developerWorks®

Using scrum methods with Rational Team Concert Version 4: Part 2. Plan and manage sprints

Millard Ellingsworth (millarde@us.ibm.com)
Software Developer
IBM Corporation

05 March 2013

Starting with Version 1.0, IBM® Rational Team Concert™ has supported the scrum project management approach. Over the years since, both this collaborative software and its support for scrum and agile teams have improved dramatically. This article, updated for Version 4.0.1, replaces previous articles and explains how to use Rational Team Concert effectively within a scrum team. Part 2 uses a hands-on example to explain how to plan and run your first sprint.

View more content in this series

Run the sprint

As discussed in the "Overview of the scrum process" section in Part 1 of this two-part series, each sprint starts by pulling items from the Product Backlog into the iteration, and then developing a Sprint Backlog of tasks that the team needs to complete. The result of sprint planning is a collection of estimated tasks, most likely assigned to various team members. But it is important to remember that the purpose of sprint planning is for the team to commit to completing a collection of Stories, thus adding new and shippable functionality to the product. The planning and estimating helps the team decide how much work will fit into the sprint so that they can make that commitment.

New webcast from Millard

A few months after publishing this piece, Millard did a 2-part webcast titled "Pragmatic advice for using Scrum with Rational Team Concert."

It combines advice on how to do scrum well with how to use Rational Team Concert to help drive the right conversations and practices.

You can find it on the Rational Education YouTube channel.

- Part 1: Project organization and configuration advice
- Part 2: Tracking and encouraging sprint progress and Millard's opinions on how to use Velocity

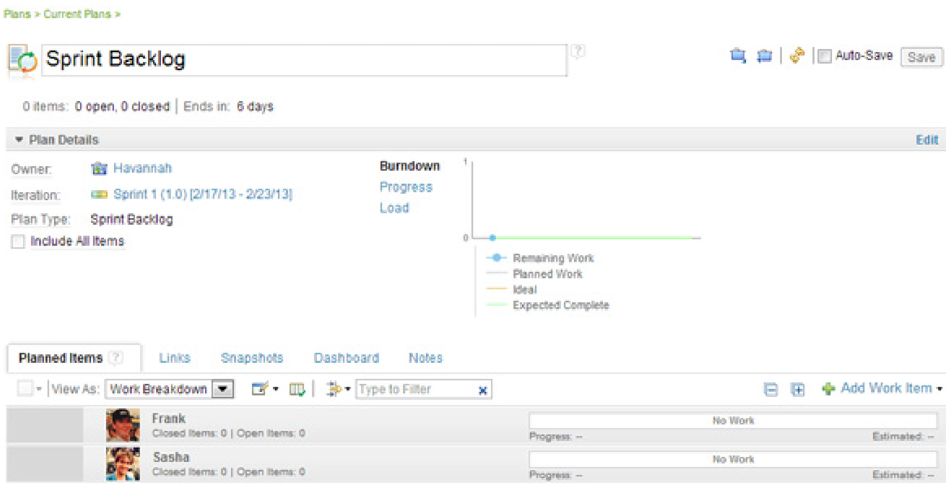
Plan the sprint

Assuming that you have Stories that are ready to be built into the product, the Sprint Planning meeting shouldn't take very long (see the links to articles in the Resources section for more on this topic). There's no rule as to how long is "too long," because it depends on the length of your sprints, the size of your team, and so on. My personal guideline is to allow about 5% of the sprint for planning. So in a 2-week sprint you should be done planning before lunch on the first day (roughly 4 of 80 hours). If you are just starting with scrum, it could take a bit longer as you get used to the practice. The first step of planning the sprint is creating the backlog.

Create the Sprint Backlog

1. Similar to the instructions in Part 1 for finding and opening the Product Backlog, use the Plans menu and **Current Plans** (or All Plans) choice to locate and open the Sprint 1 plan.
2. Click the **Plan Details** bar to display the plan's configuration. It should look something like Figure 1.

Figure 1. Default Sprint Backlog


Plans > Current Plans >

Sprint Backlog

0 items: 0 open, 0 closed | Ends in: 6 days

Plan Details

Owner: Havannah
Iteration: Sprint 1 (1.0) [2/17/13 - 2/23/13]
Plan Type: Sprint Backlog
Include All Items

View As: Work Breakdown

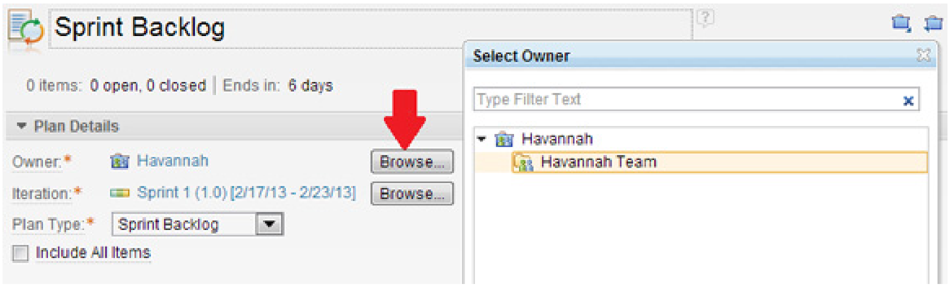
Frank
Closed Items: 0 | Open Items: 0
No Work
Progress: --   Estimated: --

Sasha
Closed Items: 0 | Open Items: 0
No Work
Progress: --   Estimated: --

The initial configuration has the Havannah project area as the plan owner, and you can see that the default Work Breakdown view shows only the project area members, not the Havannah Team members. To fix this, you need to change the plan's owner to be the team area (just as you did for the Product Backlog in Part 1).

3. Click the **Edit** link on the Plan Details bar, and then the **Browse** button on the Owner line.
4. In the Select Owner dialog window that comes up (Figure 29), select the **Havannah Team**.
5. Click **OK**. The Iteration for the plan gets reset to Unassigned. Use the **Browse** button to set it to **Sprint 1**.
6. **Save t**he changes.
7. The set of team members in the plan should change to match the team area. Click the **Plan Details** bar to close it.

Figure 2. Changing the Sprint Backlog’s owner


Sprint Backlog

0 items: 0 open, 0 closed | Ends in: 6 days

Plan Details

Owner:* Havannah

Iteration:* Sprint 1 (1.0) [2/17/13 - 2/23/13]

Plan Type:* Sprint Backlog

Include All Items

Select Owner

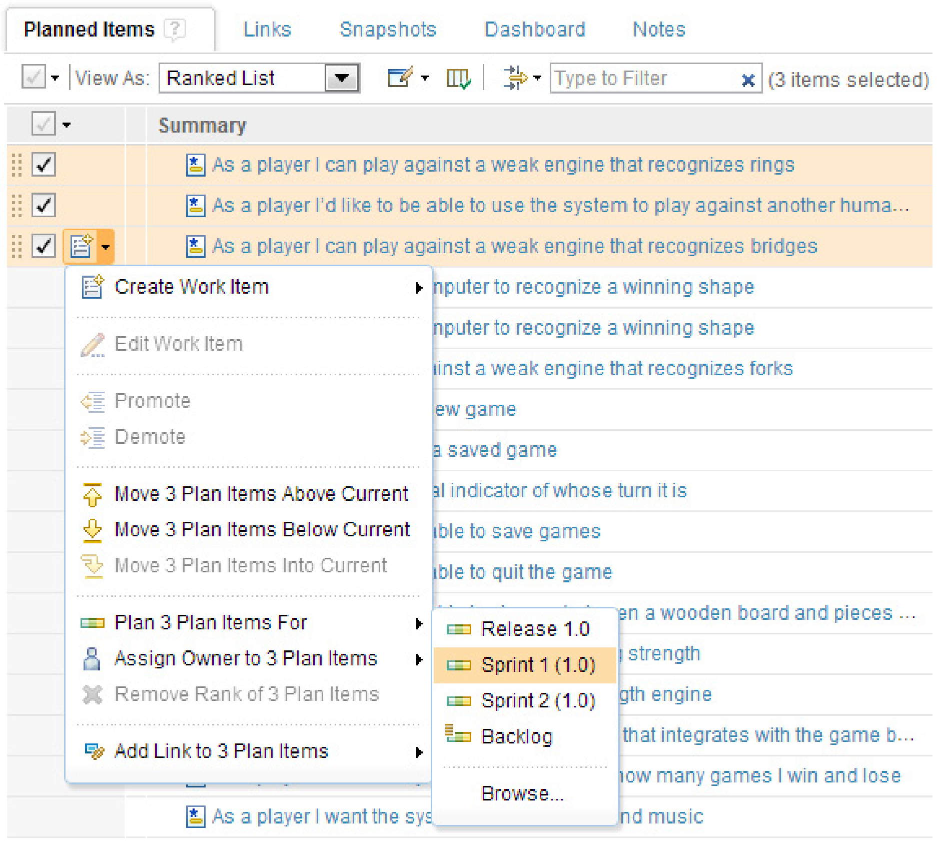
Havannah

Havannah Team

Now you have a properly configured plan, so you need to assign Stories assigned to it, and then break them down into tasks. You'll need to return to the Product Backlog plan momentarily to move the Stories to the sprint plan.

8. Using the **Plans** menu, return to the Product Backlog plans, so you can move the first three Stories to your sprint plan. Using the hover check boxes at the left side of the row, select the first three Stories, and then use the context menu to change **Planned Items for** value to Sprint 1. See Figure 3.
9. **Save** the changes. Note that the Stories will disappear from the Product Backlog, because they are now on the sprint plan.

Figure 3. Assigning Stories to Sprint 1


View As: Ranked List (3 items selected)

Summary

✓ As a player I can play against a weak engine that recognizes rings

✓ As a player I'd like to be able to use the system to play against another huma...

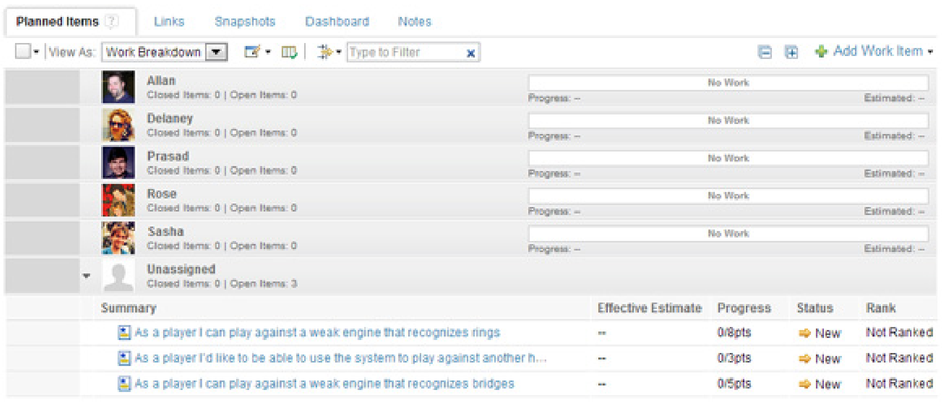
✓ As a player I can play against a weak engine that recognizes bridges

As a player I want the sys

10. Using the Plans menu, return to the **Sprint Backlog** view. You will probably need to refresh the page to make sure that the changes you just made are applied. You can use the refresh icon in the plan toolbar or just refresh the browser page.

Because the Stories don't have owners, they appear under an Unassigned user in the work breakdown shown in Figure 4. (You might need to click the Unassigned bar to open it.) It's generally considered that the team owns the Stories, although some teams choose to assign a member to own the overall Story progress. Your team will need to decide which practice it prefers, but leave them unassigned for this example.

Figure 4. Sprint Backlog with Stories to be planned



Allan
Closed Items: 0 | Open Items: 0
No Work
Progress: --
Estimated: --

Delaney
Closed Items: 0 | Open Items: 0
No Work
Progress: --
Estimated: --

Prasad
Closed Items: 0 | Open Items: 0
No Work
Progress: --
Estimated: --

Rose
Closed Items: 0 | Open Items: 0
No Work
Progress: --
Estimated: --

Sasha
Closed Items: 0 | Open Items: 0
No Work
Progress: --
Estimated: --

Unassigned
Closed Items: 0 | Open Items: 3

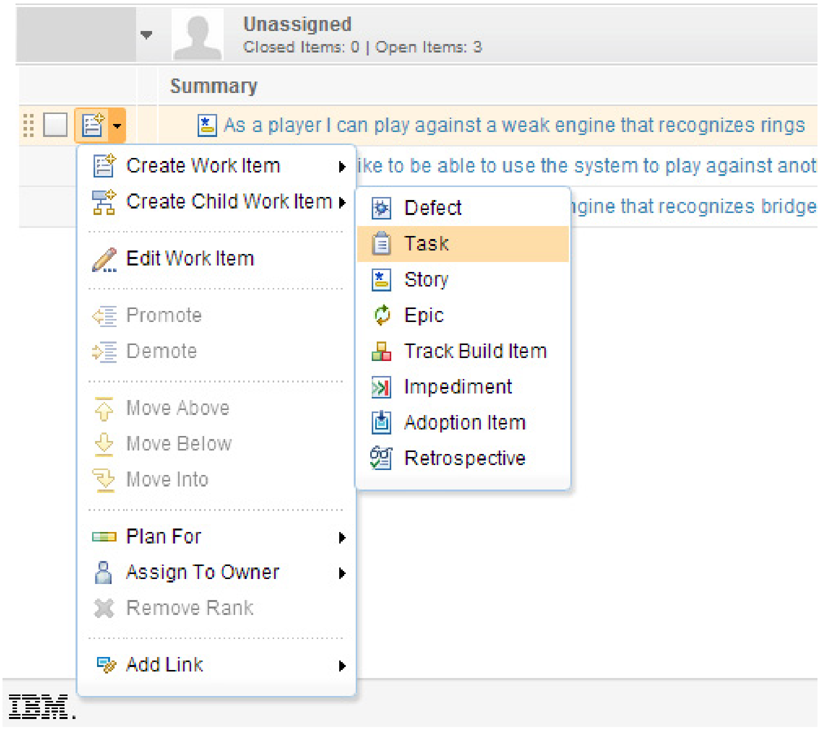
Summary	Effective Estimate	Progress	Status	Rank
As a player I can play against a weak engine that recognizes rings	--	0/8pts	New	Not Ranked
As a player I'd like to be able to use the system to play against another h...	--	0/3pts	New	Not Ranked
As a player I can play against a weak engine that recognizes bridges	--	0/5pts	New	Not Ranked

Add tasks to Stories

The next step is to break down the Stories that have been added to the sprint. This means that tasks are created for everything that needs to be performed to get the Story to "done" (that special scrum definition of *done*, which means that the feature is truly finished and can be fielded, if necessary). See the Havannah sample data document in the Downloads section for task details for each story.

1. Using the context menu in the row for the first Story (*As a player, I can play against a weak engine that recognizes rings*), choose the **Create Child Work Item** submenu and then the **Task** work item type. See Figure 5.

Figure 5. Sprint Backlog with Stories to be planned

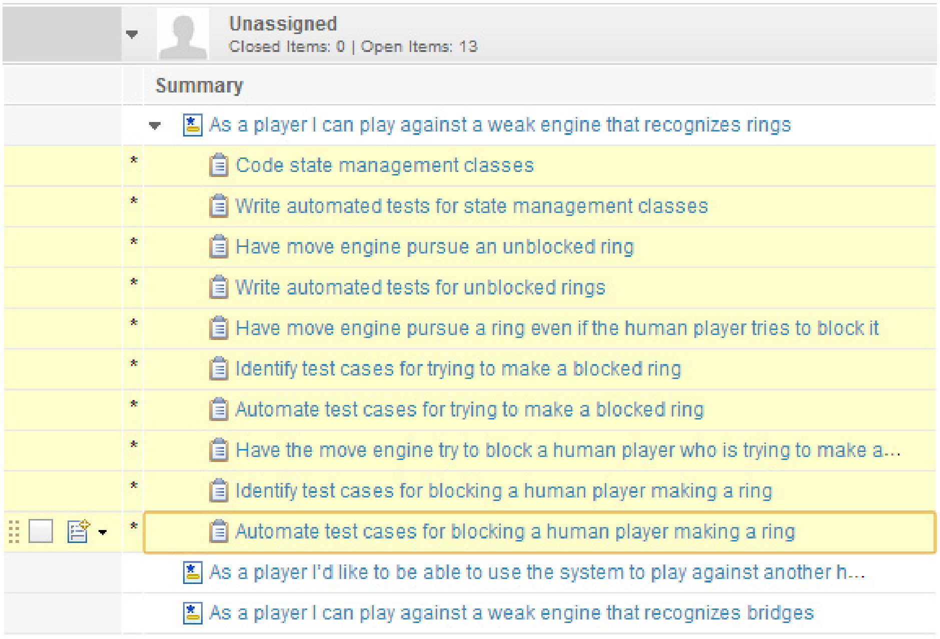


2. In the row that is added below the story, enter the task's summary:
   `Code state management classes`
3. Add the next task by using either the **Create Child Work Item** context menu choice on the Story again or the **Create Work Item** choice on the newly created task's context menu.

Note:
The way that you add the items will affect the order in which they appear. It doesn't matter which way you choose to do it. In this example, I added the first item to each Story by using Create Child Work Item. Then I added subsequent items using the Create Work Item option of the newly created item (after entering its summary). This causes the new item to be added after the current one, which most closely matches the sample data order. If you use Create Child Work Item from the Story's context menu, instead, it adds the new item to the top of the list (directly under the story) and pushes the other ones down.

4. Continue until you have entered all of the tasks for the story. Figure 6 shows the unsaved work after adding tasks to the first story.

Figure 6. Sprint Backlog with tasks for the first story


Unassigned
Closed Items: 0 | Open Items: 13

Summary
- As a player I can play against a weak engine that recognizes rings
  - * Code state management classes
  - * Write automated tests for state management classes
  - * Have move engine pursue an unblocked ring
  - * Write automated tests for unblocked rings
  - * Have move engine pursue a ring even if the human player tries to block it
  - * Identify test cases for trying to make a blocked ring
  - * Automate test cases for trying to make a blocked ring
  - * Have the move engine try to block a human player who is trying to make a...
  - * Identify test cases for blocking a human player making a ring
  - * Automate test cases for blocking a human player making a ring
- As a player I'd like to be able to use the system to play against another h...
- As a player I can play against a weak engine that recognizes bridges

In a typical Sprint Planning meeting, team members call out tasks that need to be done. Just getting them all recorded will be sufficient for now. Time estimates and assignments can come later in the meeting.

5. Continue adding tasks for the other Stories, using the Havannah sample data.

Tip:
Because of the iterative and interactive nature of sprint planning, it might work out best for your team to do these tasks and the next steps for one Story at a time. Then you can return to this section to plan the next story. There is no point in planning more work than there is time to do. Keep an eye on the Team Load indicators (see below) to know when to stop. As you get used to working with Rational Team Concert, you'll find the flow that works best for your team.

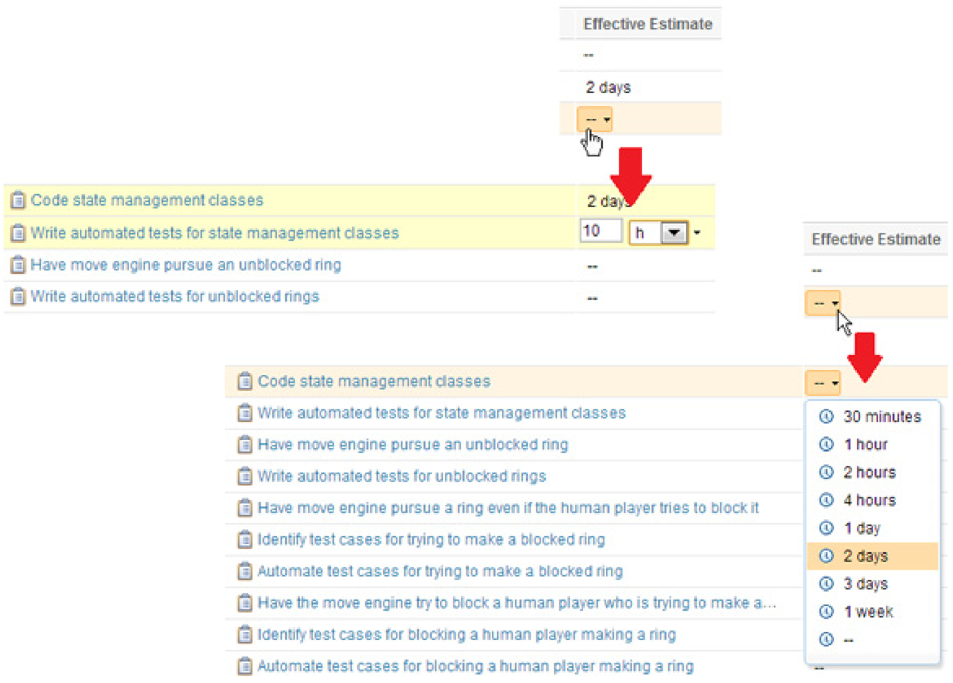
Estimate the tasks and assign owners

Now it's time to estimate the tasks that have been entered for your Stories. However, some teams prefer to assign an owner for each task first so they can estimate according to the team members' skills.

You can do the following steps in the order that fits your team's practices.

1. If you have the Sprint Backlog open, you can enter estimates directly on the plan using the drop-down menu under the Effective Estimate column. Several common estimates are available directly when you the arrow to see the options. Or if you click the double hyphen (`--`), a small editor comes up to let you enter any value (see Figure 36). Of course, having more details on the task than just a summary is always a good idea, so you can click the link to open the task, add more description, and enter the estimate there.

Figure 7. Entering time estimates on the plan



Effective Estimate
--
2 days
--

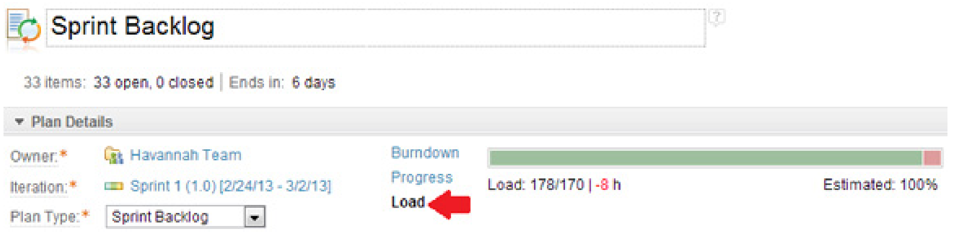
	Effective Estimate
Code state management classes	2 days
Write automated tests for state management classes	10 h
Have move engine pursue an unblocked ring	--
Write automated tests for unblocked rings	--

Effective Estimate
--
--

	Effective Estimate
Code state management classes	--
Write automated tests for state management classes	
Have move engine pursue an unblocked ring	
Write automated tests for unblocked rings	
Have move engine pursue a ring even if the human player tries to block it	
Identify test cases for trying to make a blocked ring	
Automate test cases for trying to make a blocked ring	
Have the move engine try to block a human player who is trying to make a...	
Identify test cases for blocking a human player making a ring	
Automate test cases for blocking a human player making a ring	--

When you have entered all estimates, you can see the total estimates for each story, as well as for the whole sprint. To see the total for the sprint, you need to open the **Plan Details** bar at the top of the plan, and then click the **Load** link (see Figure 8). When you do this, you should notice that the team has planned 178 hours of work but has only 170 hours available. Hold on, though. There are 5 people working for a week, so shouldn't the total hours be 200? Not quite. If you recall from Part 1, you modified Rose's commitment to the project to be available only 75% of her time, which is 30 of 40 hours. Even so that means the total should be 190.

Figure 8. The team load total is incorrect



Sprint Backlog

33 items: 33 open, 0 closed | Ends in: 6 days

▾ Plan Details

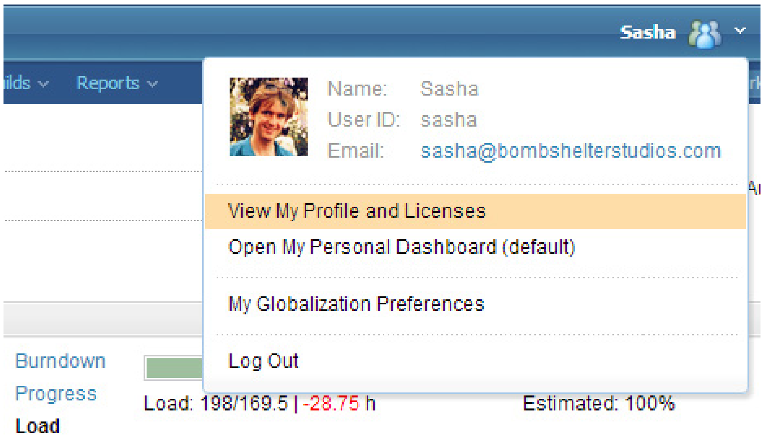
Owner:*	Havannah Team
Iteration:*	Sprint 1 (1.0) [2/24/13 - 3/2/13]
Plan Type:*	Sprint Backlog

Load: 178/170 | -8 h Estimated: 100%

In Part 1 you also added Sasha to the project area membership and the team area membership. Rational Team Concert won't let someone be assigned more than 100% total, so behind the scenes, it split Sasha's time 50/50 between the two. To fix this, you need to adjust Sasha's work allocation to give 100% of his time to the Team area. Fortunately, we're still logged in as Sasha, so this is a quick and easy fix.

2. In the user profile menu in the upper-right corner, use the **View My Profile and Licenses** option to see Sasha's profile. See Figure 9.

Figure 9. Navigate to Sasha’s profile

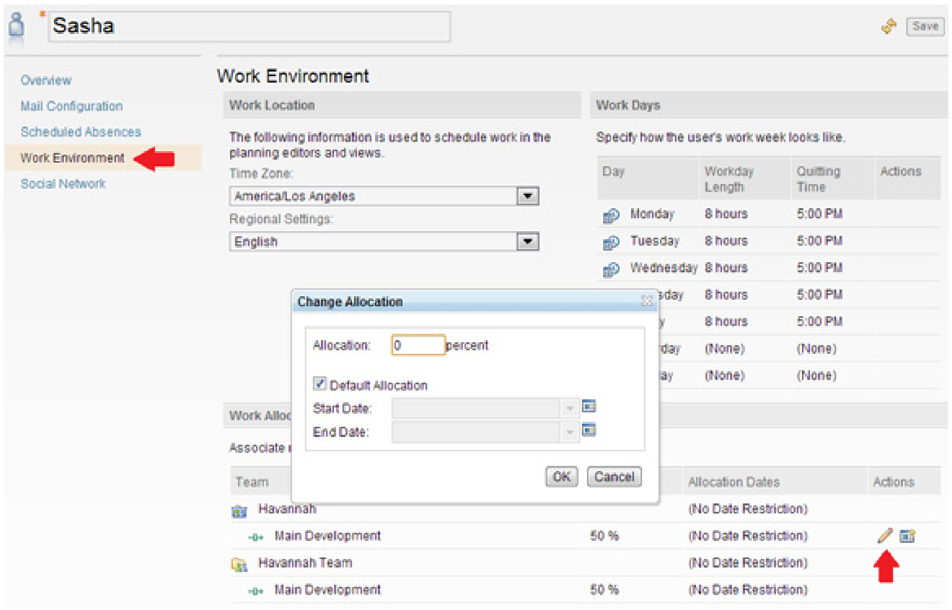

Sasha
ilds
Reports
View My Profile and Licenses
Open My Personal Dashboard (default)
My Globalization Preferences
Log Out
Burndown
Progress
Load

Name:	Sasha
User ID:	sasha
Email:	sasha@bombshelterstudios.com

Load: 198/169.5 | -28.75 h   Estimated: 100%

3. Select the work environment section, and then check the work allocation at the bottom of the page. Hover your cursor over the **Actions** column, choose the pencil icon to edit the allocation for the Havannah project area, and set it to zero (see Figure 10).
4. Use the same procedure to set the allocation for the Havannah Team to 100.
5. **Save**.
6. Use the Plans menu to return to the **Sprint Backlog**.
7. Refresh the plan to make sure that it picks up the changes in Sasha's allocation.

Figure 10. Fixing Sasha’s work allocation



The following information is used to schedule work in the planning editors and views.

Time Zone: America/Los Angeles

Regional Settings: English

Specify how the user's work week looks like.

Day	Workday Length	Quitting Time	Actions
Monday	8 hours	5:00 PM	
Tuesday	8 hours	5:00 PM	
Wednesday	8 hours	5:00 PM	
sday	8 hours	5:00 PM	
y	8 hours	5:00 PM	
rday	(None)	(None)	
ay	(None)	(None)	

Allocation: 0 percent

☑ Default Allocation

Start Date:

End Date:

Team		Allocation Dates	Actions
Havannah		(No Date Restriction)	
Main Development	50 %	(No Date Restriction)	
Havannah Team		(No Date Restriction)	
Main Development	50 %	(No Date Restriction)	

The load should now show 178 hours of work planned and 190 hours available. Sprint teams should always leave some slack, and the newer you are to the scrum process, the more slack you should allow.

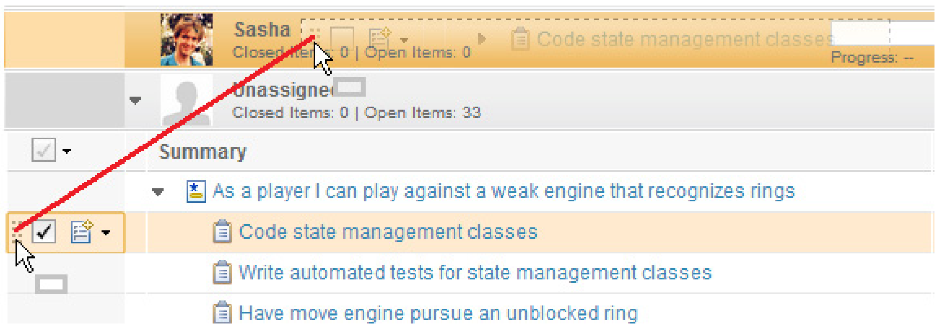
Note and tip:

If the numbers that you are seeing aren't exactly matching the example, don't worry. Rational Team Concert calculates available time based on the iteration schedule, user's work allocation, and **what time it is right now**. If you are doing this example on a Sunday as I did (a non-workday by default), and the first workday of Sprint 1 is tomorrow, then the available time should indeed show as 170 hours to start. However, what if you started this example one day, stopped, and then resumed the next day or later? If one or more of those were sprint *workdays,* the available hours will be lower. That's because some of the defined workdays are already past and those hours are no longer available for work. One important implication of this is that if you plan your sprints on Monday mornings, by noon, several of the available work hours will already be gone. The effect is that you'll never, technically, have 40 hours of work available on sprint planning weeks by the time the meeting is over.

Typically, scrum team members sign up for tasks, and the assignments can be made at the same time. Because different team members might take different amounts of time to complete a task, you may prefer to finalize a time estimate after an assignment has been made.

8. From the list in the Sprint Backlog, drag tasks to assign them to particular team members. Use the Havannah sample data document from the Downloads section to identify the assignments (see Figure 11). It is also possible to use the check boxes to select multiple items to be assigned to one person and drag them all at once.

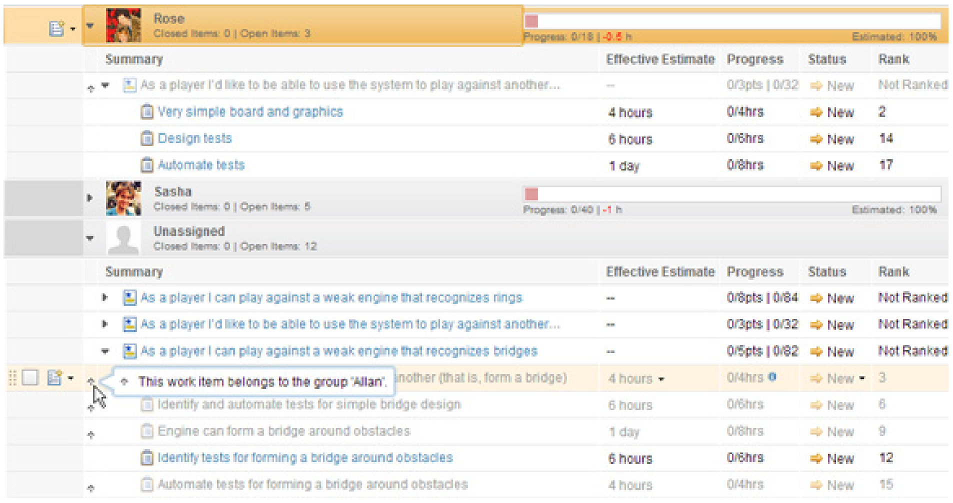
Figure 11. Assigning tasks to team members


Sasha
Closed Items: 0 | Open Items: 0
Code state management classes
Progress: --
Unassigned
Closed Items: 0 | Open Items: 33
Summary
As a player I can play against a weak engine that recognizes rings
Code state management classes
Write automated tests for state management classes
Have move engine pursue an unblocked ring

9. Continue dragging tasks to their new owners until all assignments are made. It will help to expand or collapse the Stories as needed to keep as many items on the screen at once as possible. It is not necessary to assign every task before the sprint can begin, as long as everyone is comfortable that the planned work can be completed. Of course, one way to be comfortable is to assign all of the tasks and check everyone's load bars.

When you are finished, your plan should look similar to Figure 12.

Figure 12. Finished Sprint Backlog view



Rose
Closed Items: 0 | Open Items: 3
Progress: 0/18 | -0.5 h   Estimated: 100%

Summary	Effective Estimate	Progress	Status	Rank
As a player I'd like to be able to use the system to play against another...	--	0/3pts | 0/32	New	Not Ranked
Very simple board and graphics	4 hours	0/4hrs	New	2
Design tests	6 hours	0/6hrs	New	14
Automate tests	1 day	0/8hrs	New	17

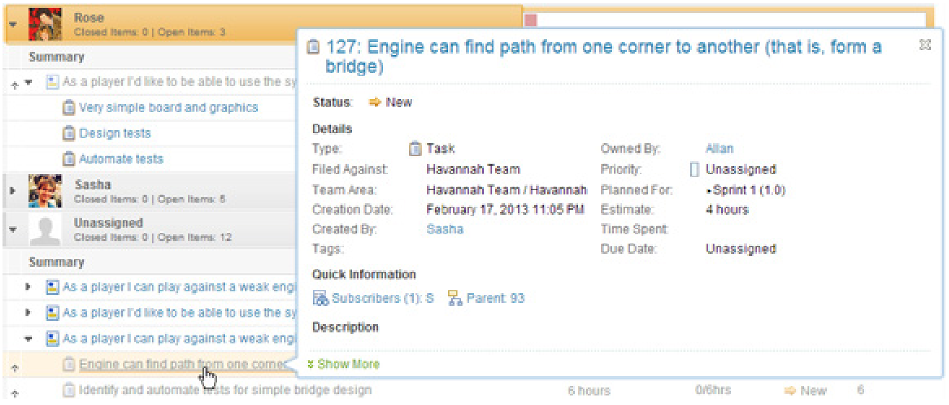
Sasha
Closed Items: 0 | Open Items: 5
Progress: 0/40 | -1 h   Estimated: 100%

Unassigned
Closed Items: 0 | Open Items: 12

Summary	Effective Estimate	Progress	Status	Rank
As a player I can play against a weak engine that recognizes rings	--	0/8pts | 0/84	New	Not Ranked
As a player I'd like to be able to use the system to play against another...	--	0/3pts | 0/32	New	Not Ranked
As a player I can play against a weak engine that recognizes bridges	--	0/5pts | 0/82	New	Not Ranked
This work item belongs to the group 'Allan'. nother (that is, form a bridge)	4 hours	0/4hrs	New	3
Identify and automate tests for simple bridge design	6 hours	0/6hrs	New	6
Engine can form a bridge around obstacles	1 day	0/8hrs	New	9
Identify tests for forming a bridge around obstacles	6 hours	0/6hrs	New	12
Automate tests for forming a bridge around obstacles	4 hours	0/4hrs	New	15

The arrows at the left of an item (see Figure 12) are **outplace indicators**. They appear when an item belongs one place but is assigned somewhere else. Hovering your cursor over the arrow shows you where the item is assigned. It is still shown here under the Story because it is a child work item of the Story. This makes it easy to glance at the Story and see the items that are not yet assigned, as well as to see who owns assigned items (without having to go look for them). Rational Team Concert supports rich hover information for items that can provide even more data (as Figure 13 shows). So do most applications that are part of the Rational solution for Collaborative Lifecycle Management (CLM).

Figure 13. Rich hover data for a task



127: Engine can find path from one corner to another (that is, form a bridge)

Status: New

Details

Type:	Task	Owned By:	Allan
Filed Against:	Havannah Team	Priority:	Unassigned
Team Area:	Havannah Team / Havannah	Planned For:	Sprint 1 (1.0)
Creation Date:	February 17, 2013 11:05 PM	Estimate:	4 hours
Created By:	Sasha	Time Spent:	
Tags:		Due Date:	Unassigned

Quick Information
Subscribers (1): S   Parent: 93

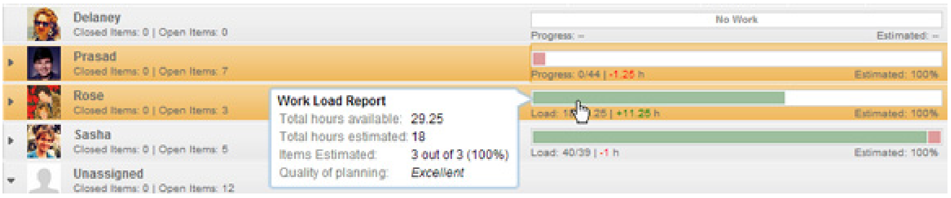
Description

Show More

Typically, *tasks* are assigned to team members, but the *Stories* remain unassigned – the team or the product owner decides when a Story is complete. One advantage to this approach is that the breakdown of a Story into tasks is always visible from this (and other) views that display both Stories (top-level work items) and tasks (often not considered top-level work items).

Also, the workload is shown for every team member, but it is not the default view (progress is the default view, because it is the one that you are going to need most of the time). Clicking on the progress bars switches between progress and load views for each user (go ahead, try it). In Figure 14, notice that Rose and Sasha have load bars displayed while Delaney and Prasad still have progress bars displayed. Hovering over a bar will open a pop-up window with details.

Figure 14. Workload bars on the iteration plan



Delaney
Closed Items: 0 | Open Items: 0
No Work
Progress: -- Estimated: --

Prasad
Closed Items: 0 | Open Items: 7
Progress: 0/44 | -1.25 h Estimated: 100%

Rose
Closed Items: 0 | Open Items: 3
Load: 1[illegible].25 | +11.25 h Estimated: 100%

Work Load Report
Total hours available: 29.25
Total hours estimated: 18
Items Estimated: 3 out of 3 (100%)
Quality of planning: *Excellent*

Sasha
Closed Items: 0 | Open Items: 5
Load: 40/39 | -1 h Estimated: 100%

Unassigned
Closed Items: 0 | Open Items: 12

This is how you can monitor team workload as tasks are estimated. To balance the workload and monitor progress, all team members must have entered their work allocations, as well as their scheduled absences (see Part 1). You could continue assigning tasks until team members no longer have time available. For scrum teams to be successful, remember to leave some slack in everyone's workload to adjust for estimates that are inadequate or for interruptions. The intent is that all of the work committed is completed, no matter what comes up, so choose a percentage of slack time that fits your team. To help assure success, leave more slack in the early stages. You can tighten it as you understand the team's rhythm and capabilities better.

Start work

The Sprint Planning meeting is over, the Sprint Backlog is full of planned work, much of it assigned, so it's time to get to building some deliverable software!

Team members are likely to use Rational Team Concert for source code management and builds, and perhaps use Rational Quality Manager to track tests and testing results, they will need the planning and work items parts only a couple of times a day to update time remaining, to close tasks, and to start new ones. If they finish early, they might need to use the query capabilities to look for unassigned work that they can do. They will probably look at the dashboard each day, at least during the Daily Scrum (also known as the *daily standup* meeting) to check the Sprint Burndown status.

The scrum master and product owner might use it a bit more often to track and help team members coordinate their work and prepare the backlog for future iterations. One thing that is valuable to team members is to organize the work that they just agreed to by using the Planned Time view within the Sprint Backlog. (If the team members use the Eclipse client, this ordering of their tasks is the same as their My Work view.) This also communicates their priorities to other team members, which can help if team members are on different work schedules and regular direct communication is difficult. In this example, if Prasad mostly pairs with Allan to create tests, it matters that Prasad can easily tell which capabilities Allan plans to develop first.

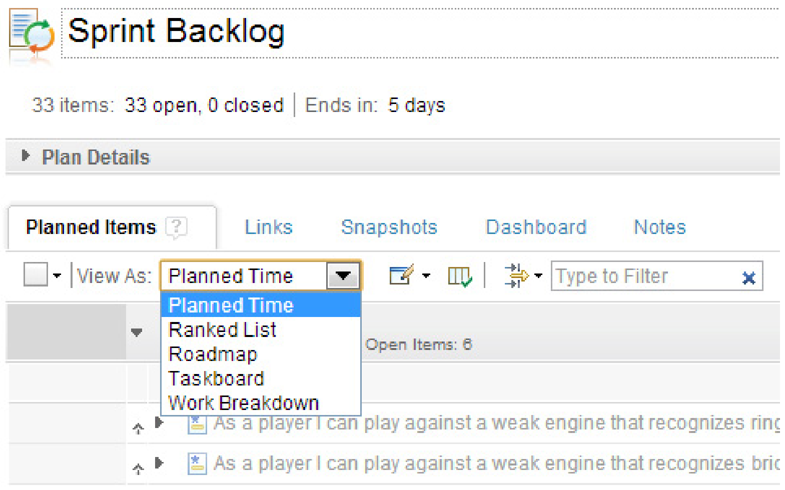
For the next part, rather than continue the do-this, do-that tutorial approach, let's just follow Sasha as he plans his work, reports progress, and works on keeping the team on track. As mentioned above, because Rational Team Concert constantly recalculates progress and time available, trying to match actions and screen captures would be difficult for the following section anyway.

Plan individual work

To follow along, log in as Sasha (use your *URI root* followed by **/ccm/web/projects/Havannah**, for example, https://localhost:9443/ccm/web/projects/Havannah), and you will be taken to the project dashboard. Use the **Plans** view to open the **Sprint 1** backlog.

The first thing Sasha does is switch to the Planned Time view, as shown in Figure 15.

Figure 15. Planned Time view selection


Sprint Backlog

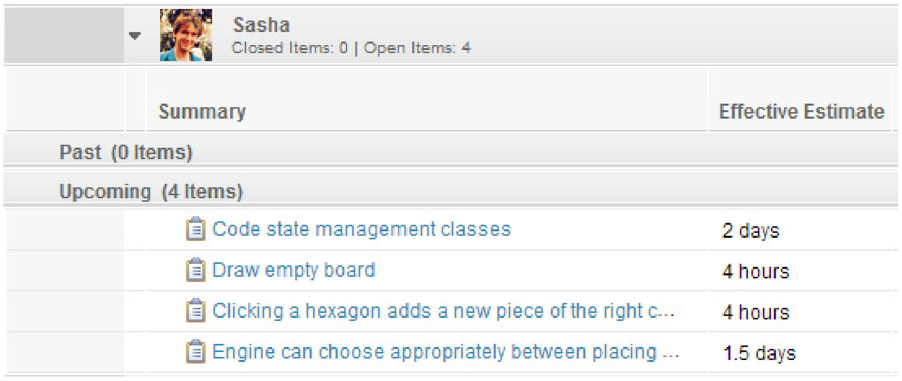
33 items: 33 open, 0 closed | Ends in: 5 days

As a player I can play against a weak engine that recognizes ring

As a player I can play against a weak engine that recognizes bric

Notice there are multiple views of the Sprint Backlog available in addition to the Work Breakdown one that you have worked with before. We'll take a look at the Taskboard view later. Given that Sasha sorts alphabetically to the bottom of the plan, he might want to close the other team members' sections while he works on his own tasks. He'll also need to open his **Upcoming** items section (as work is completed, it is moved into the **Past** items section). See Figure 16 to see Sasha's section of the Planned Time view.

Figure 16. Planned Time for Sasha


Sasha
Closed Items: 0 | Open Items: 4

Summary	Effective Estimate
Past (0 Items)	
Upcoming (4 Items)	
Code state management classes	2 days
Draw empty board	4 hours
Clicking a hexagon adds a new piece of the right c...	4 hours
Engine can choose appropriately between placing ...	1.5 days

Sasha wants to do the "Draw empty board" task first, because it's easy and he'd like to have something to show after the first day. He'll need the state management classes before he can start adding pieces, and the engine stuff will have to come last. Using the dragging technique, he moves the drawing task to the top and saves the plan.

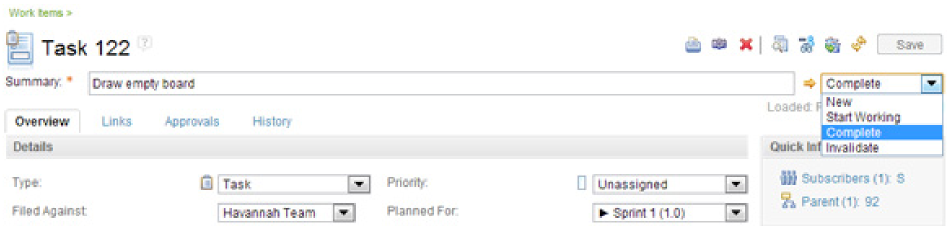
The Planned Time view makes it easy to see what work is remaining and when the owner plans to do it. As teams browse this view, they might find ways to reorder their tasks to better support their teammates. The Planned Time view can also be used to support the Daily Scrum meeting, thus making it pretty easy to discuss what's done and what's next.

To support the views discussed so far, team members should make sure that their work items properly reflect their actual states. When team members start working on a task, they should change the status to In Progress. Daily, team members should update the time remaining in each of their tasks. When a task is completed, its status should be set to Resolved. Given that the scrum method emphasizes work completed, not work started, it is preferable to start and complete a work item before moving on to the next item. This enables the team to complete Stories faster and avoids the waste associated with multitasking.

To help keep track of the amount of work still open until the Story is completed, **Time Remaining** should be recorded for each task that people worked on each day. Time Remaining is a simple text field. Therefore, if a task is worked on for multiple days, each team member needs to update the Time Remaining field **each day** according to what is left at that time for the particular task. Otherwise, the Sprint Burndown will not be accurate.

Let's pretend that some time has passed, and Sasha has completed his first task and needs to start his second one. He clicks the **Draw empty board** task to open it in the editor. He then sets the status to **Complete**, as shown in Figure 17.

Figure 17. Work Item editor and Correction field to update the estimate



Task 122

Summary: * Draw empty board

Complete

Type: Task

Priority: Unassigned

Filed Against: Havannah Team

Planned For: Sprint 1 (1.0)

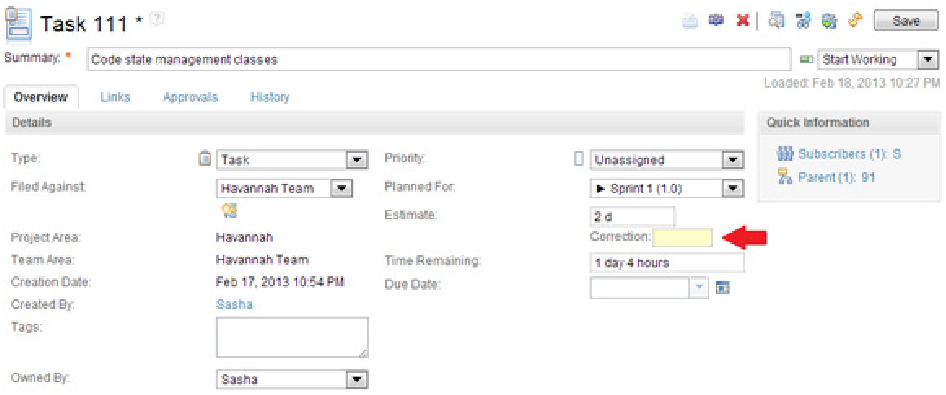
Subscribers (1): S

Parent (1): 92

Similarly, Sasha opens the next task (Code state management classes) and changes its status to **Start Working**. He also updates the time remaining to 1.5 days, because he made some progress on that task today. These sorts of small changes are important, however trivial they feel, because they update the status of the project for all stakeholders to see.

To help team members track their histories and successes in estimating tasks, a correction for the estimate should be entered as soon as it is discovered that the original estimate was incorrect. Although it is possible to edit just the estimate, that won't help the team learn.

Figure 18. Work Item editor and Correction field to update the estimate field



Task 111 *

Save

Summary: *	Code state management classes		Start Working
			Loaded: Feb 18, 2013 10:27 PM

Overview | Links | Approvals | History

Details

Type:	Task	Priority:	Unassigned
Filed Against:	Havannah Team	Planned For:	▶ Sprint 1 (1.0)
		Estimate:	2 d
		Correction:	
Project Area:	Havannah		
Team Area:	Havannah Team	Time Remaining:	1 day 4 hours
Creation Date:	Feb 17, 2013 10:54 PM	Due Date:	
Created By:	Sasha		
Tags:			
Owned By:	Sasha		

Quick Information

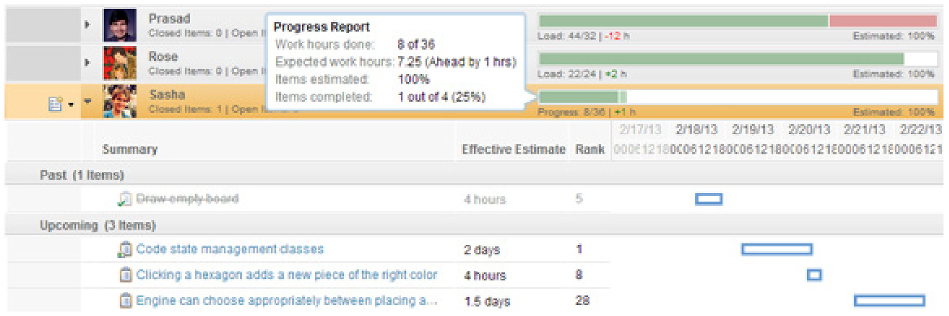
Subscribers (1): S

Parent (1): 91

Every work item has a **Discussion** feature to record progress or to capture information learned about the task. Any team member can update information in this field, and it is an excellent way to capture the history of the work item.

If you look at Sasha's Planned Time view now (Figure 19), you can see that he has a Past item and his next task is identified as in progress. He has also switched his bar to show progress rather than load, and hovering over it shows he's just a bit ahead of schedule. Not bad for the first day. The Gantt-like duration bars in the calendar segment show when he expects to do the work.

Figure 19. Sasha’s Planned Time view after the first day



Prasad
Closed Items: 0 | Open I
Load: 44/32 | -12 h	Estimated: 100%

Rose
Closed Items: 0 | Open I
Load: 22/24 | +2 h	Estimated: 100%

Sasha
Closed Items: 1 | Open I
Progress: 8/36 | +1 h	Estimated: 100%

Progress Report

Work hours done:	8 of 36
Expected work hours:	7.25 (Ahead by 1 hrs)
Items estimated:	100%
Items completed:	1 out of 4 (25%)

Summary	Effective Estimate	Rank
Past (1 Items)		
~~Draw empty board~~	4 hours	5
Upcoming (3 Items)		
Code state management classes	2 days	1
Clicking a hexagon adds a new piece of the right color	4 hours	8
Engine can choose appropriately between placing a...	1.5 days	28

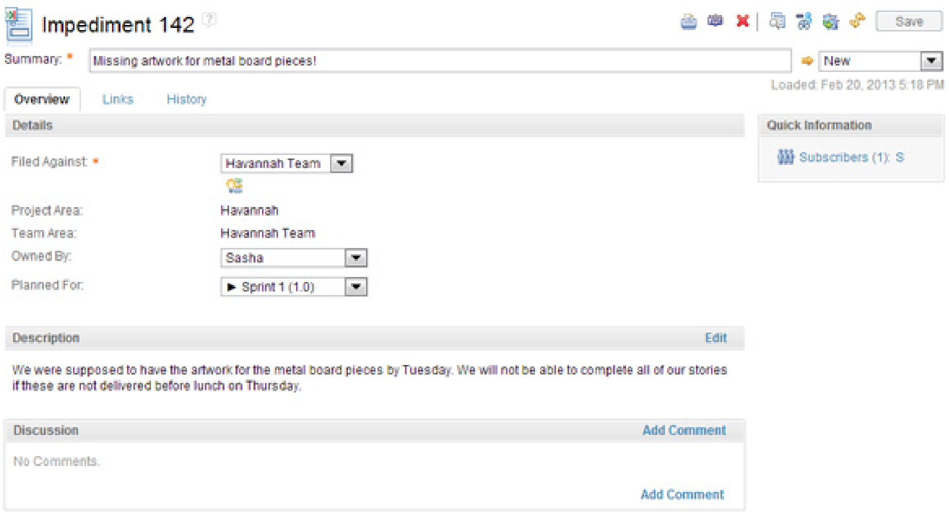
All team members should be engaged in similar activities each day.

Story progress should be tracked by starting work on it, as well. Often, no particular team member is made responsible for the Story as a whole. In that case, as the tasks under the Story are completed, the scrum master or product owner must move the Story through its workflow stages (select **New > In Progress > Implemented > Done**).

BLOCKED!

An important part of the Daily Scrum meeting is reporting items that are blocking progress. Given the short time between the Sprint Planning meeting and the Sprint Review meeting where you show off your deliverable software, you can't afford blockers, or impediments. In Rational Team Concert, you can report a blocker at any time by creating an Impediment work item. That item typically assigned to the scrum master for resolution. Figure 20 shows a sample impediment that might hamper the Havannah team. The team dashboard, discussed below, contains a widget that displays all open impediments so that everyone knows about them.

Figure 20. Impediment work item



Summary: * Missing artwork for metal board pieces! New

Loaded: Feb 20, 2013 5:18 PM

Filed Against: * Havannah Team

Project Area: Havannah

Team Area: Havannah Team

Owned By: Sasha

Planned For: Sprint 1 (1.0)

We were supposed to have the artwork for the metal board pieces by Tuesday. We will not be able to complete all of our stories if these are not delivered before lunch on Thursday.

No Comments.

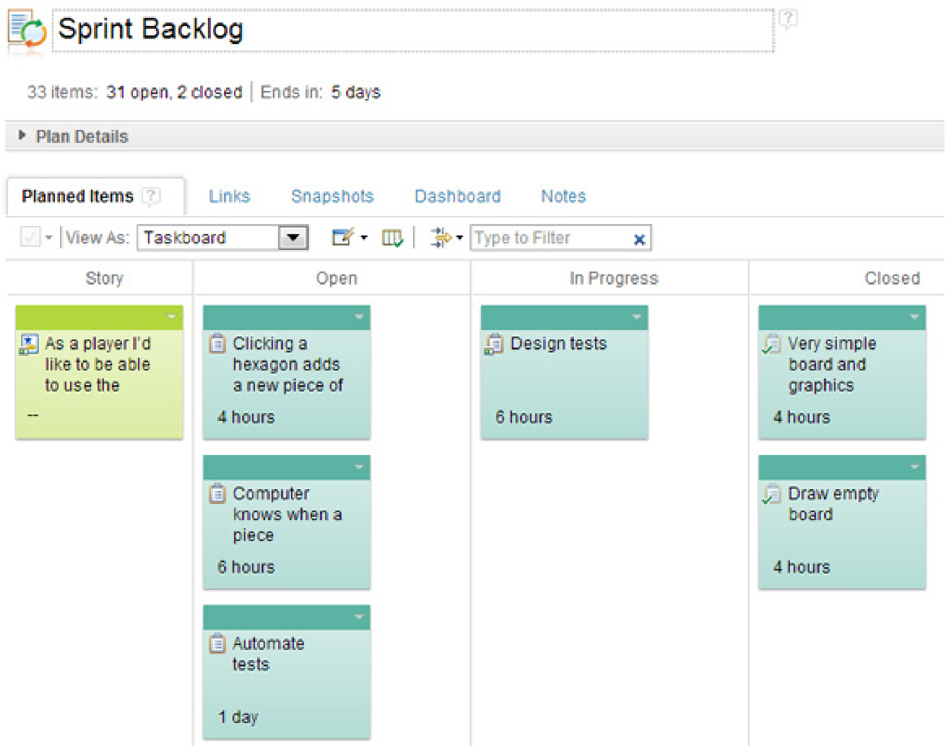
Track and report progress

After work is underway, the team will meet at least daily to review progress at the daily standup (Daily Scrum), probably using the Sprint Burndown chart discussed below. As mentioned previously, the Planned Time view works well for this also, because each person's completed work and current work are visible and easy to find. You can even run the roll call based on the order of people in the plan.

Developer's Taskboard

Another way to monitor work progress is the **Developer's Taskboard**. It is reminiscent of the classic whiteboard covered with sticky notes, which shows the tasks in columns that indicate which ones are waiting, in progress, or completed. The simple display makes it easy for people who aren't on the team to follow and understand it, too. Also, it always shows the relationship of the tasks to their parent items (typically Stories) in the left column. Team members can start working on items or set them to Completed by simply dragging them to the appropriate column. See Figure 21 for an example of a Taskboard view.

Figure 21. Developer's Taskboard



Burndown charts

The classic way to track progress in a scrum team is the burndown chart. The Sprint Burndown chart provides an instant answer to the question "Are we on track to finish all of the work that we committed to do?" Assuming that all of your team members update their work items appropriately, the trend line moves closer to zero (work remaining) as work is completed.

A dashboard for the project area is automatically created for a new scrum project, with a variety of charts, including a burndown chart (for both the sprint and the release). A dashboard is also created automatically for each team area. That one is the focus here. Although the dashboard is not a scrum artifact, it is a central point of focus for a team that uses Rational Team Concert and where you can find your burndown chart. (Each user has a personal dashboard, too, but that's a discussion for another article.)

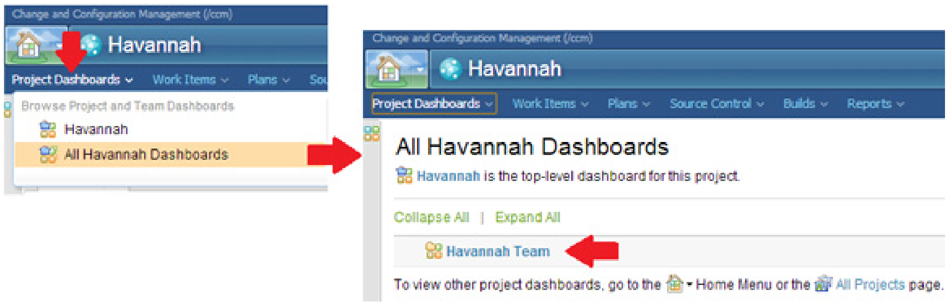
Tip:
If team members do not update their data by the end of each day, the status of their work items will not be reflected properly in the report. Although updating the status of their tasks will help the trend line to catch up at the next processing point, there is no way to update the history. This might cause the burndown chart to have unfortunate flat segments (making it seem no progress was made) and sudden, unexplainable dips in the remaining work (which often indicates that work was thrown out of the sprint). It really is important that team members keep their work items updated (which is why I keep bringing it up).

Rational Team Concert does not skip non-work days in its plotting. In a globally distributed team, it might be Sunday for you but already Monday for some of your teammates. Therefore, flat lines often represent weekend times, but they could also signal a lack of progress (or progress updates).

Team dashboard

Open the Havannah Team dashboard by using the Project Dashboards menu (Figure 22), navigating to the **All Havannah Dashboards** page, and then choose the **Havannah Team** entry.

Figure 22. Finding the team dashboard



All Havannah Dashboards

Havannah is the top-level dashboard for this project.

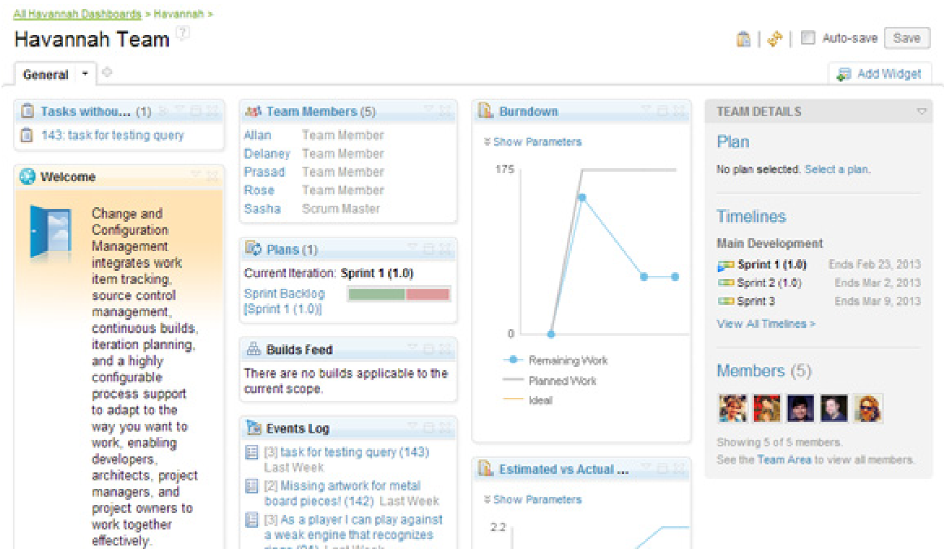
To view other project dashboards, go to the Home Menu or the All Projects page.

The default dashboard, shown in Figure 23, includes various widgets in addition to the burndown chart. You'll probably want to remove the Welcome widget eventually, because it has documentation links that you aren't likely to need after you get started. There is a Customize Your Dashboard widget that includes instructions on how to add, remove, and reorganize the dashboard. There's also a widget to track your team builds, to list the team members, to provide of a list of team plans, and more. The burndown chart is one of many built-in reports that is available in a small version that can be used in a dashboard.

You can add any of the many widgets that are included in Rational Team Concert or add Open Social gadgets. You can create additional tabs across the top of the dashboard, each with a different set of widgets. Widgets can be added that display reports (such as the burndown chart) or the results of queries, either predefined or ones that you create. Some of the report widgets, such as the Burndown widget, have links in their titles that open up a full-sized version of the report.

The best way to learn about the dashboard and what it can do is to play around with it. If you close the browser window without saving any changes, it will revert to its previous configuration when you open it again.

Figure 23. The Havannah Team dashboard



All Havannah Dashboards > Havannah >

Havannah Team

General

Tasks withou... (1)

143: task for testing query

Welcome

Change and Configuration Management integrates work item tracking, source control management, continuous builds, iteration planning, and a highly configurable process support to adapt to the way you want to work, enabling developers, architects, project managers, and project owners to work together effectively.

Team Members (5)

Allan	Team Member
Delaney	Team Member
Prasad	Team Member
Rose	Team Member
Sasha	Scrum Master

Plans (1)

Current Iteration: **Sprint 1 (1.0)**

Sprint Backlog [Sprint 1 (1.0)]

Builds Feed

There are no builds applicable to the current scope.

Events Log

[3] task for testing query (143) Last Week

[2] Missing artwork for metal board pieces! (142) Last Week

[3] As a player I can play against a weak engine that recognizes

TEAM DETAILS

Plan

No plan selected. Select a plan.

Timelines

Main Development

Sprint 1 (1.0)	Ends Feb 23, 2013
Sprint 2 (1.0)	Ends Mar 2, 2013
Sprint 3	Ends Mar 9, 2013

View All Timelines >

Members (5)

Showing 5 of 5 members.
See the Team Area to view all members.

During retrospective meetings, the team can discuss how adding information to the dashboard might help with issues that need more attention. To provide at least a small taste of how a dashboard can be customized to suit your team, let's create a query that finds tasks for the current sprint that do not have time estimates, and then install that query into the team dashboard. Usually, this query should not return any results, but when it does, you need them to be visible in the dashboard where they will be brought to everyone's attention. These items would represent work that is planned but not accounted for in anyone's estimates and not included in the burndown report. That could present a problem getting the work finished (or getting to "done" in agile parlance).

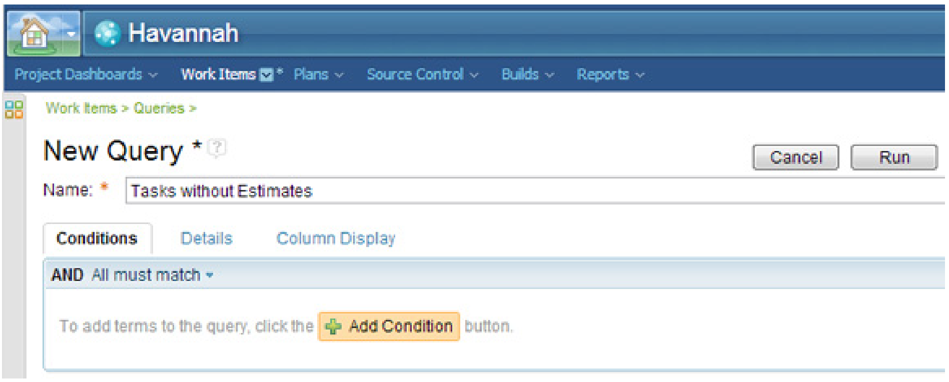
Create a custom query

First, create and test the query.

1. Use the Work Items menu and choose the **Create Query** action.
2. Enter a name for the query, for example, `Tasks without estimates`.

The page should look like Figure 24.

Figure 24. New Query page



Name: * Tasks without Estimates

AND All must match

To add terms to the query, click the Add Condition button.

Add conditions and set values

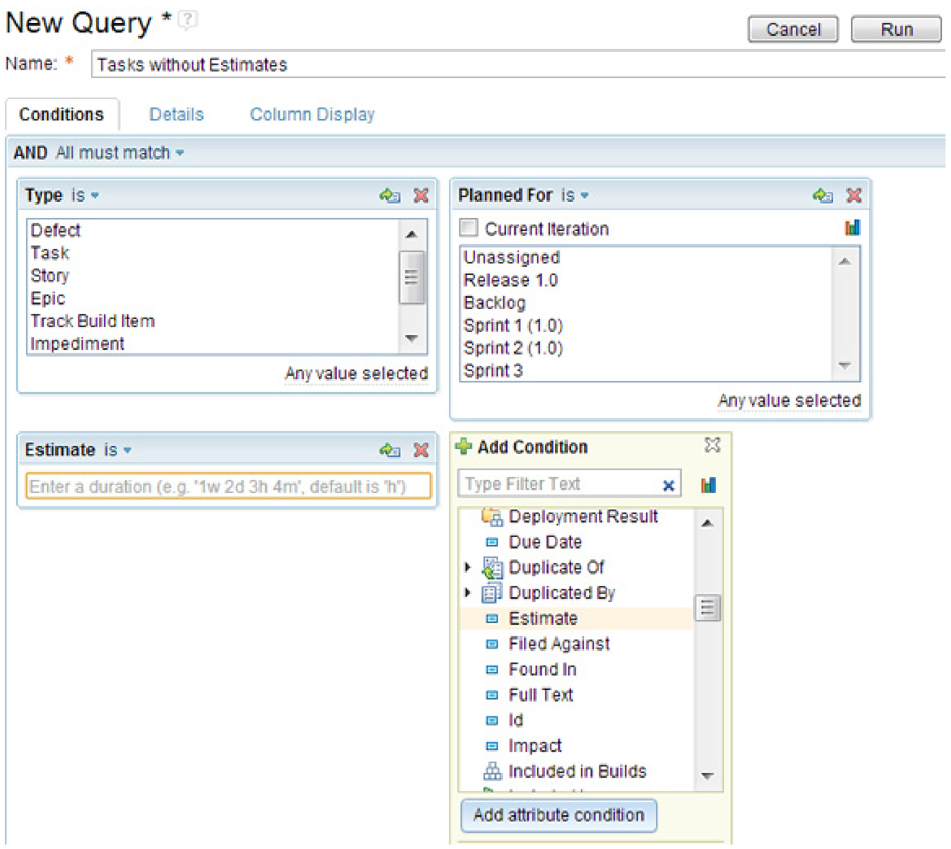
Now you need to add several conditions:

- Type = Task
- Planned For = current iteration
- Estimate is 0 h

1. Start by clicking the **Add Condition** button, and then scroll down the Add Condition widget, choose **Type**, and click the **Add attribute condition** button. A condition box will be displayed, and the Add Condition widget will move to the right, ready for you to add the next condition.
2. In the Add Condition control, scroll to and select **Planned For**, and click **Add attribute condition**.
3. Repeat the procedure to add **Estimate** to the query.

The page should now have three conditions (plus the Add Condition widget) and look similar to Figure 25.

Figure 25. Query conditions

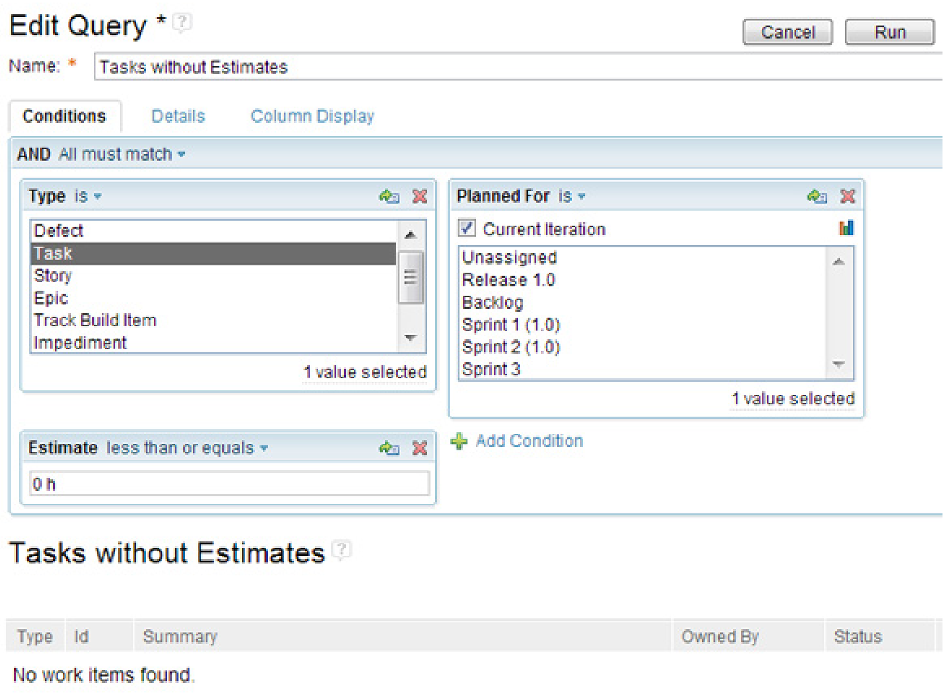


Next, set the condition values for the attributes.

4. In the Type condition (Type is…) area, select **Task**.
5. Under the Planned For condition, check the **Current Iteration** check box. This is known as a query variable. It automatically looks up the correct value based on the current iteration setting for the project timeline.

6. For the Estimate condition, enter `0 h` (for zero hours), and use the drop-down menu in the header to change the condition to **less than or equals** (just using *equals* does not work).
7. Test the query by running it (using the **Run** button in the toolbar). The result should look similar to Figure 26 (notice that I closed the Add Condition widget to save screen real estate). There are no errors, so he query seems to work. However, it is hard to be sure because it didn't find any work items (unless you cheated and didn't enter estimates for all of the items during the Sprint Planning section).
8. **Save** the query.

Figure 26. Finished custom query


Edit Query *

Name: * Tasks without Estimates

AND All must match

Type is: Defect, **Task** (selected), Story, Epic, Track Build Item, Impediment — 1 value selected

Planned For is: ☑ Current Iteration; Unassigned, Release 1.0, Backlog, Sprint 1 (1.0), Sprint 2 (1.0), Sprint 3 — 1 value selected

Estimate less than or equals: 0 h

Tasks without Estimates

Type	Id	Summary	Owned By	Status

No work items found.

Now, add a work item without an estimate so you can make sure that the query works.

9. Use the Work Items menu to create a **Task**. Enter whatever you want for a summary, set Filed Against to **Havannah Team** and Planned for to **Sprint 1**. Do not enter an estimate.
10. **Save** your changes.
11. Use the Work Items menu, and select your new query from the Recent Queries list. If it doesn't automatically run and show you the task that you just added, click the **Run** button to verify that the query works.

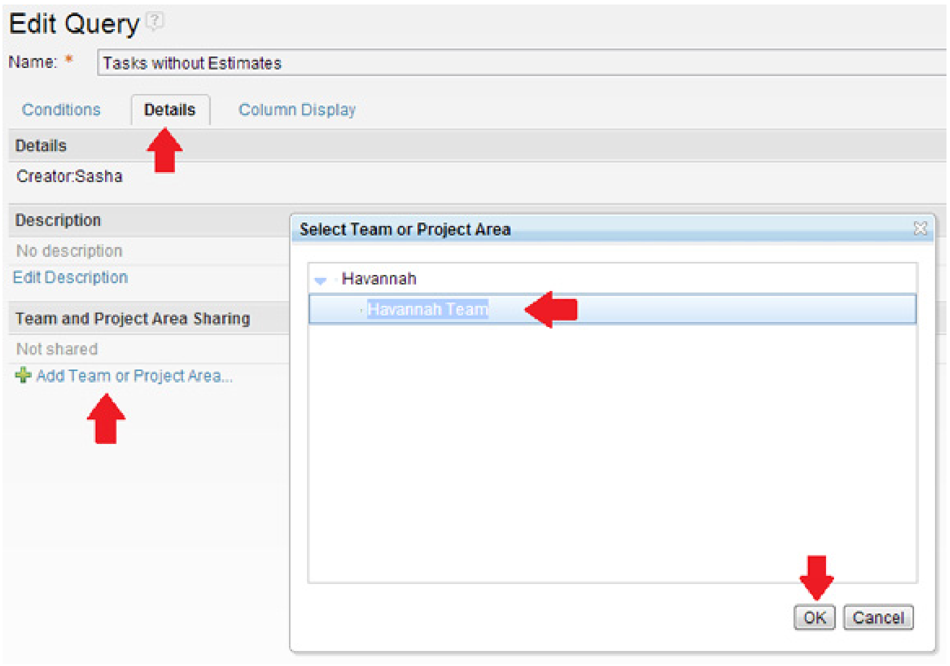
There's one more step for this query. At the moment, it is Sasha's personal query so will show only under his My Queries list. If you want to use this query in the team dashboard, it needs to be accessible to all team members. If it isn't specified that way, others will see an error about not having permission to access the query rather than the query results.

Make the query available to the rest of the team

Because you just tested the query, it's already open.

1. Click the pencil icon to edit your query.
2. Select the **Details** tab.
3. Under Team and Project Area Sharing, select the **Add Team or Project Area** link.
4. In the Select Team or Project Area dialog window, select **Havannah Team**, and click **OK**.
5. **Save**.

Figure 27. Sharing a query with the Havannah team


Edit Query

Name: * Tasks without Estimates

Conditions | Details | Column Display

Details

Creator:Sasha

Description

No description

Edit Description

Team and Project Area Sharing

Not shared

Add Team or Project Area...

Select Team or Project Area

Havannah

Havannah Team

Add the query to the team dashboard

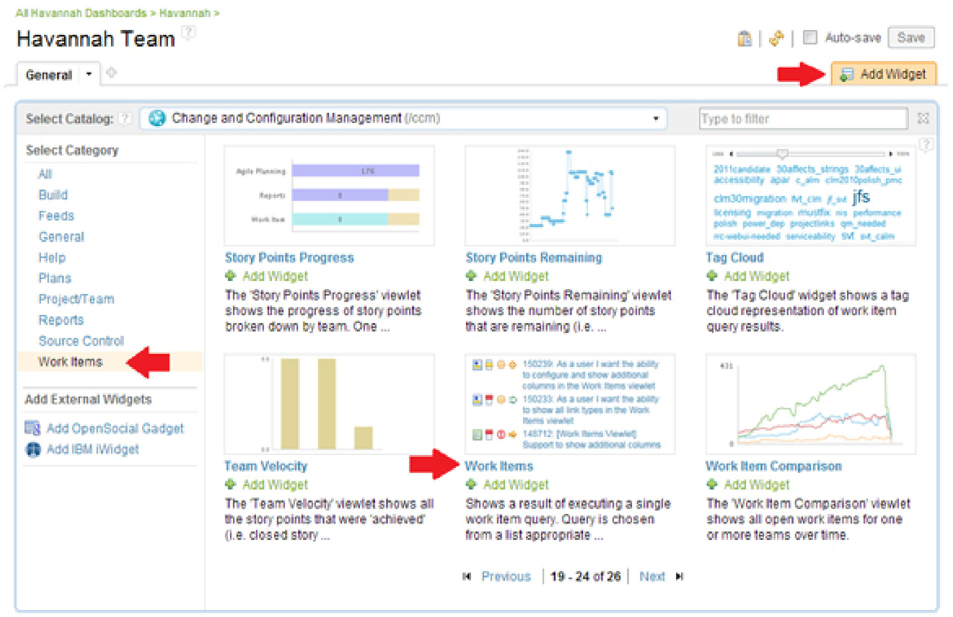
Now that you have a tested query, so you know that it works, and you have shared it with the team, you can install it into the Havannah Team dashboard.

1. Use the Project Dashboards menu to navigate to the **Havannah Team dashboard**.
2. Select the **Add Widget** link on the right side of the tab line.
3. Under the Select Category list on the left side, choose **Work Items**.

This will reduce the number of widgets quite a bit, but you'll still need to scroll to find the **Work Items** widget. (You could have typed that into the filter field, but I encourage you to browse through the widgets and read some of the descriptions to see what else interests you.)

4. Click the **Next** link at the bottom of the widget palette until the Work Items widget comes into view (see Figure 28). The widget shows the results of executing a single work item query (which you will configure for the widget).

Figure 28. Work Items widget palette



Story Points Progress
Add Widget
The 'Story Points Progress' viewlet shows the progress of story points broken down by team. One ...

Story Points Remaining
Add Widget
The 'Story Points Remaining' viewlet shows the number of story points that are remaining (i.e. ...

Tag Cloud
Add Widget
The 'Tag Cloud' widget shows a tag cloud representation of work item query results.

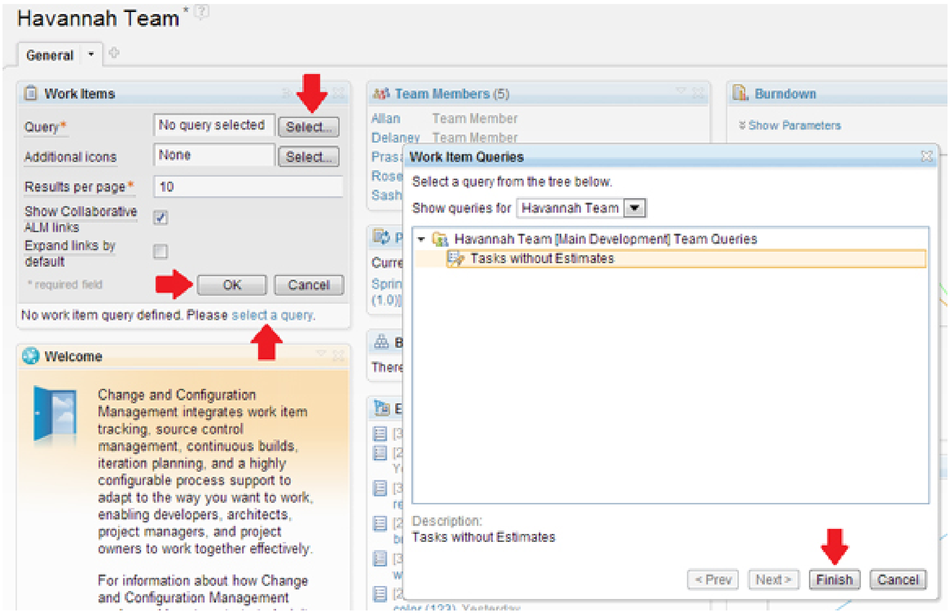
Team Velocity
Add Widget
The 'Team Velocity' viewlet shows all the story points that were 'achieved' (i.e. closed story ...

Work Items
Add Widget
Shows a result of executing a single work item query. Query is chosen from a list appropriate ...

Work Item Comparison
Add Widget
The 'Work Item Comparison' viewlet shows all open work items for one or more teams over time.

5. Under the Work Items widget section, select **Add Widget**. You will get a brief indication that the widget was added, and you might see it appear as the first widget in the upper left of the page (depending on how tall your window is when open).
6. Use the close button on the widget palette to hide it. The very first widget in the upper left should be the new one that you just added. The title will be just *Work Items*, and the message inside the widget will indicate that no work item query has been defined, along with a link to fix that.
7. Choose the **select a query link** to open the widget configuration dialog window.
8. Click the **Select** button for the Query attribute to open the Work Item Queries dialog window.
9. Open the **Havannah Team (Main Development Team)Team Queries** folder (remember, you shared the new query with the team earlier). See Figure 29.

Figure 29. Work Items widget configuration



Query* No query selected Select...
Additional icons None Select...
Results per page* 10
Show Collaborative ALM links ☑
Expand links by default ☐
* required field OK Cancel
No work item query defined. Please select a query.

Allan Team Member
Delaney Team Member

Select a query from the tree below.
Show queries for Havannah Team
Havannah Team [Main Development] Team Queries
Tasks without Estimates
Description:
Tasks without Estimates

Change and Configuration Management integrates work item tracking, source control management, continuous builds, iteration planning, and a highly configurable process support to adapt to the way you want to work, enabling developers, architects, project managers, and project owners to work together effectively.

For information about how Change and Configuration Management

10. Click **Finish** in the Work Item Queries window.
11. Click **OK** in the Work Items widget configuration window.

The query will run, and the result will be displayed in the widget. The name of the widget will change to match the query, and the number of items that match the query will be shown in parentheses. In this example, the title should say *Tasks without estimates (1)*.

12. **Save** the changes to the dashboard.

Feel free to play with the widget menus to adjust the appearance. Remove the Welcome widget if you prefer, or just move it so it isn't so prominent. If you don't like the changes you've made, just close the browser without saving them.

As your team works together and considers how to improve, you will find that the dashboard and custom queries are an invaluable way to track progress. This is just a taste of what's possible.

Filling the data warehouse

For tracking trends and historical reporting, Rational Team Concert uses data warehouse technology to collect and somewhat compress what would be a large amount of data, even for a relatively small project. If you are trying to look at reports while working through this article and find that they have no data in them, you didn't do anything wrong. The report in Figure 23 was created after monitoring the project's progress for several days. When you've just started, it's likely that the data warehousing snapshot collection hasn't run yet.

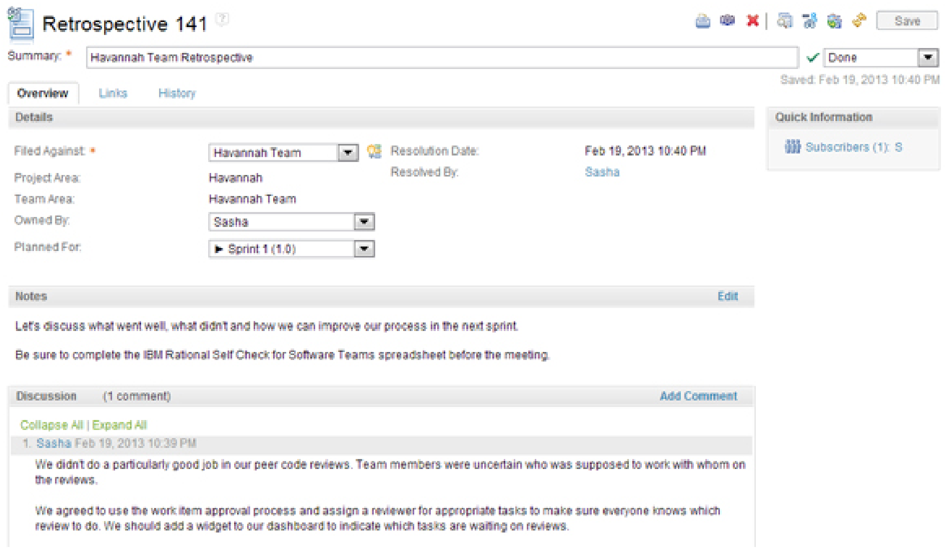
This process typically runs at midnight in the Jazz Team Server's time zone, so you need to have a server that's up all of the time for the process to run automatically. Snapshot collection can be forced from the web UI. However, this is not recommended in a production environment, because the process can be resource intensive and can affect user response during normal working hours.

Schedule the Sprint Review and Retrospective

Another important part of the scrum process is the **Sprint Review** meeting. The first part of this is the demo to the stakeholders. Using Rational Team Concert might not be part of this, because the point is to show off working software, not the list of tasks. However, it is important to capture feedback and comments from the review meeting, preferably in the sprint's Notes tab, as part of the sprint's history.

The next part of the Sprint Review meeting is the *Retrospective* (sometimes called *Reflection*). This is a chance for the team to discuss what went well, what didn't, and what they plan to do about it. Only team members should attend this meeting to keep the discussion as open as possible. The Scrum process template defines a Retrospective work item type (Figure 30) that you can use to make sure that the reflection meeting occurs and to track the team's comments and plans. A common approach is for the team to identify at least one practice to stop doing because it's not helping, at least one to keep doing because it is helping, and at least one to start doing to address an identified deficiency.

Figure 30. Work item for the retrospective



Retrospective 141

Summary: * Havannah Team Retrospective — Done

Saved: Feb 19, 2013 10:40 PM

Overview | Links | History

Details

Filed Against: *	Havannah Team	Resolution Date:	Feb 19, 2013 10:40 PM
Project Area:	Havannah	Resolved By:	Sasha
Team Area:	Havannah Team		
Owned By:	Sasha		
Planned For:	► Sprint 1 (1.0)		

Quick Information

Subscribers (1): S

Notes

Let's discuss what went well, what didn't and how we can improve our process in the next sprint.

Be sure to complete the IBM Rational Self Check for Software Teams spreadsheet before the meeting.

Discussion (1 comment)

1. Sasha Feb 19, 2013 10:39 PM

We didn't do a particularly good job in our peer code reviews. Team members were uncertain who was supposed to work with whom on the reviews.

We agreed to use the work item approval process and assign a reviewer for appropriate tasks to make sure everyone knows which review to do. We should add a widget to our dashboard to indicate which tasks are waiting on reviews.

Do it over again for the next sprint

The life of a healthy scrum team is one filled with a rhythm of success. Plan for a bit, work a while, deliver, reflect, repeat.

You have now finished your first sprint, so it's time for the next.

1. Create a plan for Sprint 2. Proceed the same way that you did when you created the plan for Sprint 1. Working with the product owner, document the sprint goal on the Notes tab.
2. Then start adding items to this sprint, using the same approach as for Sprint 1.

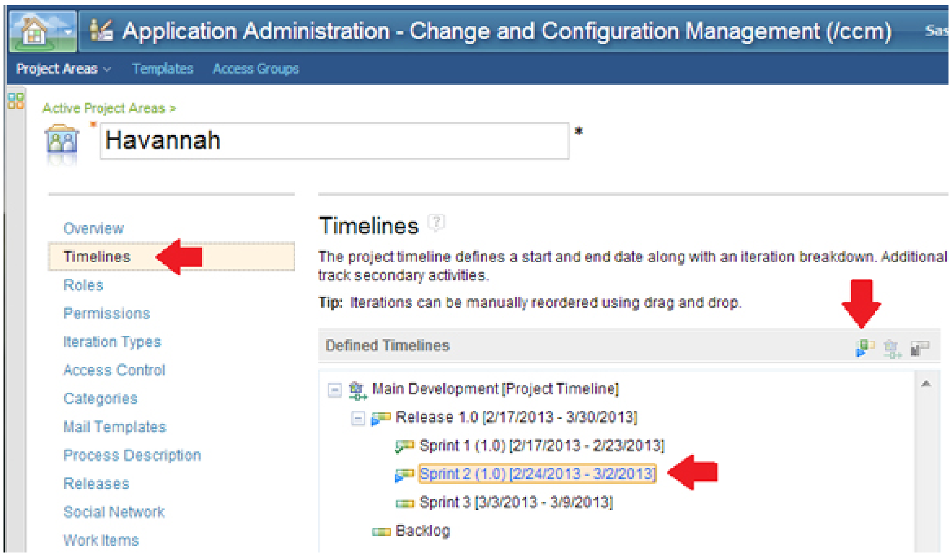
If a Story is not completed in the current sprint, there are various practices to solve this problem. One option is to schedule it for the new sprint (just automatically move the work forward, assuming that it is still the next most important thing to do). Some teams put it on the backlog. Maybe it comes back into the next sprint, or maybe priorities have changed and it will be delayed.

Set the next sprint as current

Even though you created dates for the sprints, Rational Team Concert does not automatically shift to make the *next* sprint the current one just because time has marched forward. You must manually adjust which sprint is considered current.

1. Using the Tools menu, manage the Havannah project. (You must be logged in as the project area administrator, scrum master or product owner. In this case, either Sasha or Frank.) This opens the **Application Administration – Change and Configuration Management (/ccm)** window (the same one that you used to add users to the project much earlier).
2. Select the **Timelines** section.
3. Open the timelines until you find the iteration that you want to make the current one. For this example, select **Sprint 2**.
4. Click the current iteration icon in the upper right of the window (see Figure 31). Sprint 1 will get a check mark to indicate that it is finished, and the current iteration mark will move to Sprint 2.
5. **Save** your changes.

Figure 31. Setting the current iteration


Application Administration - Change and Configuration Management (/ccm)

Project Areas Templates Access Groups

Active Project Areas >

Havannah

Overview
Timelines
Roles
Permissions
Iteration Types
Access Control
Categories
Mail Templates
Process Description
Releases
Social Network
Work Items

Timelines

The project timeline defines a start and end date along with an iteration breakdown. Additional track secondary activities.

Tip: Iterations can be manually reordered using drag and drop.

Defined Timelines

Main Development [Project Timeline]
Release 1.0 [2/17/2013 - 3/30/2013]
Sprint 1 (1.0) [2/17/2013 - 2/23/2013]
Sprint 2 (1.0) [2/24/2013 - 3/2/2013]
Sprint 3 [3/3/2013 - 3/9/2013]
Backlog

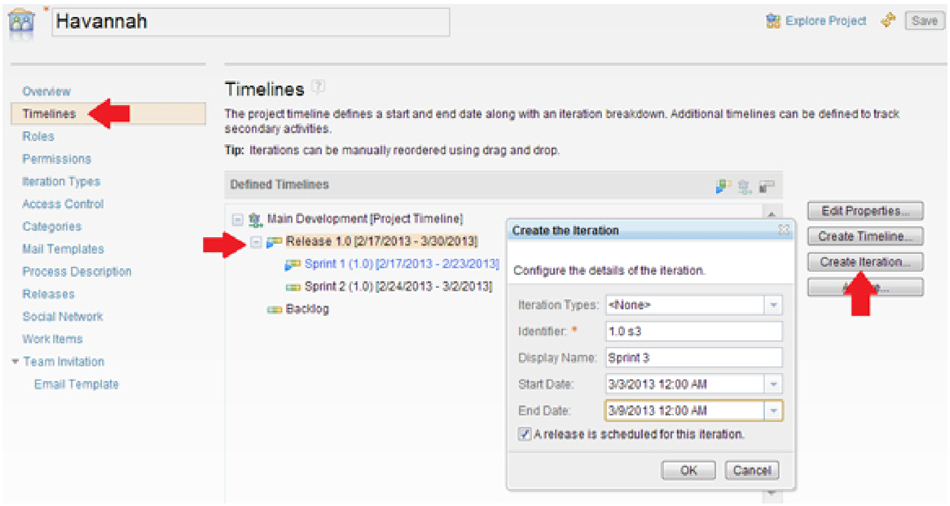
Add more iterations

When you need to add a new iteration for the project, just follow these steps:

1. Open the Application Administration window as described in the first step above.
2. Click the **Timelines** link to display the timelines.
3. Expand the timeline until you see the iteration list.
4. Select the parent iteration, under which you'll create the new iteration.

5. Click the **Create Iteration** button.
6. Enter an identifier (the usual syntax is the release ID, the letter `s`, and the sprint number, for example: `1.0 s3`. All that matters is that it is unique.
7. Enter a display name, adjust the dates, and click **OK**.
8. **Save** the project area changes.

Figure 32. Creating an iteration



Timelines

The project timeline defines a start and end date along with an iteration breakdown. Additional timelines can be defined to track secondary activities.

Tip: Iterations can be manually reordered using drag and drop.

Defined Timelines

- Main Development [Project Timeline]
  - Release 1.0 [2/17/2013 - 3/30/2013]
    - Sprint 1 (1.0) [2/17/2013 - 2/23/2013]
    - Sprint 2 (1.0) [2/24/2013 - 3/2/2013]
  - Backlog

Create the Iteration

Configure the details of the iteration.

Iteration Types: <None>
Identifier: * 1.0 s3
Display Name: Sprint 3
Start Date: 3/3/2013 12:00 AM
End Date: 3/9/2013 12:00 AM
[✓] A release is scheduled for this iteration.

Even though you've covered a lot of ground in this exercise, there's so much more to Rational Team Concert and Collaborative Lifecycle Management than scrum-based agile planning (and there's even more to agile planning than has been described here). For starters:

- There are many more useful things that you can do with dashboards. They can be configured with enough information to help everyone know where the project stands.
- Consider using Jazz source control and the Jazz build engine to power your continuous integration efforts.
- There is an Eclipse client integration that provides work item and source control integration. Planning features are available in the Eclipse client, but the web UI is generally preferable for planning tasks (and that is where new planning development happens).
- There are also integrations for Microsoft Visual Studio.
- IBM® Rational® Requirements Composer is a complete requirements management application that can link requirements with the Stories in Rational Team Concert that deliver the requirements.
- IBM® Rational® Quality Manager is a comprehensive test planning, execution, and reporting application that can create defects in Rational Team Concert for tests that fail.

Acknowledgements

Many thanks to Kohji Ohsawa and Stephanie Bagot for their technical reviews and suggestions. Thanks to Mike Cohn for permission to use the Havannah example.

Downloads

Description	Name	Size
Sample files	Havannah_sample_data.zip	125KB

Resources

Learn

- Check these resources for more information related to this article:
  - Get Ready to Sprint with Rational Team Concert: Keep your agile team on track and productive by focusing on stories they can finish, also by Millard Ellingsworth (IBM developerWorks, December 2012).
  - Progressive refinement of user stories in the Product Backlog by Mike Cohn (developerWorks, January 2013).
  - Browse other Rational software articles about agile development and check the Agile transformation section of developerWorks.
  - Learn more about scrum project management on the Scrum Alliance website
  - Learn more from Mike Cohn's books, *Agile Estimating and Planning* (Prentice Hall, 2005) and *User Stories Applied: For Agile Software Development* (Addison-Wesley Professional, 2004) or on his website.
- Find out more about Rational Team Concert:
  - Find Rational Team Concert articles and links to many other resources on IBM developerWorks, and check the product overview page, features and benefits, system requirements, and the user information center.
  - Check the Rational Team Concert page on Jazz.net.
  - Watch the Using Rational Team Concert in a globally distributed team webcast or a demonstration of the Dashboards and reports, or listen to the podcast about IBM Rational Team Concert and Jazz.
- Explore the Rational software area on developerWorks for technical resources, best practices, and information about Rational collaborative and integrated solutions for software and systems delivery.
- Improve your skills. Check the Rational training and certification catalog, which includes many types of courses on a wide range of topics. You can take some of them anywhere, anytime, and many of the Getting Started ones are free.

Get products and technologies

- Download Rational Team Concert from Jazz.net (site access requires registration, no charge). You can try it **free** on up to 10 users for as long as you want (requires registration). If you'd prefer, you can try it in the sandbox instead, without installing it on your own system.
- Evaluate IBM software in the way that suits you best: Download it for a trial, try it online, use it in a cloud environment, or spend a few hours in the SOA Sandbox learning how to implement service-oriented architecture efficiently.

Discuss

- Participate in the forum at Jazz.net (requires registration, which is free). This is also where you can enter and review enhancement requests and bug reports.
- Get connected with your peers and keep up on the latest information in the Rational community.

- Rate or review Rational software. It's quick and easy.
- Share your knowledge and help others who use Rational software by writing a developerWorks article. Find out what makes a good developerWorks article and how to proceed.
- Follow Rational software on Facebook, Twitter (@ibmrational), and YouTube, and add your comments and requests.
- Ask and answer questions and increase your expertise when you get involved in the Rational forums, cafés, and wikis.

About the author

Millard Ellingsworth

Millard Ellingsworth lives in the hills west of Portland, Oregon, where he works on developing the IBM Rational Collaborative Lifecycle Management community, improving how teams work together to build software that matters. During the small pockets of free time that leaves him, he divides his attention between playing golf, noodling on the guitar, woodworking, and tinkering with Android development. You can follow him on Twitter as @millard3 and on Google+.