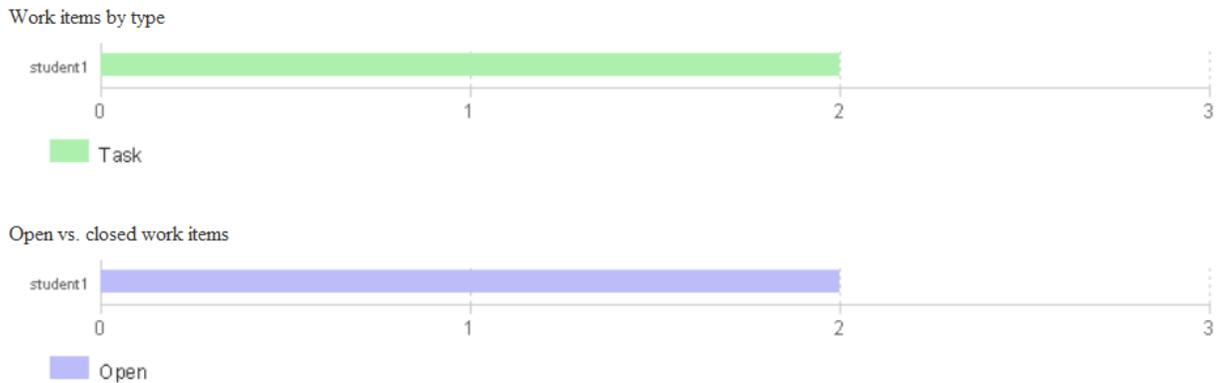
- a. The first action **Export...** allows you to export the report to a variety of formats.



- b. You can make this report your default report with the middle action: . When you set a default report for a specific user, that report opens any time that user is logged into the Web interface and clicks the **Reports** tab.
- c. Selecting this action  allows you to modify the report. Click it now.
- d. The report parameters dialog displays. Adjust the parameters by selecting Current Contributor as the Owner.



- e. Click **Run** and you will see the modified report that shows the Work Items Owned by the Current Contributor i.e. the user you are logged in as



4. Now create a new personal report based on the report above that will be titled: *Work Items Assigned to Me*. Then you will make it your default report.

a. Click **Create Report**.

The screenshot shows the Rational Team Concert interface. The top navigation bar includes 'Dashboards', 'Project Areas', 'Work Items', 'Plans', 'Source Control', 'Builds', and 'Reports'. On the left, a sidebar contains 'My Reports' and 'Shared Reports' (Build, Repository, Source Control, Work Items, Blocking Work Items). The 'Create Report' button is highlighted with a red box. The main area displays a report titled 'Work Items by Owner\*' with a bar chart for 'student1' showing 1 Task.

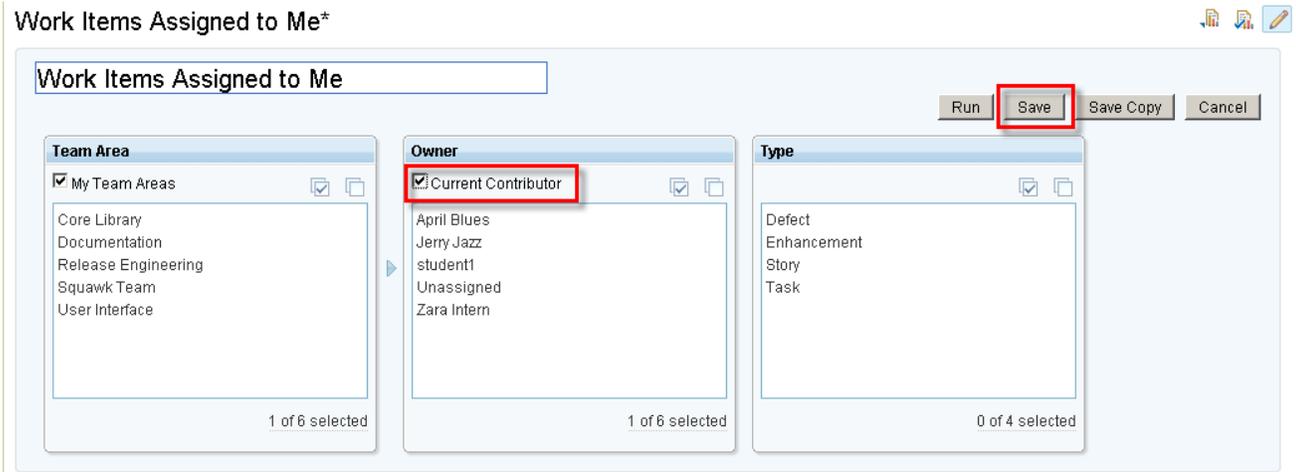
b. Fill this dialog in as show here. Be sure to select the **My Reports** folder. Click on **Save**.

The screenshot shows the 'Report' dialog box. The 'Name' field is 'Work Items Assigned to Me'. The 'Description' field is empty. The 'Folder' is 'My Reports'. The 'Template' list includes 'Work Items by Owner', which is selected. The 'Save' button is highlighted with a red box.

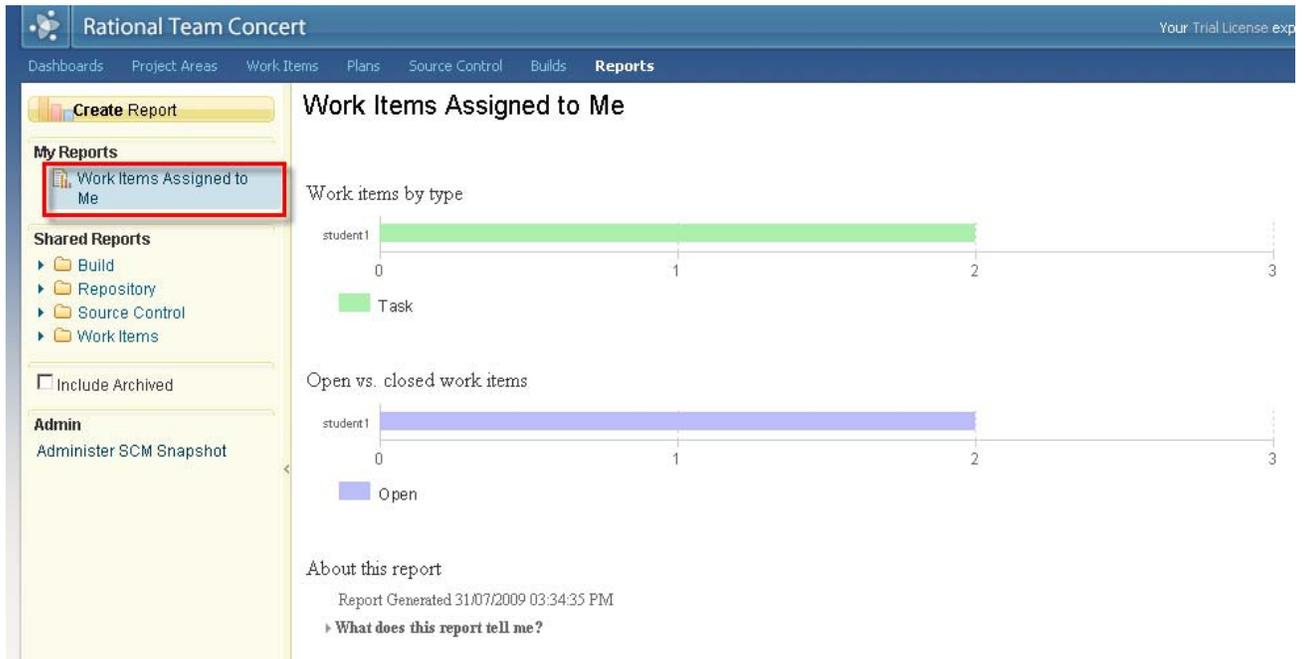
c. Now customize the report by selecting the edit icon in the upper right corner



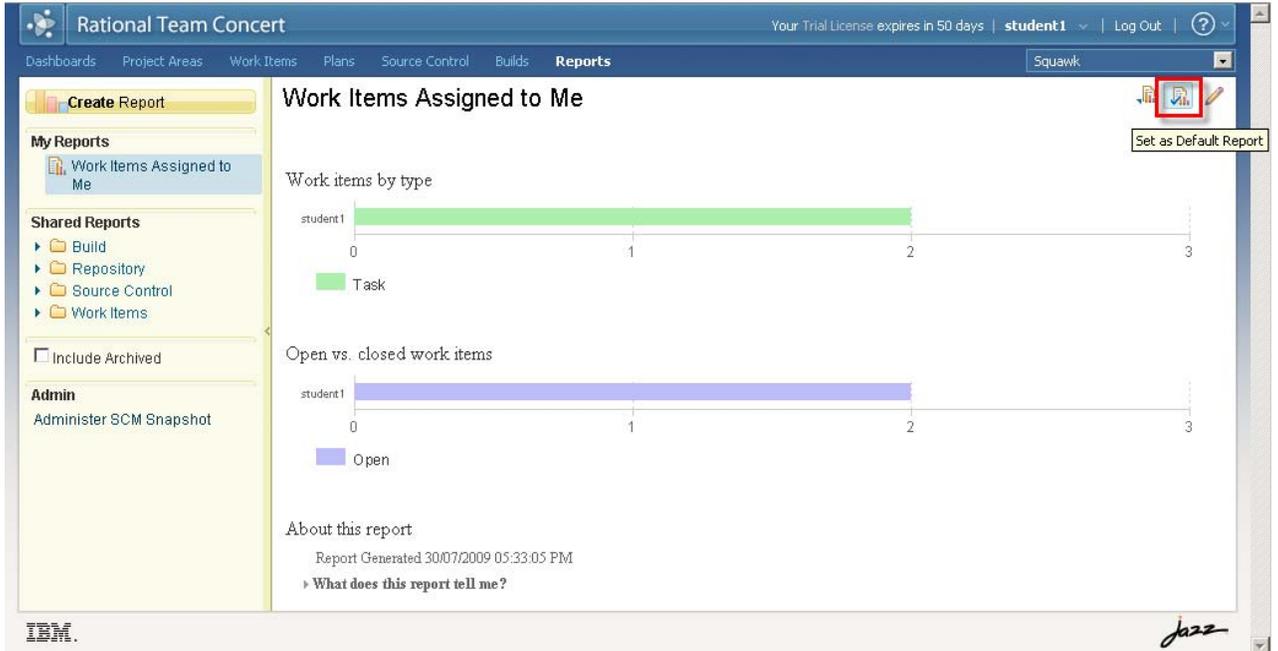
- d. In the report parameter dialog select yourself in the **Owner** box and click **Save**.



- e. Click **Run** to see your report. Note that your report is listed under **My Reports**.



- f. Make this report your default report with the “Set as default” action: .



The screenshot shows the Rational Team Concert interface. The main content area displays the report 'Work Items Assigned to Me'. The report includes two horizontal bar charts for the user 'student1':

- Work items by type:** A green bar representing 'Task' items, with a value of 2.
- Open vs. closed work items:** A purple bar representing 'Open' items, with a value of 2.

Below the charts, the report generation details are shown: 'Report Generated 30/07/2009 05:33:05 PM'. A button labeled 'Set as Default Report' is highlighted with a red box in the top right corner of the report area.

### Conclusion



In this Lab you were exposed to powerful capabilities available out-of-box in Jazz that allows your team to keep track of the status of a project and how these capabilities can enable effective decision-making.

## Lab 12 Be Agile with Advanced SCM

In this lab you will learn how to use advanced capabilities of the Rational Team Concert Source Control Management (SCM) component. You can suspend and resume change sets, when you are interrupted by unexpected (and high priority) work. Also you can share a change set with teammates through repository workspace (not a stream), when you do not want to deliver the change set to the entire team.



### Lab Scenario

So far, you have finished the core implementation of your squawker. To complete the implementation, you will start an important (but not high priority) task to write Javadoc for your squawker. It will take several days to complete since making quality documentation is often more difficult than making quality working code, thus you can't finish the task in a day.

The next day, a teammate suggests a valuable enhancement that affects your code. After team discussion, you will prioritize the enhancement as high, and switch the current work to the new enhancement work. You will then use the suspend capability to revert and store your incomplete work, and start addressing the enhancement.

You will want a peer review of the fix before delivering the change to the team's stream. Share the code with your teammate by sharing a change set on your repository workspace.

After finishing the new high priority work, you will resume the suspended work.

### Important!

Ensure that you carry out this Lab assuming the role and identity of the user you previously created. The instructions for all labs will denote **student<N>**, which you should replace **<N>** with your assigned id number. Sample screenshots in all labs will use the id **student1**. If you are unsure please check with the instructor.



### 12.1 Start working on low priority long task

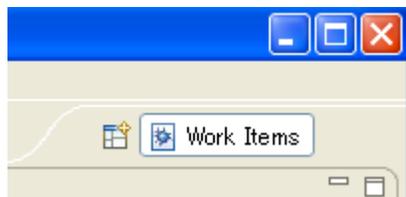
\_\_1. If you don't already have Rational Team Concert running, do the following:

- \_\_a. Open Rational Team Concert by double-clicking the Team Concert shortcut  on the Windows Desktop.

- \_\_b. In the **Workspace Launcher** window type `C:\Workspaces\student<N>` (replacing <N> with your id number). Click **OK**.

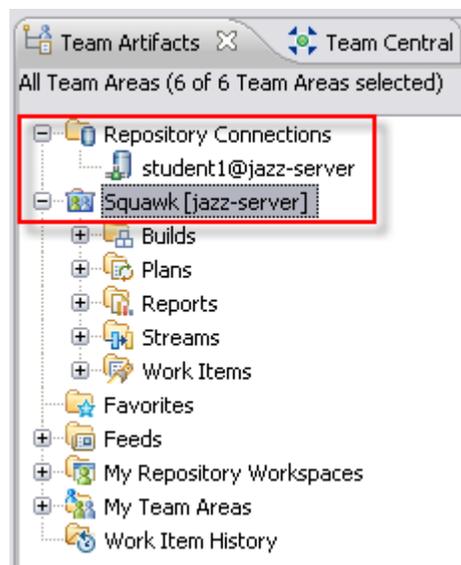


- \_\_c. Switch to the **Work Items** perspective.

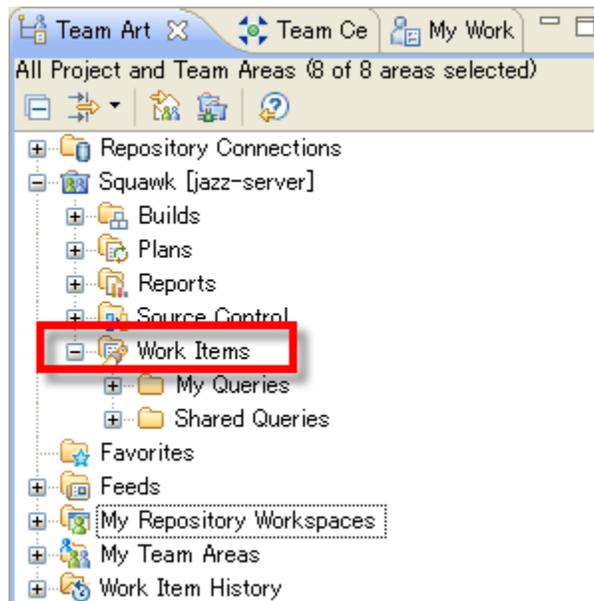


- \_\_2. Create a new **Task** work item.

- \_\_a. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), ensure that you are connected to the **Squawk** project area as `student<N>`



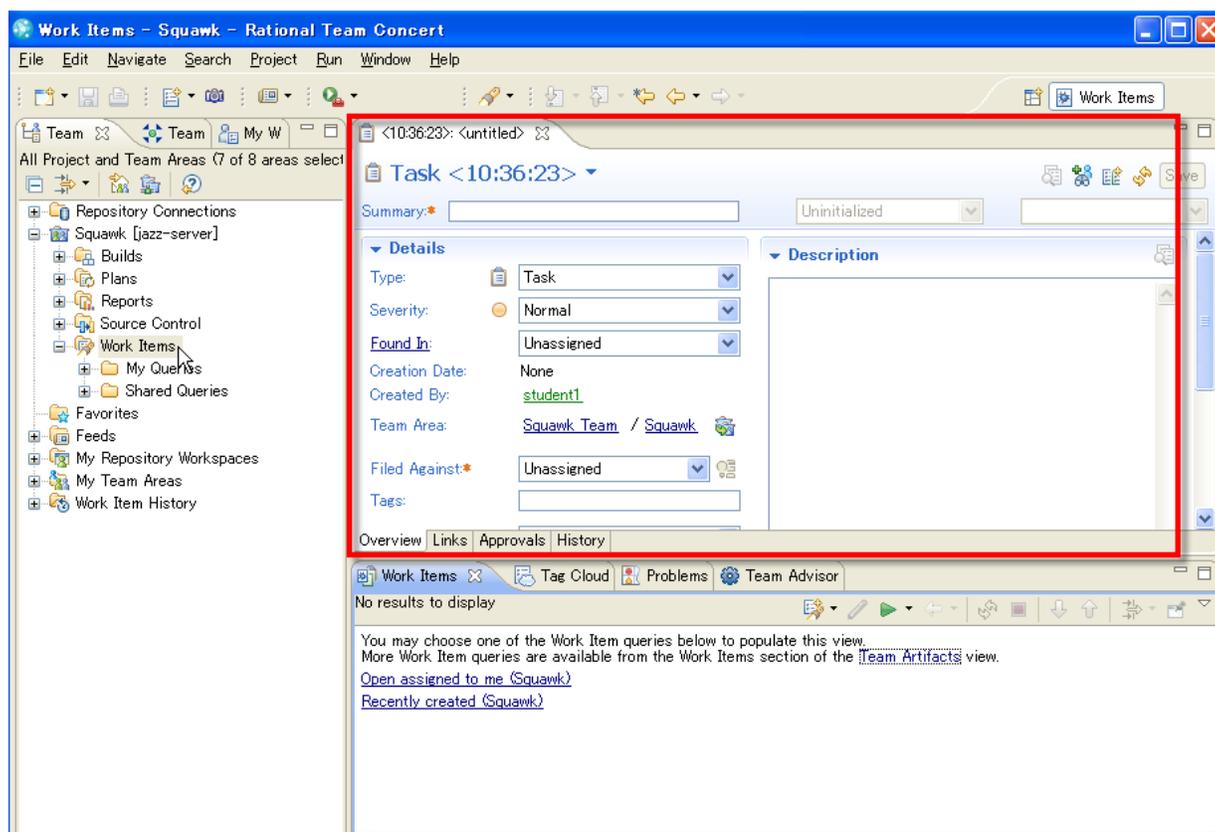
- \_\_b. Expand the **Squawk** project area and then expand **Work Items**.



- \_\_c. Right-click **Work Items** and select **New → Work Item....**

- \_\_d. In the **Create Work Item** window select the **Task**, and click **Finish**.

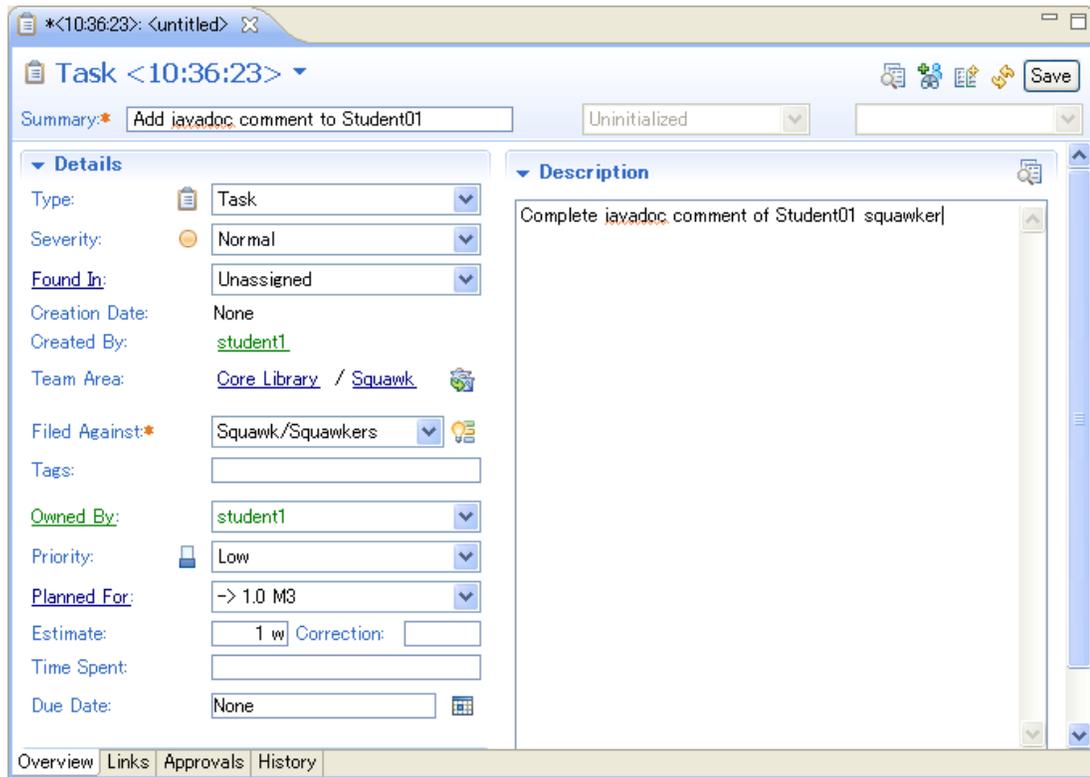
\_\_e. The Work Item editor contains the new task:



\_\_f. For the details of the Task:

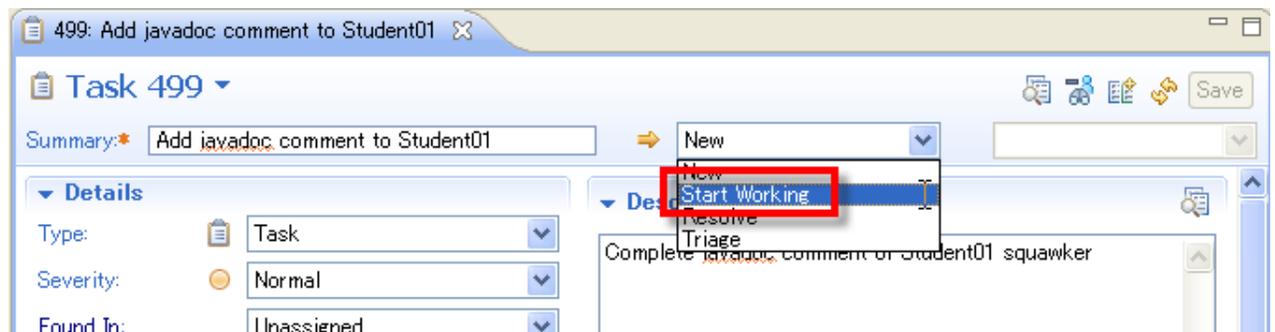
- \_\_i. Type Add javadoc comment to Student<N> in the **Summary** (replace student<N> with student id respectively)
- \_\_ii. Type Complete javadoc comment of Student<N> squawker in the **Description**.
- \_\_iii. Set **Filed Against** to Squawk/Squawkers.
- \_\_iv. Set **Owned By** to student<N>
- \_\_v. Set **Priority** to Low
- \_\_vi. Set **Planned For** to ->1.0 M3
- \_\_vii. In **Estimate**, type 1 week
- \_\_viii. Click **Save** in the right corner of the **Work Item** editor view.

\_\_g. Your completed task should look similar to this:

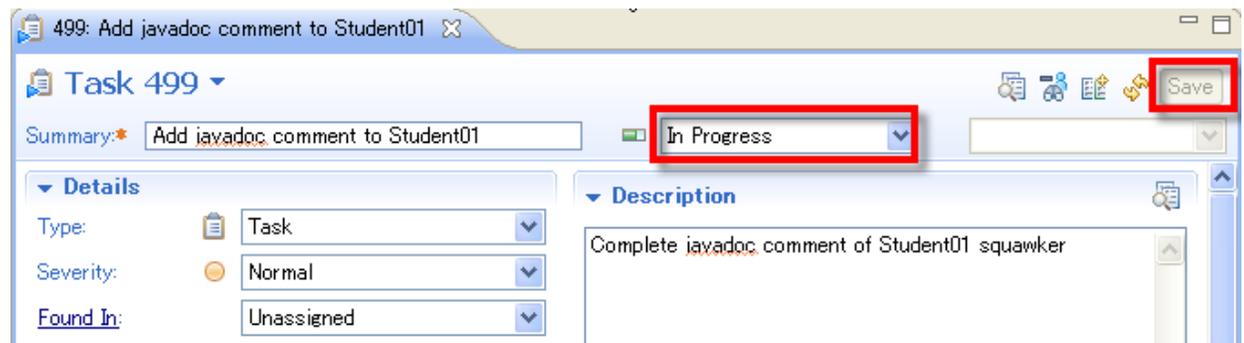


\_\_3. Start working with the task.

\_\_a. On the Work Item Editor, click the status drop down, and select **Start Working**.

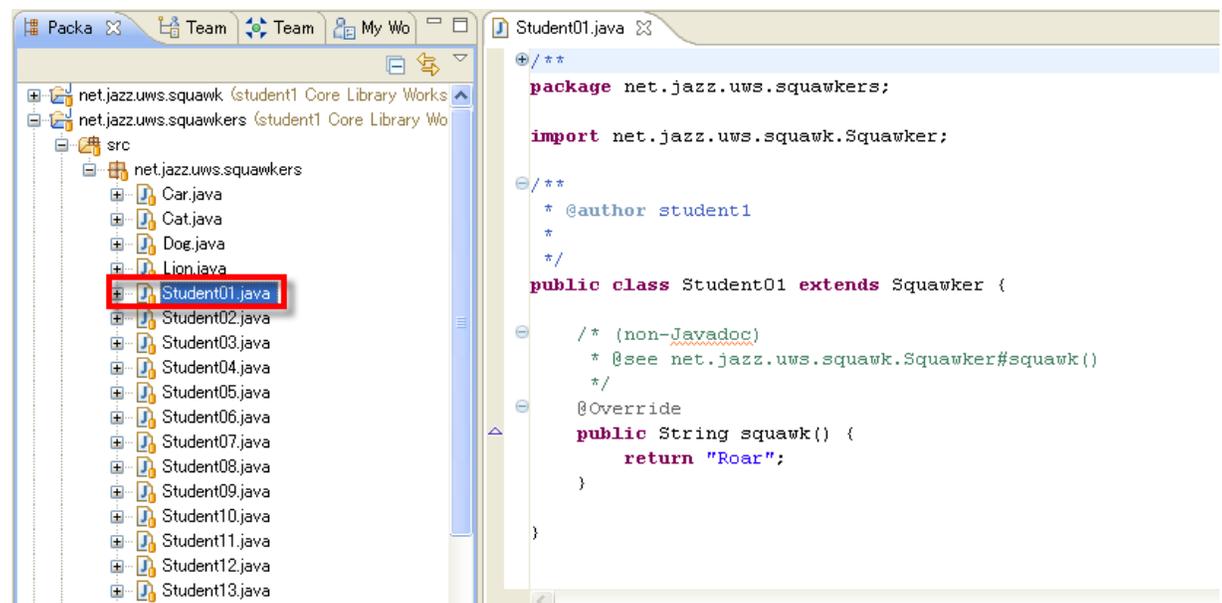


- \_\_b. Click **Save**, then the status will change to **In Progress**.



- \_\_c. Close the Work Item editor.

- \_\_d. In the **Package Explorer** view, expand to the **net.jazz.uws.squawkers/src/net.jazz.uws.squawkers** Java package and double-click the **Student<N>.java** class file which you created in Lab 4 (replace **<N>** with your assigned id number). This will open the Java Editor.



- \_\_e. Modify the Javadoc comment above the class definition. You should not complete the comment at this time.

```

+ /**
package net.jazz.uws.squawkers;

import net.jazz.uws.squawk.Squawker;

/**
 * Student01 goes ... |
 * @author student1
 */
public class Student01 extends Squawker {

    /* (non-Javadoc)
     * @see net.jazz.uws.squawk.Squawker#squawk()
     */
    @Override
    public String squawk() {
        return "Roar";
    }
}
    
```

- \_\_f. Save the class. (Ctrl-S or )
- \_\_g. Close the Java Editor.



**Lab Scenario**

The change you have made so far is not yet completed, thus you will not deliver it to the team’s stream.

The next day, a teammate suggests a valuable enhancement that affects your code. The team discusses the idea – its value and impact to the schedule – with the aid of Agile Planning in Rational Team Concert. After team discussion, you will prioritize the enhancement as high, and switch the current work to the new enhancement work.

You will then use the suspend capability to revert and store your incomplete work then start addressing the enhancement.

## 12.2 Instructor Demo - Team Lead creates a new Story for enhancement

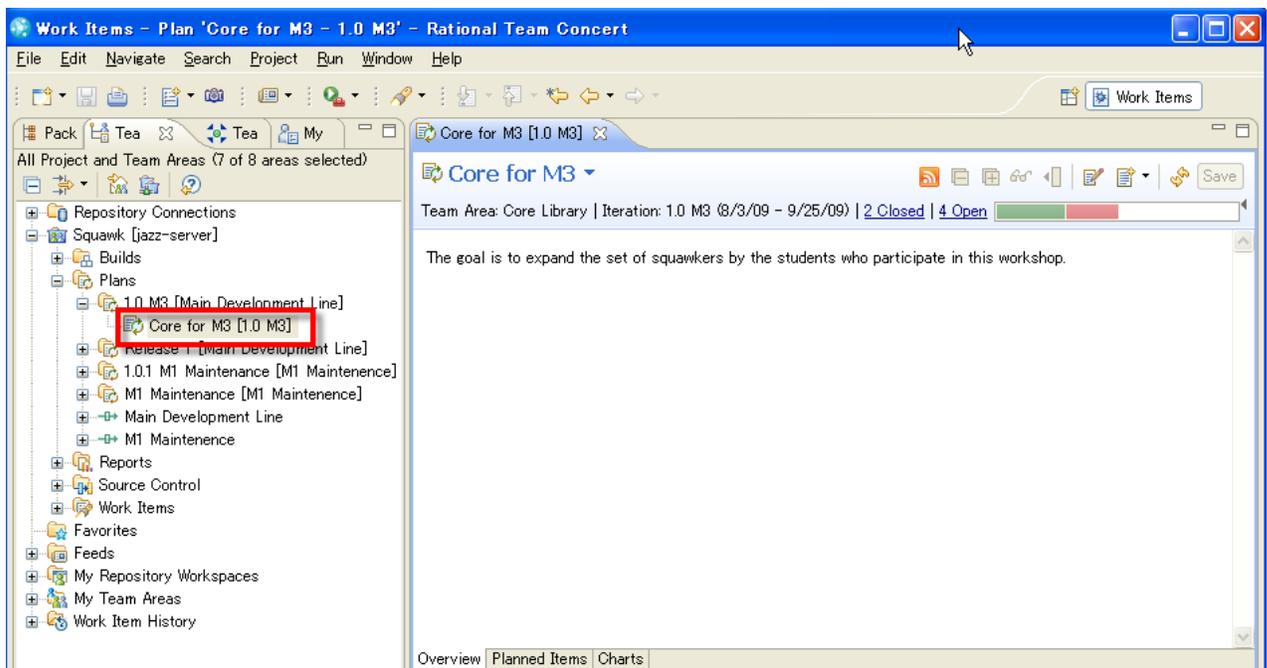


### Instructor Demo

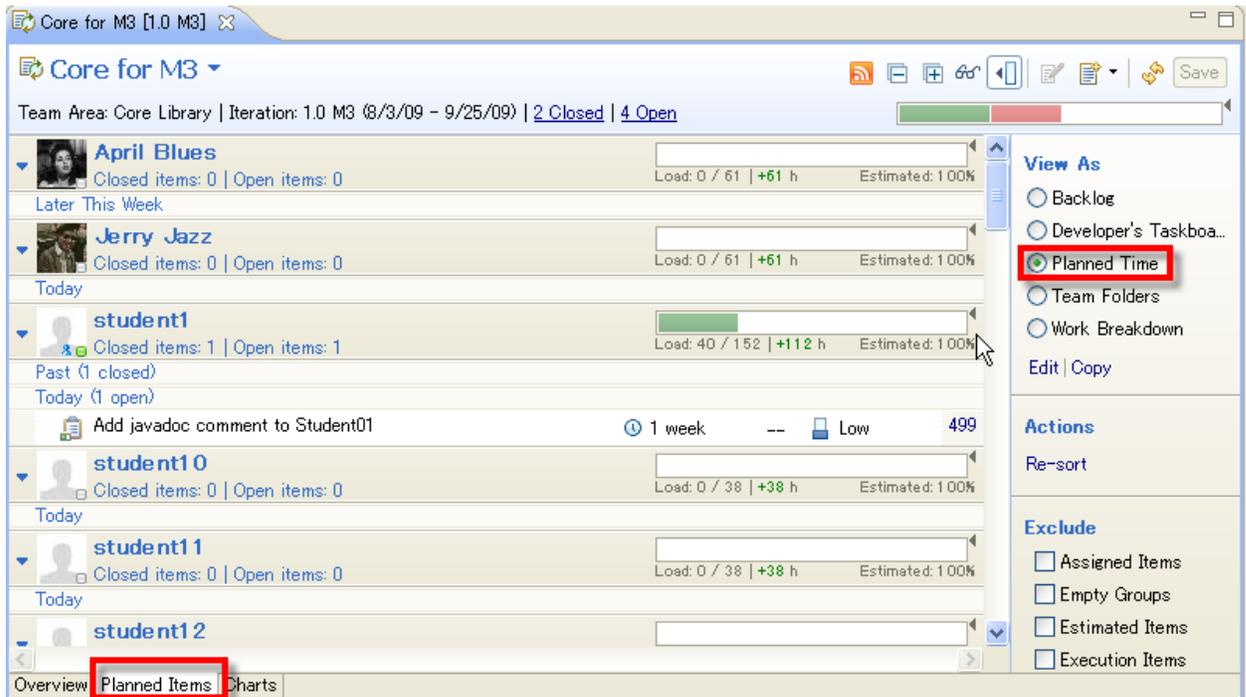
Section 12.2 is performed by the instructor as a demo. In this section, the instructor will create a new story for the enhancement.

## 12.3 Review work load availability of the team

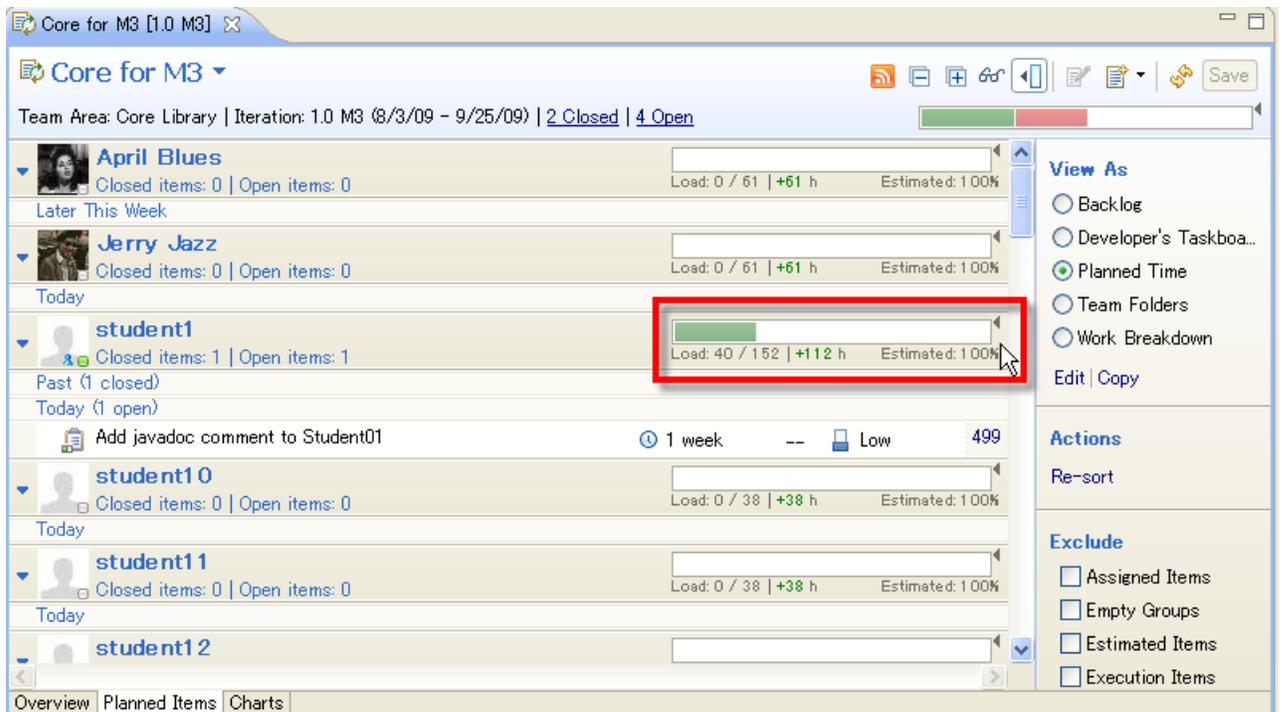
1. In the **Team Artifacts** view, select **Squawk->Plans->1.0 M3 [Main Development Line]->Core for M3 [1.0 M3]**. Double-click to open the **Core for M3** plan.



\_\_2. Select the Planned Items tab on the Core for M2 [1.0 M2] plan, and set View As to Planned Time.



\_\_3. Check the load indicator, and make sure that you can complete the new task within this iteration.





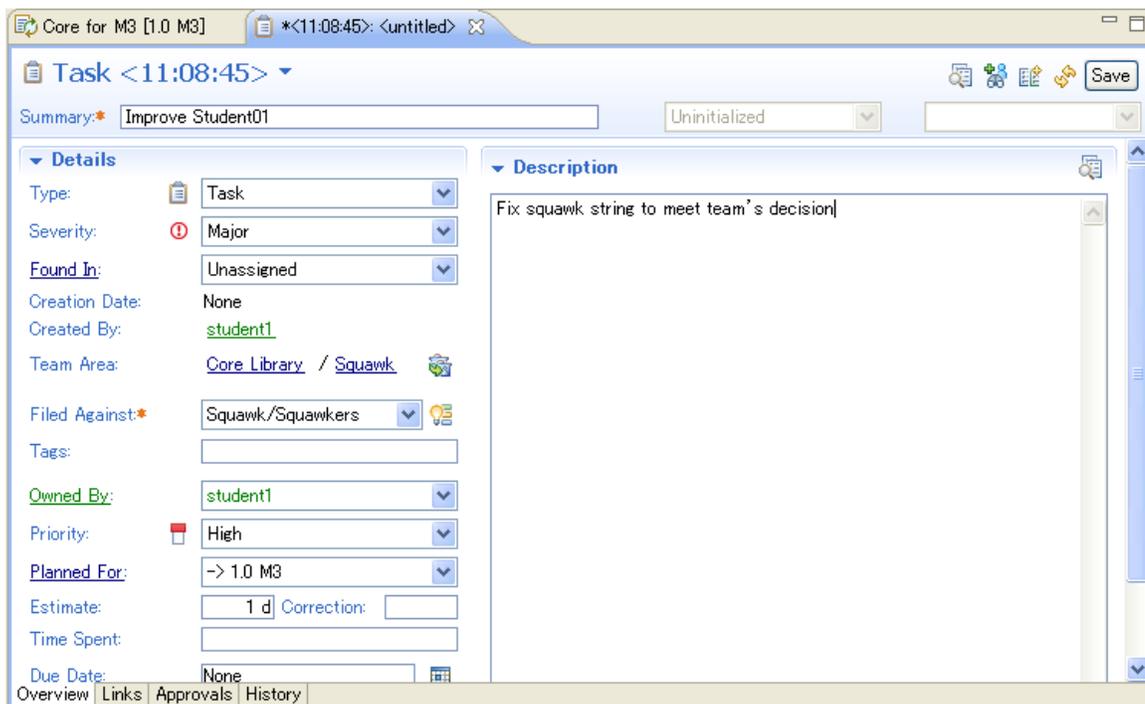
### Progress / Load Indicator

Note that, the load bar, on the top right of the team member folder, visibly shows the work load capacity of the team member. You will see the bar is green and has free (white) space, thus you decide to complete this unexpected defect within this iteration.

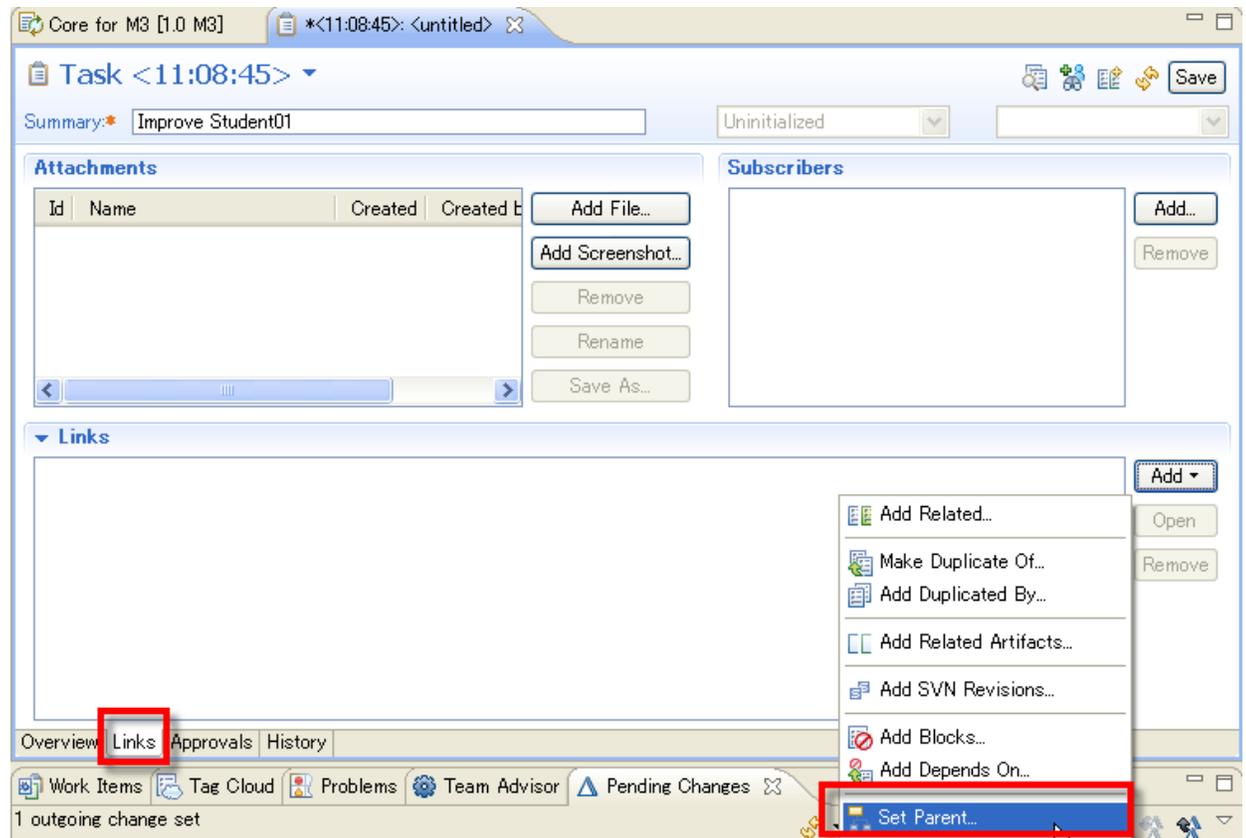
Rational Team Concert provides quick view of the 'real health' of a project, which encourage the team to embrace valuable changes with a realistic probability of success.

- \_\_4. Create a child enhancement for your work to address the new enhancement request
  - \_\_a. In the **Team Artifacts** view, expand the **Squawk** project area and right-click **Work Items** and select **New → Work Item....**
  - \_\_b. In the **Create Work Item** window select **Task**, and click **Finish**.
  - \_\_c. For the details of the enhancement:
    - \_\_i. Type `Improve Student<N>` in the **Summary** (replace `student<N>` with the student id respectively)
    - \_\_ii. Type `Fix squawk string to meet team's decision` in the **Description** field.
    - \_\_iii. Set **Severity** to `Major`.
    - \_\_iv. Set **Filed Against** to `Squawk/Squawkers`.
    - \_\_v. Set **Owned By** to `student<N>`
    - \_\_vi. Set **Priority** to `High`
    - \_\_vii. Set **Planned For** to `->1.0 M3`
    - \_\_viii. In **Estimate**, type `1 day`

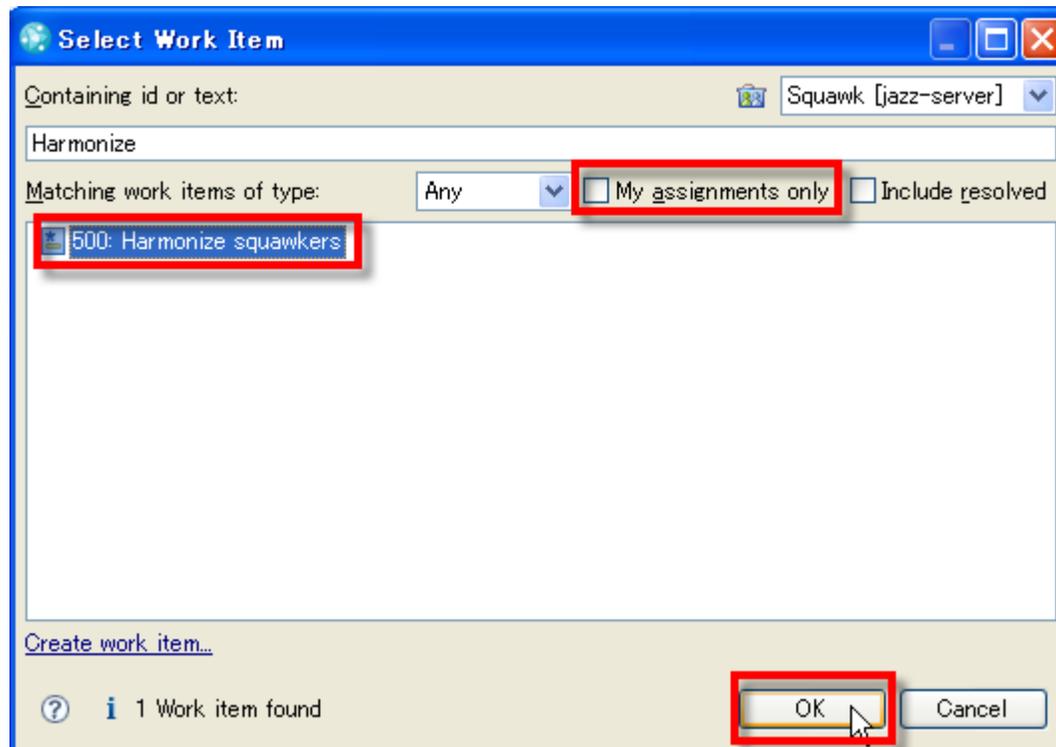
\_\_d. Your completed enhancement should look similar to this:



- \_\_e. Click the **Links** tab of the Enhancement.
- \_\_f. Click the **Add** button in the **Links** area of the Enhancement work item and select **Set Parent...**



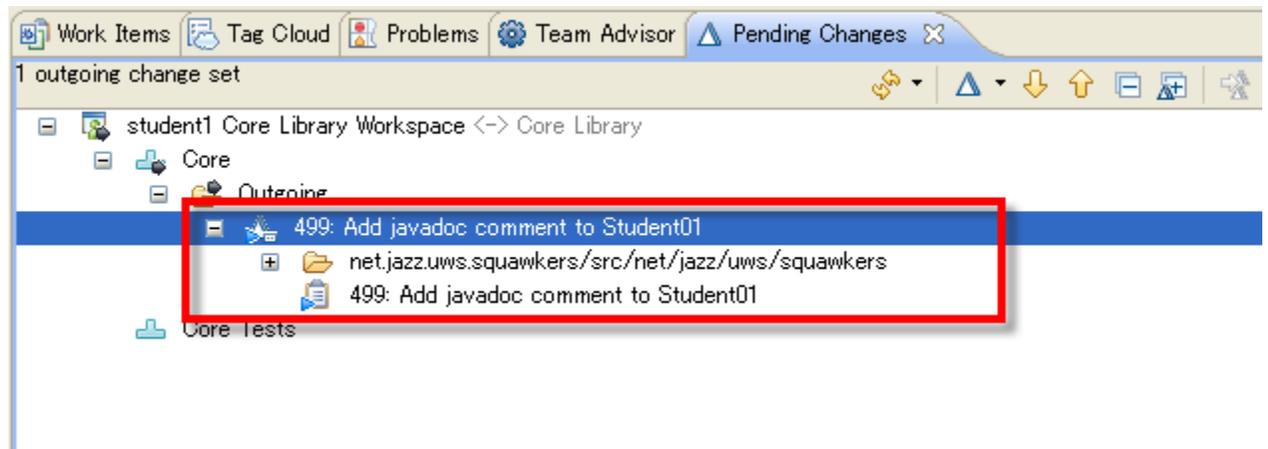
- \_\_g. On the **Select Work Item** dialog, select the Story “Harmonize squawkers” which the instructor created, and click **OK**. If the work item is not shown, , confirm that **My assignment only** is off, and type `Harmonize` into the text field.



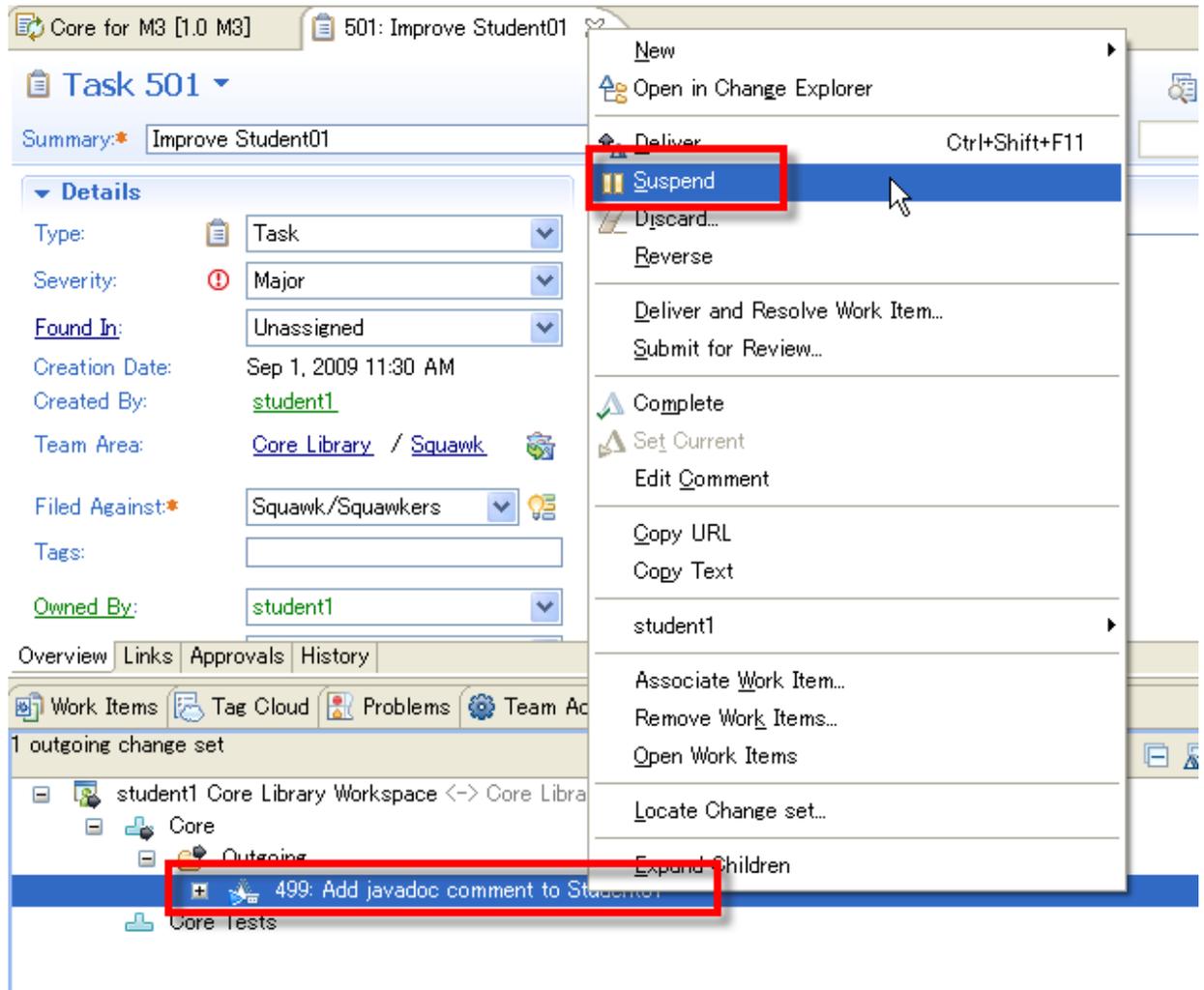
- \_\_h. Save the work item.

## 12.4 Suspend current work, and start working on the new task.

- \_\_1. Suspend the unfinished javadoc task.
  - \_\_a. Open the **Pending Changes** view (**Window** → **Show View** → **Other** → **Jazz Source Control** → **Pending Changes**) and confirm that the change set is associated to work item **Add comment to Student<N>**. If the change set is not associated to the work item, right-click the change set and select **Associate Work Item...** .



- \_\_b. Right-click the incomplete change set **Add comment to Student<N>**, and select **Suspend**.



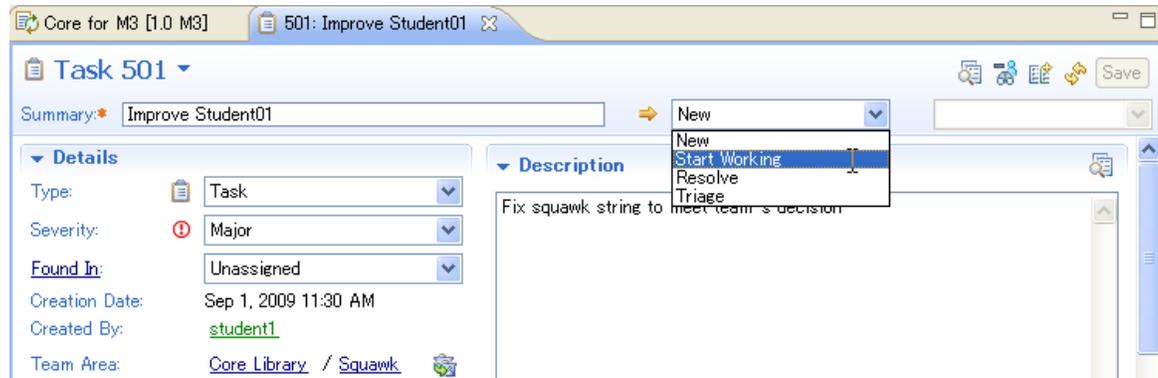
### Suspend and Store a work



It was not easy, by using former SCM tools, to suspend and store an unfinished work. That involves many manual and error prone procedures like copying the code, creating temporary patch file and storing it to local folder or creating SCM branch and so on.

As you see, Rational Team Concert provides easy and effective way to accomplish this task, so user can agilely and safely switch to a high prioritized work.

- \_\_2. Start working on the new task.
- \_\_a. In the Work Item Editor of your task, set the state to **Start Working**, and save the work item.



- \_\_b. Close the work item editor.

## 12.5 Complete the task, collaborating with teammates by 'code'.



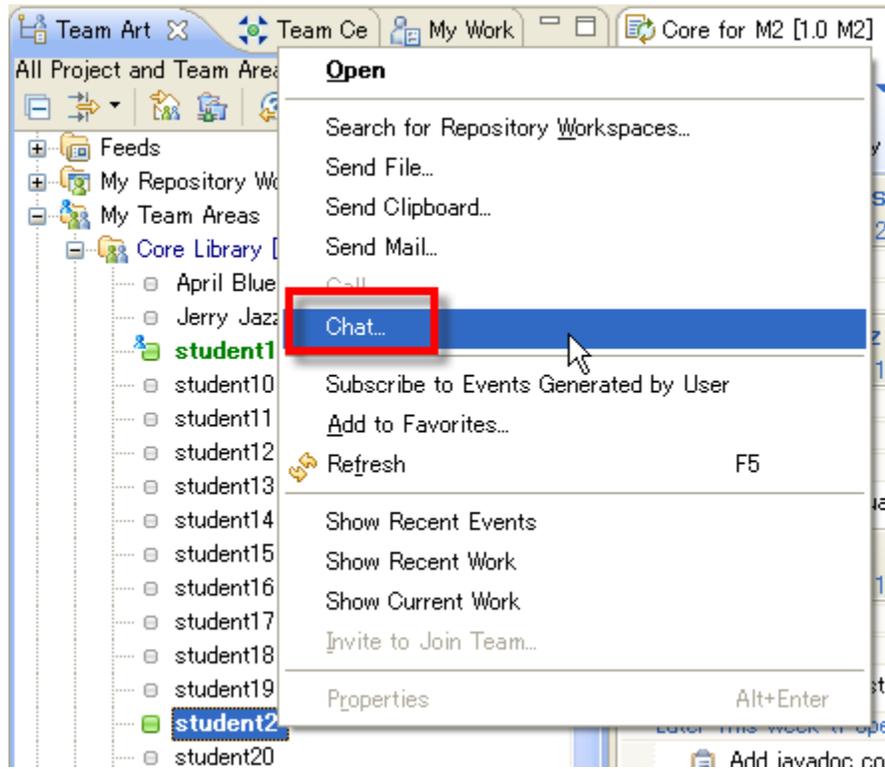
**Lab Scenario**

To accomplish perfect harmony, chat with a teammate on how to harmonize your squawkers (e.g. suffix with same number of '!'s), then fix your squawker accordingly.

Before delivering your change to the team's stream, check if your squawker and teammate's squawker work well together, by sharing the change set with the teammate through a repository workspace.

- \_\_1. Chat with other teammate about implementation detail

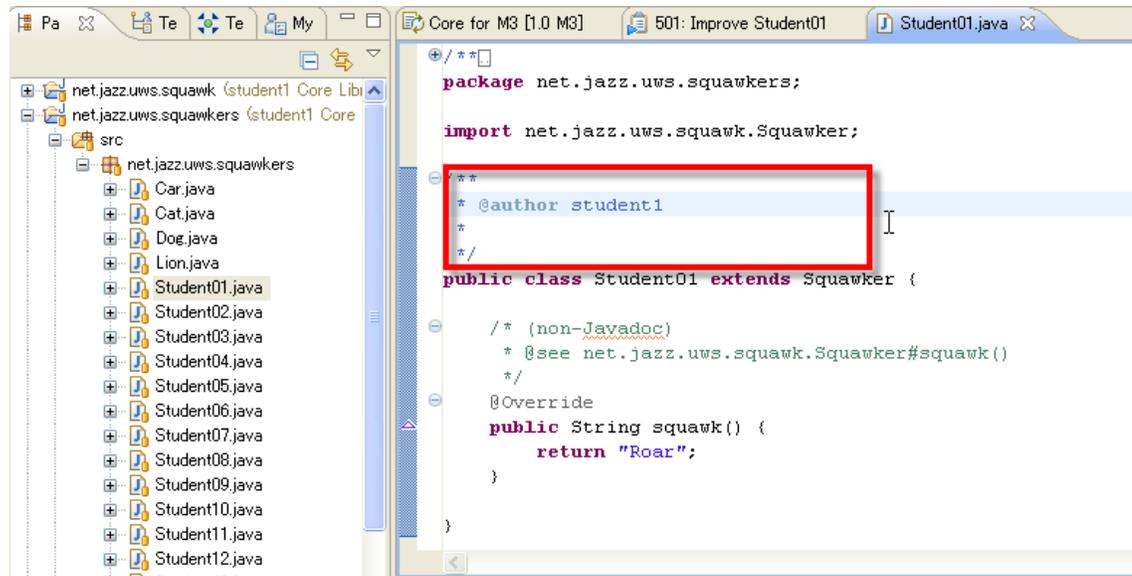
- \_\_a. In the **Team Artifacts** view, expand the **Squawk** project area, **My Team Areas** → **Core Library** and right-click **student<M>** (neighbor student) and select **Chat**



- \_\_b. Discuss about how to harmonize your squawkers (e.g. suffix with same number of "!"s).

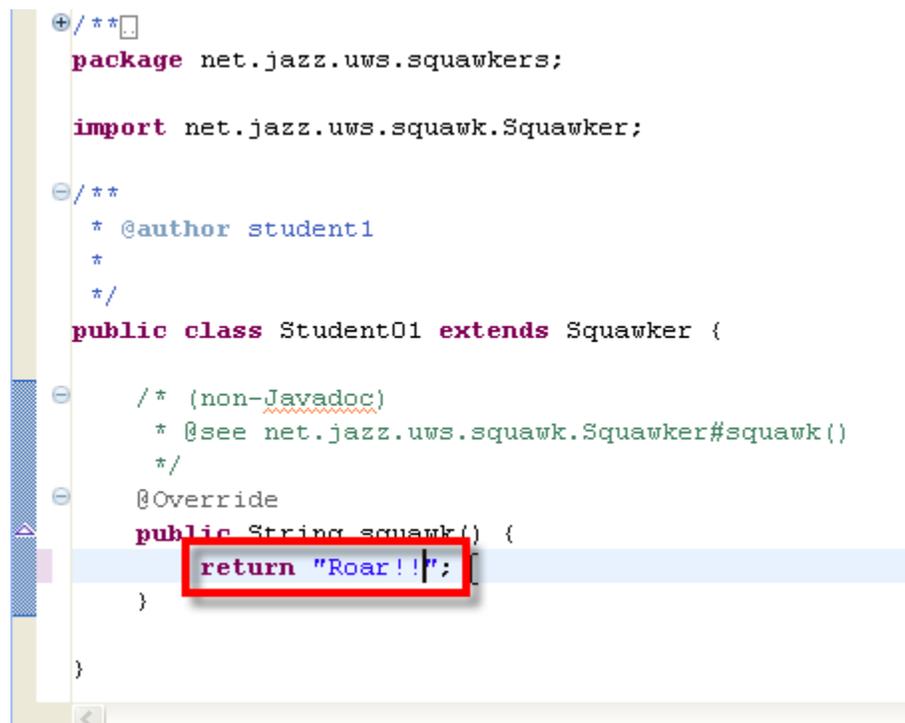
\_\_2. Modify the code.

- a. In the **Package Explorer** view, expand to the **net.jazz.uws.squawkers/src/net.jazz.uws.squawkers** Java package and double-click the **Student<N>.java** class file (replace <N> with your assigned id number). This will open the Java Editor.

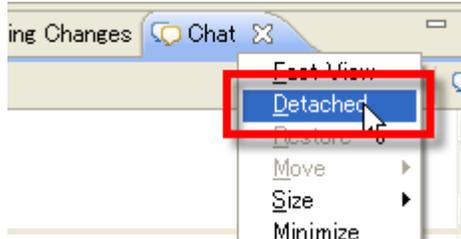


Note that the code is reverted to the current version in the team stream (see the javadoc comment) as a result of the **suspend**.

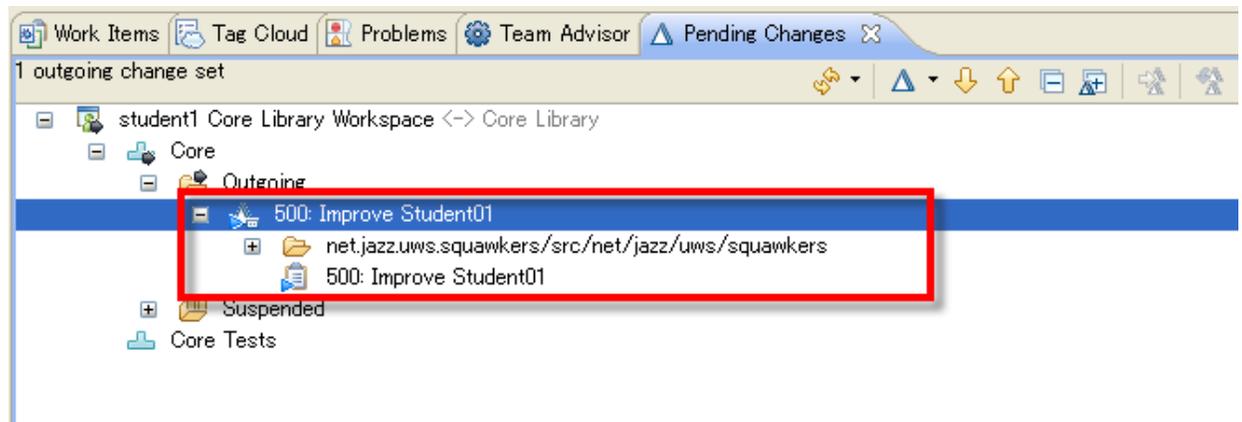
- b. Modify the line of `squawk()` method, according to the decision you and your teammate made.



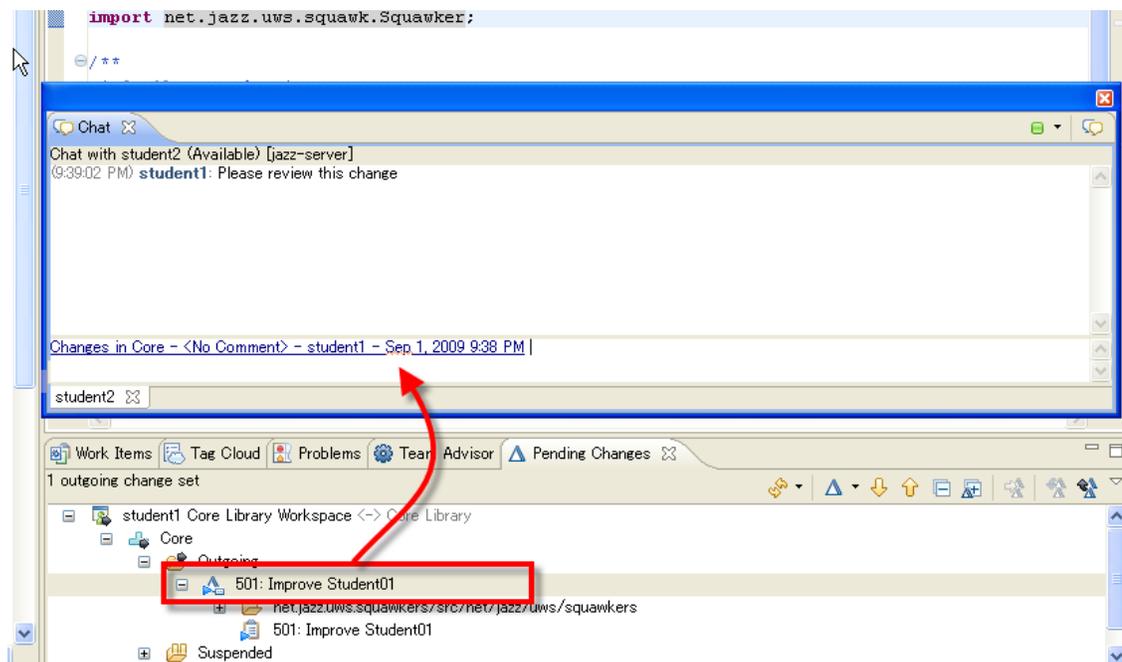
- \_\_c. Save the java code.
- \_\_3. Send link of the change-set by chat to share undelivered change-set with peer.
  - \_\_a. Right-click **Chat** view tab header, and select **Detached**



- \_\_b. Move the detached **Chat** view to an appropriate place where you can see both the **Chat** view and the **Pending Changes** view.
- \_\_c. In the **Pending Changes** view confirm that the change set is associated to work item **Improve Student<N>**. If the change set is not associated to the work item, right-click the change set and select **Associate Work Item...**



- \_\_d. In the **Pending Changes** view, expand **Core Library Workspace** → **Core** → **Outgoing**, then drag the change set (**Improve Student<N>**) and drop it into the input field of the **Chat** view, then press the **Enter** key to send the link.

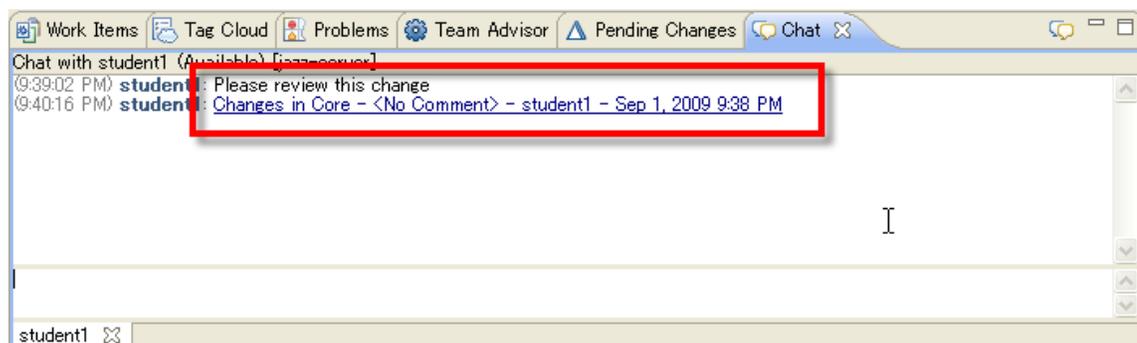


### Important!



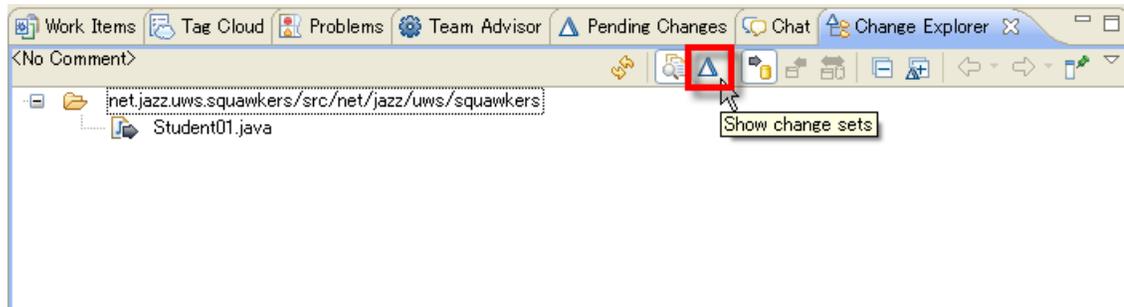
Before you confirm that a change set works well, delivering it to the team's stream may cause negative side effects (e.g. breaking the build). Instead of using team's stream to share the change, you can share the change set through a repository workspace and not affects any other teammates.

- \_\_4. Investigate received change set
- \_\_a. In the **Chat** view, click the link you received.

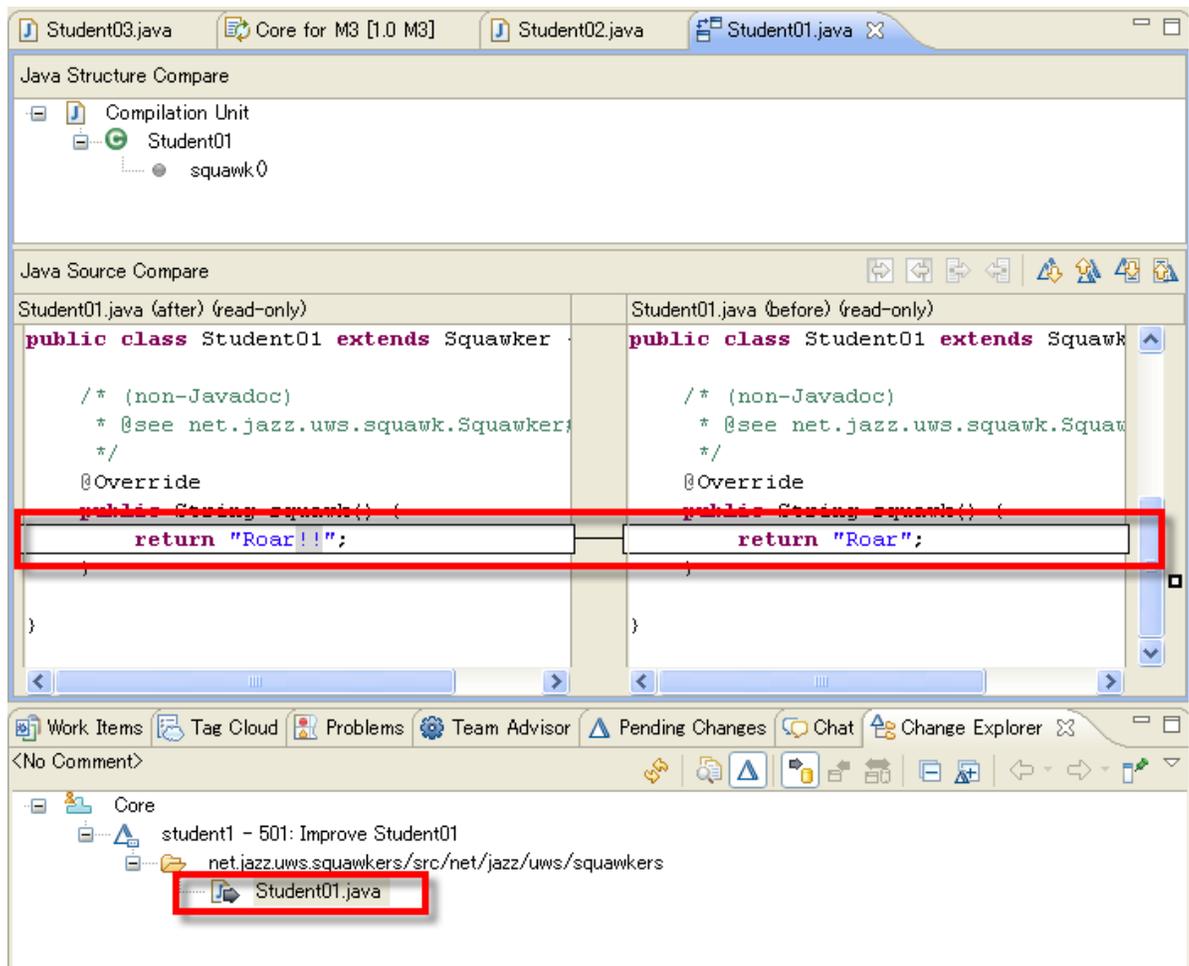


\_\_b. This opens the **Change Explorer** view.

\_\_c. In the **Change Explorer** view, click the  button to view change set

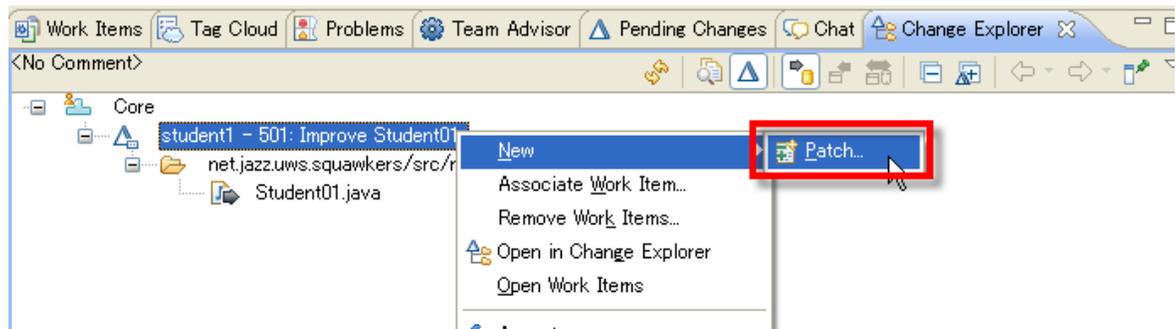


\_\_d. Expand the change set hierarchy, then double-click **Student<M>.java**, and check the code differences in the **Java Structure Compare** editor

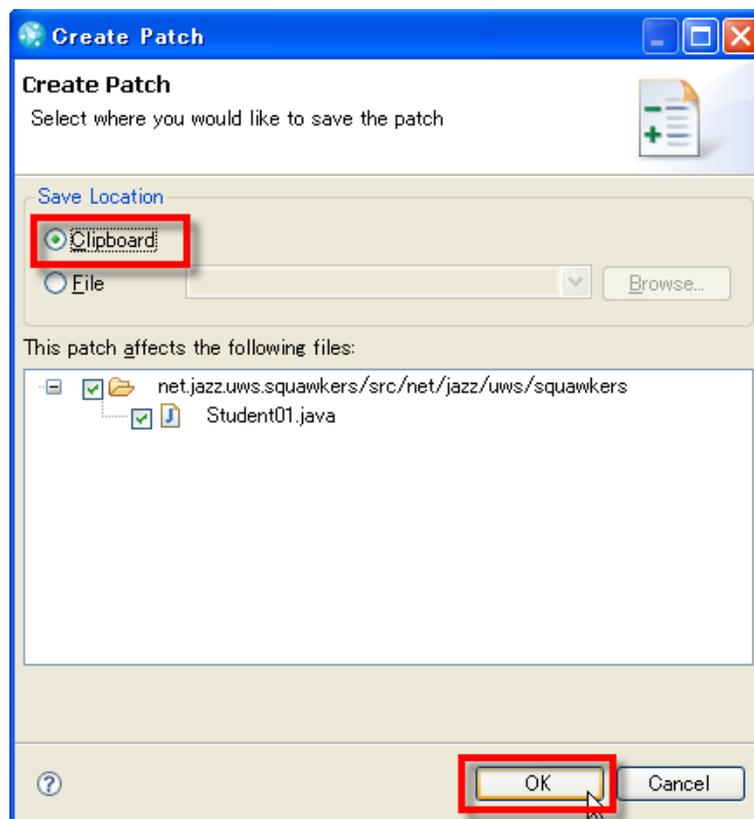


\_\_5. Accept the change set as a patch to test the code in the local workspace

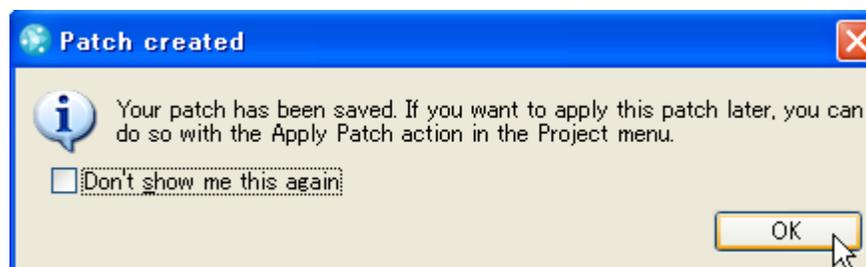
- \_\_a. In the **Change Explorer** view, right-click the change set and select **New → Patch...**



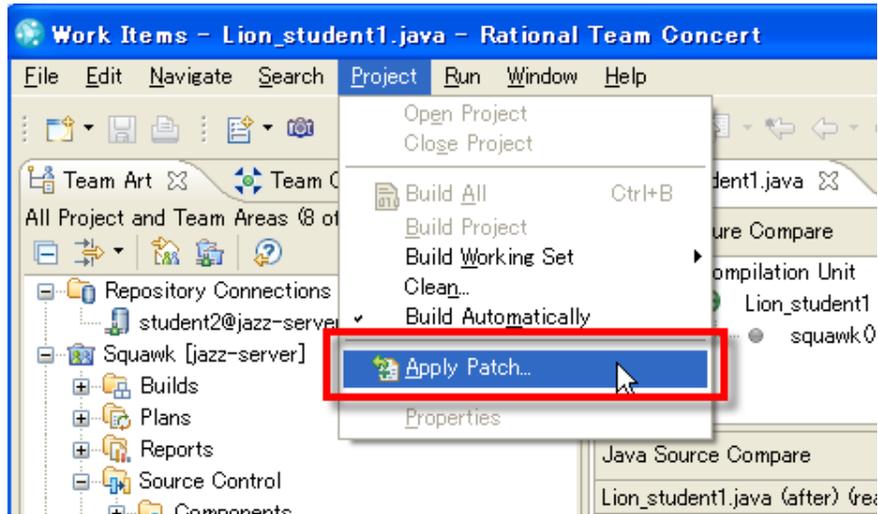
On the **Create Patch** dialog, set **Save Location** to **Clipboard**, and click **OK**.



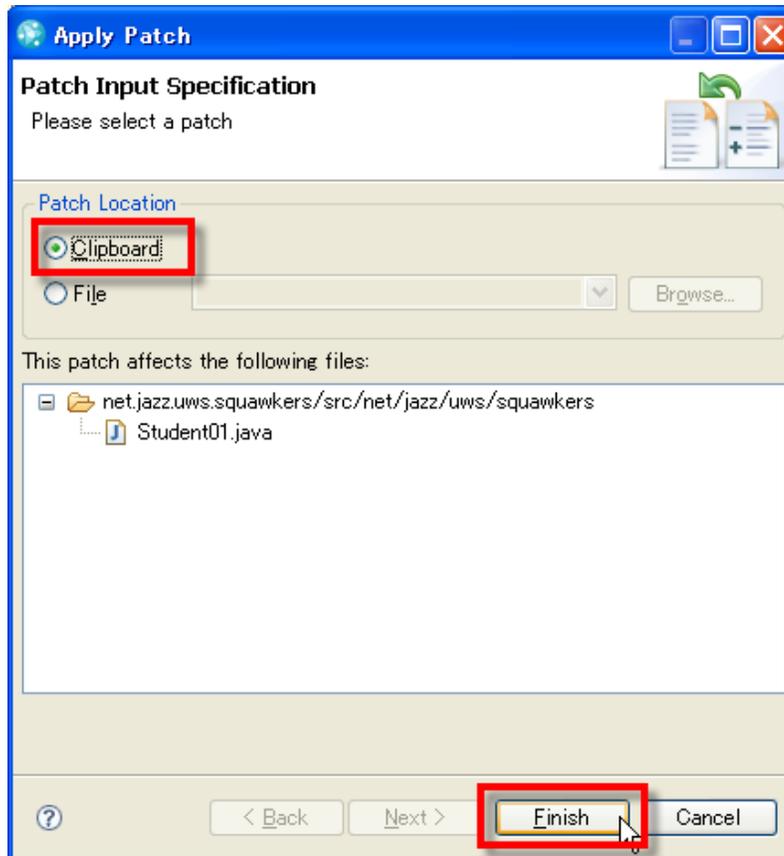
- \_\_b. Click **OK**, on the **Patch created** message dialog.



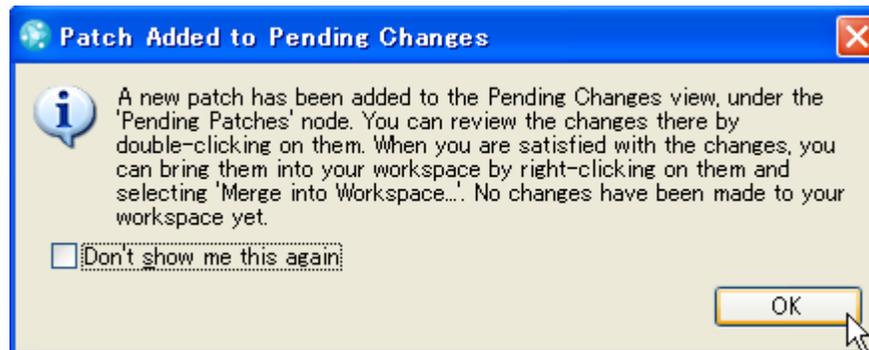
\_\_c. On the workbench menu, select **Project** → **Apply Patch....**



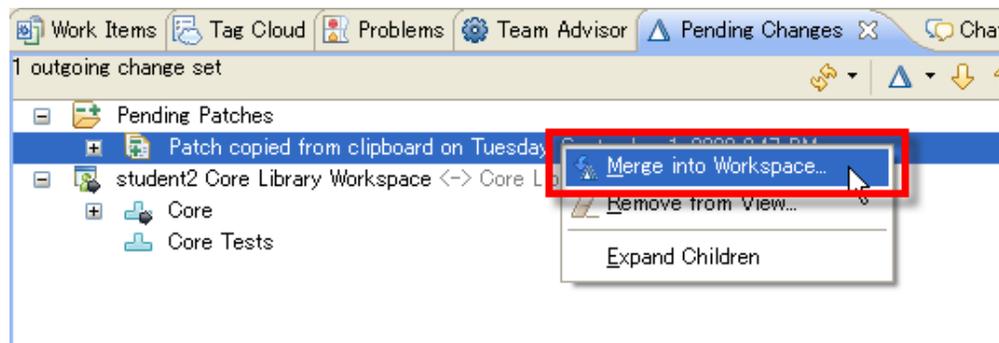
\_\_d. On the **Apply Patch** dialog, set **Patch Location** to **Clipboard** and click **Finish**.



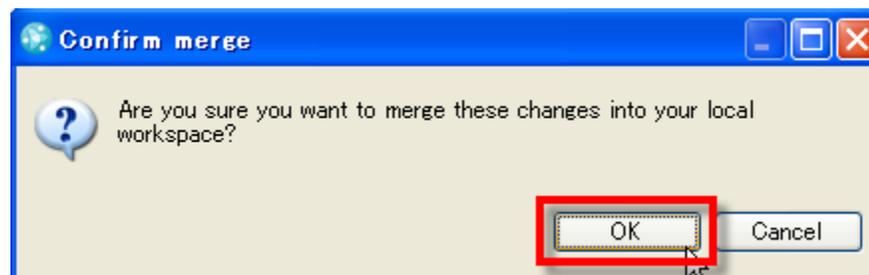
- \_\_e. Click **OK**, on the **Patch Added to Pending Changes** message dialog.



- \_\_f. On the **Pending Changes** view, expand the **Pending Patches**, right-click the **Patch copied from clipboard on .....**, and select **Merge into Workspace**.



- \_\_g. Click **OK** on the **Confirm merge** message dialog.

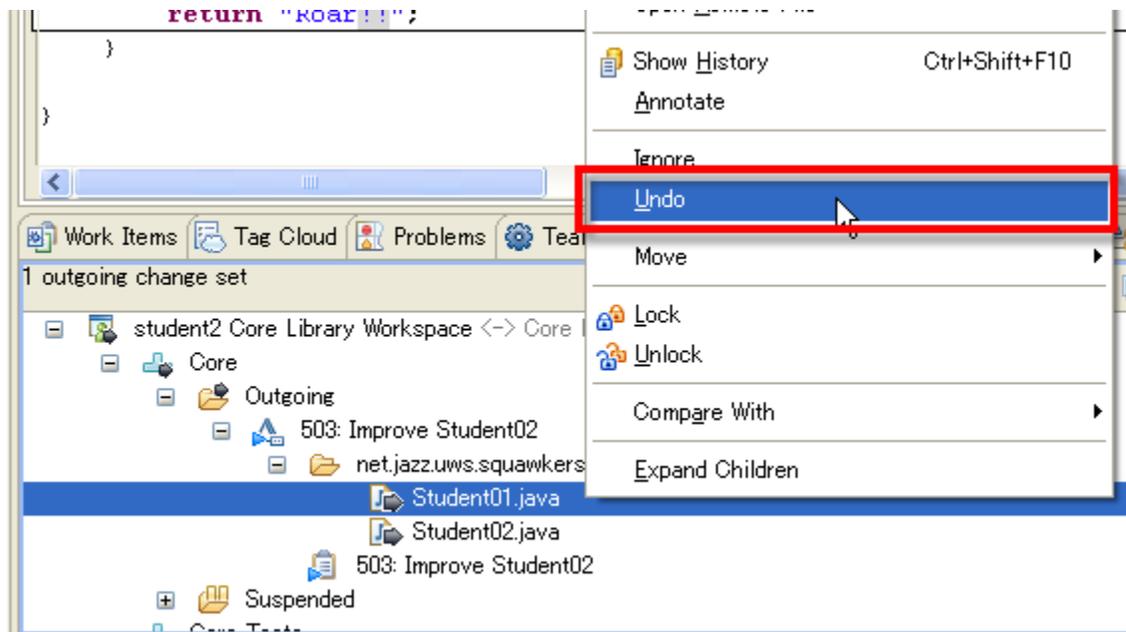




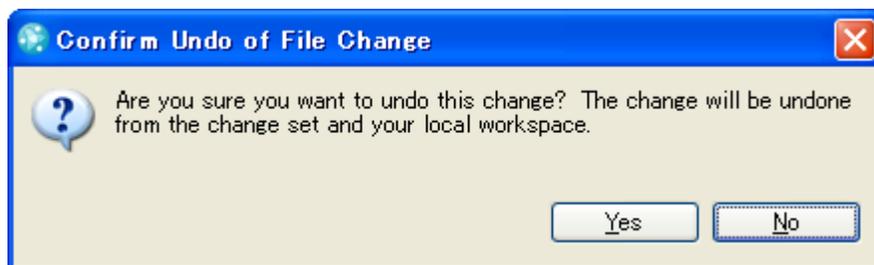
After you accept the temporary patch into your local workspace, a new class will be appear in your Package Explorer view. You can build and test the code locally to confirm that the changes you and your teammate made, work well together.

You may share temporary work without Team Concert, e.g. sending code snippets by e-mail and merging the snippet into code manually, etc. , but it is not an easy or reliable technique at all.

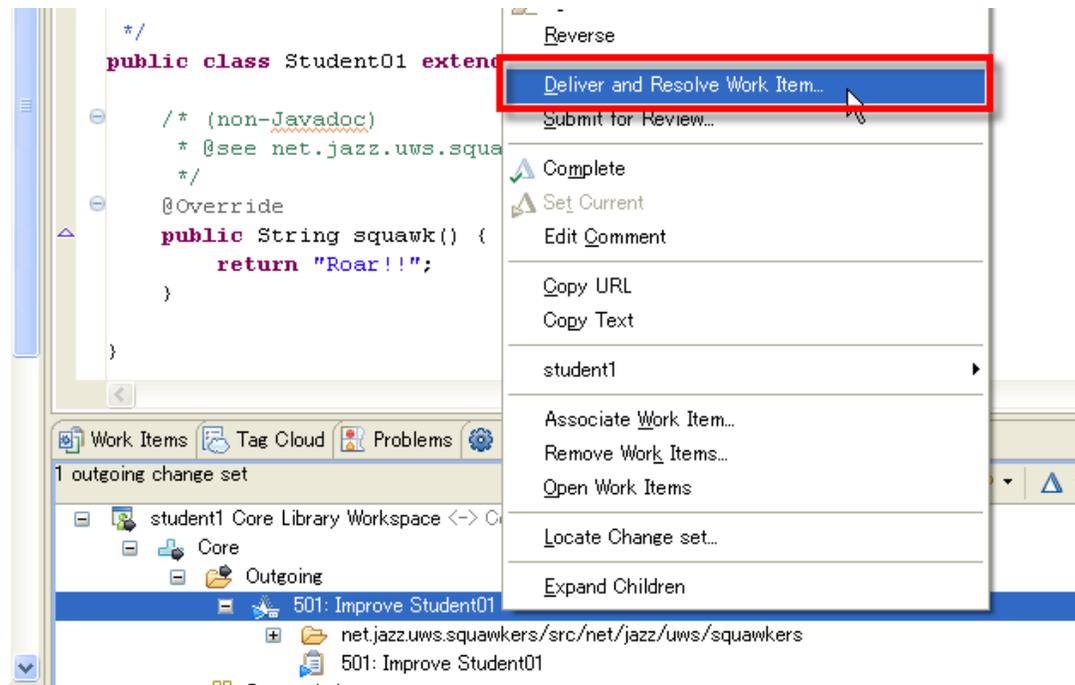
- \_\_6. Undo the patch
  - \_\_a. In the **Pending Changes** view, expand the **Outgoing** folder hierarchy completely and find the file you accepted as a patch.  
 (Be careful not to select your code. The following screenshot is for student2, who is collaborating with student1.)
  - \_\_b. Right click on the file, and select **Undo**.



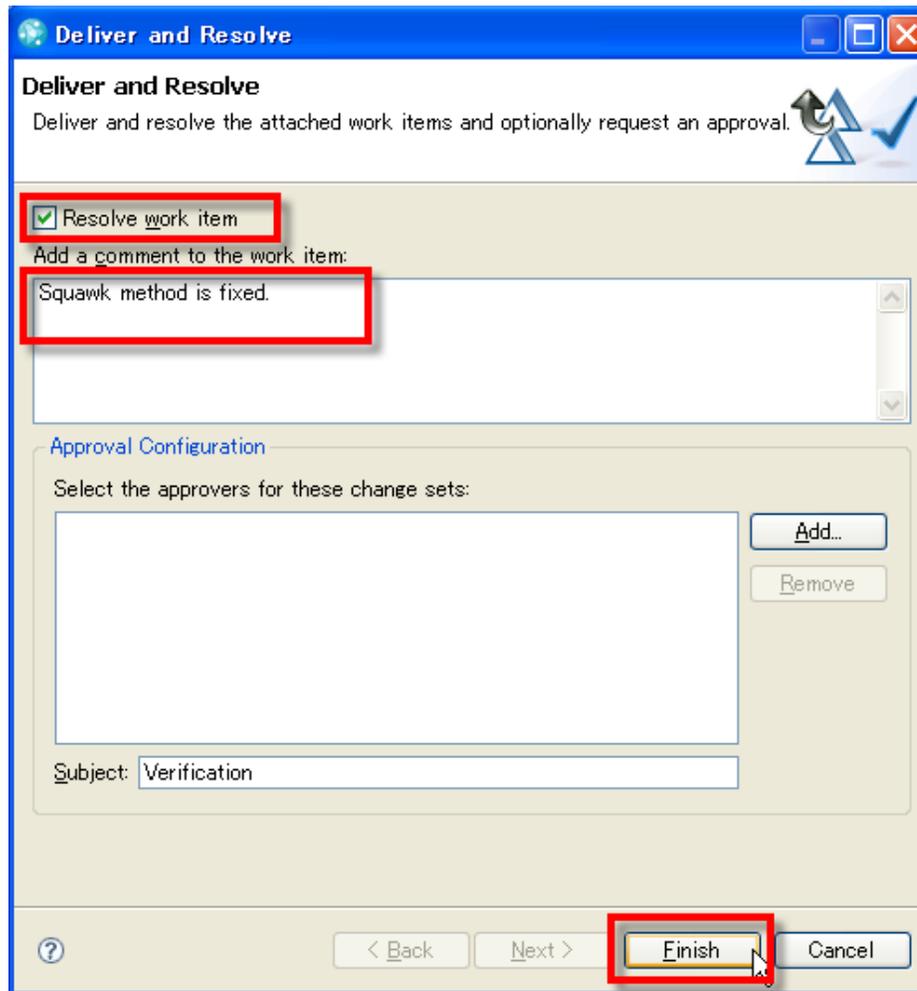
- \_\_c. Click **Yes** on the **Confirm Undo of File Change** dialog.



- \_\_7. Deliver your squawker code and resolve your task work item
- \_\_a. In the **Pending Changes** view, right-click your change set and select **Deliver and Resolve Work Item**.



- b. On the **Deliver and Resolve** dialog, add a comment into the text box, and click **Finish**.

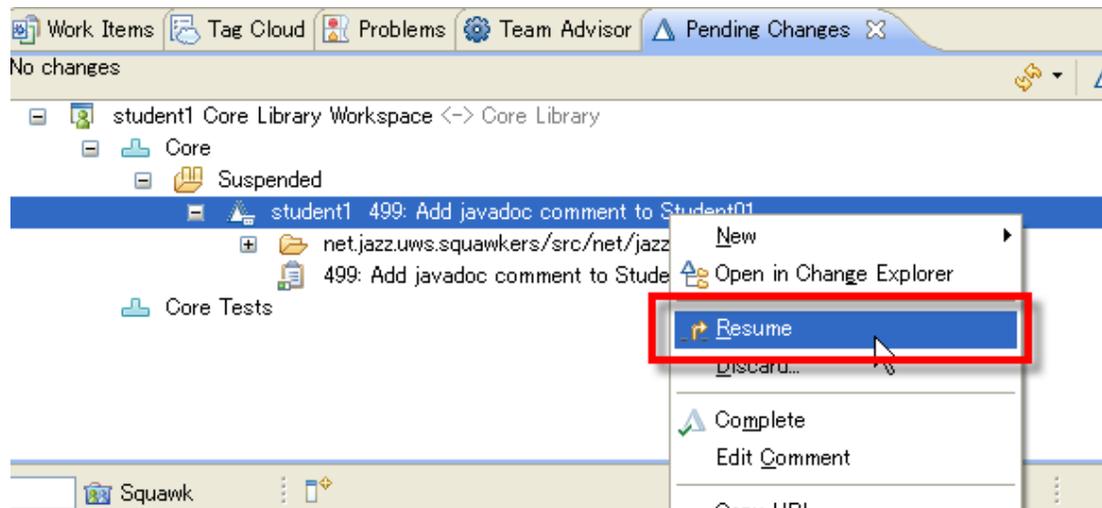


## 12.6 Resume the suspended work

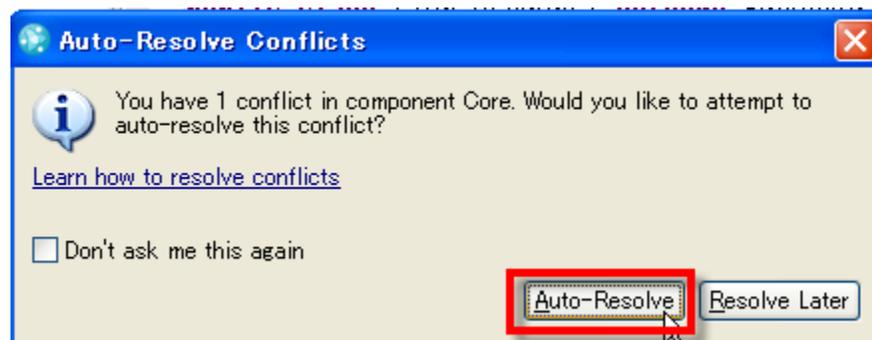


**Lab Scenario**  
You have completed important work successfully. So, it's time to resume suspended work and complete it.

- \_\_1. Resume the suspended change set
- \_\_a. In the **Pending Changes** view, expand **Core Library Workspace** → **Core** → **Suspended**, then right-click **Add javadoc comment to Student<N>**, and select **Resume**.



- \_\_b. On the **Auto-Resolve Conflicts** dialog, click **Auto-Resolve**.



#### Conflict and Auto-Resolve

The changes you made were to the same file, therefore this conflict occurs. The changes were to different lines, therefore Team Concert can automatically resolve the conflict (auto merge).

- \_\_c. In the **Package Explorer** view, double-click your squawker to open the code in the Java Editor.

- \_\_d. Confirm that the two change sets (unfinished javadoc and fixed squawk method) were merged correctly.

```
Core for M3 [1.0 M3] Student01.java X
/**
 *
 */
package net.jazz.uws.squawkers;

import net.jazz.uws.squawk.Squawker;

/**
 * Student01 goes ....
 * @author student1
 */
public class Student01 extends Squawker {

    /* (non-Javadoc)
     * @see net.jazz.uws.squawk.Squawker#squawk()
     */
    @Override
    public String squawk() {
        return "Roar!!";
    }
}
```

- \_\_e. Finish the javadoc comment, then deliver the change set and resolve the javadoc task work item.

**Conclusion**



Congratulation! You have experienced the advanced features of the Rational Team Concert SCM component, and observed how Rational Team Concert helps teams to be agile and produce maximum value.

## Appendix A: Lab 2 Planning your work - Demo

### Demo Scenario

Our project has a high-level release plan called the **Backlog**. This plan defines what the project wants to accomplish in this release. In general, the plan items tend to be broadly defined. Stories are created to further refine the plan items. The stories are prioritized so the stories are completed in the right order. As part of the iteration preparation, the project team will assign the story to the iteration plan. This story is then broken down into tasks for team members to complete

For this demo, the instructor will use the Web UI. The actions can also be performed in the Eclipse UI. We expect many people will use the agile planning tools via the Web UI.



In this scenario, the instructor, playing the role of team lead, will now examine the release plan to see what content is required for the **M3** milestone. There is a high level Plan Item work item type, called **Create a broader collection of squawkers**. A Story work item is created by the team lead as a child of the Plan Item. This Story is called **Create today's Squawkers** and prioritized so it is one of the most important stories in the release plan. As part of the discussion for the iteration, it is added to the iteration plan 1.0 M3. When adding the Story to the iteration plan, the instructor shows how the plans can be viewed in different formats depending on what kind of work you want to perform.

After the demonstration by the instructor, the students will create tasks against this story in the iteration plan.

Later, after the students have created tasks for their new Squawkers, the team lead will check that all the tasks are child-tasks of the parent Story **Create today's squawkers** and correct if required. The team lead will distribute the plan via Chat.



### Important!

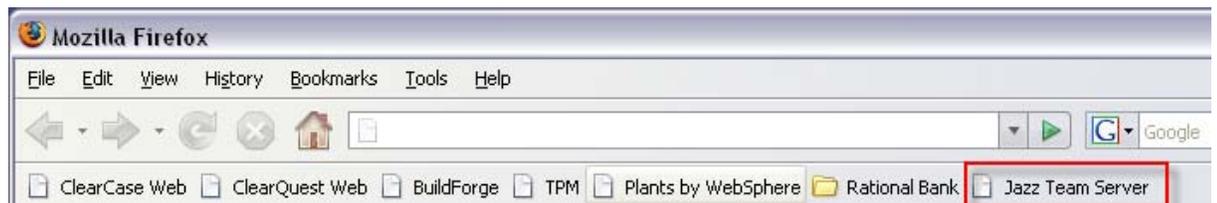
All steps in this section are to be performed by the instructor as a demonstration.

## A2.1 Create new plans



**Team role**  
The instructor will now perform the role of Jerry, the Team Lead.

- \_\_1. Connect to the Squawk project as Jerry Jazz in the Web UI.
  - \_\_a. Open the Web UI with the Firefox Internet Browser
    - \_\_i. Open the **Firefox** internet browser by double-clicking the **Mozilla Firefox** shortcut  on the **Windows Desktop**.
    - \_\_ii. Click the **Jazz Team Server** shortcut on the bookmarks toolbar.



- \_\_b. If a **Security Alert** is displayed, click **Yes** or **OK** to proceed.
- \_\_c. The Web UI uses a secure connection. Enter Jerry's id (jazzy) and password (jazzy). Click **Log In** to access the project areas.

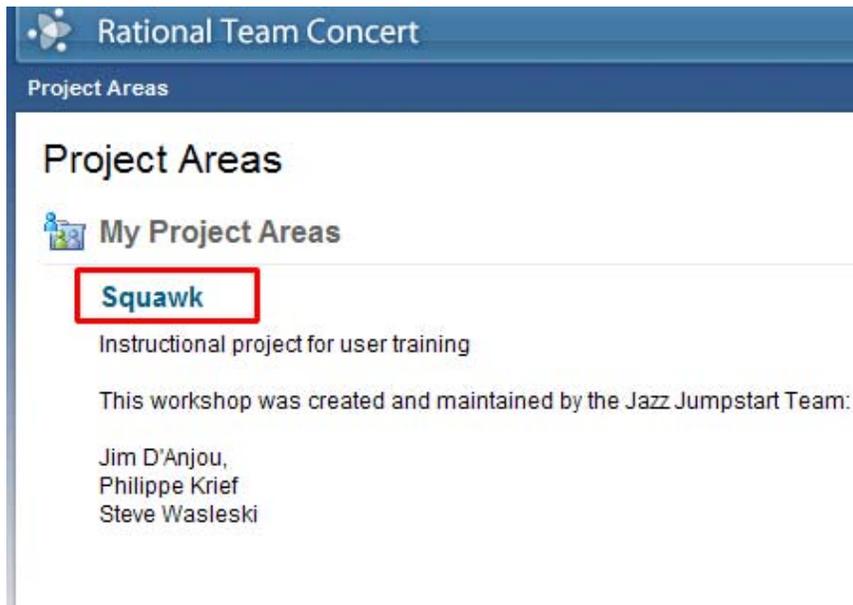


IBM

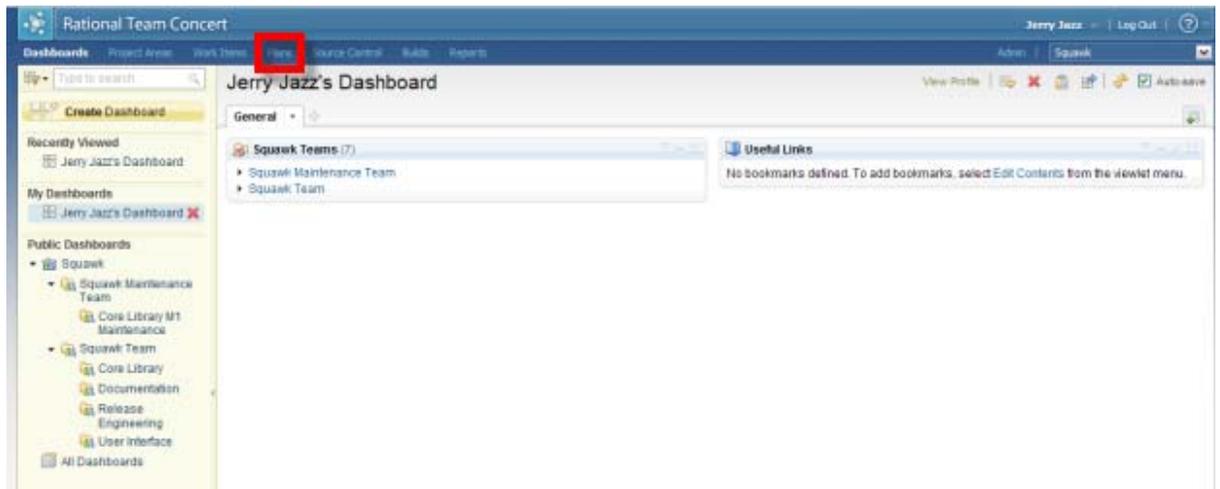
Rational. software

Licensed Material - Property of IBM Corp. © Copyright IBM Corp. and its licensors 2008, 2009. All Rights Reserved. IBM, the IBM logo, Jazz, and Rational are trademarks of IBM Corporation, in the United States, other countries, or both. Built on Eclipse is a trademark of Eclipse Foundation, Inc. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

- \_\_d. Choose from one of the project areas. Click the **Squawk** project area



- \_\_e. You will see Jerry's personal dashboard when you login. Select **Plans**.



At this stage, the instructor acts as a product owner, responsible for making sure that the team has a common vision, and establishing priorities. The instructor will start by examining the plan items and create a story to match the plan items

- \_\_2. Open the current project Backlog and create a new story against an existing plan item



This is a plan of type Product Backlog. It shows plan work items such as Plan Item and Story only. The plan **only** shows high-level plan work items, and will not show the lower level execution items (for example, the tasks)

- \_\_a. In the **Release 1** section, click the **Backlog** plan.

The screenshot shows the Rational Team Concert web interface. The browser address bar displays the URL: <https://jazz-server:9443/jazz/web/projects/Squawk#action=com.ibm.team.apt.search&predef=myCurrent&ttq=1248649627000>. The page title is "My Current Plans". The left sidebar shows a "Create Plan" button and a "Recently Viewed" list containing items like "Doc for M1 [1.0 M1]", "Backlog [Release 1]", "M1 1.0.1 Core Library Plan [1.0.1 M1 Maintenance]", and "Core for M3 [1.0 M3]". The main content area displays a table of plans:

1.0.1 M1 Maintenance 15 Jan 2008 - 31 Dec 2009			
Name	Team Area	Progress	
M1 1.0.1 Core Library Plan	Core Library M1 Maintenance	0/0 h	
Release 1 15 Jan 2008 - 31 Dec 2009			
Name	Team Area	Progress	
Backlog	Squawk Team	80/135 Story Points	<div style="width: 60%;"></div>

- \_\_b. Click the **Overview** tab to read about this release.

The screenshot shows the "Backlog" overview page. The breadcrumb is "My Current Plans >". The page title is "Backlog". Below the title, it says "Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 7 Open". There are three tabs: "Overview" (highlighted with a red box), "Planned Items", and "Charts". A progress bar shows "Progress: 80/135 Story Points".

Application Overview The Squawk application is a teaching application used in workshops and demos of Rational Team Concert.

It is the simplest collaborative application you can imagine - it consists of a set of "squawker" classes that output or "squawk" some text to the console. For example, a lion squawks with a roar and Fred squawks with "Wassap?" In a workshop a participant authors his or her own squawkers to say what participant wants them to say. A few lines of code and you are done. The application UI aggregates all the squawkers into a squawk party and the cacophony of unusual noise.

This plan represents the overall release plan to be completed over a set of iterations called milestones. The development is underway and a som

- \_\_c. Click the **Planned Items** tab to view the current work items allocated to this plan.
- \_\_d. Initially, the plan will be set to **View As: Teams** which shows the assignment of work to all teams for Release 1.

**Backlog**  
 Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 7 Open  
 Overview | **Planned Items** | Charts  
 Progress: 80/135 Story Points

View As: Teams

**Squawk Team**  
 Closed Items: 12 | Open Items: 7  
 Progress: 80/135 Story Points

Item	Points	Progress	Priority
Streamline personal build	5 pts	0/0	High
▶ Create a broader collection of squawkers	0/5		High
Create application build environment	5 pts	0/0	High
Create core application based on initial prototype	5 pts	0/0	High
Predefine template squawker classes	5 pts	0/0	Medium
▶ Create a .net implementation of the Squawker application	20 pts	22/22	Medium
Define application documentation	3 pts	0/0	Medium
Define a handful of example squawkers	3 pts	0/0	Medium
Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned
Support theme based squawk parties	8 pts	0/0	Unassigned
Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned
Set up a project status web page	5 pts	0/0	Unassigned
Create a basic user interface	12 pts	0/0	Unassigned
Test Core Library team process change for iteration M2 endgame	5 pts	0/0	Unassigned

**Core Library**  
 Closed Items: 2 | Open Items: 0  
 Progress: 2/2 Story Points

Item	Points	Progress	Priority
▶ Create a .net implementation of the Squawker application	20 pts	22/22	Medium
Define a Jazz build definition that runs MS Build	1 pt	0/0	Unassigned
C# implementation	1 pt	0/0	Unassigned

**Documentation**  
 Closed Items: 0 | Open Items: 0  
 Progress: 0/0 Story Points

**Release Engineering**  
 Closed Items: 0 | Open Items: 0  
 Progress: 0/0 Story Points

**User Interface**  
 Closed Items: 0 | Open Items: 0  
 Progress: 0/0 Story Points

\_\_e. In the **View As** entry field, select **Backlog** to view the current backlog for Release 1.

**Backlog**  
 Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 7 Open

Overview | **Planned Items** | Charts

Progress: 80/135 Story Points

View As: **Backlog**

	Create futuristic squawkers	High	5 pts	0/0
	Create a broader collection of squawkers	High		0/5
	Predefine template squawker classes	Medium	5 pts	0/0
	Support other types of technologies like audio and video squawkers	Unassigned	20 pts	0/0
	Support theme based squawk parties	Unassigned	8 pts	0/0
	Allow new squawkers to be defined from a user interface instead of code	Unassigned	12 pts	0/0
	Set up a project status web page	Unassigned	5 pts	0/0

\_\_f. Find the Plan Item called **Create a broader collection of squawkers**.

**Backlog**

Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 7 Open

Overview | **Planned Items** | Charts

Progress: 80/135 Story Points | Estimated: 100%

View As: **Backlog**

	Create futuristic squawkers	High	5 pts	0/0	493
	<b>Create a broader collection of squawkers</b>	High		0/5	466
	Predefine template squawker classes	Medium	5 pts	0/0	484
	Support other types of technologies like audio and video squawkers	Unassigned	20 pts	0/0	471
	Support theme based squawk parties	Unassigned	8 pts	0/0	470
	Allow new squawkers to be defined from a user interface instead of code	Unassigned	12 pts	0/0	469
	Set up a project status web page	Unassigned	5 pts	0/0	468

\_\_g. Click on this Plan Item to open.

**Plan Item 466** Save

Summary: \* Create a broader collection of squawkers New

Overview **Links** Approvals History Loaded: 27 Jul 2009 00:27:06

Details		Quick Information	
Type:	Plan Item	Owned By:	Jerry Jazz
Severity:	Normal	Priority:	High
Found In:	Unassigned	Planned For:	Release 1
Creation Date:	14 Aug 2008 20:28:08	Estimate:	<input type="text"/> Correction: <input type="text"/>
Created By:	Jerry Jazz	Time Spent:	<input type="text"/>
Project Area:	Squawk	Due Date:	<input type="text"/>
Team Area:	Squawk Team		
Filed Against:	Squawk		
Tags:	<input type="text"/>		

**Description** Edit

A rich set of squawkers is needed to be able to demo the application.

**Discussion** Add Comment

No Comments. Add Comment



The instructor reads the Plan Item and decides to create a new Story called **Create today's squawkers** as a child of the Plan item **Create a broader collection of squawkers**. There is already a Story called **Create futuristic squawkers** - that is a child of the Plan Item. However, we do need a new Story.

\_\_h. Read the Plan Item and note there are already child work items. Hover over the **Children** section to see the child work item.

Quick Information

Subscribers (1): JJ

**Children (1): 493**

Children (1)  
Story 493: Create futuristic squawkers

\_\_i. Use your browser Back button  to return to the Plan.

\_\_3. Create the new Story **Create today's squawkers.**

\_\_a. Switch to the **Work Breakdown** view

---

 **Backlog**

Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2008)

[Overview](#) [Planned Items](#) [Charts](#)

---

View As:  ▼

-   Backlog
-   Iterations
-   Teams
-   **Work Breakdown**
-   Support other types of technologies like audio and video
-   Support theme based squawk parties
-   Allow new squawkers to be defined from a user interface

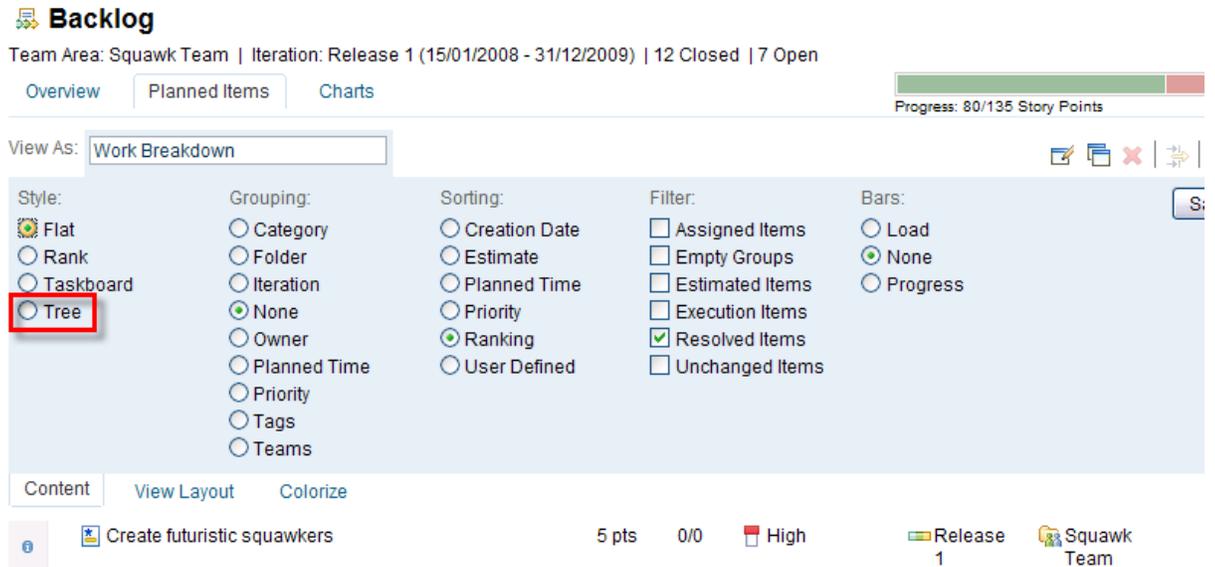
---

\_\_b. You will need the plan to be represented in **Tree** view mode and *may* need to complete the next steps.

\_\_i. Move your cursor on the right side of the **View as:** field to display the actions buttons.

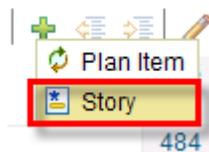
Then click on the **Edit View** button 

\_\_ii. Click the **Tree** style from the left column, make sure **Resolved Items** is selected under Filter, and **Save** the display configuration.



\_\_c. Click the Insert New Work Item icon  to add a new story to the Release 1 backlog.

\_\_d. Select **Story**



\_\_e. Add the following to the new Story

<00:53:37> Save and Close Save Close

Summary:

Filed Against: Squawk  Quick Information: No Links.

Tags:

Owned By: Unassigned

Priority:  Unassigned

Planned For: Release 1

Story Points: 1 pt

Progress:   
Progress: 0 / 0 h Estimated: -

Description:

Discussion: No Comments.

\_\_i. **Summary:** Create today's squawkers

\_\_ii. **Filed Against:** Squawkers

\_\_iii. **Tag:** squawker

\_\_iv. **Story Points:** 5pts

\_\_v. **Description:** Create new squawkers for today's PoT

\_\_f. Select **Save and Close**

<00:53:37> **Save and Close** Save Close

Summary: Create today's squawkers

Filed Against: Squawk/Squawkers  Quick Information: No Links.

Tags: squawker

\_\_g. Make this story a child of the Plan item

\_\_i. Drag the Story **Create today's squawkers** on to the top of the Plan Item **Create a broader collection of squawkers**.

Overview | **Planned Items** | Charts

Progress: 80/140

View As: Work Breakdown

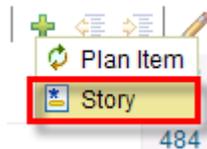
	<b>Create a broader collection of squawkers</b>	0/5	High	Release 1	
	Predefine <b>Create today's squawkers</b>	5 pts	0/0	Medium	Release 1
	Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	Release 1
	Support theme based squawk parties	8 pts	0/0	Unassigned	Release 1
	Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	Release 1
	Set up a project status web page	5 pts	0/0	Unassigned	Release 1
	<b>Create today's squawkers</b>	5 pts	0/0	Unassigned	Release 1

The instructor creates another new Story called **Create documentation for today's squawkers** as a child of the Plan Item **Create a broader collection of squawkers**

\_\_h. Create a new Story called **Create documentation for today's squawkers**.

\_\_i. Click the **Insert New Work Item** icon to add a new Story to the Release 1 backlog.

\_\_ii. Select **Story**



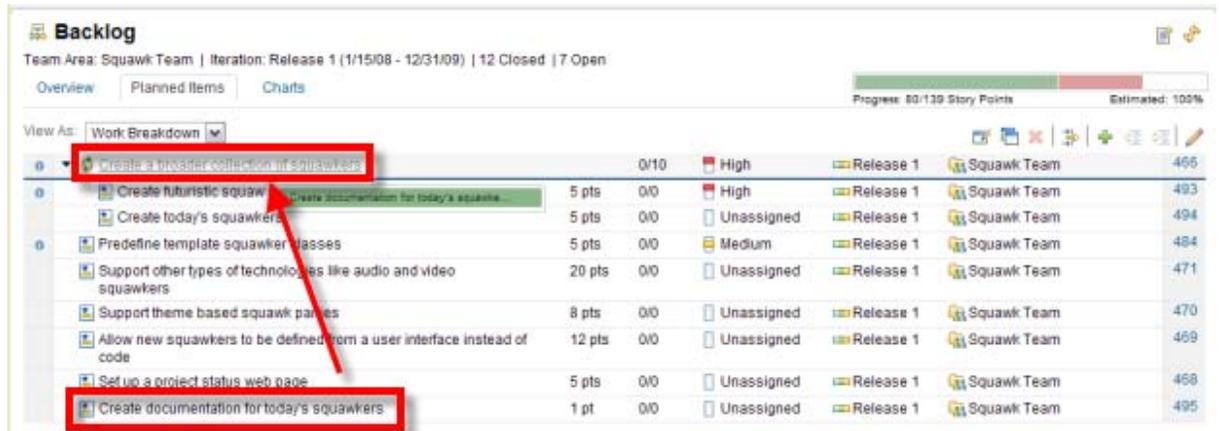
- \_\_i. Add the following to the new Story
  - \_\_i. **Summary:** Create documentation for today's squawkers
  - \_\_ii. **Filed Against:** Squawkers Documentation
  - \_\_iii. **Tags:** squawker documentation
  - \_\_iv. **Story Points:** 1pts
  - \_\_v. **Description:** Create documentation for the new squawkers for the PoT
- \_\_j. Select **Save and Close**

The screenshot shows a software development tool interface for creating a story. The window title is "<11:31:42>". The main content area contains the following fields and controls:

- Summary:** A text input field containing "Create documentation for today's squawkers".
- Filed Against:** A dropdown menu showing "...awkers Documentation".
- Tags:** A text input field containing "squawker, documentation".
- Owned By:** A dropdown menu showing "Unassigned".
- Priority:** A dropdown menu showing "Unassigned".
- Planned For:** A dropdown menu showing "Release 1".
- Story Points:** A dropdown menu showing "1 pt".
- Progress:** A section showing "No work planned", "Progress: 0 / 0 h", and "Estimated: -".
- Description:** A large empty text area.
- Discussion:** A section showing "No Comments" and an "Add Comment" button.

In the top right corner, there are two buttons: "Save and Close" (highlighted with a red box) and "Save Close".

- \_\_4. Make this story a child of the Plan Item **Create a broader collection of squawkers**
  - \_\_a. Drag the story **Create documentation for today's squawkers** to the Plan Item **Create a broader collection of squawkers**



- \_\_b. Confirm that all your new stories are children of the Plan Item **Create a broader collection of squawkers** by selecting the arrowhead





The team lead now looks at the plan in a view mode called Backlog. This view lists all the plan items regardless of their parent/child relationships. The story **Create today's squawkers** has not been prioritized yet. The team lead drags the story to the top of the prioritized list. Note how the priority moves from Unassigned to High automatically. This relative prioritization will be retained regardless of which plan view you use.

\_\_5. Prioritize the Story **Create today's stories**

\_\_a. On the plan, **Planned Items** tab, select **View As: Backlog**

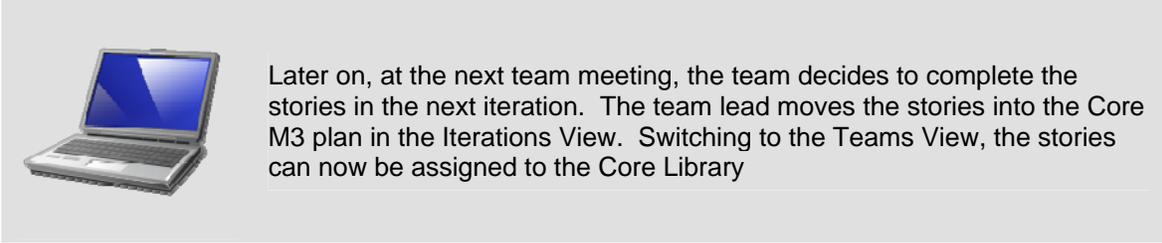
Story	Priority	Points	Progress	Estimate
Create futuristic squawkers	High	5 pts	0/0	493
Create a broader collection of squawkers	High	5 pts	0/11	466
Predefine template squawker classes	Medium	5 pts	0/0	484
Support other types of technologies like audio and video squawkers	Unassigned	20 pts	0/0	471
Support theme based squawk parties	Unassigned	8 pts	0/0	470
Allow new squawkers to be defined from a user interface instead of code	Unassigned	12 pts	0/0	469
Set up a project status web page	Unassigned	5 pts	0/0	468
Create documentation for today's squawkers	Unassigned	1 pt	0/0	495
Create today's squawkers	Unassigned	5 pts	0/0	494

\_\_b. Drag new Story **Create today's squawkers** to between the top two items (note the dark line will appear between the items)

Story	Priority	Points	Progress	Estimate
Create futuristic squawkers	High	5 pts	0/0	493
Create a broader collection of squawkers	High	5 pts	0/11	466
Predefine template squawker classes	Medium	5 pts	0/0	484
Support other types of technologies like audio and video squawkers	Unassigned	20 pts	0/0	471
Support theme based squawk parties	Unassigned	8 pts	0/0	470
Allow new squawkers to be defined from a user interface instead of code	Unassigned	12 pts	0/0	469
Set up a project status web page	Unassigned	5 pts	0/0	468
Create documentation for today's squawkers	Unassigned	1 pt	0/0	495
Create today's squawkers	Unassigned	5 pts	0/0	494

\_\_c. The priority of **Create today's squawkers** has changed to **High**

Story	Priority	Points	Progress	Estimate
Create futuristic squawkers	High	5 pts	0/0	493
Create today's squawkers	High	5 pts	0/0	494
Create a broader collection of squawkers	High	5 pts	0/11	466
Predefine template squawker classes	Medium	5 pts	0/0	484



- \_\_6. Move the stories into the Core M3 plan
  - \_\_a. On the plan, **Planned Items** tab, select **View As: Iterations**

**Backlog**  
 Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 8 Open

Overview | **Planned Items** | Charts

View As: **Backlog** (dropdown menu open with **Iterations** highlighted)

		awkers	High
		wkers	High
		Create a broader collection of squawkers	High
		Predefine template squawker classes	Medium
		Support other types of technologies like audio and video squawkers	Unassigned
		Support theme based squawk parties	Unassigned

- \_\_b. Select the arrow next to the Plan Item **Create a broader collection of squawkers** to expose the Story **Create today's squawkers**.

**Backlog**  
 Team Area: Squawk Team | Iteration: Release 1 (15/01/2008 - 31/12/2009) | 12 Closed | 8 Open

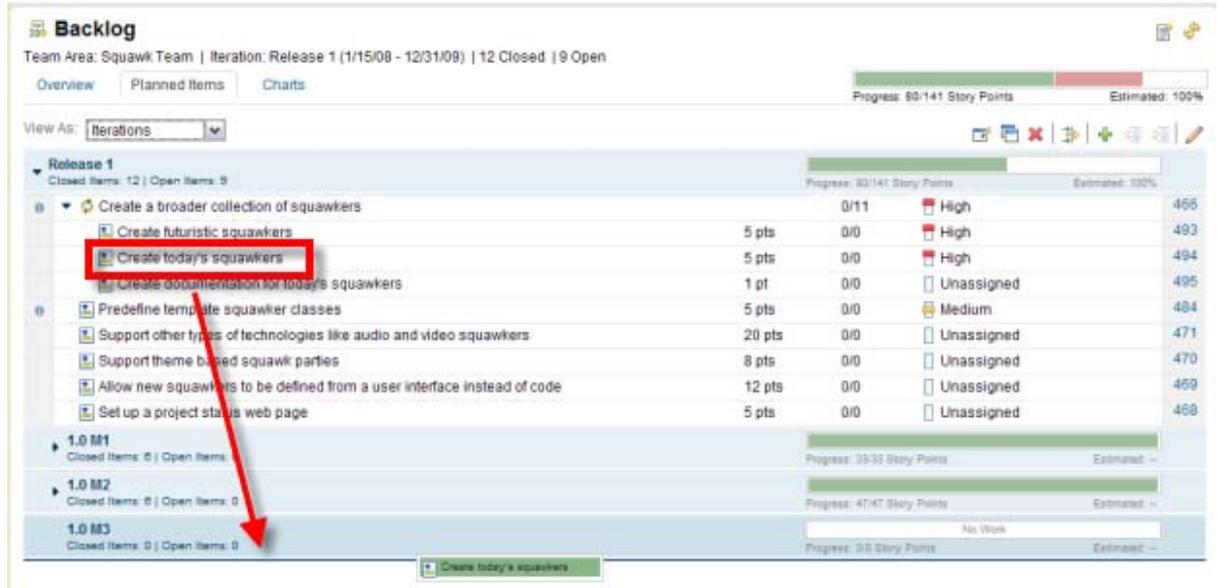
Overview | **Planned Items** | Charts

Progress: 80/140 Story Points | Estimated: 100%

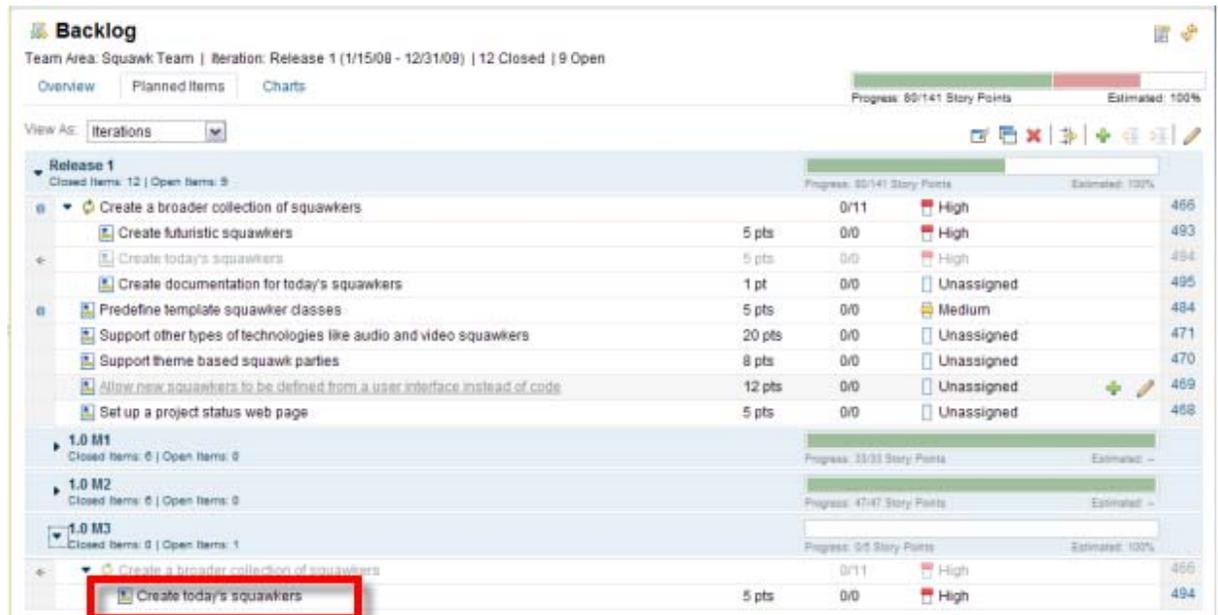
View As: **Iterations**

Item	Progress	Points	Open	Priority	Estimate
<b>Release 1</b>	Progress: 80/141 Story Points		9		Estimated: 100%
▶ Create a broader collection of squawkers	0/11			High	466
▶ Predefine template squawker classes	5 pts	0/0		Medium	484
▶ Support other types of technologies like audio and video squawkers	20 pts	0/0		Unassigned	471
▶ Support theme based squawk parties	8 pts	0/0		Unassigned	470
▶ Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0		Unassigned	469
▶ Set up a project status web page	5 pts	0/0		Unassigned	468
<b>1.0 M1</b>	Progress: 33/33 Story Points		0		Estimated: --
<b>1.0 M2</b>	Progress: 47/47 Story Points		0		Estimated: --
<b>1.0 M3</b>	No Work		0		Estimated: --

- c. Select the story **Create today's squawkers** in the Release 1 area and drag into the 1.0 M3 area



- d. The iteration plan will look as follows



- 7. Move the story **Create documentation for today's squawkers** to 1.0 M3

\_\_a. Drag the story to the 1.0 M3 iteration

**Backlog**  
Team Area: Squawk Team | Iteration: Release 1 (1/15/08 - 12/31/09) | 12 Closed | 9 Open

Overview | Planned Items | Charts

View As: Iterations

Item	Points	Progress	Priority	Estimate
Release 1		Progress: 80/141 Story Points		Estimated: 100%
Create a broader collection of squawkers		0/11	High	466
Create futuristic squawkers	5 pts	0/0	High	493
Create today's squawkers	5 pts	0/0	High	494
Create documentation for today's squawkers	1 pt	0/0	Unassigned	495
Predefine template squawker classes	5 pts	0/0	Medium	484
Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	471
Support theme based squawk parties	8 pts	0/0	Unassigned	470
Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	469
Set up a project status web page	5 pts	0/0	Unassigned	468
1.0 M1		Progress: 33/33 Story Points		Estimated: --
1.0 M2		Progress: 47/47 Story Points		Estimated: --
1.0 M3		Progress: 3/5 Story Points		Estimated: 100%
Create a broader collection of squawkers		0/11	High	466
Create today's squawkers	5 pts	0/0	High	494

\_\_b. Your Plan will now look as follows

**Backlog**  
Team Area: Squawk Team | Iteration: Release 1 (1/15/08 - 12/31/09) | 12 Closed | 9 Open

Overview | Planned Items | Charts

View As: Iterations

Item	Points	Progress	Priority	Estimate
Release 1		Progress: 80/141 Story Points		Estimated: 100%
Create a broader collection of squawkers		0/11	High	466
Create futuristic squawkers	5 pts	0/0	High	493
Create today's squawkers	5 pts	0/0	High	494
Create documentation for today's squawkers	1 pt	0/0	Unassigned	495
Predefine template squawker classes	5 pts	0/0	Medium	484
Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	471
Support theme based squawk parties	8 pts	0/0	Unassigned	470
Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	469
Set up a project status web page	5 pts	0/0	Unassigned	468
1.0 M1		Progress: 33/33 Story Points		Estimated: --
1.0 M2		Progress: 47/47 Story Points		Estimated: --
1.0 M3		Progress: 3/5 Story Points		Estimated: 100%
Create a broader collection of squawkers		0/11	High	466
Create today's squawkers	5 pts	0/0	High	494
Create documentation for today's squawkers	1 pt	0/0	Unassigned	495



Note the Plan Item **Create a broader collection of squawkers** is grayed out, while the two new stories are in normal text. The grayed-out text indicates that this work item is owned by another part of the plan: Release 1.

## \_\_8. View the stories in the Teams View



When you created the 2 stories, you selected the **Filed Against** attribute based on the **Work Item Category** or the functional area of the project to which the story belongs to. Each category is associated with a team area whose members are responsible for developing that component.

The team lead now shows the work item categories created and their associated team areas and then shows the plan in a view mode called **Teams**. This view lists all work items belonging to each team within the project based on the work item's category and the category's associated team area.

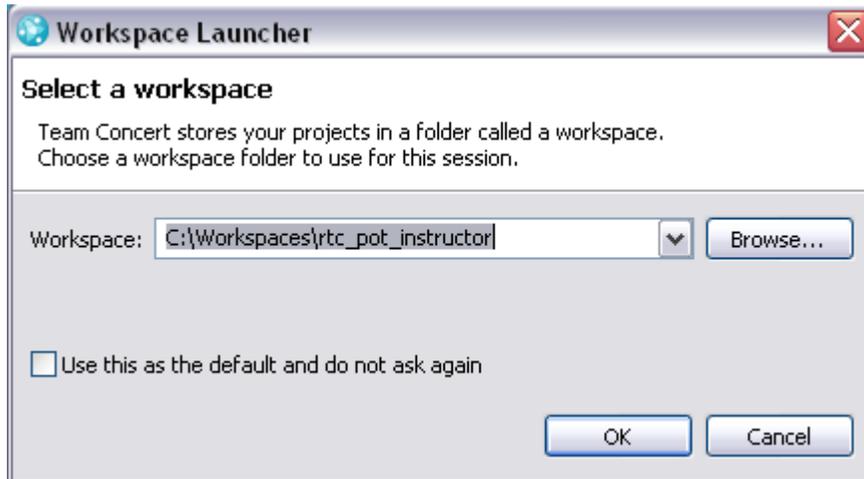
- \_\_a. To view the work item categories, the instructor must use the Eclipse client. The instructor must use the `C:\Workspaces\rtc_pot_instructor` Eclipse workspace. If Rational Team Concert is not yet running, do the following:

- \_\_i. Open Rational Team Concert by double clicking the Team Concert shortcut

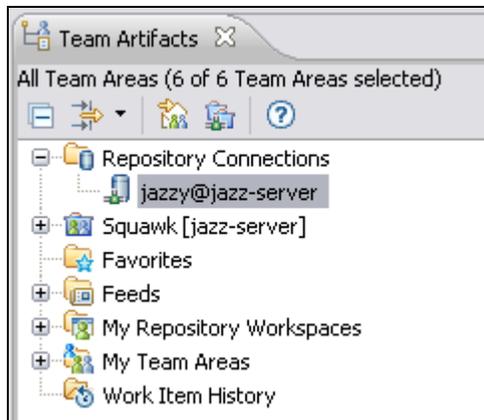


on the Windows Desktop.

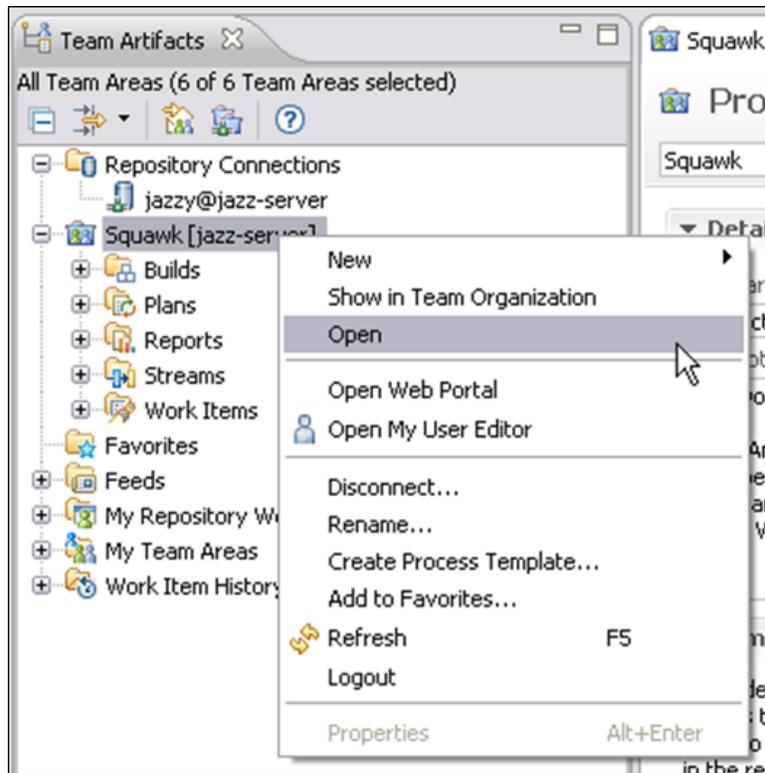
- \_\_ii. In the Workspace **Launcher** dialog that is displayed, make sure to select the instructor’s workspace: C:\Workspaces\rtc\_pot\_instructor. Click **OK**.



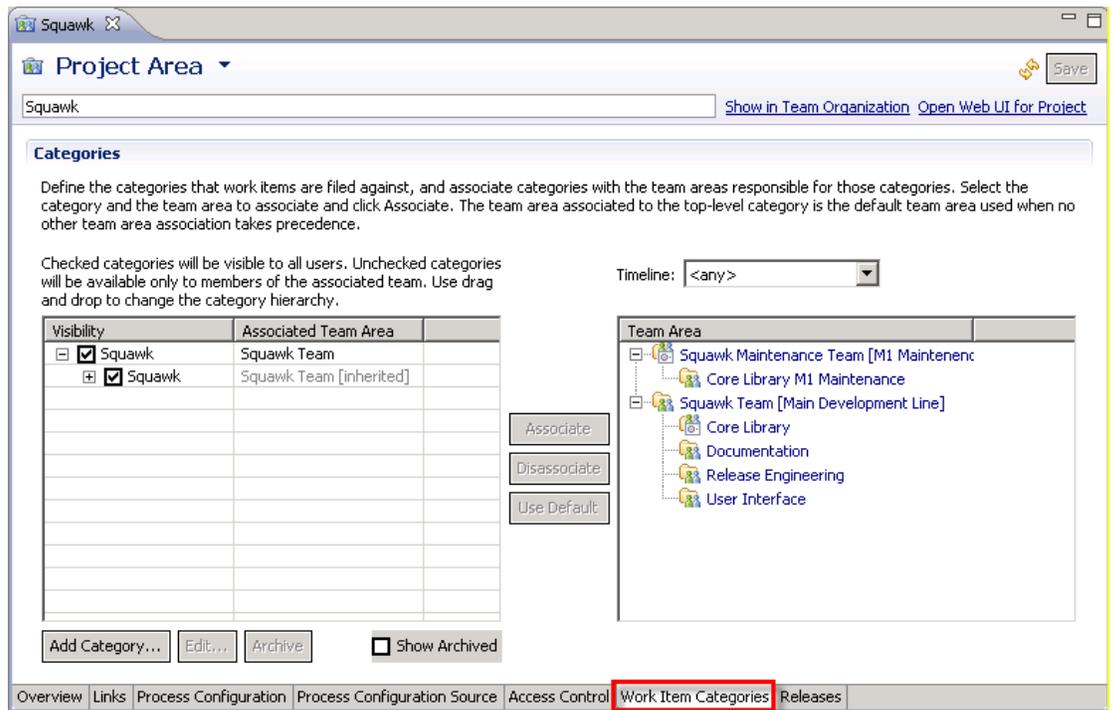
- \_\_iii. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), ensure that you are connected to the **Squawk** Project area as *jazzy*.



- \_\_b. Open the project area to view the work item categories and their associated team areas
- \_\_i. Right-click the **Squawk** project area in the **Team Artifacts** view and select **Open**.

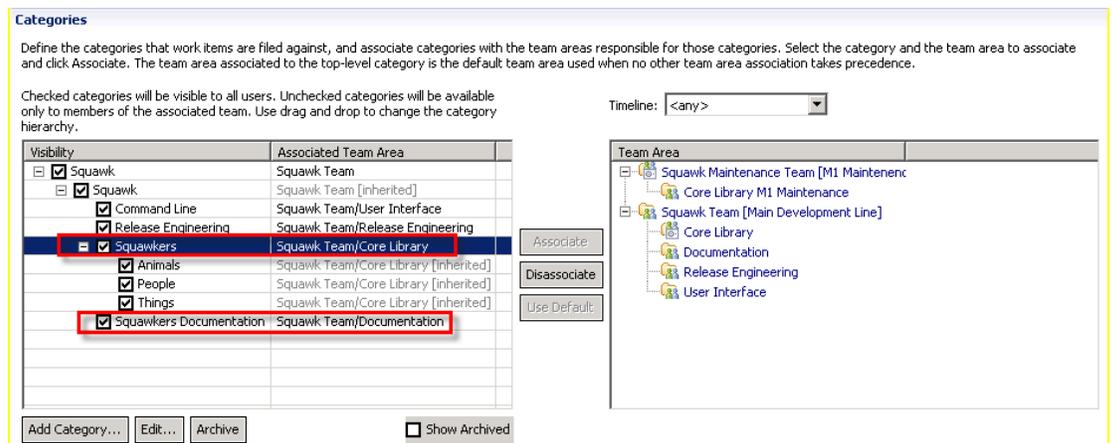


- \_\_ii. Select the **Work Item Categories** tab in the Project Area Editor.



- \_\_iii. Click the **+** button to expand all the sub categories.

- \_\_iv. The **Associated Team Area** column in the category section displays the team area associated with each category. For example the Squawk/Squawkers category is associated with the Core Library Team Area and the Squawkers Documentation category is associated with the Documentation Team Area.

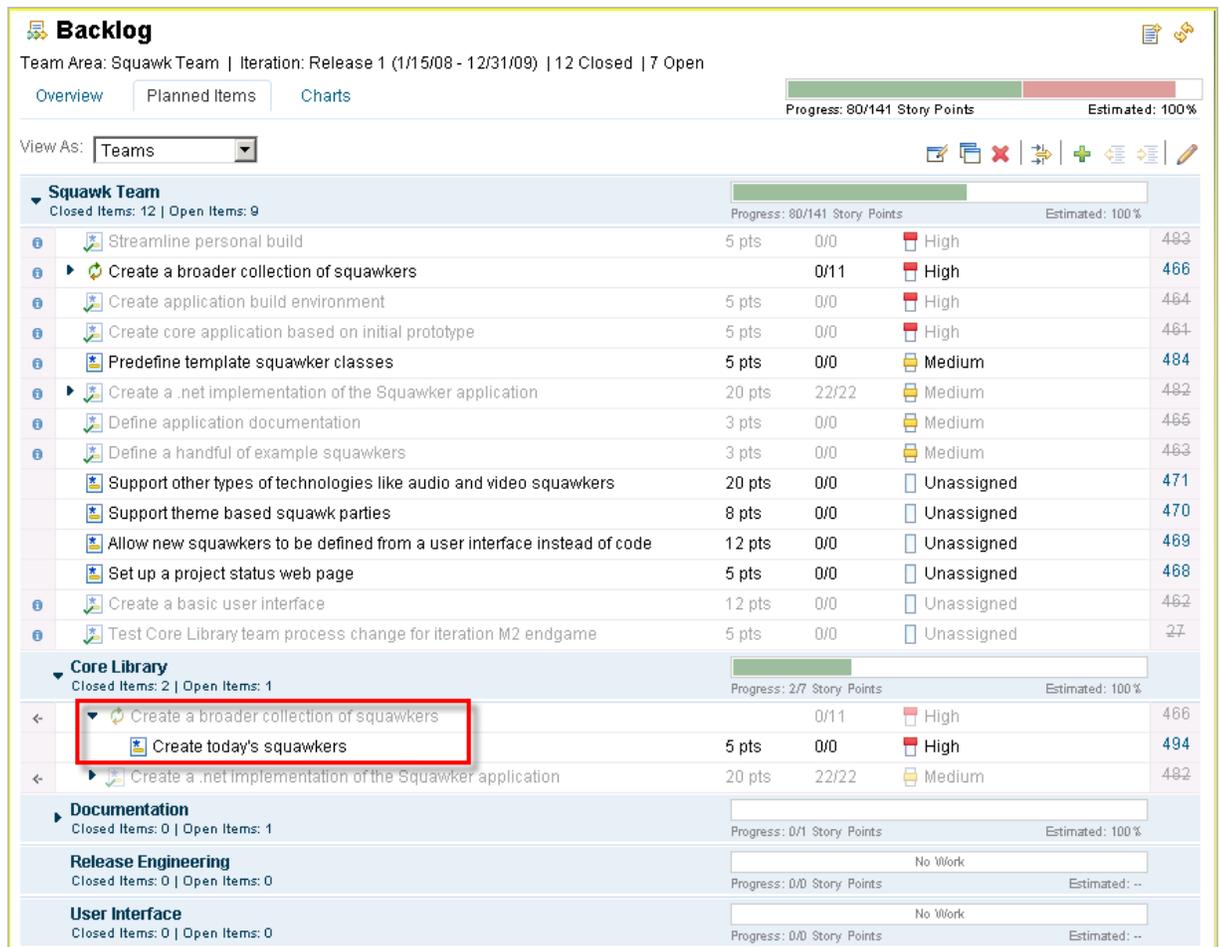


c. In the Web UI, click the **Planned Items** tab on the Plan and select **View As: Teams**

The screenshot shows the IBM Backlog interface for the 'Squawk Team' in 'Release 1 (1/15/08 - 12/31/09)'. The 'Planned Items' tab is active. A 'View As:' dropdown menu is open, with 'Teams' selected and highlighted by a red box. The main backlog table lists items with their points, progress, and priority. A progress bar at the top indicates 80/141 Story Points completed (100% estimated).

Item	Points	Progress	Priority	Estimate
Work Breakdown	0/11	0%	High	466
Predefine template squawker classes	5 pts	0/0	Medium	484
Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	471
Support theme based squawk parties	8 pts	0/0	Unassigned	470
Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	469
Set up a project status web page	5 pts	0/0	Unassigned	468
<b>1.0 M1</b>		33/33 Story Points		Estimated: --
<b>1.0 M2</b>		47/47 Story Points		Estimated: --
<b>1.0 M3</b>		0/6 Story Points		Estimated: 100%
Create a broader collection of squawkers	0/11	0%	High	466
Create today's squawkers	5 pts	0/0	High	494
Create documentation for today's squawkers	1 pt	0/0	Unassigned	495

- d. The Filed Against attribute for the Story **Create today's squawkers** is set to Squawk/Squawkers. This work item category is associated with the **Core Library** team area. Click the twisty icon  to expand the **Core Library** section. Select the arrow next to the Plan Item **Create a broader collection of squawkers** to expose the Story **Create today's squawkers**.

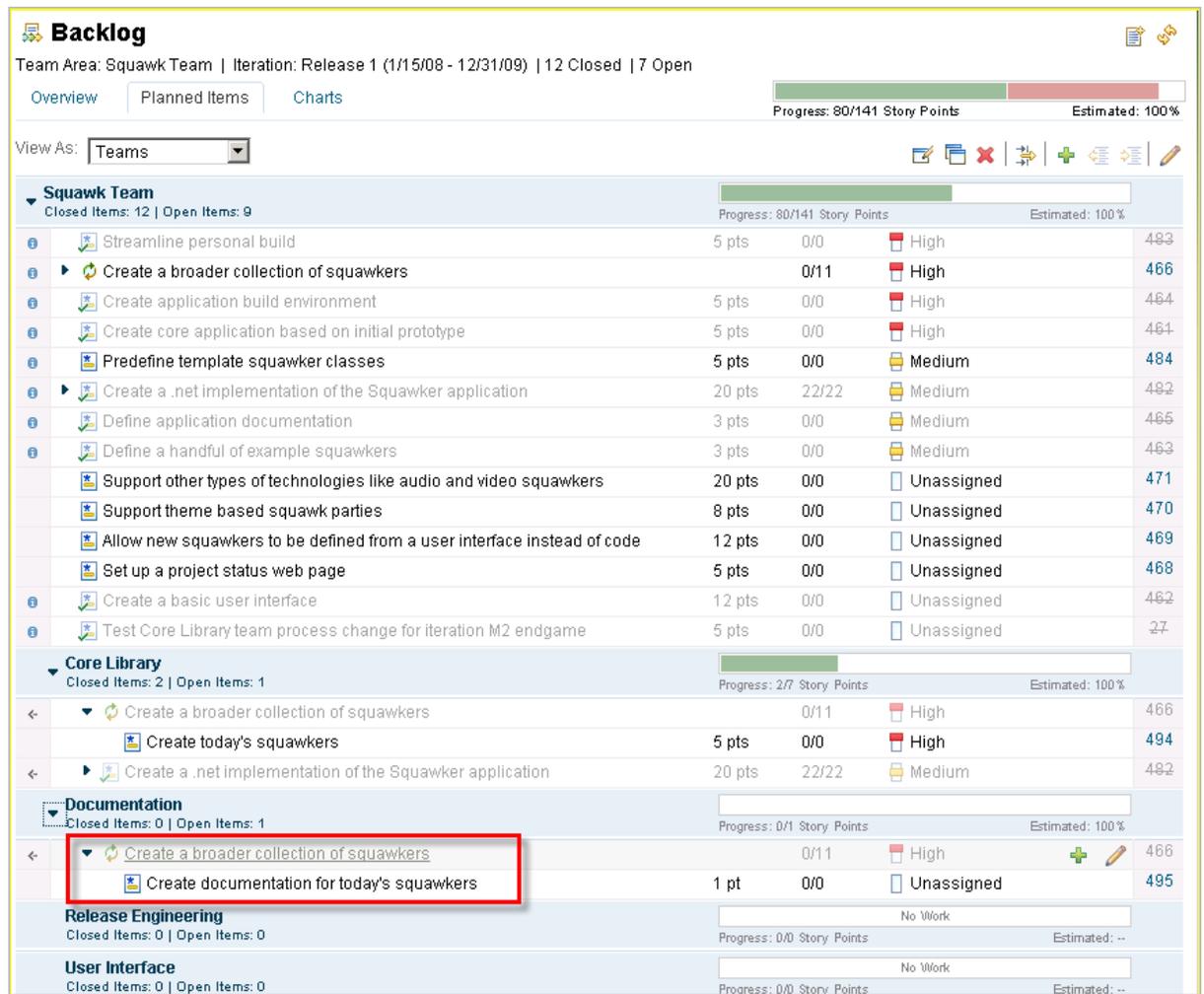


**Backlog**  
 Team Area: Squawk Team | Iteration: Release 1 (1/15/08 - 12/31/09) | 12 Closed | 7 Open  
 Overview | Planned Items | Charts  
 Progress: 80/141 Story Points Estimated: 100%

View As: Teams

Squawk Team		Progress: 80/141 Story Points		Estimated: 100%	
Closed Items: 12   Open Items: 9					
	Streamline personal build	5 pts	0/0	High	483
	Create a broader collection of squawkers		0/11	High	466
	Create application build environment	5 pts	0/0	High	464
	Create core application based on initial prototype	5 pts	0/0	High	461
	Predefine template squawker classes	5 pts	0/0	Medium	484
	Create a .net implementation of the Squawker application	20 pts	22/22	Medium	482
	Define application documentation	3 pts	0/0	Medium	465
	Define a handful of example squawkers	3 pts	0/0	Medium	463
	Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	471
	Support theme based squawk parties	8 pts	0/0	Unassigned	470
	Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	469
	Set up a project status web page	5 pts	0/0	Unassigned	468
	Create a basic user interface	12 pts	0/0	Unassigned	462
	Test Core Library team process change for iteration M2 endgame	5 pts	0/0	Unassigned	27
Core Library		Progress: 2/7 Story Points		Estimated: 100%	
Closed Items: 2   Open Items: 1					
	Create a broader collection of squawkers		0/11	High	466
	Create today's squawkers	5 pts	0/0	High	494
	Create a .net implementation of the Squawker application	20 pts	22/22	Medium	482
Documentation		Progress: 0/1 Story Points		Estimated: 100%	
Closed Items: 0   Open Items: 1					
Release Engineering		No Work		Estimated: --	
Closed Items: 0   Open Items: 0		Progress: 0/0 Story Points			
User Interface		No Work		Estimated: --	
Closed Items: 0   Open Items: 0		Progress: 0/0 Story Points			

- \_\_e. The Filed Against attribute for the Story **Create documentation for today's squawkers** is set to Squawkers Documentation. This work item category is associated with the **Core Library** team area. Click the twisty icon  to expand the **Documentation** section. Select the arrow next to the Plan Item **Create a broader collection of squawkers** to expose the Story **Create documentation for today's squawkers**.



**Backlog**  
Team Area: Squawk Team | Iteration: Release 1 (1/15/08 - 12/31/09) | 12 Closed | 7 Open

Overview | Planned Items | Charts

Progress: 80/141 Story Points Estimated: 100%

View As: Teams

Team Area	Item Name	Points	Progress	Priority	Estimated
<b>Squawk Team</b>	Streamline personal build	5 pts	0/0	High	483
	▶ Create a broader collection of squawkers		0/11	High	466
	▶ Create application build environment	5 pts	0/0	High	484
	▶ Create core application based on initial prototype	5 pts	0/0	High	484
	▶ Predefine template squawker classes	5 pts	0/0	Medium	484
	▶ Create a .net implementation of the Squawker application	20 pts	22/22	Medium	482
	▶ Define application documentation	3 pts	0/0	Medium	465
	▶ Define a handful of example squawkers	3 pts	0/0	Medium	463
	▶ Support other types of technologies like audio and video squawkers	20 pts	0/0	Unassigned	471
	▶ Support theme based squawk parties	8 pts	0/0	Unassigned	470
	▶ Allow new squawkers to be defined from a user interface instead of code	12 pts	0/0	Unassigned	469
	▶ Set up a project status web page	5 pts	0/0	Unassigned	468
	▶ Create a basic user interface	12 pts	0/0	Unassigned	462
	▶ Test Core Library team process change for iteration M2 endgame	5 pts	0/0	Unassigned	27
<b>Core Library</b>	Create a broader collection of squawkers		0/11	High	466
	▶ Create today's squawkers	5 pts	0/0	High	494
	▶ Create a .net implementation of the Squawker application	20 pts	22/22	Medium	482
<b>Documentation</b>	▶ Create a broader collection of squawkers		0/11	High	466
	▶ Create documentation for today's squawkers	1 pt	0/0	Unassigned	495
<b>Release Engineering</b>	No Work		0/0		--
<b>User Interface</b>	No Work		0/0		--

- \_\_9. You have demonstrated how to look at your backlog for **Release 1**, create new stories, make them children of a Plan Item **Create a broader collection of squawkers**, assign them to the **1.0 M3** iteration and look at your teams work items. The students will now create tasks against these stories.

## A2.5 Review plans and distribute

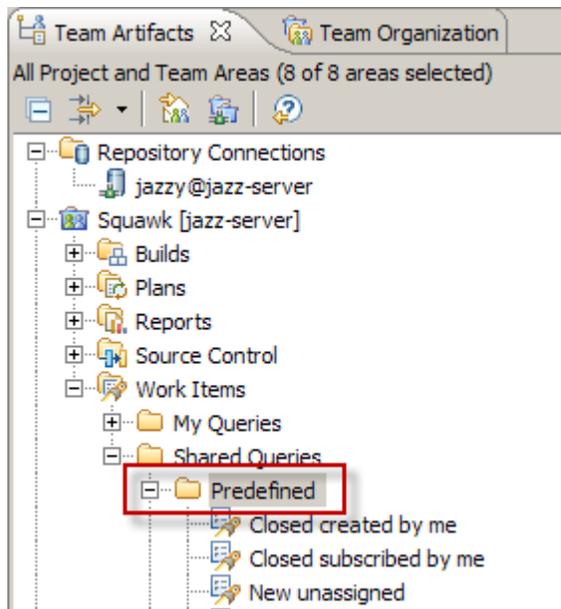


Now all the students have completed creating their tasks, the instructor will check that all the tasks are children of the stories. If any tasks are not associated with the correct story, or are orphaned, move them to the correct story.

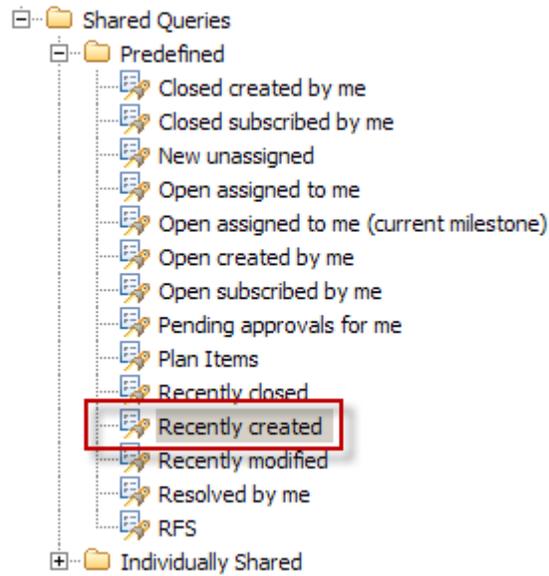
The next step is to send out the story to all students via the chat window.

\_\_1. Check the new tasks created by the students are all children of the story **Create today's squawkers**.

\_\_a. In the **Team Artifacts** view, expand the Squawk project and select **Work Items**→**Shared Queries**→**Predefined**.



- \_\_b. Run the **Recently Created** query by double-clicking the query and check each student has created the relevant tasks.



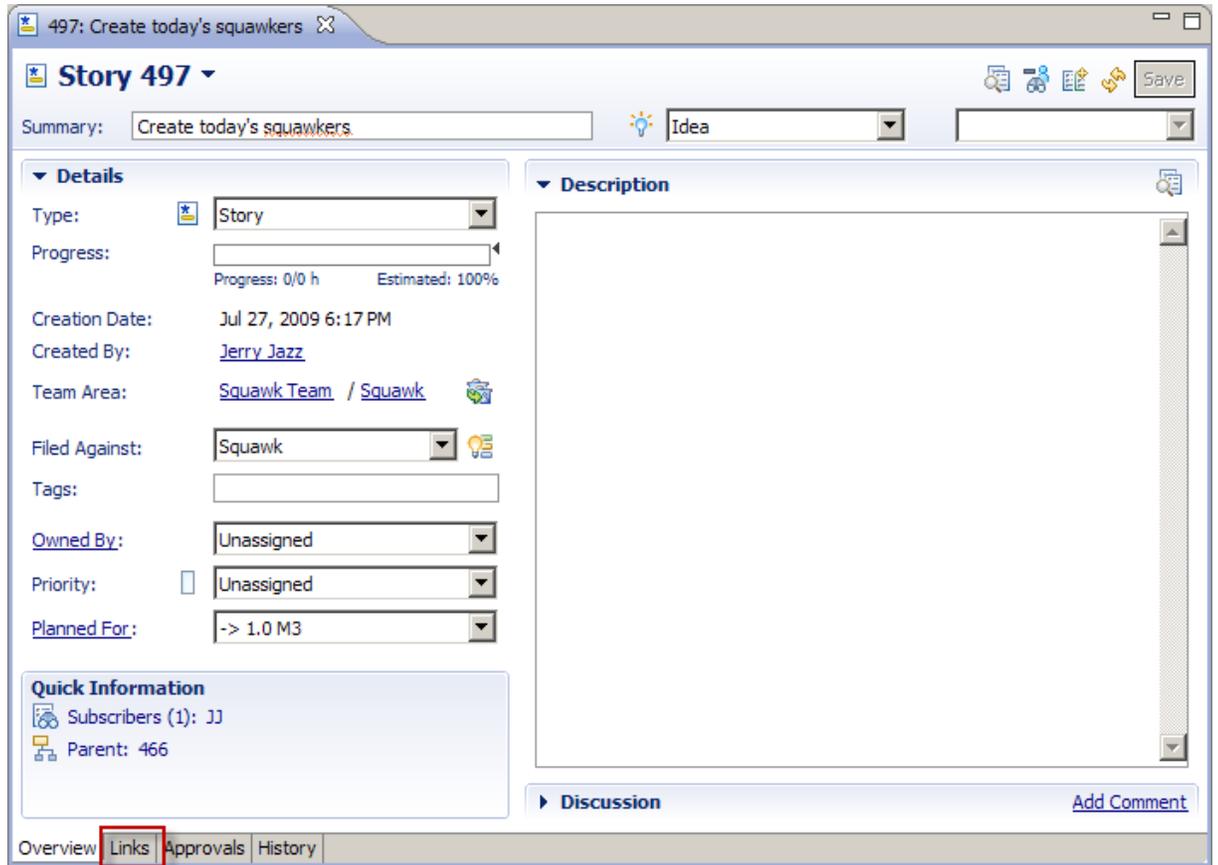
- \_\_c. Find the **Create today's squawkers** story in the Work Items View

The screenshot shows the 'Work Items' view. The toolbar includes 'Tag Cloud', 'Problems', and 'Team Advisor'. Below the toolbar, it says 'Found 1 work item - Recently created'. A table displays the work item details:

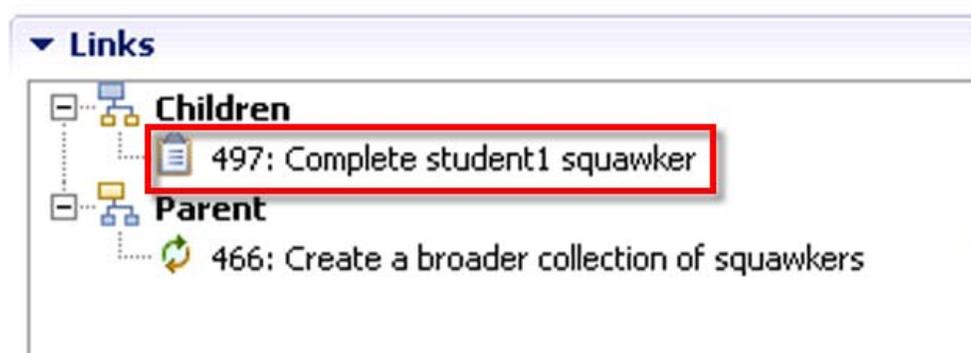
I..	Status	P	S	Summary	Ownr
497	Idea			Create today's squawkers	Ur

The row containing the work item 'Create today's squawkers' is highlighted with a red rectangular box.

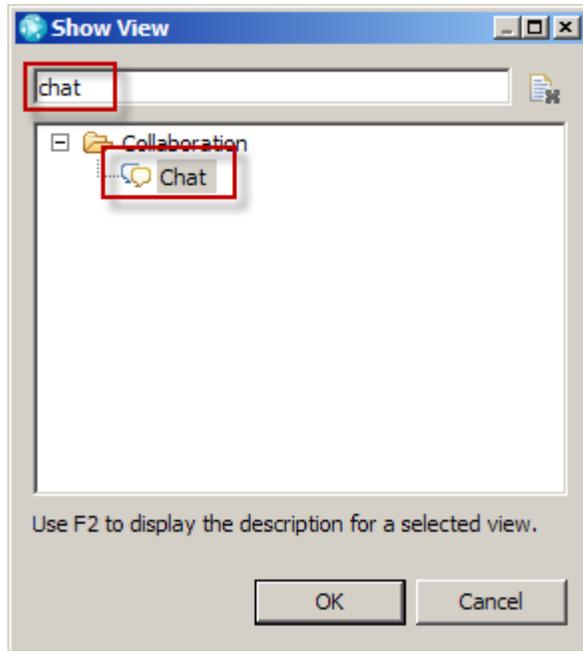
- \_\_d. Open the **Create today's squawkers** story
- \_\_e. Select the **Links** tab



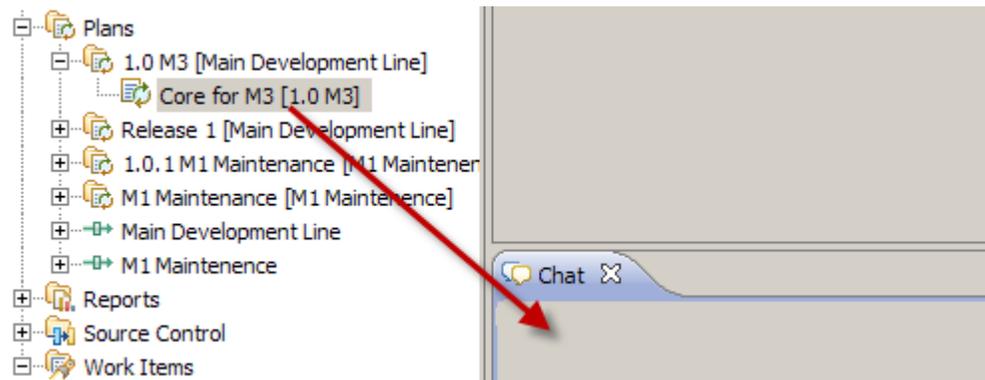
- \_\_f. Check that all the students have created a child task in the Links area and correct as required



- \_\_2. Send the Core for M3 Plan to all students
  - \_\_a. Open the Chat window from the menu **Window→Show View→Other...** and type in Chat



- \_\_b. Select all the students in the chat window
- \_\_c. Drag the **Core for M3** plan from the **Team Artifacts** view into the **Chat** window



---

## Appendix B: Lab 5      Remembering Well Known SCM Configurations – Demo



### Demo Scenario

Thus far, you have contributed your own squawker class and delivered your work so that other people can use it. You have not been working in isolation. Your teammates have been creating and delivering their own squawkers. As part of the last exercise, you accepted these changes from other members of your team bringing your code up-to-date with everyone else. The instructor will now play the role of the Team Lead, creating baselines and snapshots, which can be retrieved and used at any point in the future.



### Lab Scenario

After the instructor creates baselines and snapshots, you will explore these new objects by querying their contents. The replace operation provides a convenient way to reconfigure your workspace.



### Important!

All steps in this section are to be performed by the instructor as a demo.

## B5.1 Creating baselines for the Core Library Component

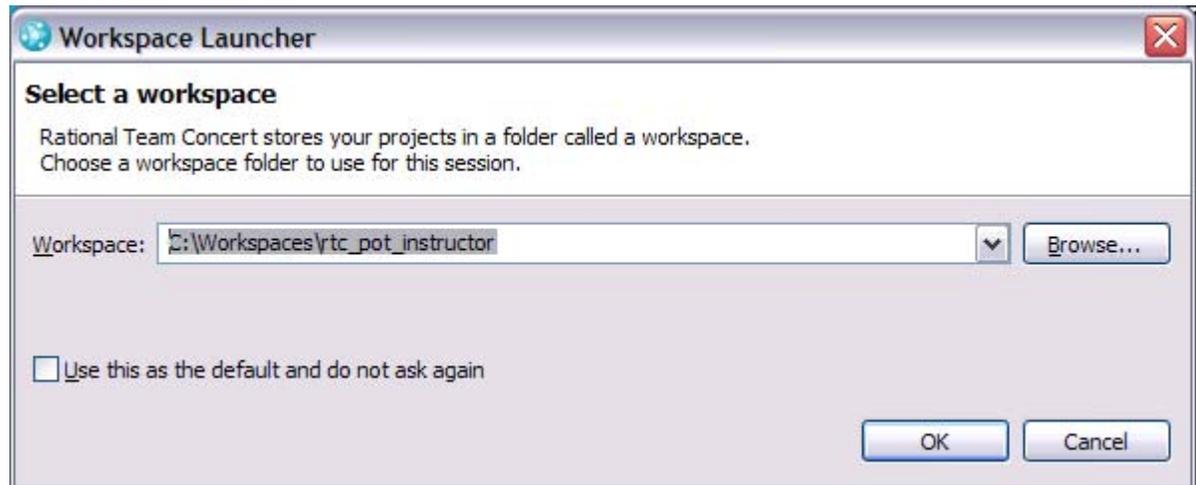


### Team role

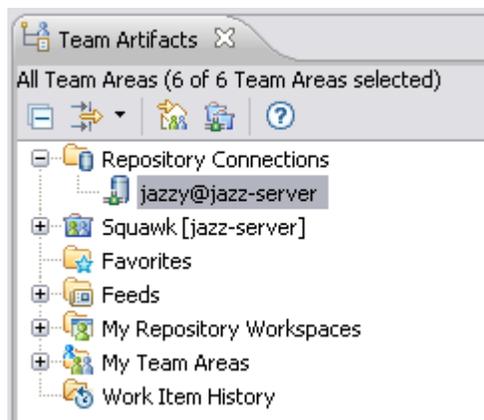
The instructor will now perform the role of **Jerry**, the Team Lead.

- \_\_1. The instructor must use the `C:\Workspaces\rtc_pot_instructor` Eclipse workspace. If Rational Team Concert is not yet running, do the following:
  - \_\_a. Open Rational Team Concert by double-clicking the Team Concert shortcut  on the Windows Desktop.

- \_\_b. In the Workspace **Launcher** dialog that is displayed, make sure to select the instructor's workspace: C:\Workspaces\rtc\_pot\_instructor. Click **OK**.

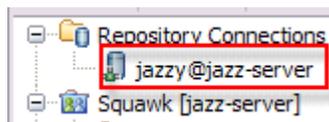


- \_\_c. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), ensure that you are connected to the **Squawk** Project area as *jazzy*.

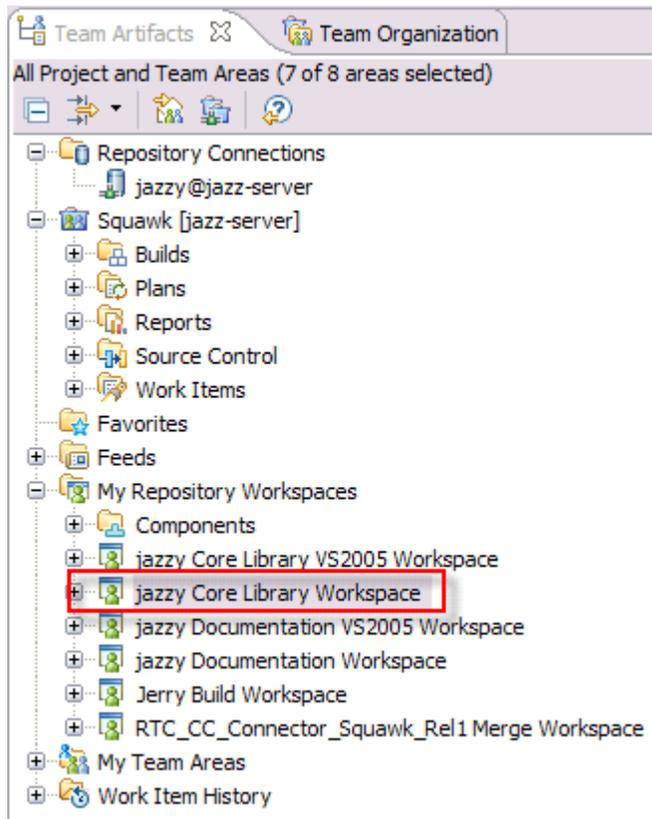


- \_\_2. Load your workspace jazzy Core Library Workspace

- \_\_a. In the **Team Artifacts** view (**Window → Show View → Team Artifacts**), ensure that you are connected to the **Squawk** project area as *jazzy*

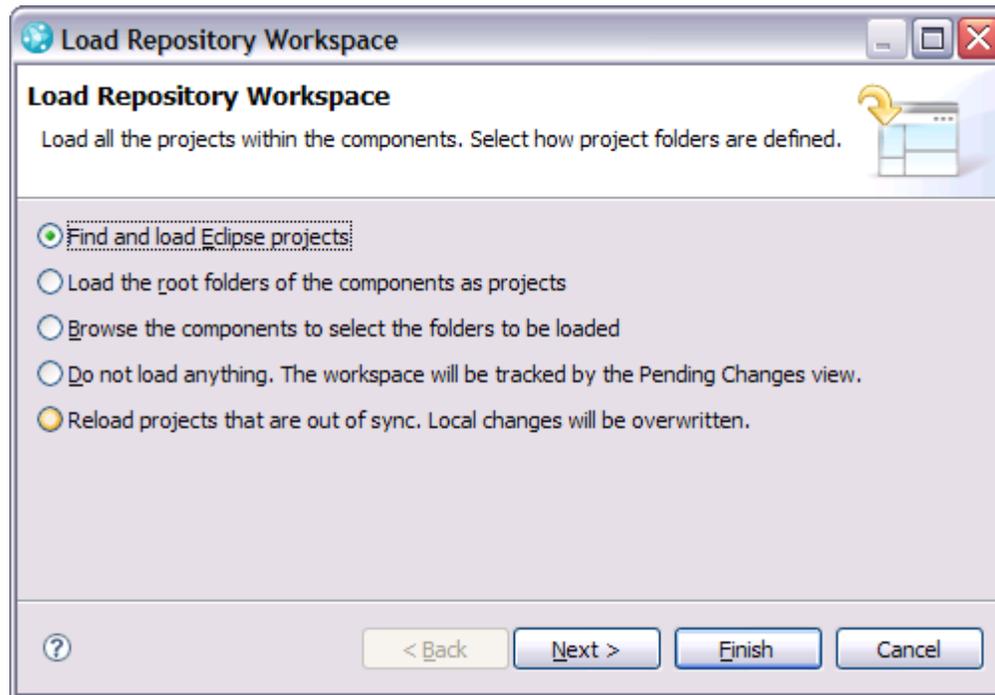


- \_\_b. Expand **My Repository Workspaces** and select **jazzy Core Library Workspace**.

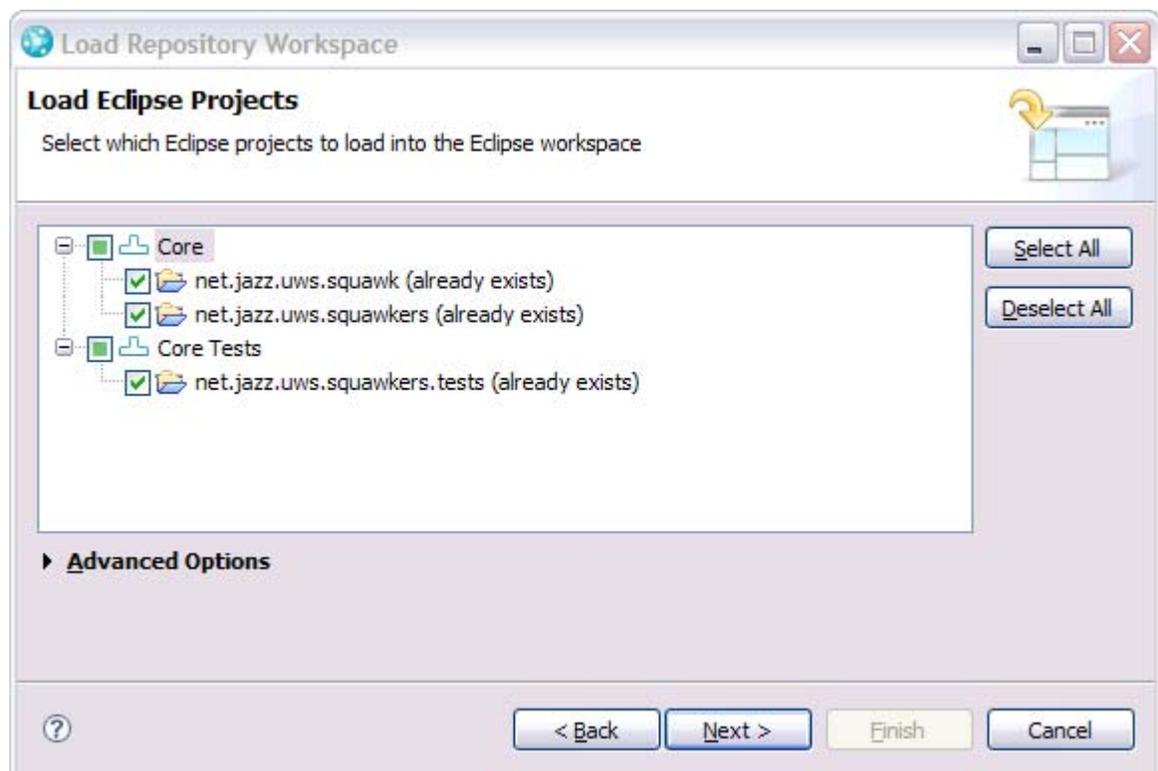


- \_\_c. Right-click the **jazzy Core Library Workspace** stream and select **Load....**

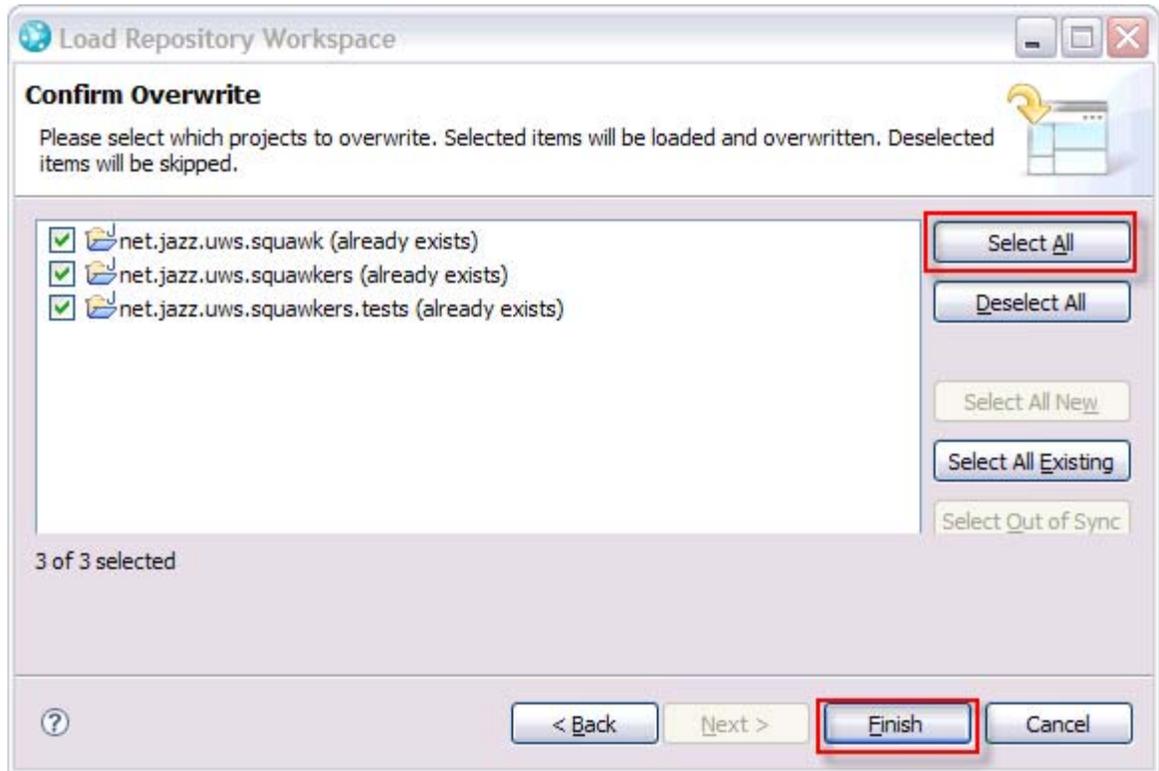
- \_\_\_d. On the **Load Repository Workspace** window, select **Find and load Eclipse projects** and click **Next**.



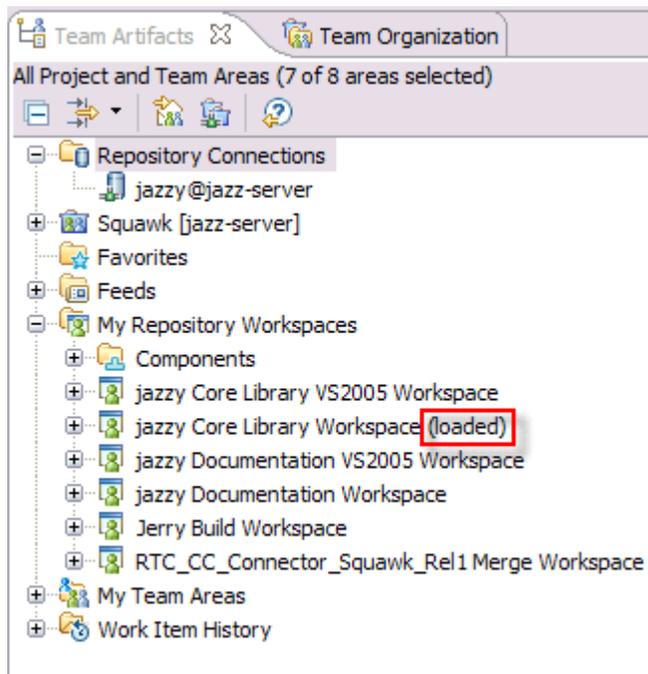
- \_\_\_e. On the **Load Eclipse Projects** window, check that all components are selected and click **Next**.



- \_\_f. If your current Eclipse workspace contains some of these components, the **Confirm Overwrite** window will pop up. Click on **Select All** then click **Finish**.



- \_\_g. Your repository workspace should now be visible as loaded under **My Repository Workspaces** in the **Team Artifacts** view.



- \_\_3. Verify that Jerry's repository workspace delivered/accepted all changes to/from the Core Library Stream.



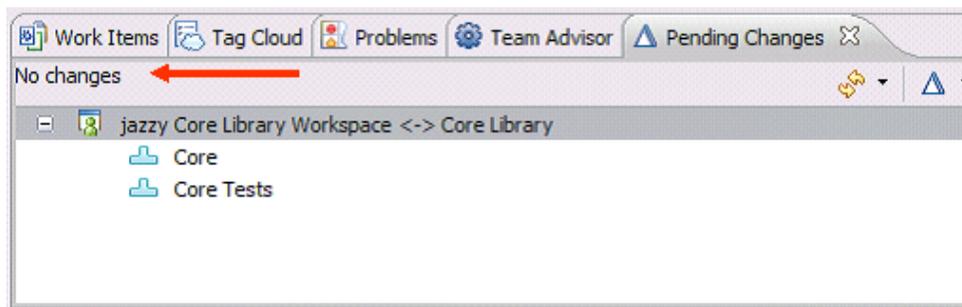
**Important!**

This demo requires that your Core Library repository workspace is up-to-date; with no incoming or outgoing changes remaining in the Pending changes view.

If there are any incoming change sets in the Pending Changes view, right-click those change sets and accept them.

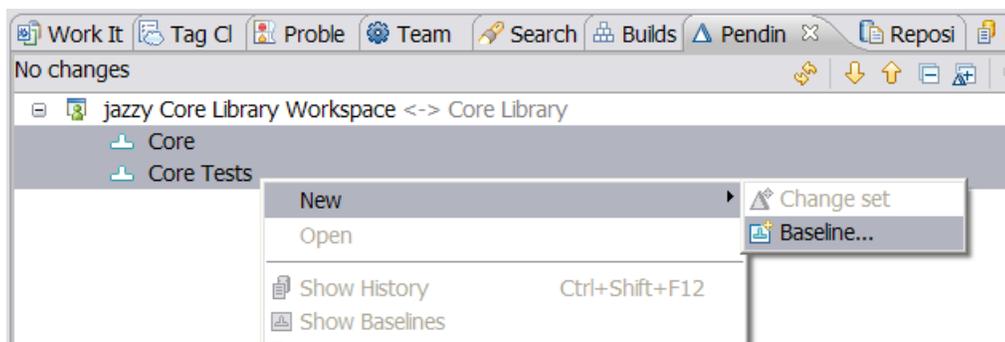
Similarly, if there are any outgoing change sets in the Pending Changes view, right-click and deliver them.

- \_\_a. From the **Pending Changes** view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), accept all incoming changes and deliver any outgoing changes. Proceed to next step once **"No changes"** are pending.

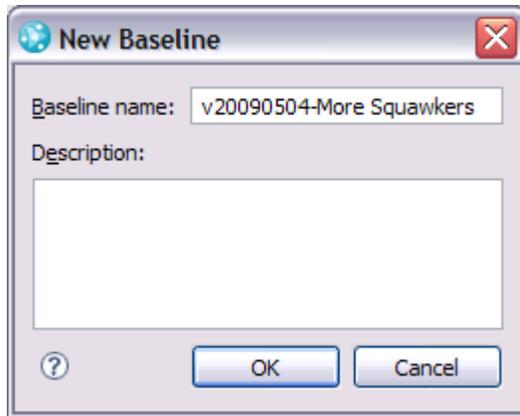


- \_\_4. From Jerry's Core Library Repository Workspace, create a baseline for the Core and Core Tests component and deliver it to the **Core Library** stream.

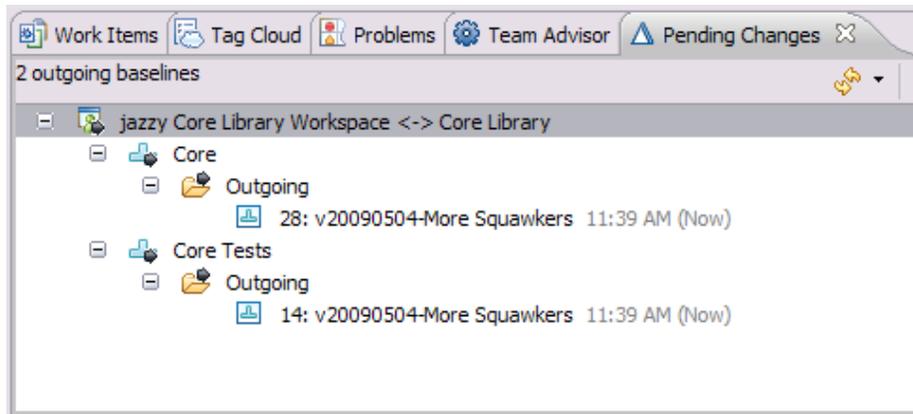
- \_\_a. From the **Pending Changes** view, select the Core Component and Core Tests components, right-click and select **New → Baseline**. Note that this action is also available from other places that components are visible, such as the **Team Artifacts** view or inside a repository workspace editor.



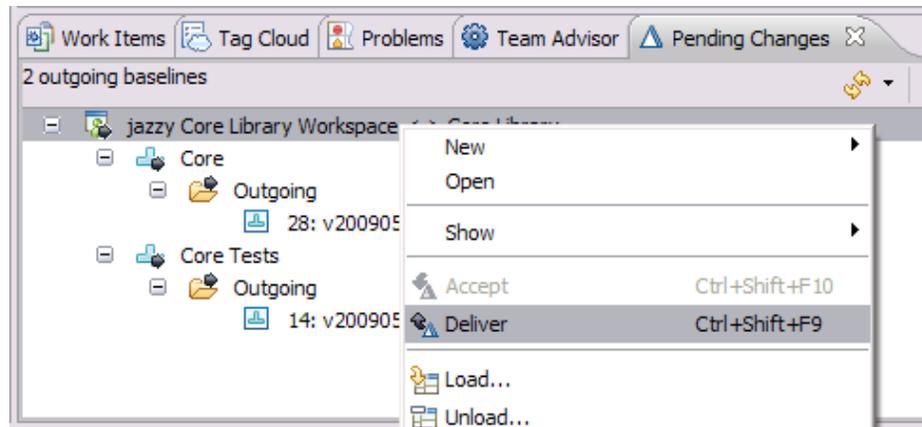
- \_\_i. On the New Baseline dialog, type a name of the form v<yyyymmdd>-More Squawkers.
- \_\_ii. Click **OK**.



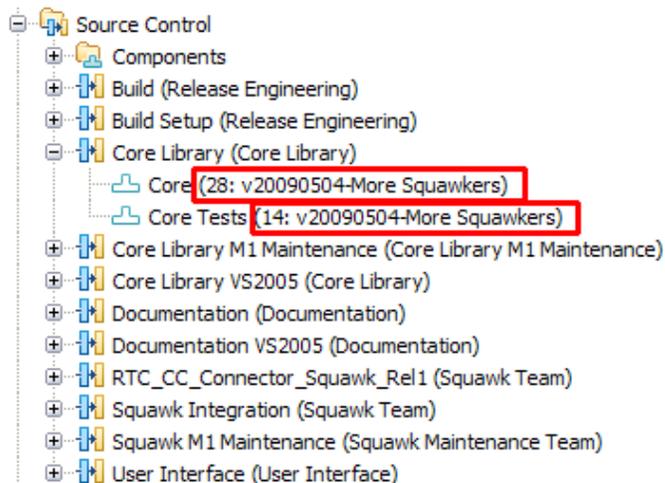
- \_\_b. There is now an outgoing change-set for the Core component. The change-set contains the component's new baseline.



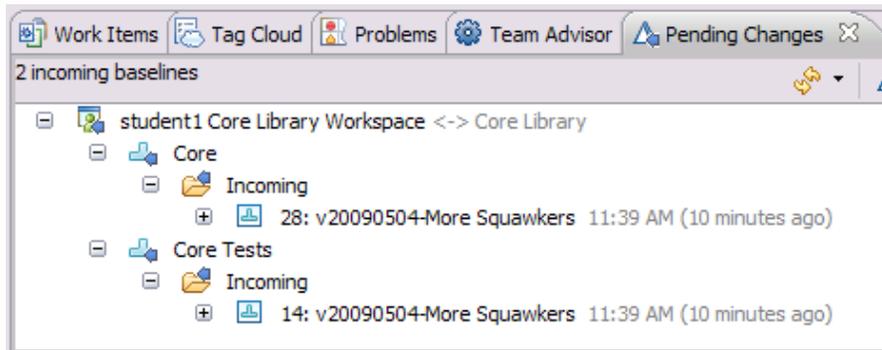
- \_\_c. Right-click the workspace and select **Deliver**. Note that you are just delivering the baseline tag for the current state of Jerry's repository workspace to the stream. All the included change-sets have already been delivered in the previous lab exercise by each student.



- \_\_5. At the end of this section, the Core Library stream should display the new baseline as its foundation configuration



- \_\_\_6. At the end of this section, all students should have incoming changes ready to be accepted in their **Pending Changes** view. They may need to refresh their view to see the incoming change-sets.



## B5.2 Taking a snapshot of the Core Library Repository Workspace

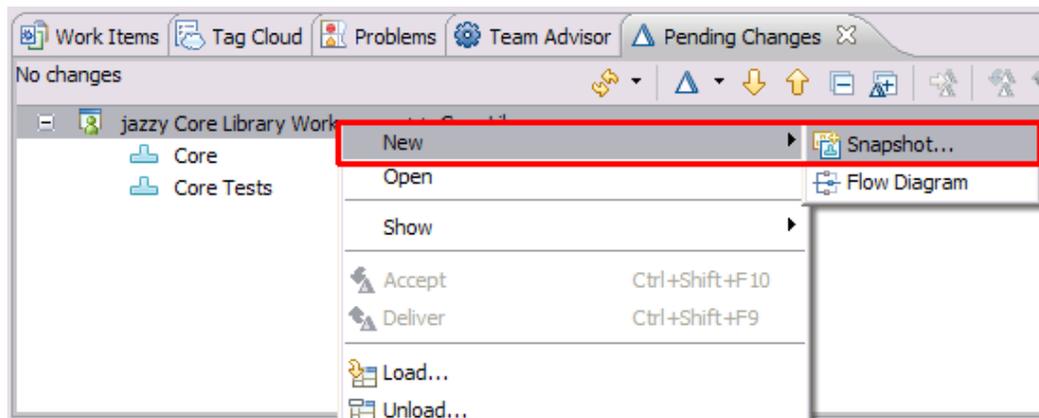


### Team role

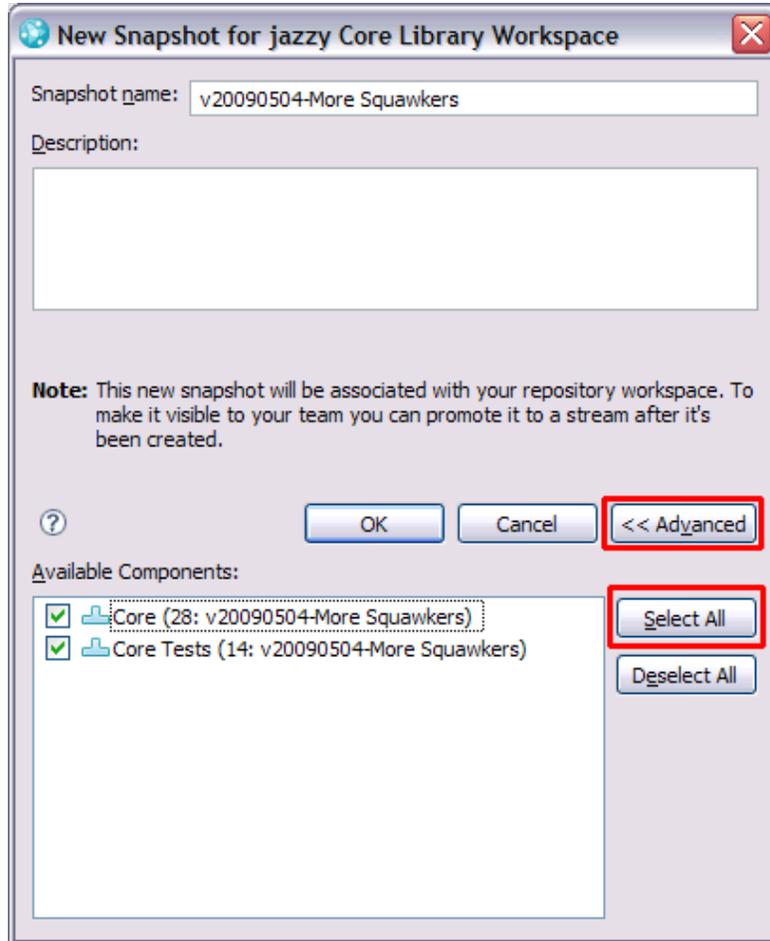
The instructor will now performing the role of **Jerry**, the Team Lead.

\_\_1. Take a snapshot of Jerry's **Core Library Repository Workspace**

\_\_a. Still in the **Pending Changes** view, right-click the **Core Library** repository workspace and click **New->Snapshot**.



- \_\_b. In the **New Snapshot** window:
  - \_\_i. Name the snapshot using the same convention as the baselines, that is, V<yyyymmdd>-More Squawkers .
  - \_\_ii. Click **Advanced** and note that you can selectively choose the components to include in the snapshot. **Select All** components. The snapshot will capture this information.
  - \_\_iii. Click **OK**.



As a result of that, the search view will show up listing all the snapshots you have in your Core Library workspace.



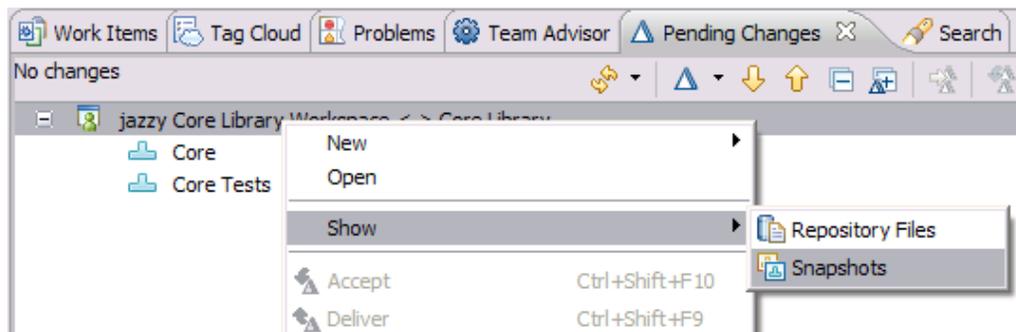
## B5.3 Promote the snapshots to the appropriate streams



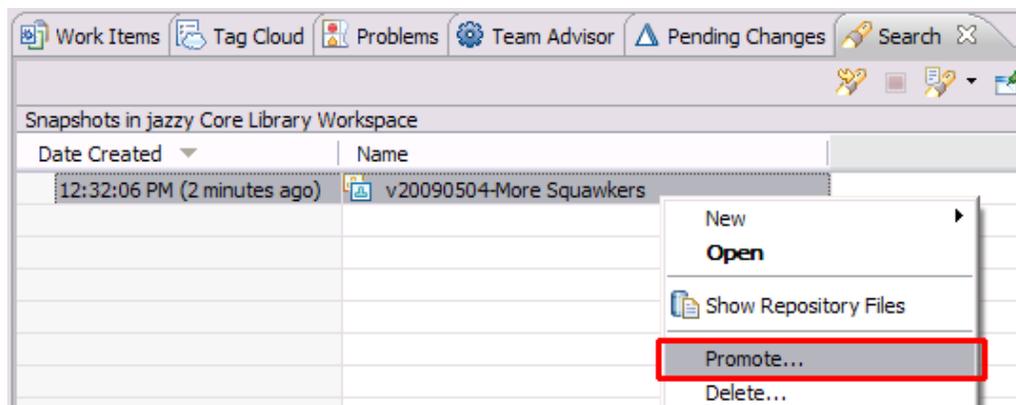
### Snapshot promotion

The snapshot created in section 5.2 is associated with your personal repository workspace. Even though all the change-sets and baselines have been delivered, the snapshot tag is not yet visible at the stream level.

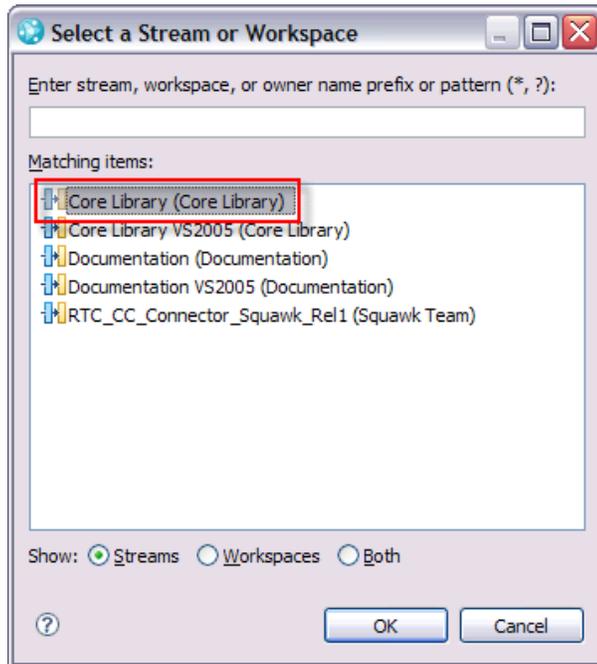
- \_\_1. Promote the new **Core Library** snapshot to the **Core Library** stream
  - \_\_a. If the Search view doesn't display the snapshots stored in your Core Library workspace then, in the **Pending Changes** view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), right-click Jerry's **Core Library** repository workspace and click **Show Snapshots**.



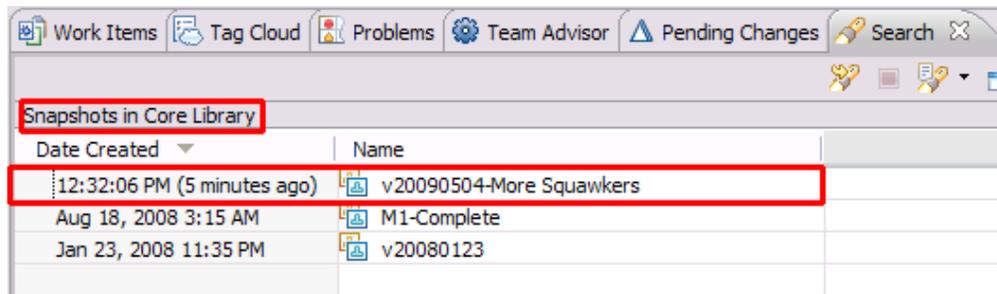
- \_\_b. From the resulting **Search** view, right-click the snapshot you created in section 5.2, and click **Promote**.



- c. In the **Select a Stream or Workspace** window select the **Core Library** stream and click **OK**.



- d. The Core Library stream now has information on the new snapshot.



---

## Appendix C: Lab 6      User's View of Build – Demo



### Lab Scenario

By now you have developed a new squawker, performed some unit tests and created baselines and snapshots. You are now ready to build your application with help of the Team Concert Build Engine.

Your project team will then have a built application you should all be happy with.



### Important!

All steps in this section are to be performed by the instructor as a demo.

### C6.1 Start the Team Concert Build Engine

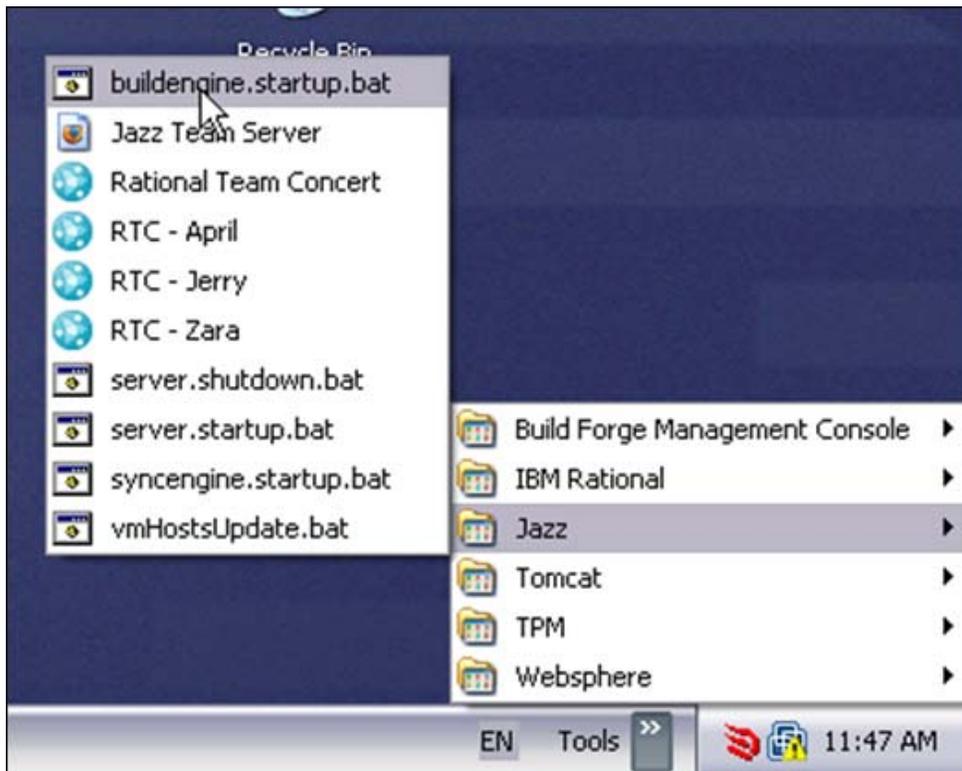
The build engine, which is a specialized Team Concert client, is included with the client installation. Your instructor will start the build engine.



### Team role

The instructor will now perform the role of **Jerry**, the Team Lead.

1. Open the **Tools** menu, navigate to **Jazz** and click **buildengine.startup.bat** to start the build engine.



**Starting the build engine manually (not applicable for Visual Studio\*)**

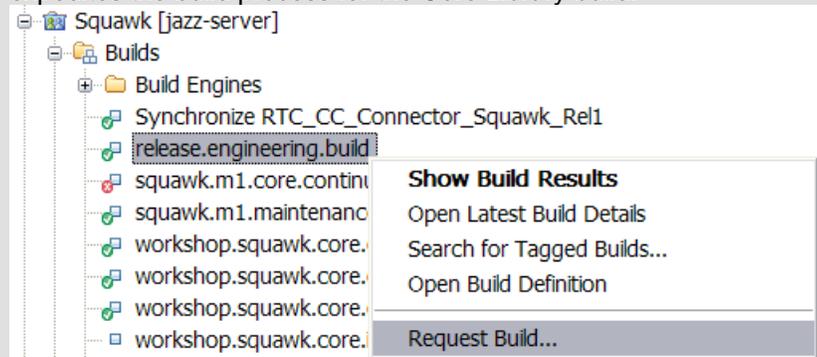
You can also start the build engine manually by navigating to the install location of the build engine executable (C:\Jazz\TeamConcert\buildsystem\buildengine\eclipse) and entering the following command in a command window:

```
jbe -repository https://jazz-server:9443/jazz -userId
build -pass jazz -engineId workshop.build.engine -
sleepTime 3
```

**IMPORTANT:** If you are running outside the VM image there is one additional step. The following only needs to be run one time after installing or updating the RTC build system.



- After starting the build engine request the following build definition to prime the build with the build scripts and supporting files. This expedites the build process for the Core Library build.

**Build User**

The user id build is a Team Concert user with an assigned repository workspace for each type of build. This ensures that the workspaces are configured correctly.

You can run builds with any user id. You actually will run a build with your user id when requesting a private build.

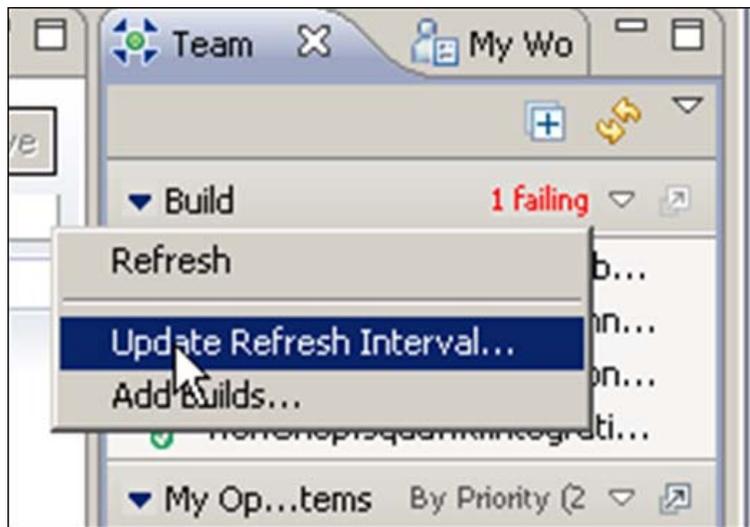
## C6.2 Requesting a Build

In this section, your instructor will request a build. It is also possible to schedule builds to run at specific times.

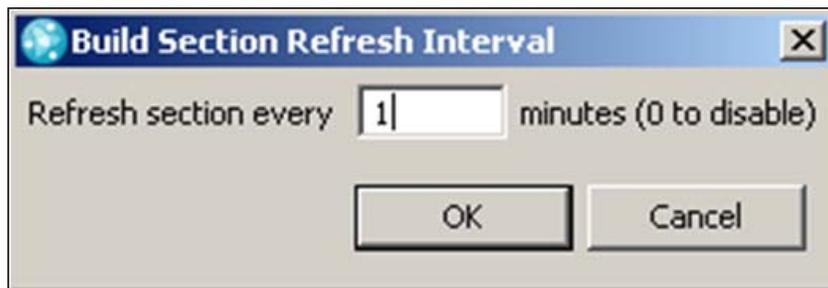
- 1. Enable build alerts in the **Team Central** to get notified even if you are working outside of Rational Team Concert (i.e. another application has focus). These alerts come in the form of 'fly-ins' in the lower right corner of the windows.



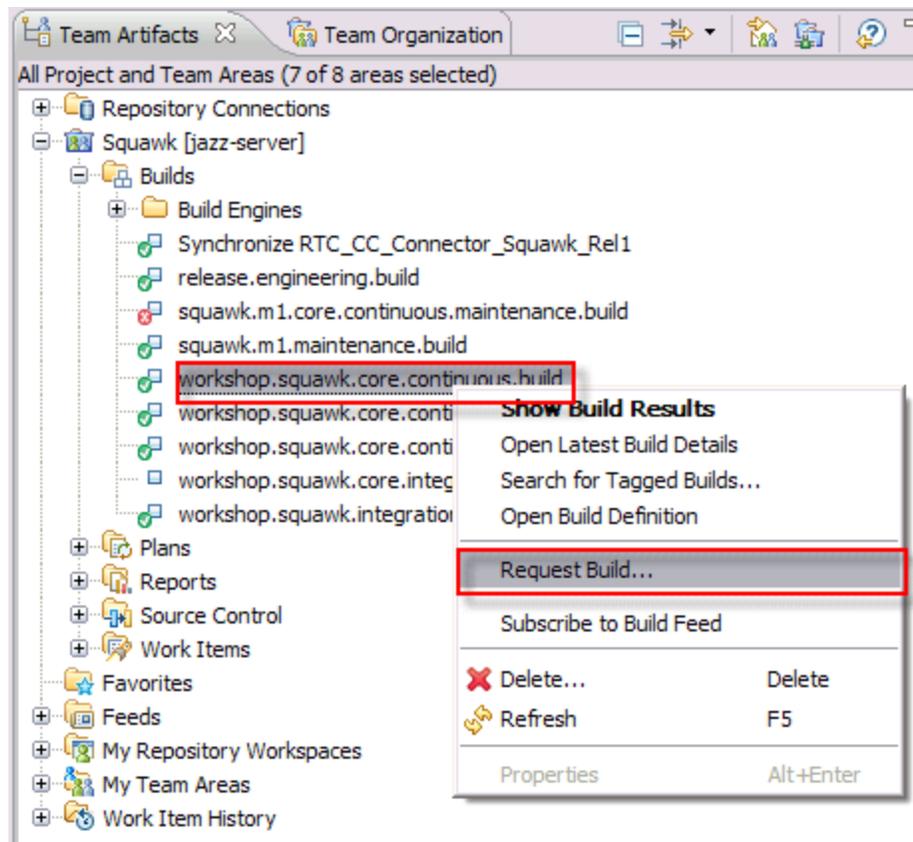
- a. In the **Team Central** view, select the drop down menu in the **Build** section toolbar. Select **Update Refresh Interval...**



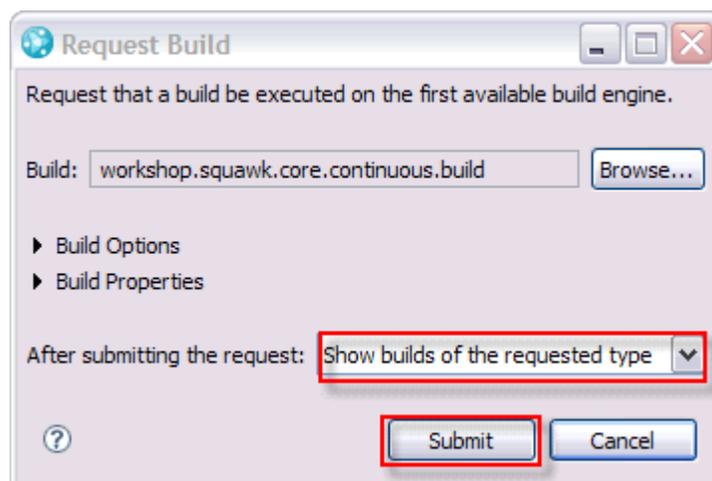
- b. When the **Build Section Refresh Interval** dialog appears, set the refresh value to 1 minute and click **OK**. The default value of zero disables the notification.



- \_\_2. Request a build in the **Team Artifacts** view for the workshop build definition.
- \_\_a. In the **Team Artifacts** view, navigate to **Squawk → Builds**. Right-click *workshop.squawk.core.continuous.build* then select **Request Build...**



- \_\_b. When the **Request Build** dialog appears, make sure that **Show Builds of the requested type** is selected and click **Submit**.



- \_\_\_3. The **Builds** view shows the build you just submitted. Here you see the latest build is in a pending state meaning that it has not started yet. Your build request may show a different progress state.

Build	Label	Progress	Estimated Completion	Start Time	Duration
workshop.squawk.core.co...		Pending			
workshop.squawk.core.co...	B20080514-162...	Completed		May 14, 2008 4:28:0...	18 seconds
workshop.squawk.core.co...	B20080514-162...	Completed		May 14, 2008 4:22:2...	23 seconds
workshop.squawk.core.co...	B20080514-1621	Completed		May 14, 2008 4:21:3...	2 seconds
workshop.squawk.core.co...	B20080514-1621	Completed		May 14, 2008 4:21:1...	1 second
workshop.squawk.core.co...	B20080514-160...	Completed		May 14, 2008 4:07:3...	1 minute, 6 seconds

- \_\_\_4. When the build completes you will receive an alert as a temporary window in the lower right area of your screen.



**Build Troubleshooting Information**

Your instructor can check the build request status in the command window where the build engine was started. So if the build did not start check, have your instructor verifying the build engine command window to see if it received the request.

- \_\_\_5. Open the completed build from the **Builds** view by double clicking it. Inspect your changes and perhaps others delivered since the last build.

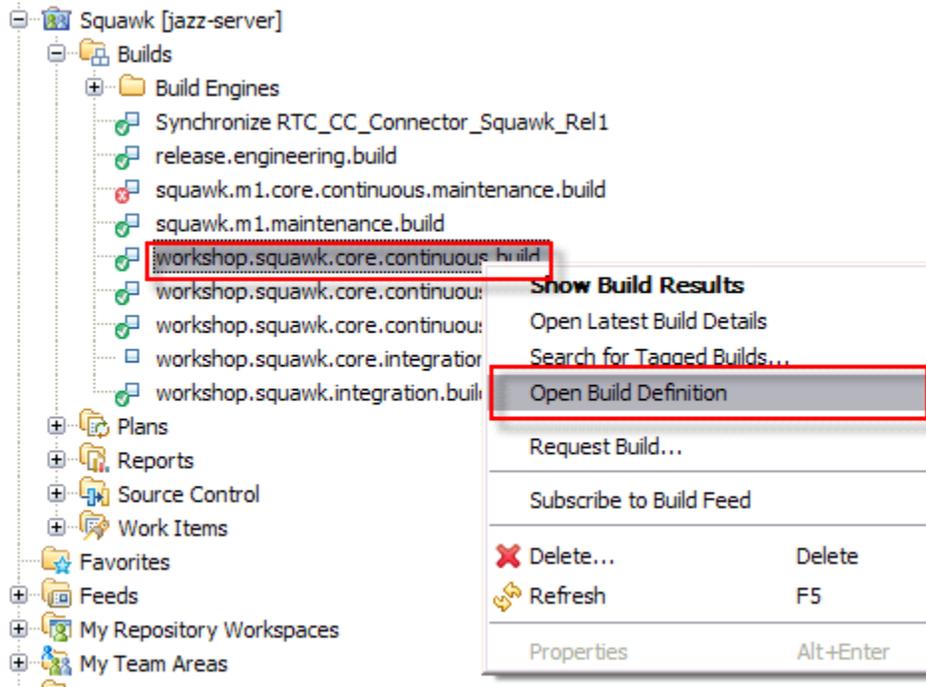
The screenshot shows the 'Build workshop.squawk.core.continuous.build B20090812-1154-workshop.squawk.core.continuous.build' page. The build status is 'Completed' with a duration of 42 seconds, starting at 11:54:10 AM and completing at 11:54:52 AM on August 12, 2009. A status trend bar shows a sequence of red and green bars, indicating a successful build. The page is divided into several sections:

- Reported Work Items:** None reported against this build. Links for 'Create a new work item' and 'Associate an existing work item' are provided.
- Contribution Summary:**
  - Downloads: [1 download](#)
  - External Links: [1 link](#)
  - Logs: [1 log](#)
  - Repository Workspace: [workshop.squawk.core.build.workspace.v2](#)
  - Snapshot: [workshop.squawk.core.continuous.build\\_B20090812-1154](#)
  - JUnit: [39 tests, 0 failures, 0 errors](#)
  - Changes: [Show changes](#)
  - Compile: [0 errors, 0 warnings](#)
  - Work items: [1 included in build](#)
- General Information:**
  - Requested by: Jerry Jazz
  - Build Definition: [workshop.squawk.core.continuous.build](#)
  - Build Engine: [workshop.build.engine](#)
  - Build History: [39 builds](#)
  - Tags:
  - Deletion allowed
- Associated Release:** Released builds are available as choices in the work item 'Found In' field. [Create a release to associate with this build](#)

The bottom navigation bar includes tabs for Overview, Activities, Completion, JUnit, Logs, Downloads, External Links, and Properties.

Note: You can also open the build details from the build definition: In the **Builds** section of the *Squawk* project right-click the build definition *workshop.squawk.core.continuous.build* and select **Open Latest Build Details**.

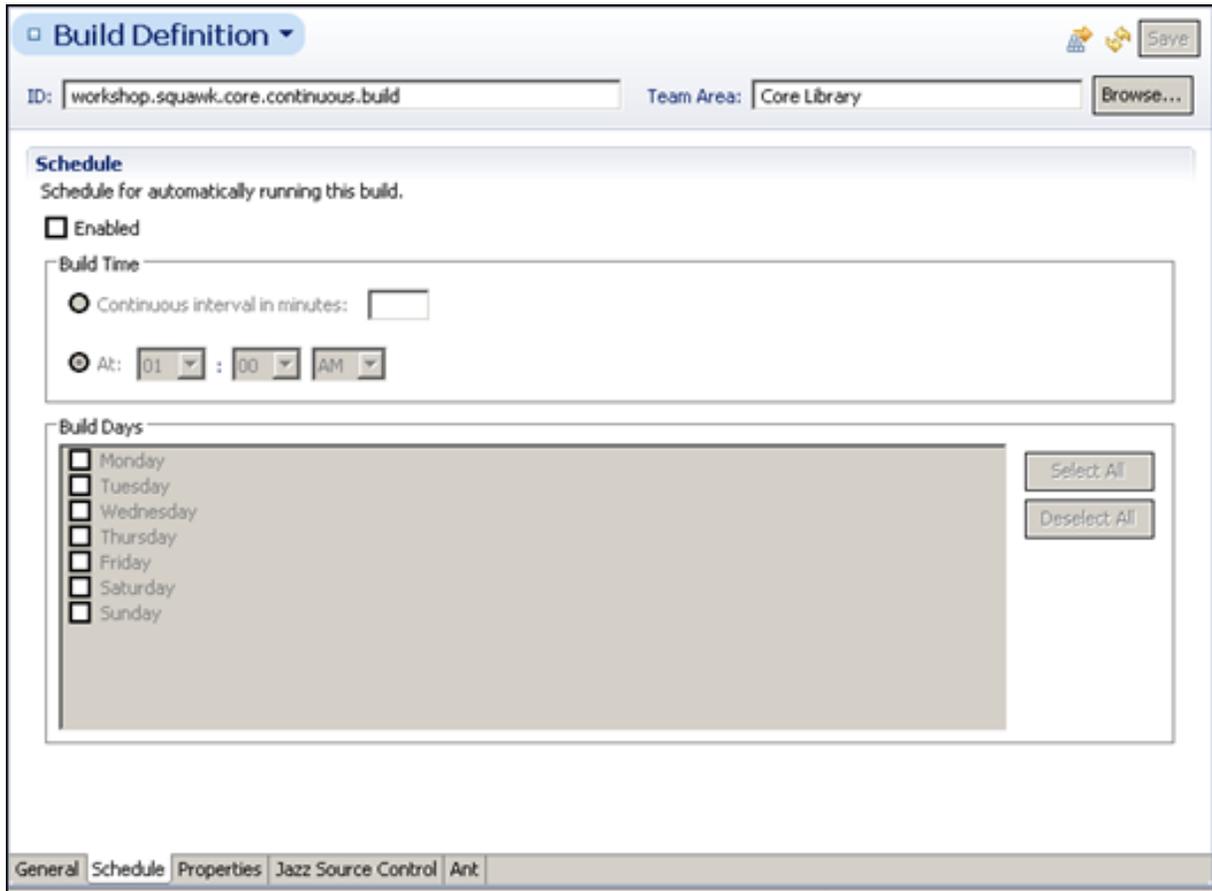
- \_\_\_6. Select the build definition *workshop.squawk.core.continuous.build* for the Squawk project then right-click and select **Open Build Definition** to open it.



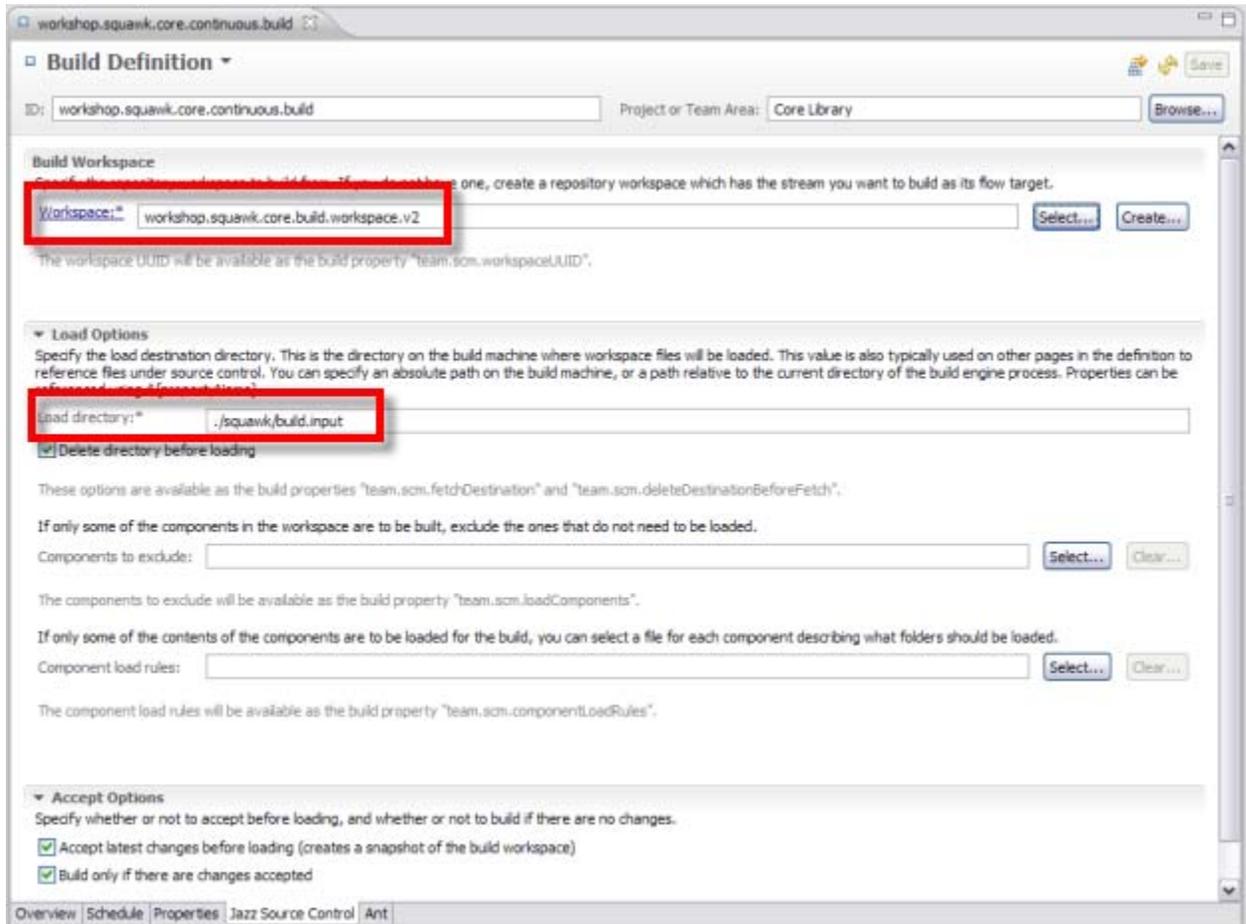
\_\_7. Review the information in the **Overview** tab.

The screenshot shows a web browser window with two tabs. The active tab is titled "workshop.squawk.core.continuous.build". The page content is titled "Build Definition" and includes a "Save" button. Below the title, there are input fields for "ID:" (containing "workshop.squawk.core.continuous.build") and "Project or Team Area:" (containing "Core Library"), with a "Browse..." button next to the latter. A "General Information" section follows, containing a "Description:" field with the text: "workshop.squawk.core.continuous.build", "Engine startup command:", and "jbe -repository https://localhost:9443/jazz -userId build -pass jazz -engineId workshop.build.engine -sleepTime 3". Below this is a checked checkbox for "Ignore warnings when computing overall status". The "Supporting Build Engines" section contains a list of engines with checkboxes: "junit" (unchecked), "RationalBuildForgeConnector" (unchecked), "RTC\_CC\_Connector\_Squawk\_Rel1.synchronizer" (unchecked), and "workshop.build.engine" (checked). To the right of this list are buttons for "Select All", "Deselect All", and "Create Engine". The "Pruning Policy" section has an unchecked checkbox for "Prune build results" and two input fields: "Successful build results to keep:" (value 10) and "Failed build results to keep:" (value 10). At the bottom, a navigation bar includes tabs for "Overview", "Schedule", "Properties", "Jazz Source Control", and "Ant".

- \_\_8. Select the **Schedule** tab and note how you can define the build to run on a specific frequency of your team's choice.



- \_\_9. Select the **Jazz Source Control** tab and note that the workspace name is defined here as well as the file system directory where the workspace content is loaded to after the workspace is updated with any changes in the stream.



---

## Appendix D: Lab 8 Endgame and a Tightened Process – Demo

In this lab exercise you will move to the *End Game* iteration of Milestone *M3* and experience some differences in the team's process for the end game. In the *M3 End Game* iteration the *Core Library* team has customized their process so that changes can be delivered only if their team lead has approved the work item associated with the delivery.



### Lab Scenario

As you approach your final milestone, the process for the iteration gets stricter. For example, you might insist that all tests must run to completion and without error before you are allowed to deliver any changes.

### D8.1 Review the process for End Game



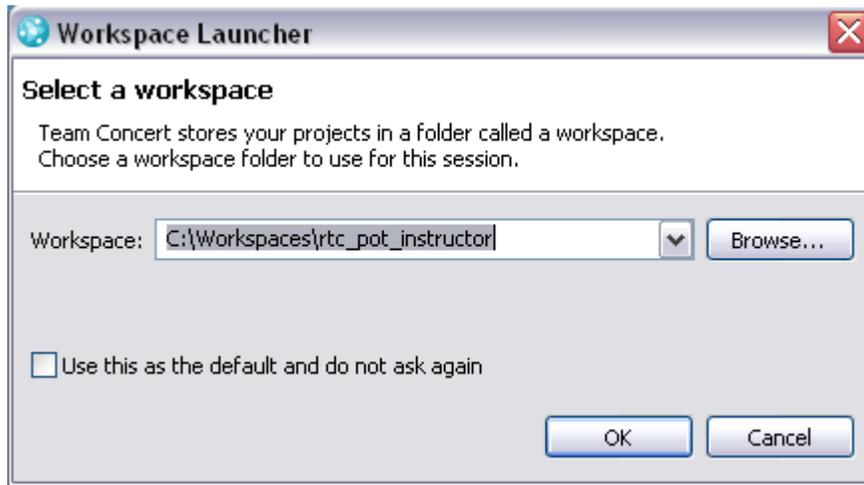
### Team role

The instructor will now perform the role of **Jerry**, the Team Lead.

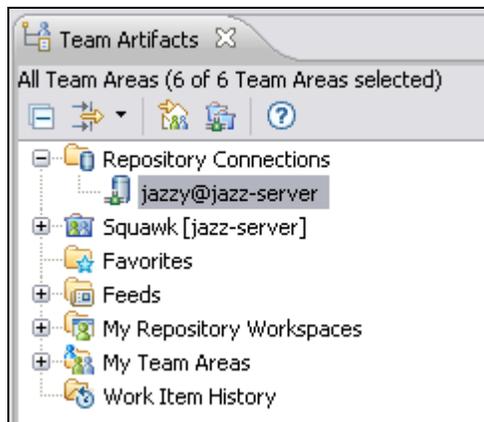
\_\_1. The instructor must use the `C:\Workspaces\rtc_pot_instructor` Eclipse workspace. If Rational Team Concert is not yet running, do the following:

\_\_a. Open Rational Team Concert by double clicking the Team Concert shortcut  on the Windows Desktop.

- \_\_b. In the Workspace **Launcher** dialog that is displayed, make sure to select the instructor's workspace: `C:\Workspaces\rtc_pot_instructor`. Click **OK**.

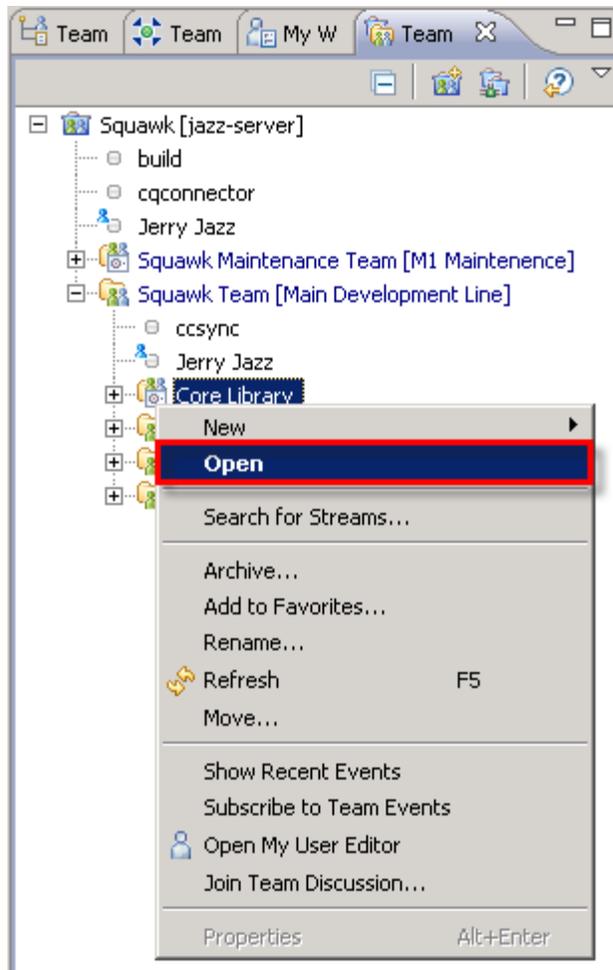


- \_\_c. In the **Team Artifacts** view (**Window** → **Show View** → **Team Artifacts**), ensure that you are connected to the **Squawk** Project area as *jazzy*.



\_\_2. Open the process editor for End Game iteration

\_\_a. Open the **Core Library** team area from the **Team Organization** view (**Window → Show View → Team Organization**) or from **My Team Areas** in the **Team Artifacts** view



- \_b. Click the **Process Customization** tab from the Team Area Editor for **Core Library**.

The screenshot shows the IBM Team Area Editor interface for the 'Core Library' team area. The 'Process Customization' tab is highlighted in red at the bottom of the window. The interface includes a 'Details' section with 'Summary' and 'Description' text boxes, a 'Members' section with a table of roles and a list of users, and a 'Timeline' section with a dropdown menu set to 'Main Development Line [inherited]'. A 'Process Description' section is also visible, stating that the team does not specify a custom process description.

**Members**

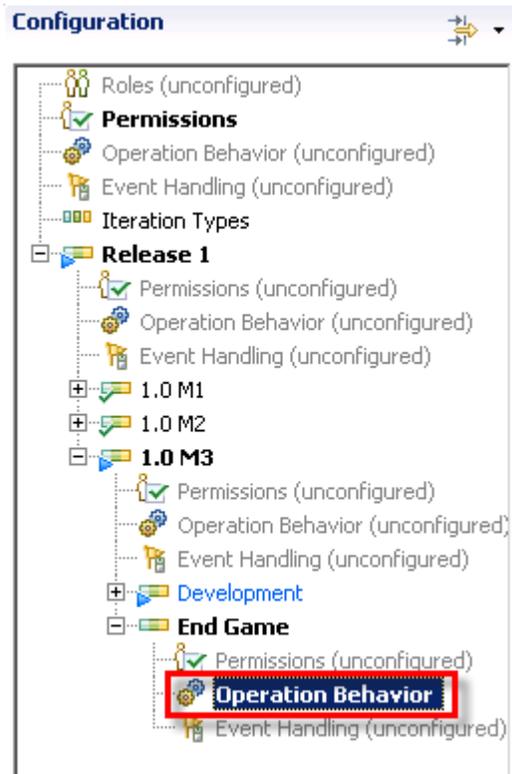
Roles determine a user's permissions as well as any preconditions and follow-up actions that are run for team operations. The roles assignments below are also valid in all child team areas. Unless configured otherwise, all users in the repository play the 'default' role.

Name	Process Roles
April Blues	contributor
Jerry Jazz	teamlead, contributor
student1	contributor
student10	contributor

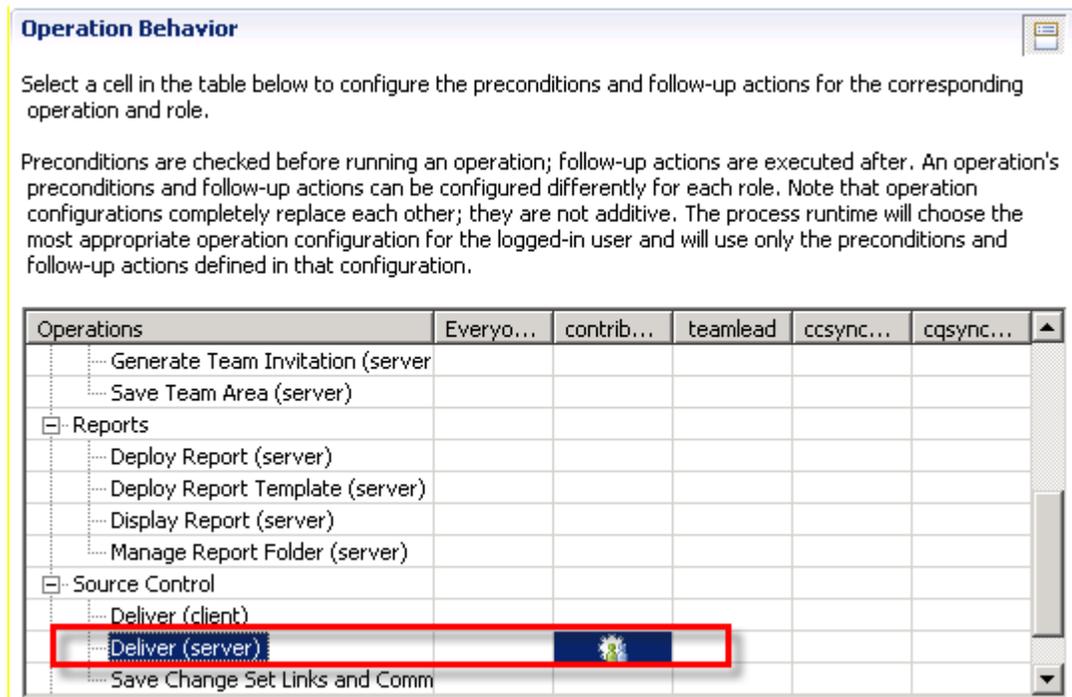
Buttons: Add..., Create..., Remove

Navigation: Overview | Links | **Process Customization** | Process Customization Source

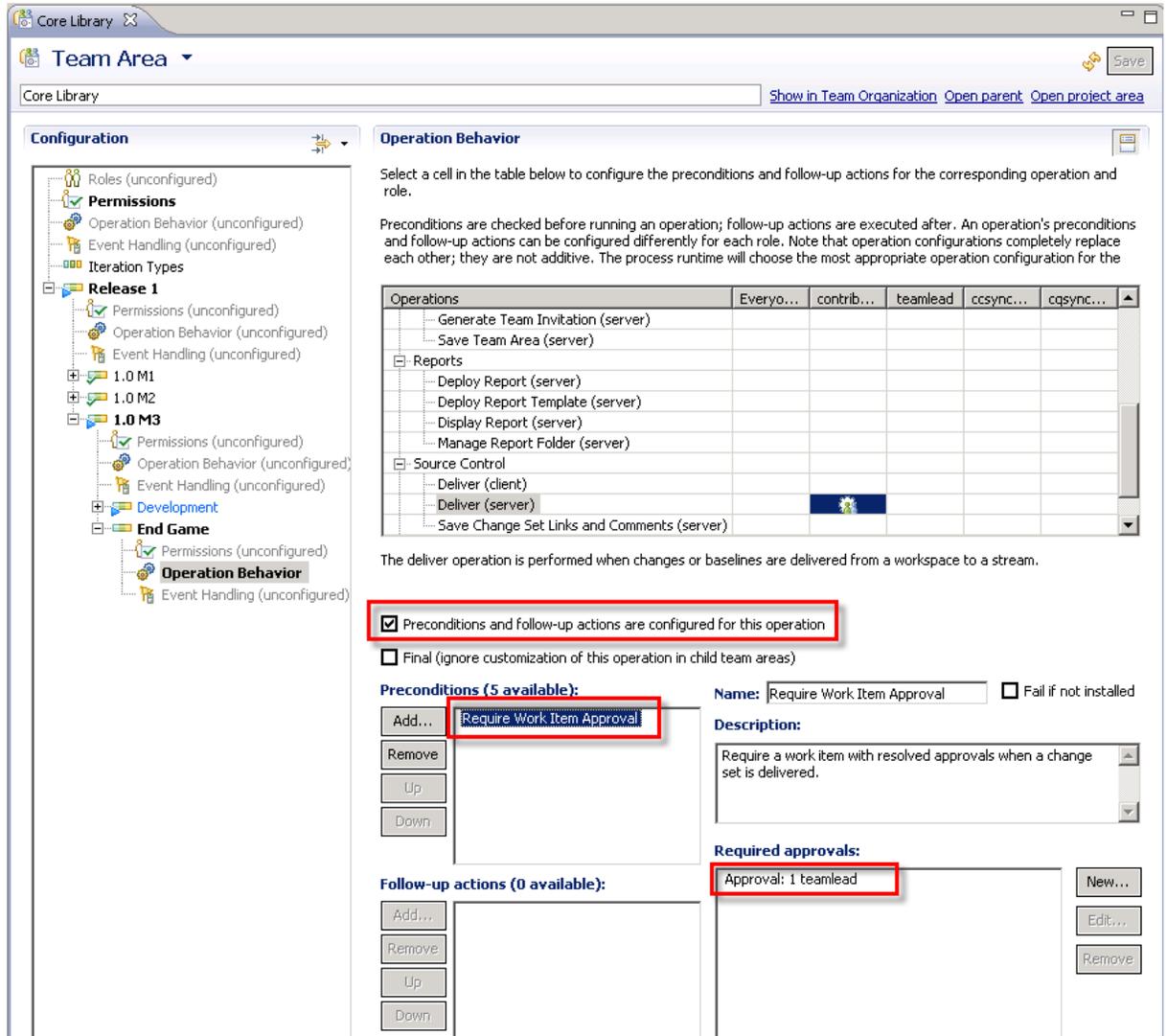
- c. Expand **Release 1** → **1.0 M3** → **End Game** and select **Operation Behavior**.



- d. In the **Operation Behavior** section scroll down and confirm that the **Contributor** role for **Source Control** → **Deliver (server)** has a precondition configured.



- e. Confirm that the precondition **Require Work Item Approval** has been added to the **Preconditions** list and that the **Required Approvals** section is set to **Approval: 1 teamlead**



This process restriction in the End Game iteration enforces a tighter restriction on code delivery. It ensures that the contributors cannot deliver code without approvals for the associated work item from their team leads.

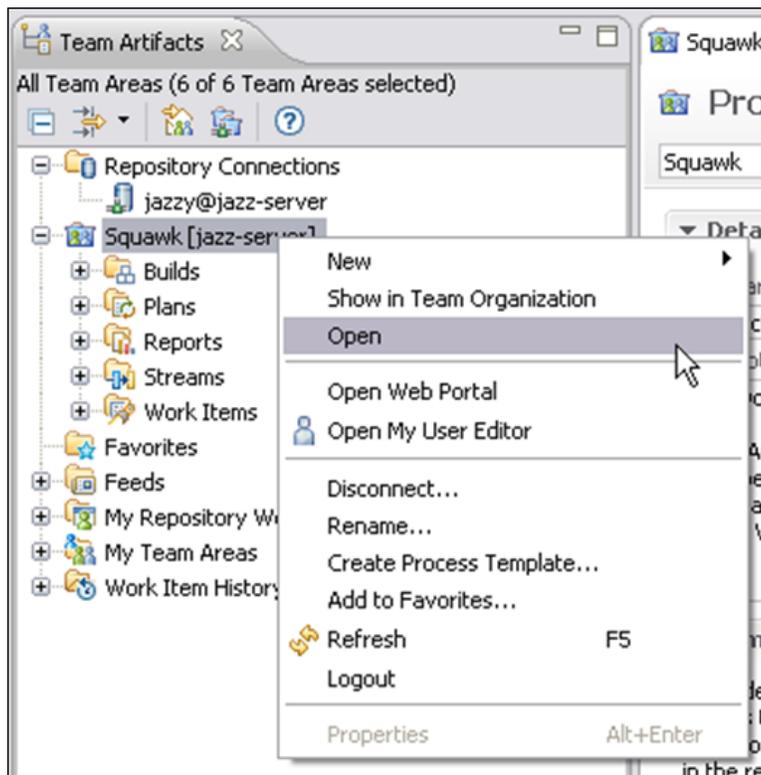
## D8.2 Change the process behavior



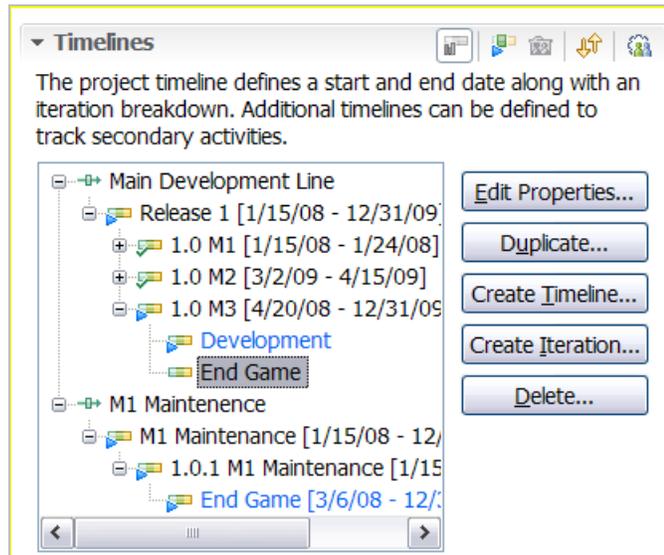
### Team role

The instructor will now perform the role of **Jerry**, the Team Lead.

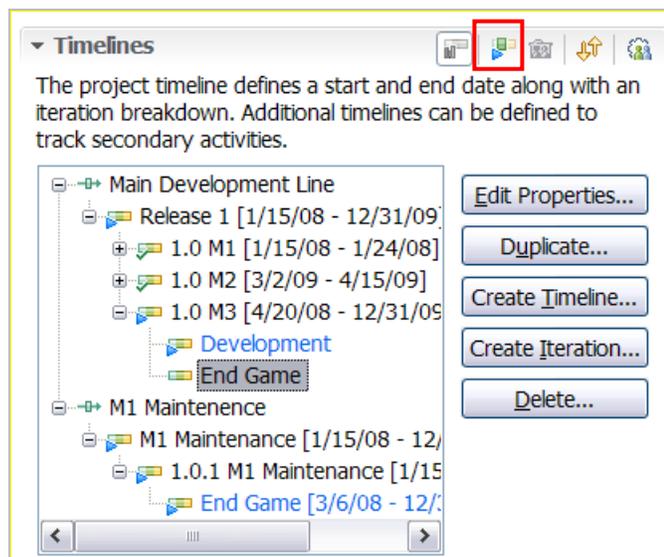
- \_\_1. Set the current iteration to the End Game
  - \_\_a. Right-click the **Squawk** project area and select **Open**.



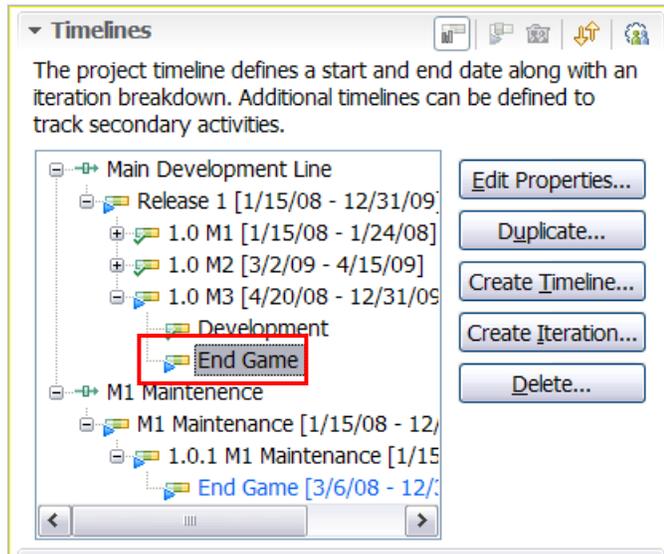
- \_\_b. In the **Timelines** section of the **Squawk** project area editor, expand the **Main Development Line** → **Release 1** → **1.0 M3** and select **End Game** iteration.



- \_\_c. Click the **Set the Selected Iteration as Current** toolbar button



- \_\_d. The updated iteration structure should look like the picture below.



- \_\_e. Click **Save**.
- \_\_f. Remind the students to close Team Concert before performing section 8.3 in order to ensure their workspaces are refreshed with the current process configuration. Depending on how fast they get to the delivery, it is possible that their workspace still hasn't picked up the process state change and therefore not require the approval on delivery.



**Important!**  
Section 8.3 is to be performed by the student. Refer to the corresponding section in the student portion of this workbook.

## D8.4 Approve the End Game Work Items



**Team role**  
The instructor will now perform the role of **Jerry**, the Team Lead.

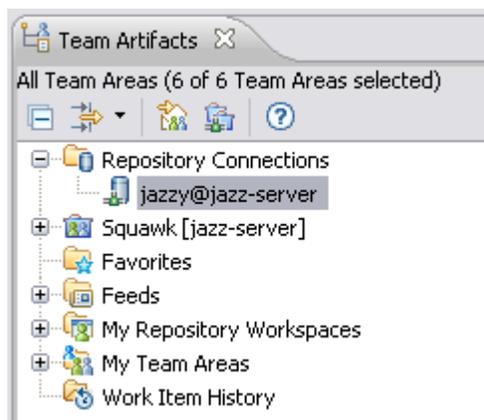
- \_\_1. The instructor must use the `C:\Workspaces\rtc_pot_instructor` Eclipse workspace. If Rational Team Concert is not yet running, do the following:

- \_\_a. Open Rational Team Concert by double clicking the Team Concert shortcut  on the Windows Desktop.
- \_\_b. In the Workspace **Launcher** dialog that is displayed, make sure to select the instructor's workspace: C:\Workspaces\rtc\_pot\_instructor.



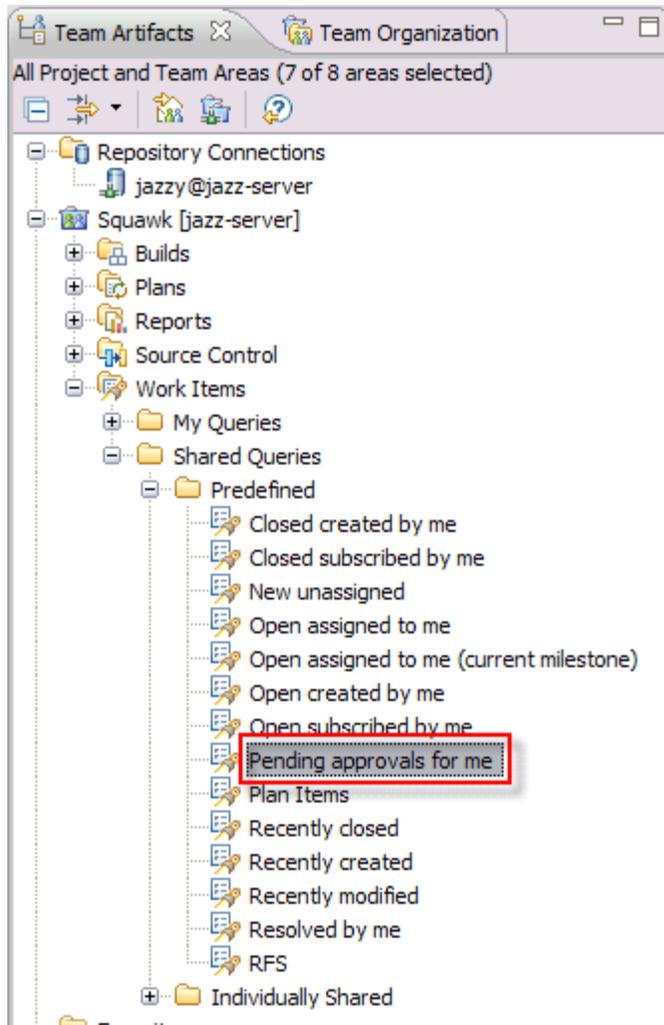
Click **OK**.

- \_\_c. In the **Team Artifacts** view (**Window** → **Show View** → **Team Artifacts**), ensure that you are connected to the **Squawk** Project area as *jazzy*.



\_\_2. Find Work Items awaiting approval

- \_\_a. Expand the **Squawk** Project Area, then expand **Work items** → **Shared Queries** → **Predefined** and double-click the **Pending approvals for me** query.



- \_\_b. In the Work Items results view double click the first work item to open it.
- \_\_c. Optionally, select the **Links** tab, open the change set associated with the work item, then go to the change explorer and double-click the changed source file which will open the compare window. This demonstrates how easily a team lead can check the changes for a developer before approving the work item.

- \_\_d. On the open work item, select the **Approvals** tab, change the Approval State to **Approved** and click **Save**.

The screenshot displays the 'Approvals' section for work item 494. The table below shows the current approval state:

Approver	State	Due
Approval for End Game change to my student1 squawker Jerry Jazz	Approved	

The dropdown menu for the 'State' column is open, showing the following options:

- Approved
- Rejected
- Pending

The 'Save' button is highlighted in the top right corner. Below the 'Approvals' section, the 'Work Items' section shows a table with one row for work item 494, status 'In Progress', and summary 'End Game change to my student1 squawker'.

I..	Status	P	S	Summary	Owned By
494	In Progr...			End Game change to my student1 squawker	student1

- \_\_e. Repeat the above steps for each of the Work Items with pending approvals.



**Important!**  
Section 8.5 is to be performed by the student. Refer to the corresponding section in the student portion of this workbook.

---

## Appendix F: Lab 10 Integrating with Other SCM Systems – Demo

### Important!

All steps in this section are to be performed by the instructor as a demo.

This is an **optional** module; if the ClearCase® Connector is not a relevant feature for the audience, the instructor can skip this Lab.

If it is a relevant feature, the instructor can either:

Demo all sections of this document



- **(Recommended)** In the interest of time, demo only sections 10.4 and 10.6 to show the synchronization results. The instructor should then perform the steps in the other sections of this script while students work on Labs 6 and 7.
  - While students work on Lab 6, instructor performs steps in sections 10.1 and 10.3.
  - Instructor demos 10.4 steps, before proceeding to Module 7 slides.
  - While students work on Lab 7, instructor performs steps in section 10.5.
  - Instructor demos 10.6 steps, before proceeding to Module 8 slides.
  - Skip section 10.2.

In this lab the instructor will demonstrate how to synchronize code changes between Rational Team Concert and IBM Rational ClearCase. If your company is already using Rational ClearCase, this lab will demonstrate how you can take advantage of these two different tools working together.



### Lab Scenario

The Squawk Application decomposes into five components: Core, Core Tests, UI, UI Tests and Documentation.

In Lab 4, the students created their own Squawker Class (Student<N>.java) and modified files in the Core and Documentation components. Let's assume that these two components are maintained in Rational Team Concert. The instructor will demonstrate how to synchronize (export) these changes from Rational Team Concert to Rational ClearCase.

In ClearCase, the instructor will modify files in the UI component and demonstrate how to synchronize (import) changes from Rational ClearCase to Rational Team Concert.

And yes, it is possible modify the same component in Rational Team Concert and Rational ClearCase. At synchronization time, manual merge might be required. Both Rational Team Concert and Rational ClearCase provide merging tools to assist on resolving merge conflicts.

Playing the role of the Team Lead, the instructor will demonstrate how to monitor changes performed in Rational Team Concert and/or Rational ClearCase and decide when it is time to integrate the code changes.

Before starting the demo, the instructor should highlight some of the key steps required to configure the synchronization engine.

### Definition of “ClearCase Synchronized Streams”

ClearCase Synchronized Streams enable the capabilities of Rational Team Concert to work on Eclipse projects that are under Rational ClearCase source control. ClearCase Synchronized Streams use the Jazz build engine to manage the synchronization.

ClearCase Synchronized Streams are not designed to import every version of an artifact from Rational ClearCase to Rational Team Concert. Rather, they provide support for importing the versions selected by a Rational ClearCase view configuration (a UCM stream or a branch and label pair) to a Jazz stream and then exporting change sets from that stream to the Rational ClearCase view. This stream-based approach takes advantage of similarities between Rational ClearCase and Rational Team Concert to facilitate ongoing work in both environments.

### Definition of “Merge Workspace”

A merge workspace is an ordinary repository workspace into which the ClearCase Synchronized Streams is loaded.



### Team role

The **instructor** will now performing the role of Jerry, the Team Lead.

---- NOT REQUIRED TO DEMO ----

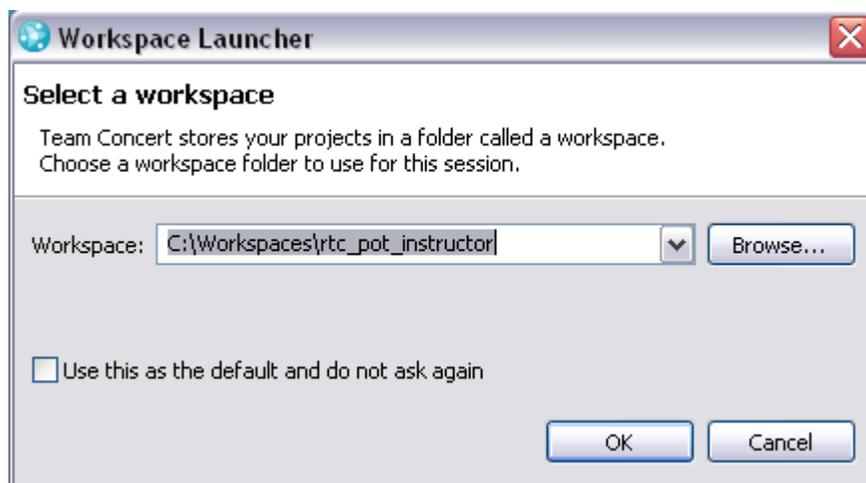
---- BUT MUST BE PERFORMED BY INSTRUCTOR AS SETUP STEP ----

## F10.1 Preparing to Synchronize Team Concert and ClearCase repositories

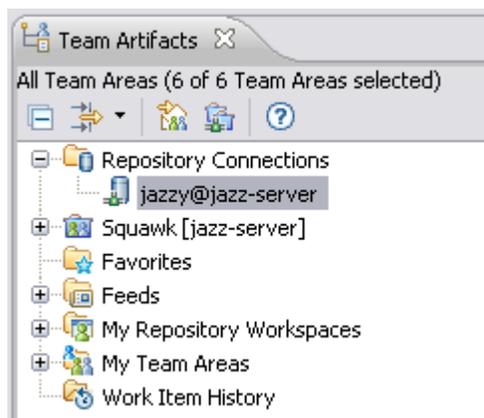
The instructor will use Jerry's **Core Library Repository Workspace** to deliver to the ClearCase Synchronized stream.

\_\_1. The instructor must use the `C:\Workspaces\rtc_pot_instructor` eclipse workspace. If Rational Team Concert is not yet running, do the following:

- \_\_a. Open Rational Team Concert by double-clicking the Team Concert shortcut  on the Windows Desktop.
- \_\_b. In the Workspace **Launcher** dialog that is displayed, make sure to select the instructor's workspace: `C:\Workspaces\rtc_pot_instructor`. Click **OK**.



\_\_c. In the **Team Artifacts** view (**Window → Show View → Team → Team Artifacts**), ensure that you are connected to the **Squawk** Project area as *jazzy*.



- \_\_2. Verify that Jerry's repository workspaces delivered/accepted all changes to/from the Core Library and Documentation Streams.

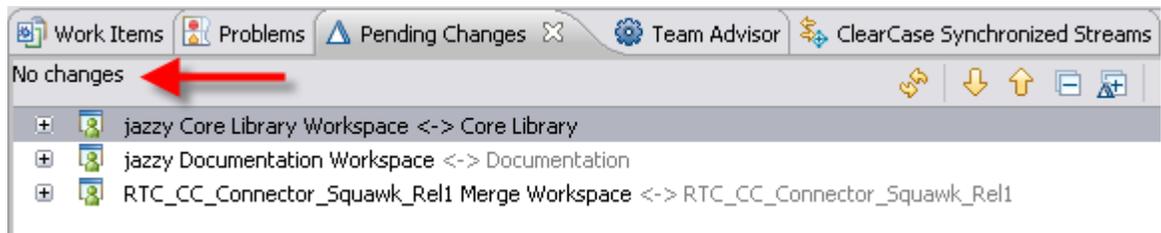
**Important!**

This demo requires that your **Core Library** and **Documentation** repository workspaces are up-to-date, with no incoming or outgoing changes remaining in the **Pending changes** view.

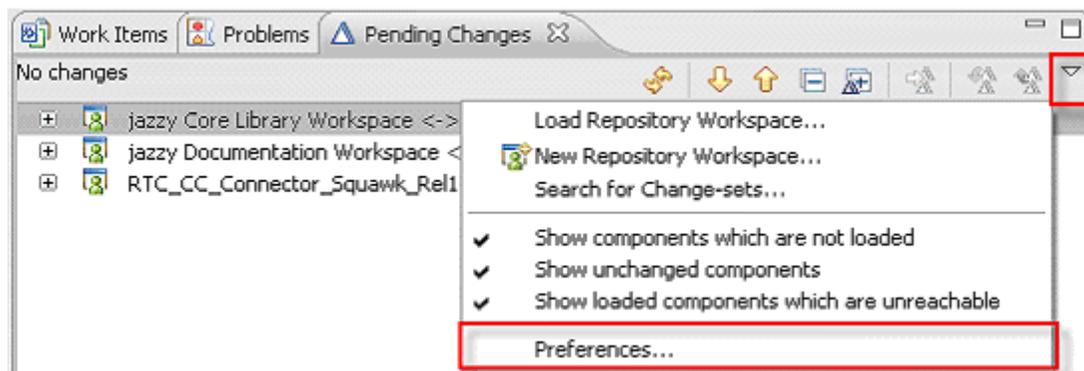
If there are any incoming change sets in the **Pending Changes** view, right-click those change sets and accept them.

Similarly, if there are any outgoing changes sets in the **Pending Changes** view, right-click and deliver them.

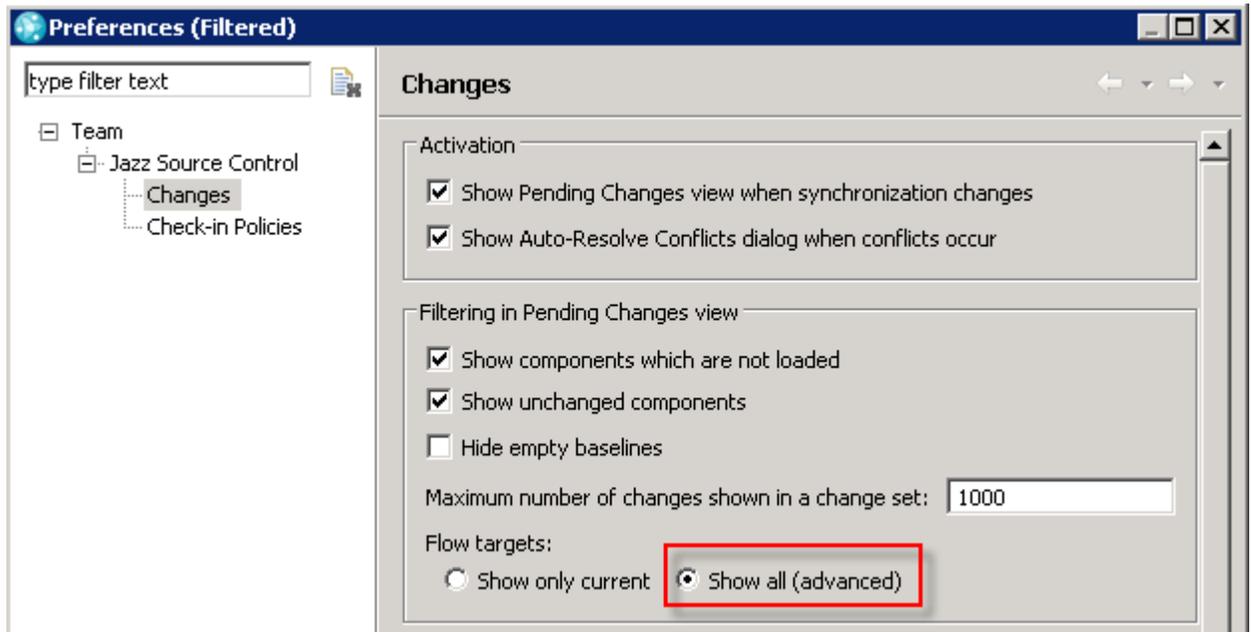
- \_\_a. From the **Pending Changes** view (**Window** → **Show View** → **Other** → **Jazz Source Control** → **Pending Changes**), accept all incoming changes and deliver any outgoing changes. Proceed to next step once "No changes" are pending.



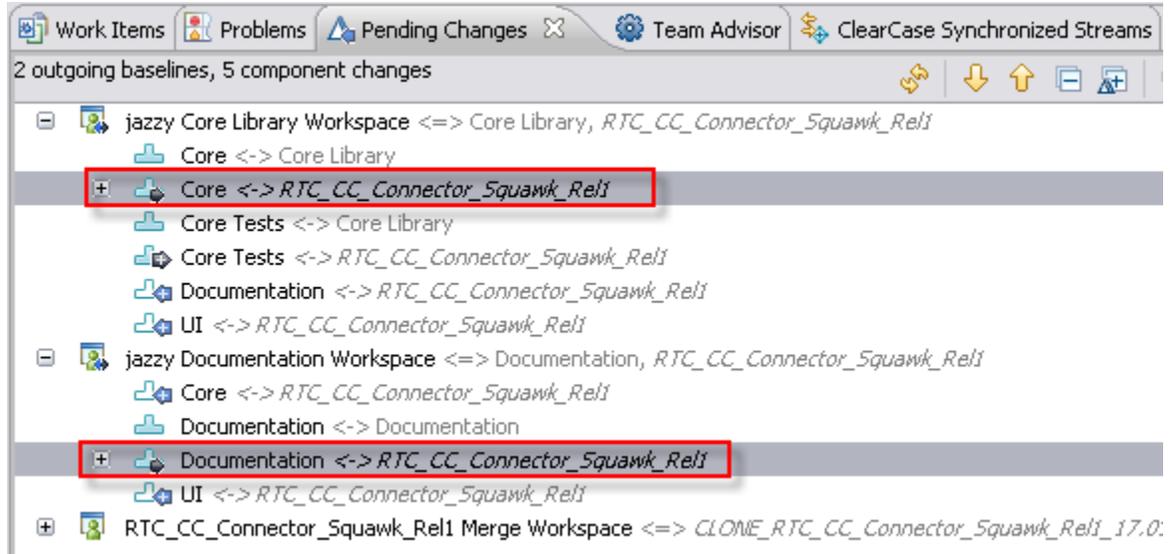
The **Pending Changes** view by default only displays the default flow target for each one of the Repository Workspaces (e.g. jazzy **Core Library Workspace** Repository Workspace flows to/from **Core Library** Stream). Click the menu icon (  ) to modify the **Pending Changes** view **Preferences** to display all flows.



- \_\_b. In the Preferences dialog, select **Show all (advanced)** Flow Targets option. Click **OK**.



- \_\_3. Verify that Jerry's **Core Library** and **Documentation Repository Workspaces** have an alternate flow defined to the ClearCase Synchronized Stream (RTC\_CC\_Connector\_Squawk\_Rel1).



---- NOT REQUIRED TO DEMO ----

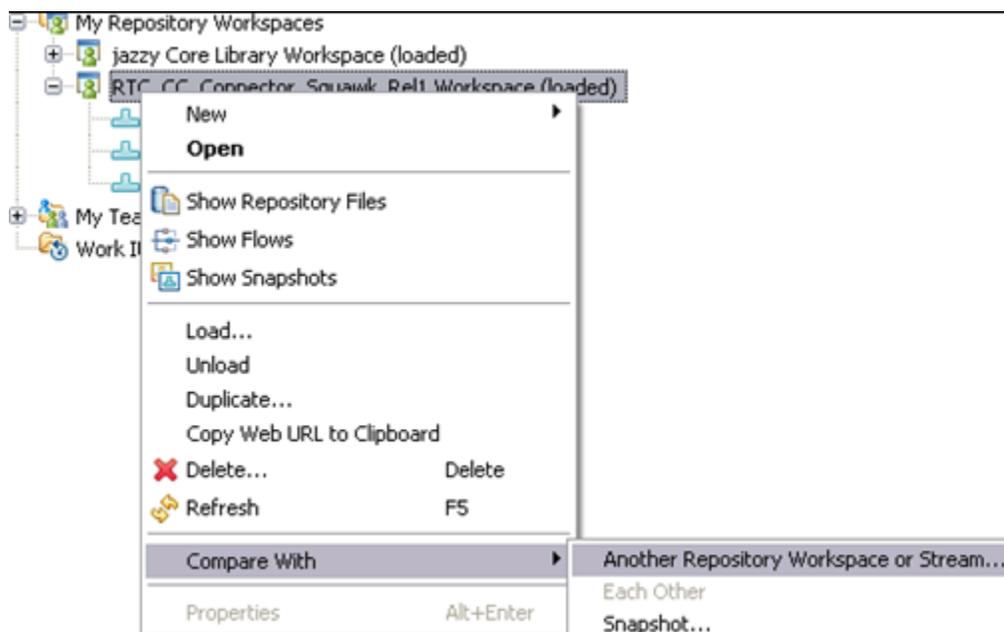
---- INSTRUCTOR CAN SKIP THIS SECTION ----

## F10.2 Browsing and Comparing Baselines

Prior to starting the synchronization process, the instructor, as the Team Lead, will query for changes since last synchronization to better understand the impact of delivering changes at this time.

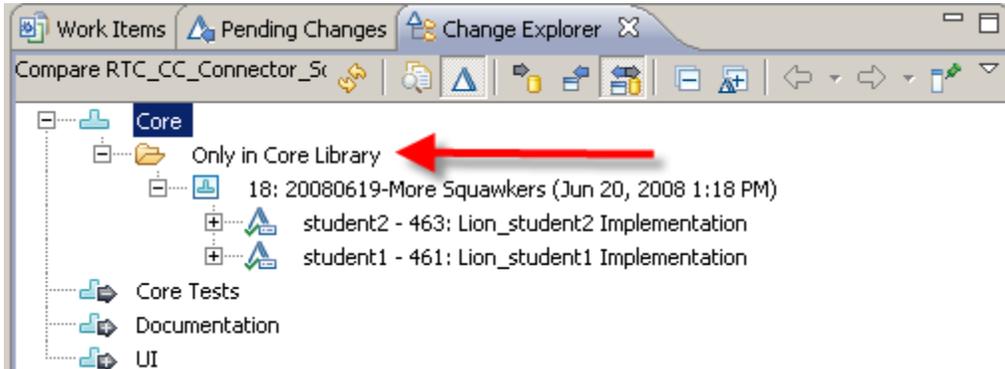
The ClearCase UCM Components were initially loaded with the same Team Concert code base that you started with in Lab 1; Rational ClearCase and Rational Team Concert were not synchronized since. In Lab 4, students created their own Squawker classes using their private Repository Workspace; at the end of the module all students delivered their final code changes to the Core Library Stream. There should be a new Squawker class in the Core Library Stream for each student in this room ready to be delivered to ClearCase. Let's review the code changes in the Core Library Stream and compare with what is currently in the ClearCase Merge Repository Workspace.

- \_\_\_1. In the **Team Artifacts** view (**Window → Show View → Other → Team → Team Artifacts**), expand **My Repository Workspaces**, to show the ClearCase Merge Repository Workspace: **RTC\_CC\_Connector\_Squawk\_Rel1 Merge Workspace**
- \_\_\_2. Right-click the ClearCase Merge Repository Workspace (**RTC\_CC\_Connector\_Squawk\_Rel1 Merge Workspace**) and select **Compare With → Another Repository Workspace or Stream**.



- \_\_\_a. Select the **Core Library** stream and click **OK**.

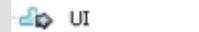
- \_\_\_3. The **Change Explorer** view opens and displays the comparison results. Expand the **Core** component to verify that new Squawkers were found only in the Core Library stream and are not yet in the ClearCase Merge Repository Workspace.



**What the icons next to the other component names mean**



 **Core Tests** The **Core Tests** component is one of the components of the Core Library Stream; however is not a component added to the ClearCase Repository Workspace.

 **Documentation**  
 **UI** The **Documentation and UI** Components are components added to the ClearCase Repository Workspace; however these two components are not in the Core Library Stream.



By default, the Pending Changes view checks for stream changes every 10 minutes, so there can be up to a 10 minute delay before changes to the synchronized stream become visible as incoming change sets for the ClearCase Workspace. You can click the  icon in the Pending Changes view toolbar to force an immediate check for new incoming and outgoing change sets.

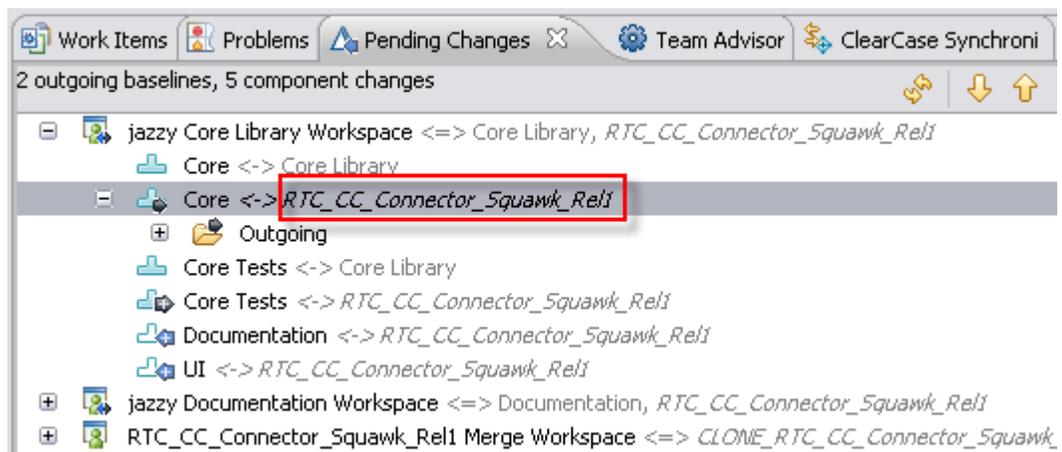
---- NOT REQUIRED TO DEMO ----

---- BUT MUST BE PERFORMED BY INSTRUCTOR AS SETUP STEP ----

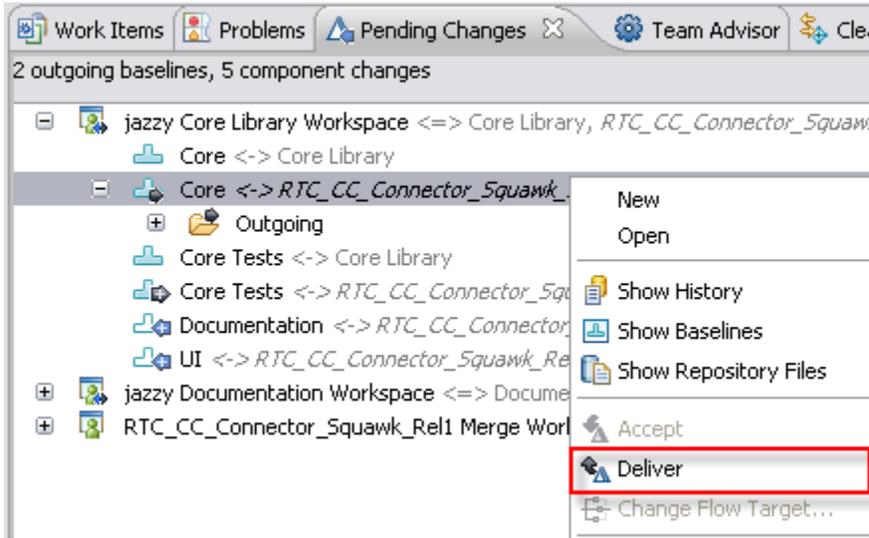
### F10.3 Synchronize (export) Team Concert code changes to ClearCase.

In this demo, the instructor will be exporting the latest baseline available for the **Core** component in the **Core Library** stream to ClearCase.

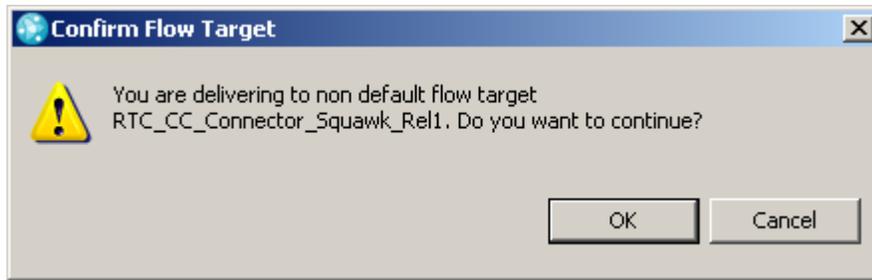
- \_\_1. In the Pending Changes view (**Window → Show View → Other → Jazz Source Control → Pending Changes**), expand **jazzy Core Library Workspace**.
  - \_\_a. There are two target flows defined for the Core component in the context of this repository workspace.
  - \_\_b. Select the **Core** component that flows to the ClearCase Synchronized Stream (**RTC\_CC\_Connector\_Squawk\_Rel1**). It should be the only one with Outgoing changes.



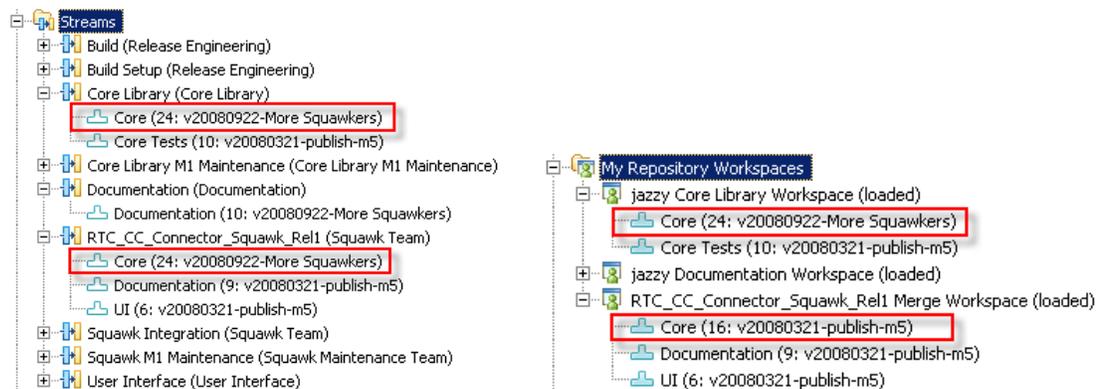
- 2. Right-click the **Core** component that flows to the ClearCase Synchronized Stream (**RTC\_CC\_Connector\_Squawk\_Rel1**) and select **Deliver**.



- a. Acknowledge the warning message and confirm that you want to proceed. Click **OK**.



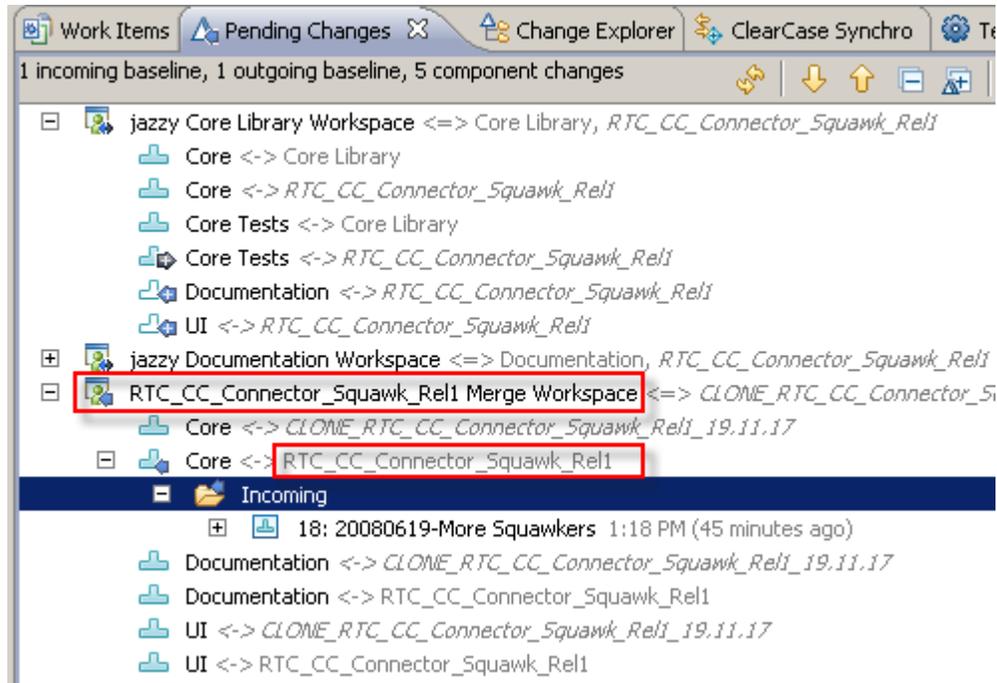
- b. At the end of the deliver operation, the **Core Library** Stream and ClearCase Synchronized Stream (**RTC\_CC\_Connector\_Squawer\_Rel1**) have a matching configuration. However the **jazzy Core Library Workspace** and **RTC\_CC\_Connector\_Squawer\_Rel1 Merge Workspace** are still selecting different Baselines.





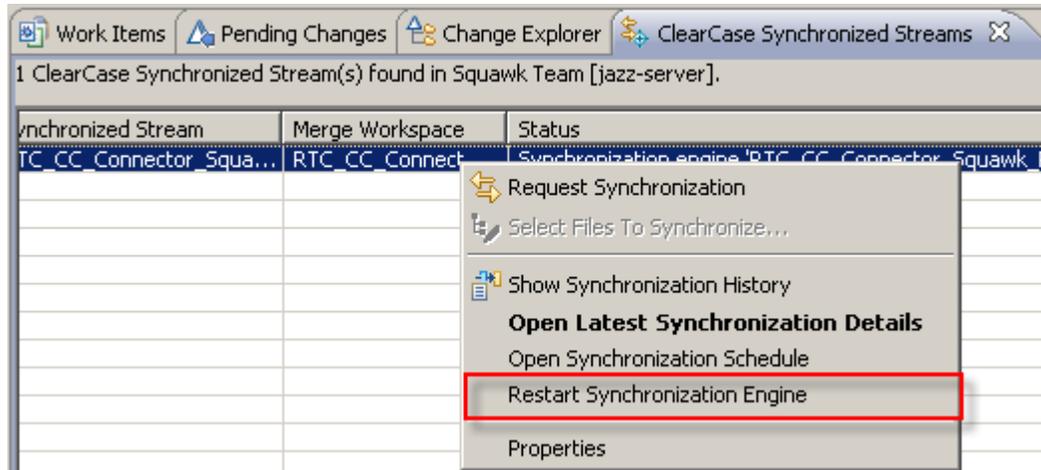
You can optionally deliver changes for the Documentation component. If you choose to include the Documentation component in the synchronization, repeat all steps in this section using the **jazzy Documentation Workspace**.

- \_\_\_c. In the **Pending Changes** view, the **RTC\_CC\_Connector\_Squawer\_Rel1 Merge Workspace** shows that there are incoming changes from the ClearCase Synchronized Stream. Right-click the **Incoming** folder and select **Accept**.



- \_\_\_d. After the accept operation is complete, the **RTC\_CC\_Connector\_Squawer\_Rel1** ClearCase Synchronized Stream and corresponding Merge Workspace are now configured with the same set of baselines. The changes to the **RTC\_CC\_Connector\_Squawer\_Rel1** ClearCase Synchronized Stream trigger the synchronization with ClearCase.

- \_\_e. This is the first time that the Team Concert and ClearCase synchronization is being performed. The synchronization process is not running. In the **ClearCase Synchronized Streams** view (**Window > Show view > Other > Jazz source control → ClearCase Synchronized Streams**), right-click the **RTC\_CC\_Connector\_Squawer\_Rel1** stream and select **Restart Synchronization Engine**.



- \_\_i. Optionally, verify that a DOS-Prompt window was initiated and the Synchronization Engine is waiting for new requests.

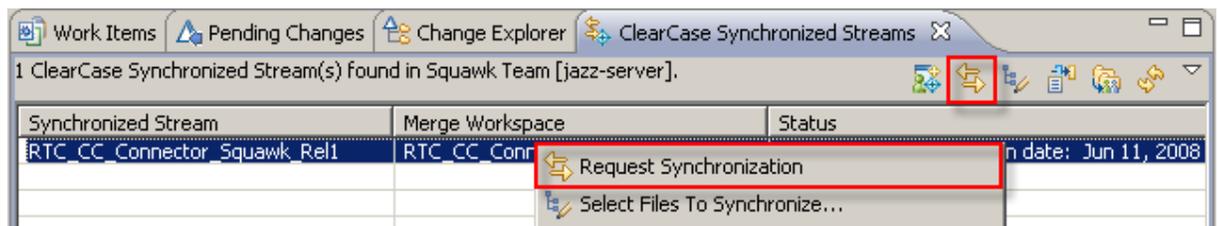
```

C:\> synchronization engine for stream RTC_CC_Connector_Squawk_Rel1
2008-06-20 13:57:14 [Jazz build engine] Running build loop...
2008-06-20 13:57:14 [Jazz build engine] Waiting for request...
    
```

If the Synchronization Engine for the ClearCase Synchronized Stream is not running the Status column in the ClearCase Synchronized Streams view displays the following message.

Status
Synchronization engine 'RTC_CC_Connector_Squawk_Rel1.synchronizer' is not active, may need to be...

- \_\_f. In the **ClearCase Synchronized Streams** view, to initiate the synchronization, right-click the **RTC\_CC\_Connector\_Squawer\_Rel1** stream and select **Request Synchronization**.



- \_\_i. The synchronization build engine discovers the request, which can take up to a minute.

```

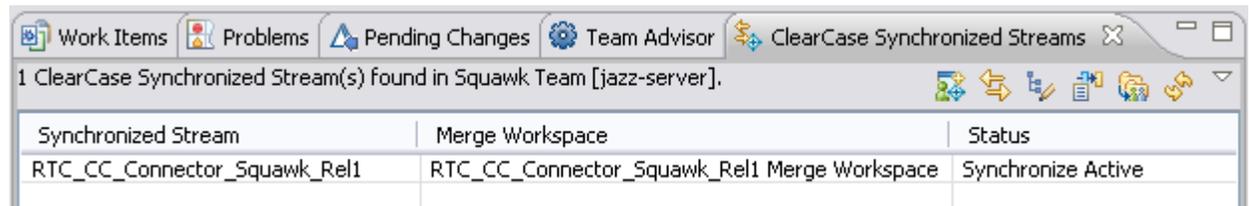
c:\ synchronization engine for stream RTC_CC_Connector_Squawk_Rel1
2008-06-20 13:57:14 [Jazz build engine] Running build loop...
2008-06-20 13:57:14 [Jazz build engine] Waiting for request...
2008-06-20 13:57:21 [Jazz build engine] Found a user request for build definition
n "Synchronize RTC_CC_Connector_Squawk_Rel1".
  
```

- \_\_g. You can use the Refresh icon  to monitor the synchronization progress. The Status column indicates whether the operation succeeded and whether any merges are required.



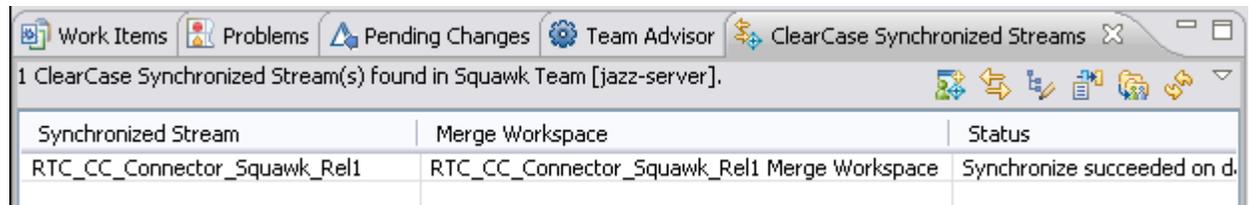
1 ClearCase Synchronized Stream(s) found in Squawk Team [jazz-server].

Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize Pending



1 ClearCase Synchronized Stream(s) found in Squawk Team [jazz-server].

Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize Active



1 ClearCase Synchronized Stream(s) found in Squawk Team [jazz-server].

Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize succeeded on d.

Status can be any of:

**Initialized successfully**

The stream has been created, but not yet synchronized with Rational ClearCase.

**Synchronize pending**

A synchronization request is been made but has not yet started.

**Synchronize active**



A synchronization request has been accepted by the synchronization process, and the synchronization process is underway.

**Synchronize succeeded**

The most recent synchronization was successful.

**Merge required**

The same file or folder has been changed in both Jazz and Rational ClearCase.

**Synchronize failed**

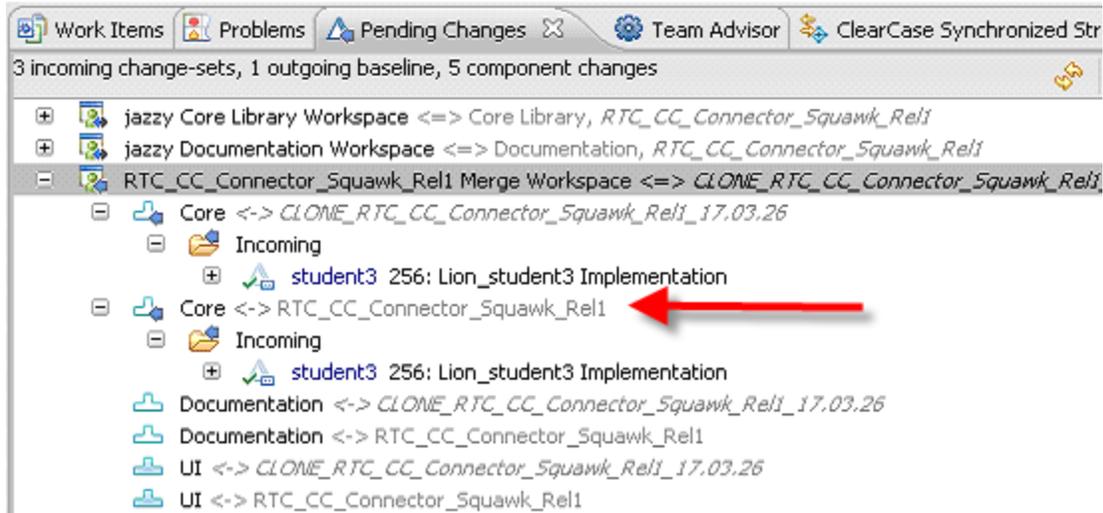
The most recent synchronization failed. Synchronization log available for review.

- \_\_h. Core Component synchronization is now complete. New Squawkers added to the Core Component in Team Concert were also added to the corresponding UCM Core Component in ClearCase.



If you included the Documentation component in the synchronization, this component is also up to date in ClearCase.

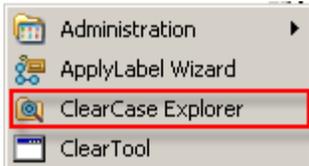
- \_\_3. Changes delivered to the ClearCase Synchronized Stream (**RTC\_CC\_Connector\_Squawk\_Rel1**) are now displayed as incoming changes to the Merge Repository Workspace. **Accept** all incoming changes from the ClearCase Synchronized Stream (**RTC\_CC\_Connector\_Squawk\_Rel1**).



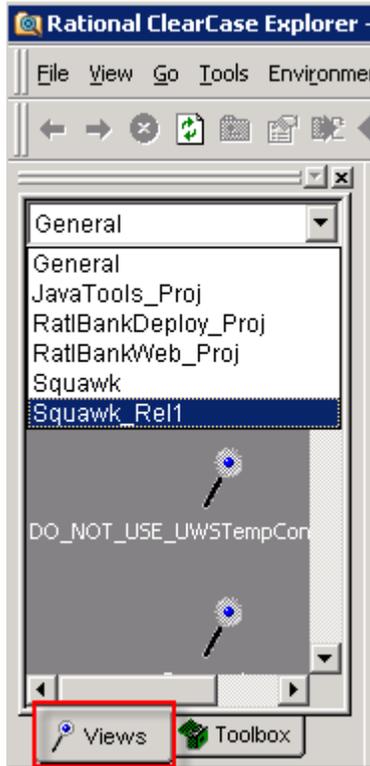
If synchronizing the Documentation component, accept the incoming changes for this component that flows from the Merge Repository Workspace (RTC\_CC\_Connector\_Squawk\_Rel1)

---- DEMO THE FOLLOWING STEPS ----

## F10.4 Show synchronization results (from Team Concert to ClearCase)

- \_\_1. Inspect Team Concert Changes in ClearCase
- \_\_a. **Select Tools > IBM Rational > IBM Rational ClearCase > ClearCase Explorer** from the Windows taskbar.
- 
- \_\_b. In ClearCase Explorer, select **View → Refresh View Shortcuts** menu options.

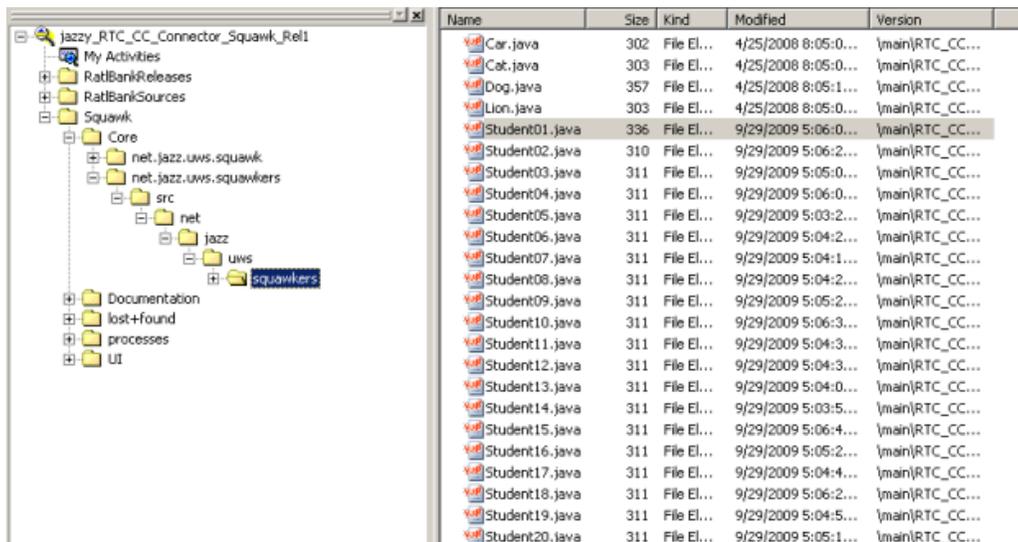
- c. Select the **Views** Tab. In this tab, select the **Squawk\_Rel1** Project page.



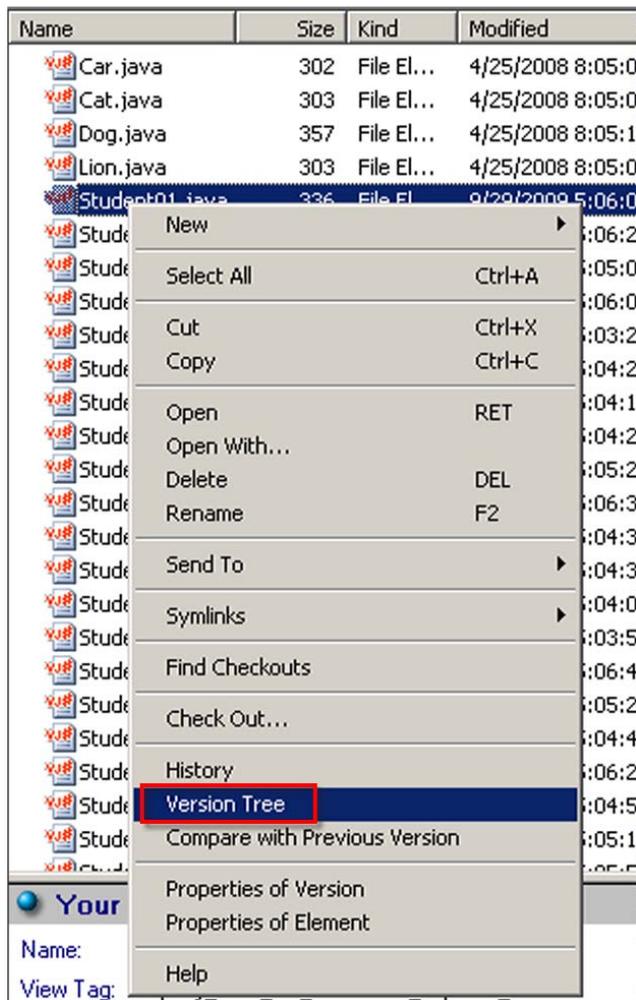
- d. Select the **jazyz\_RTC\_CC\_Connector\_Squawk\_Rel1** view shortcut.



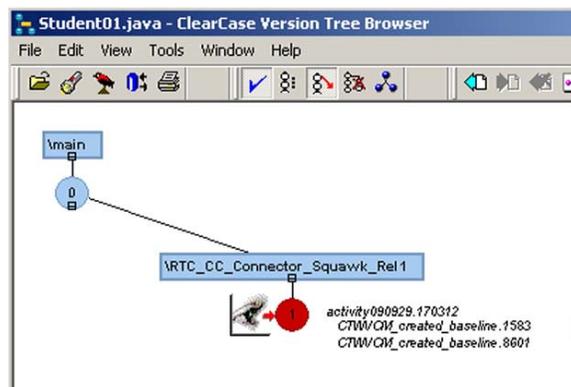
- e. Expand folders to locate the **Student<N>** class created in Team Concert.



- \_\_f. Optionally, right-click the **Student<N>.java** file and select **Version Tree** menu option.



- \_\_g. Verify that the **Student<N>.java** file was added to source control in the **RTC\_CC\_Connector\_Squawk\_Rel1** UCM stream.



- \_\_h. Close the ClearCase Version Tree browser.
- \_\_i. Exit ClearCase Explorer.

**---- NOT REQUIRED TO DEMO ----**

**---- BUT MUST BE PERFORMED BY INSTRUCTOR AS SETUP STEP ----**

## **F10.5 Synchronize (import) ClearCase source control file changes to Team Concert**

- \_\_1. Run script `C:\ClearCase_Storage\script\rtc_cc_changes.bat` to perform the following ClearCase command operations.



```
cleartool mkview -tag alex_Squawk_Rell_Integration -stream
Squawk_Rell_Integration@\Squawk -stgloc -auto

cleartool startview alex_Squawk_Rell_Integration

M:

cd alex_Squawk_Rell_Integration

cleartool mount \Squawk

cd Squawk\UI\net.jazz.uws.squawk.ui\src\net\jazz\uws\squawk\ui

cleartool ls MessageSquawker.java

cleartool mkact -nc ClearCase_Change

cleartool setact ClearCase_Change

cleartool co -nc MessageSquawker.java

Echo /* ClearCase Change - %Date% */ > temp.txt

Type MessageSquawker.java >> temp.txt

Ren MessageSquawker.java MessageSquawker.bak

Move temp.txt MessageSquawker.java > Nul

cleartool ci -nc MessageSquawker.java

cleartool setact -none

cleartool chproj -blname_template basename Squawk_Rell@\Squawk

cleartool mkbl -nc -view alex_Squawk_Rell_Integration
Squawk_Rell_v20080418

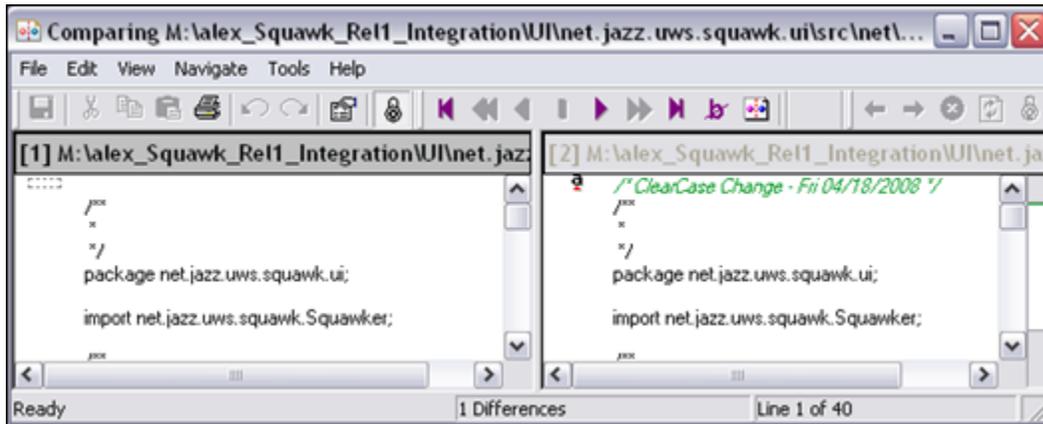
cleartool chstream -recommended -default
Squawk_Rell_Integration@\Squawk

cd \

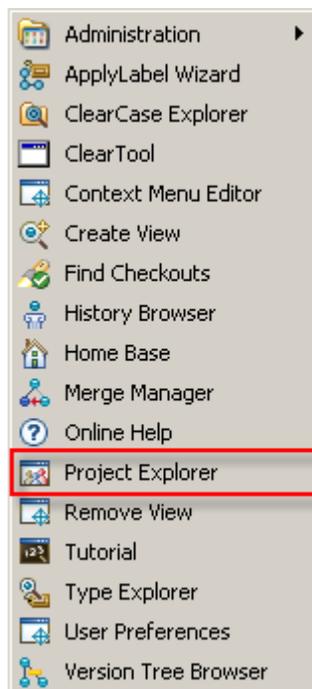
cleartool endview alex_Squawk_Rell_Integration

cleartool rmview -tag alex_Squawk_Rell_Integration
```

- \_\_\_2. The script above added the line highlighted in green to the **MessageSquawker.java** file. The MessageSquawker.java file is in one of the files of the UI component.

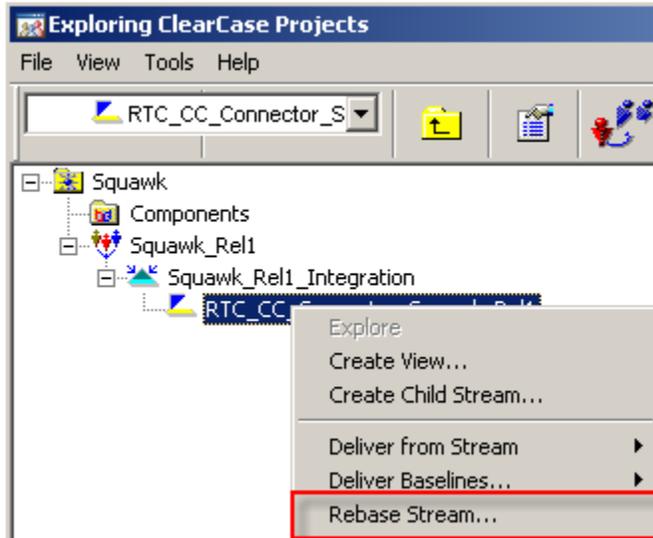


- \_\_\_3. In ClearCase, whenever you rebase the **RTC\_CC\_Connector\_Squawk\_Rel1** UCM stream, the next time that the Team Concert synchronization build engine runs, changes from ClearCase are propagated to the **RTC\_CC\_Connector\_Squawer\_Rel1** Synchronized Stream and corresponding Merge Workspace in Team Concert.
- \_\_\_4. If **ClearCase Project Explorer** is not already running, click **Tools → IBM Rational → IBM Rational ClearCase → Project Explorer** from the Windows taskbar.

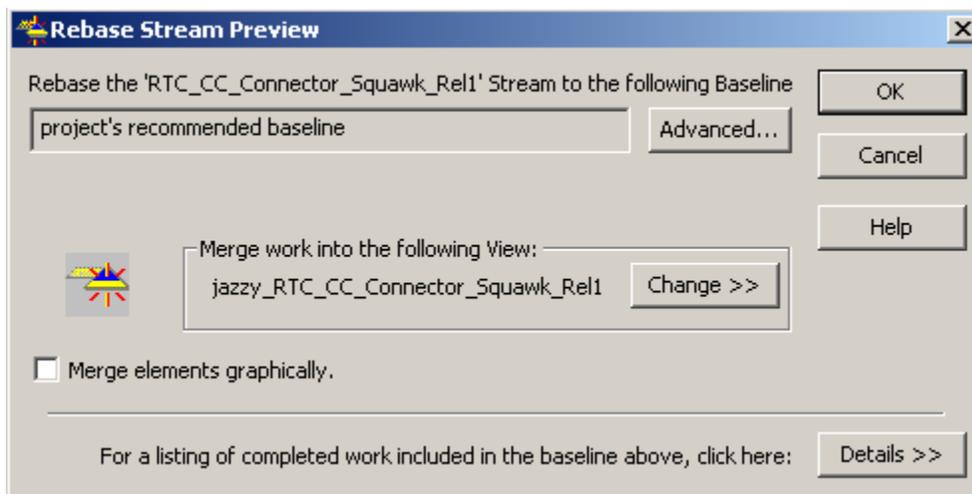


- \_\_\_5. In Project Explorer, select **View → Show All Project VOBs** menu options. Make sure to select the **Squawk** PVOB.

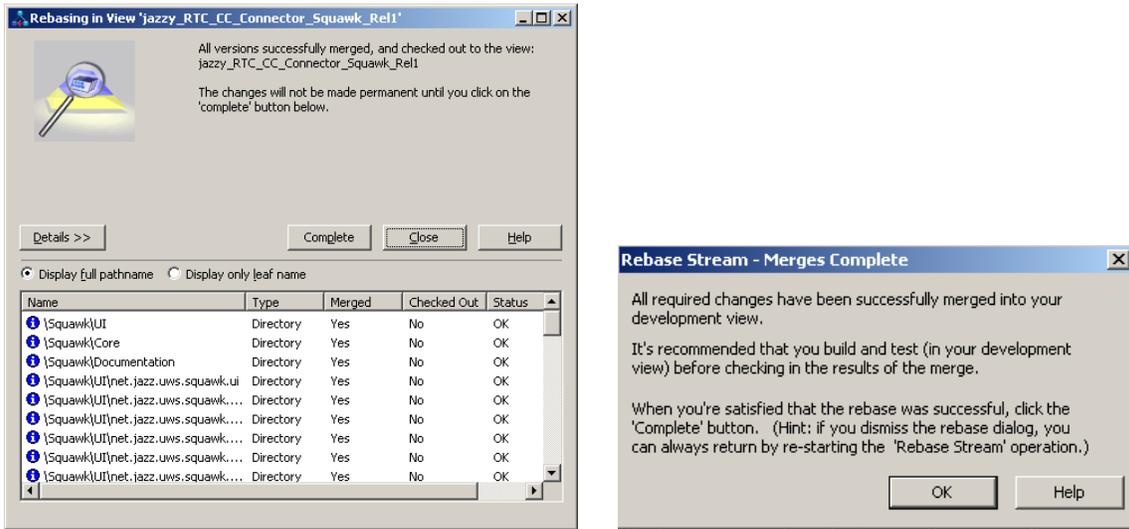
- \_\_6. Expand the **Squawk** PVOB, right-click **RTC\_CC\_Connector\_Squawk\_Rel1** UCM stream and select **Rebase Stream**.



- \_\_7. Click **OK** to initiate the rebase operation.

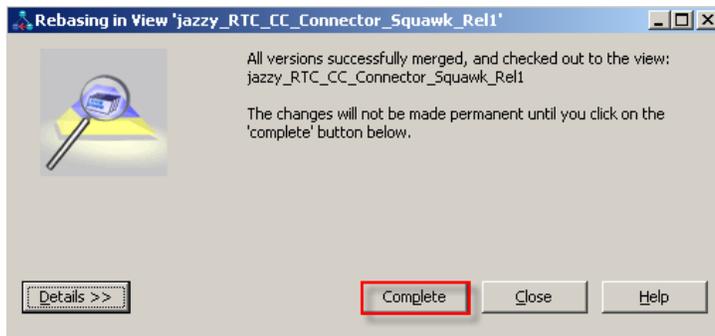


\_\_8. Click the **Details** button to verify the rebase operation progress. Click **OK**.

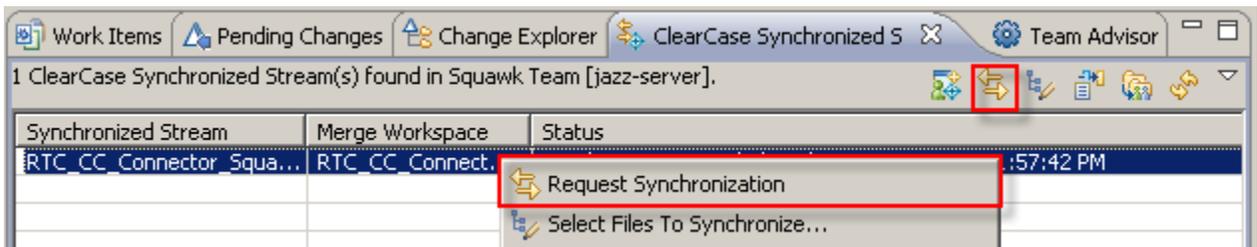


\_\_9. Click **Complete**.

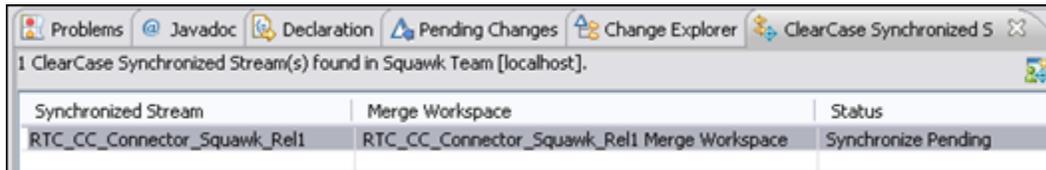
- \_\_a. Click **OK** to close the dialog that warns of the existence of other views attached to the stream.
- \_\_b. Click **Close**.



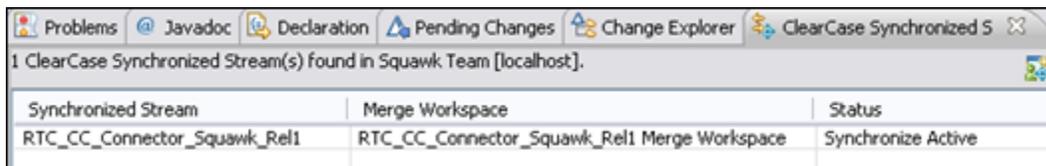
\_\_10. To request immediate synchronization, right-click the **RTC\_CC\_Connector\_Squawer\_Rel1** stream and select **Request Synchronization**. The synchronization build engine discovers the request, which can take up to a minute.



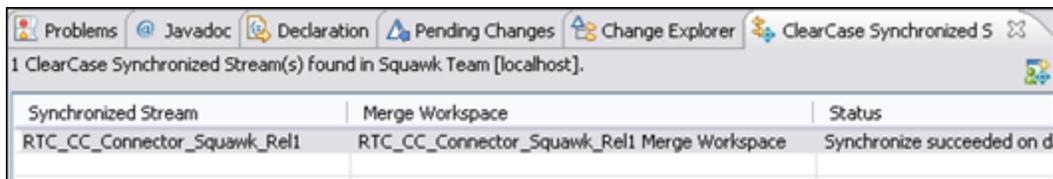
- \_\_11. You can use the Refresh icon  to monitor the synchronization progress. The Status column indicates whether the operation succeeded and whether any merges are required.



Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize Pending



Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize Active



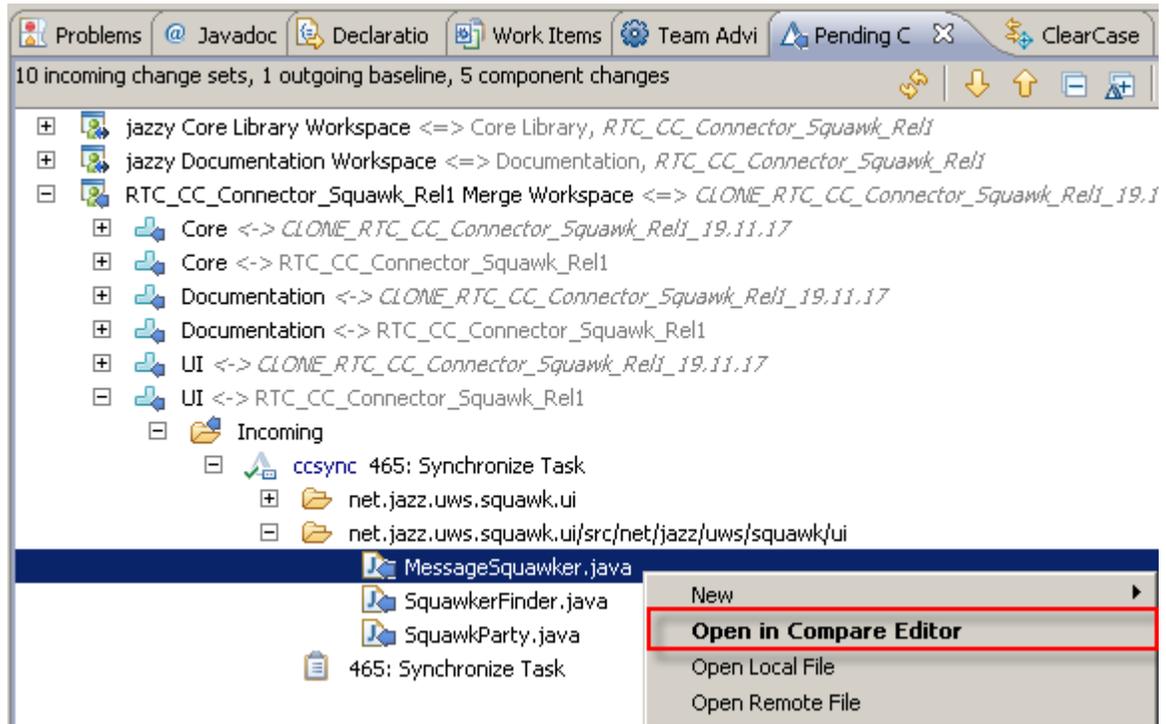
Synchronized Stream	Merge Workspace	Status
RTC_CC_Connector_Squawk_Rel1	RTC_CC_Connector_Squawk_Rel1 Merge Workspace	Synchronize succeeded on d

---- DEMO THE FOLLOWING STEPS ----

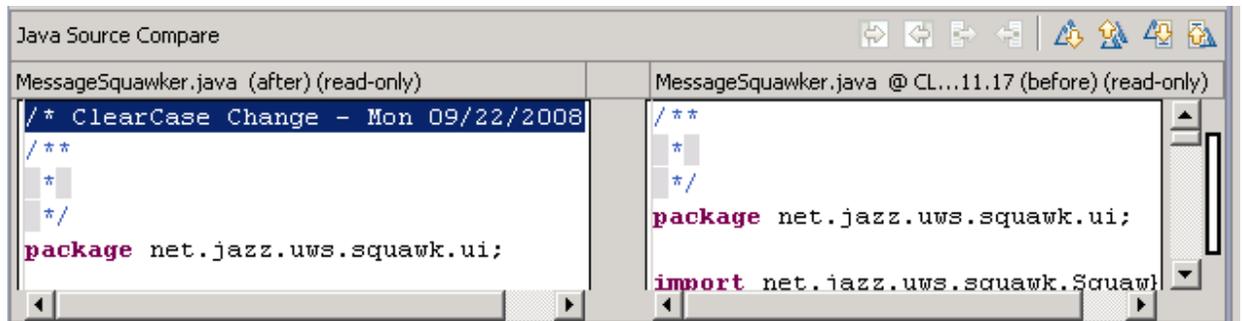
## F10.6 Show synchronization results (from Team Concert to ClearCase)

- \_\_1. Once the synchronization is complete, from the **Pending Changes** view (**Window → Show View → Pending Changes**), expand **RTC\_CC\_Connector\_Squawk\_Rel1 Merge Workspace**. Expand the UI and then expand the **Incoming** folder and sub-folders to locate the **MessageSquawker.java** file change. This is the incoming change from ClearCase.

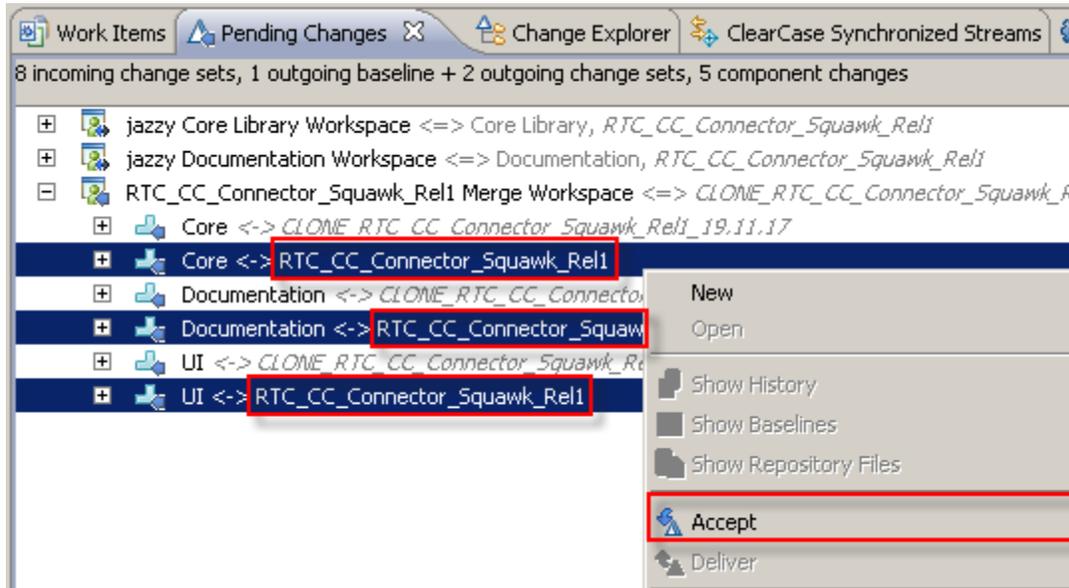
\_\_a. To inspect the incoming change, right-click and select **Open in Compare Editor**.



\_\_b. Verify changes from ClearCase are incorporated in Team Concert.



- \_\_2. Back to the **Pending Changes** view, right-click the **UI, Core** and **Documentation** components that have incoming changes for **RTC\_CC\_Connector\_Squawk\_Rel1** Stream and select **Accept**.



- \_\_a. Synchronization of changes from ClearCase to Team Concert is now complete.

### Conclusion

The Team Concert ClearCase Workspaces provide seamless interoperability between Rational Team Concert and Rational ClearCase. If your company is already using ClearCase, you can integrate the two solutions to your best advantage.

After the synchronization is complete:



- Development teams using Team Concert will have access to the UI component changes from ClearCase. In this example, developers can accept changes into their UI repository workspaces to access the latest changes from ClearCase.
- Development teams using ClearCase will have access to the Core and Documentation component changes from the Jazz repository. In this example, changes in the **RTC\_CC\_Connector\_Squawk\_Rel1** UCM Stream should now be delivered and baselined in the UCM integration stream. Developers can then rebase to incorporate those changes into their ClearCase views/streams.

---

## Appendix G: Lab 12 Be Agile with Advanced SCM - Demo



### Demo Scenario

In this scenario, the instructor, playing the role of team lead, will create a new story to track a valuable enhancement request.

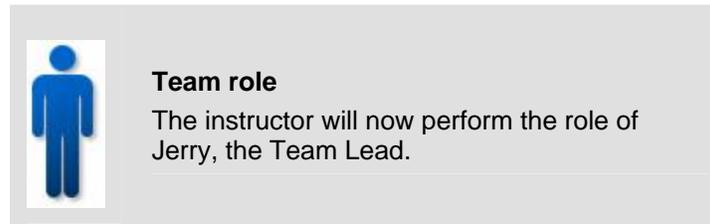
After the demonstration by the instructor, the students will create tasks against this story in the iteration plan.



### Important!

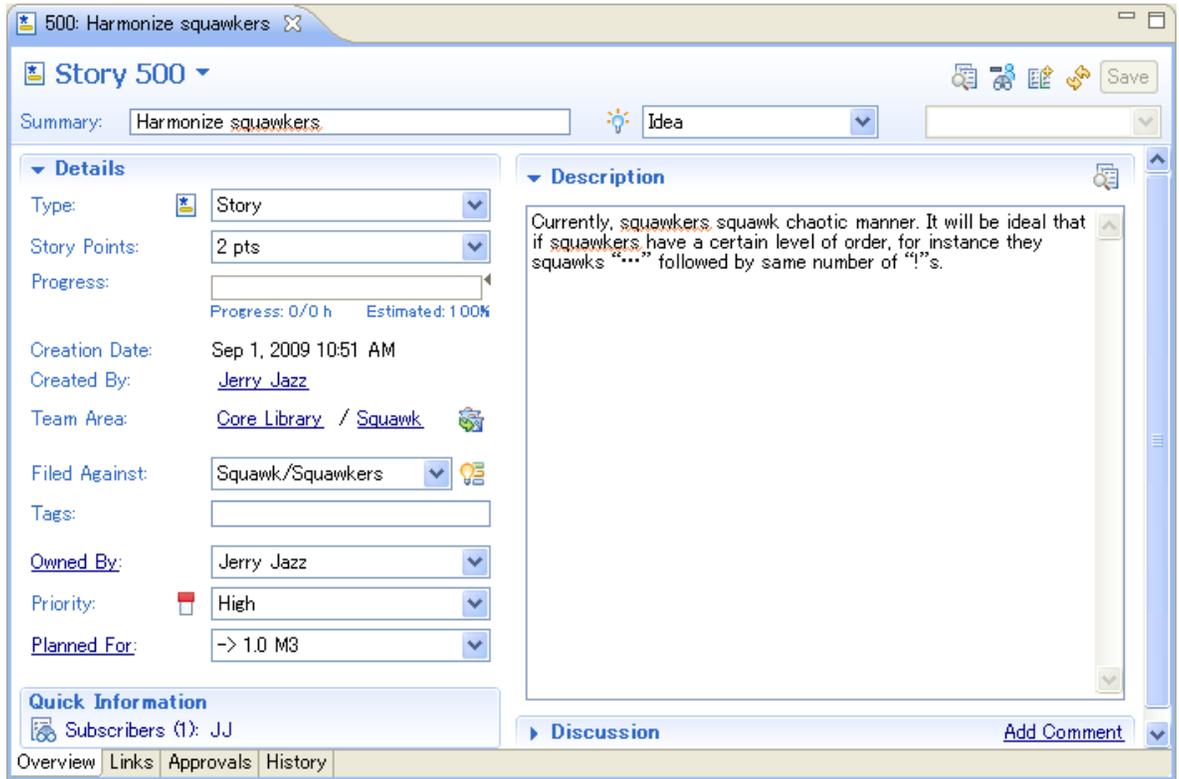
All steps in this section are to be performed by the instructor as a demonstration.

## G12.2 Instructor Demo - Team Lead creates a new Story for enhancement



- \_\_1. Create a new Story
  - \_\_a. In the **Team Artifacts** view, right-click **Work Items**, and select **New → Work Item...**
  - \_\_b. In the **Create Work Item** window select the **Story**, and click **Finish**.
  - \_\_c. For the details of the Story:
    - \_\_i. Type Harmonize squawkers in the **Summary**
    - \_\_ii. Type  
Currently, squawkers squawk chaotic manner. It will be ideal that if squawkers have a certain level of order, for instance they squawks "... " followed by same number of "!"s. in the **Description**
    - \_\_iii. Set **Story Point** to 2pt
    - \_\_iv. Set **Filed Against** to Squawk/Squawkers
    - \_\_v. Set **Owned by** to Jerry Jazz
    - \_\_vi. Set **Priority** to High
    - \_\_vii. Set **Planned For** to ->1.0 M3
    - \_\_viii. Click **Save** in the right corner of **Work Item** editor view.

\_\_d. Completed task should look similar to this:



---

## Appendix H. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have

been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

---

## Appendix I. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	AIX	CICS	ClearCase	ClearQuest	Cloudscape	Jazz
Cube Views	DB2	developerWorks	DRDA	IMS	IMS/ESA	
Informix	Lotus	Lotus Workflow	MQSeries	OmniFind	Build Forge	
Rational	Redbooks	Red Brick	RequisitePro	System i	Sametime	
<i>System z</i>	<i>Tivoli</i>	<i>WebSphere</i>	<i>Workplace</i>	<i>System p</i>	<i>Lotus Notes</i>	

Adobe, Acrobat, Portable Document Format (PDF), and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. See Java Guidelines

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Other company, product and service names may be trademarks or service marks of others.





© Copyright IBM Corporation 2009. All rights reserved.

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.

