Jira Workflows for Business Teams
Rachel Wright - jirastrategy.com

1. Introduction

Earlier today I removed 28 unused workflows from a Jira application. How do you end up with that many workflows people aren’t even using? How does that happen?

At one company, project leads were constantly asking for new custom workflows in hopes they would more accurately reflect their team’s real-life process. As new workflows were built, the old ones were disassociated with their projects and left to rot as inactive Jira resources.

At another company, business teams were stuck using the workflows built for software development teams! Even the computer helpdesk team had a *dev* workflow which didn’t fit their *support* process at all.

I’m teaching this class so you can build smart workflows that your business teams will actually use. Well designed workflows should compliment a team’s process, NOT dictate it!

We’ll cover the following topics:

* First, we’ll review the different types of workflows, their limitations, and how business workflows are different from software workflows. I’ll show you some samples you can adapt to fit the needs of your company.
* Next, we’ll talk about two commonly confused statuses: resolved and closed. I’ll share a mistake I’ve made and seen at other companies.
* After, we’ll cover best practices, like: good naming, appropriate use of transition behaviors, and the importance of roles and groups.
* Then, we’ll discuss planning for a custom workflow using a phased approach. We’ll interview a business team member to understand their process and think about how to best represent it as a Jira workflow.
* Then, we’ll build the sample workflow while discussing important concepts like: editing tools, drafts, transitions, and transition behaviors.
* Next, we’ll talk about using add-ons to extend workflow functionality. I’ll share some of my favorite plugins that I just can’t live without.
* Finally, I’ll share additional resources and downloads so you can continue learning to build great workflows. Don’t miss the course activity where you’ll design a custom workflow that you can compare with mine.

Alright, let’s get started!

1. Workflow Types

A workflow is a standard set of statuses and transitions that each issue follows through in its lifecycle. Statuses are simply steps and transitions represent the movement between steps.

Statuses take an idea from "conception" to "completion" or a task from "to do" to "done." Each project can have its own workflow and each issue type within a project can have its own workflow as well. For example:

* A “Task” issue type might require a very simple workflow, with simple statues like "Open", "In Progress" and "Closed."
* A Human Resources project might have an “Onboarding” issue type that requires additional statuses for steps that occur in the new hire process.

This is a vanilla Jira Server instance. If you’re using Jira Cloud, your view will look a little different, but the concepts are the same. Let’s look at the workflows automatically added when I installed Jira and created a sample project.

*After logging in as an administrator: Click the “cog” icon at the top right, select the “Issues” link, and click the “Workflow” link in the left sidebar.*

This “Workflows” admin page shows all active workflows assigned to projects, and inactive workflows not in use.

The first workflow was automatically created by Jira when I created a project using a template. Jira has templates for Business teams and Software teams. Let’s take a look.

*In the main navigation, click “Projects” then “Create project” to see the project templates. Choose a project type and click the “Next” button to see a preview of the default workflow for that type.*

When you create a new project using a template, Jira automatically creates the default workflow and creates any needed statuses.

Let’s explore the default workflow templates.

Jira Server and Jira Cloud both come with templates to help you get started building new projects. These templates include all the needed project elements including a Workflow and Workflow Scheme.

Jira also comes with two additional workflows: one called “Jira (Read-only System Workflow)” and one called “classic default workflow.” You can see in the screenshot that the “read only” one can’t be edited. Let’s take a closer look at it.

Here’s the “read only” workflow in diagram mode. There are 5 statuses: Open, In Progress, Resolved, Closed, and Reopened.

Here’s the same workflow in text mode. Text mode shows the transitions between statuses. For example, in the “Open” status:

* you can click the “Start Progress” transition to move to the “In Progress” status,
* you can click “Resolve Issue” transition to skip to the “Resolved” status, or
* you can click “Close Issue” transition to skip all the way to the final “Closed” status.

We’ll talk more about transitions later.

This workflow was built for software teams. It also restricts who can resolve and close issues. You’ll want to use a different type of workflow for business teams.

Before I became a Jira administrator, I was a web developer using Jira to track software tasks. I’m very familiar with the software development life cycle, and no matter what your development process looks like, software workflows tend to look fairly similar to each other.

In general it goes like this this: You receive a request, review the requirements, write the code, test the code, and deploy the code.

Business workflows however tend to be very different depending on the team and type of work to accomplish. Let’s address some example differences.

**Business Workflow Differences**

Business workflows sometimes require extra steps to handle things like: approvals, research, analysis, and dependencies from other parties. Sometimes a workflow is *expected* to move forwards and backwards multiple times. A purchase negotiation is an example. A vendor may propose a cost and your company may submit a counter proposal until a final cost is agreed upon. This could last multiple rounds.

Also, a business workflow might be conditional on other factors. For example, the Finance department might have a specific process for paying a vendor invoice but an abbreviated process for reimbursing an employee. The end result is the same (everyone gets paid) but the initial steps needed may vary based on the situation. The business workflow needs to support all these possibilities but in the simplest way possible.

Let’s look at some business workflow examples.

This is a simple Legal workflow built in Jira Cloud. In this example, the Legal team receives a request to sign a contract. They review the contract, sign it, and close the issue. Easy!

This next workflow shows a sample process for managing a tangible asset, like a computer. In this example, a user requests a new computer by creating a new issue. Once the request is approved, the machine is ordered. The order goes through the purchase, receipt, and configuration process. When the user receives the new computer, the record for the old computer is manually updated to indicate it’s been retired.

As you can see, you can use:

* the default workflows,
* the project template workflows, or
* you can create your own custom one from scratch.

You can also import sample workflows from the Atlassian Marketplace at: [marketplace.atlassian.com](http://marketplace.atlassian.com).

Next, we’ll talk about a common mistake regarding two confusing default statuses.

1. Resolved ≄ Closed

See video.

1. Do’s and Don’ts

In this lesson, we’ll talk about good things to do when creating your own workflows and also what not to do. Let’s stat with the Do’s.

**DO**

Naming in very important in Jira. Smart naming of workflows, statuses, and transitions ensure concepts are clear for your users and things are easy to find for administrators.

* You should name your workflow to describe the type of life cycle process it supports, not the project that uses it. Example: “Task Lifecycle” or “Onboarding” is better than “HR Workflow.”
	+ You’ll want to create workflows that are sharable, not one workflow per Jira project.
* A Status name should be short and reflect a current state in time.
	+ Long, multi-word names are harder to query and may be truncated on certain screens.
	+ Good status names immediately tell a user what is occurring and where they are in the process.  For example, "Pending Review", "In Review", or "Awaiting Review".
* A Transition name should be short and reflect an action taken.
	+ Good transition names immediately tell a user what action to perform to progress an issue.
		- For example, for an issue in "Pending Review" status, a good transition name would be:  "Review Complete."
	+ Bad transition names confuse the user about how to move forward.
		- A bad example is the word: "Review."  A transition button should signify the start or end of an action.  The word "Review" is ambiguous.  If a user clicks "Review," does that mean they should start a review or that the review has already occurred? I don’t know and your users won’t know either!
* Have you ever worked on a project that didn’t go as planned? Have you ever had to start a task over? Be sure to factor these occurrences into your workflow. Give users options to abandon or stop progress on issues at appropriate times.
	+ For example, if a request is no longer needed, give users the ability to jump all the way to the final “Closed” status. Don’t make them click through every step in the workflow to close an issue.
* Give users appropriate options to fix improperly transitioned issues too.
	+ For example, include a "reopen" transition, in the final status, to fix issues that were accidentally or improperly closed.
* Next, use transition restrictions sparingly. Do you really need to restrict who can do what? Jira records who clicked what transition when, so in most cases this is enough of an audit trail – no need to make it more complicated.
* But if you need to restrict an action, always use project roles or Jira groups.
	+ In the screenshot example, this transition can only be clicked by a Jira application admin or administrators in the project.
	+ Using roles and groups makes your workflow more maintainable in the future.
	+ Avoid specifying names of individual users in Transition behaviors.

In the beginning, keep workflows as simple as possible, until you've uncovered a deficiency or process step that needs special attention.

Now let’s talk about what not to do.

**DON’T**

* Don’t use more statuses or transitions than are needed. You only want to add a workflow step for statuses that will be queried or reported on.
	+ Not every little “thing” to do needs a status in the workflow.
* Don’t use the "Closed" status before an issue is in its final state. "Closed" should indicate no remaining work is needed.
* Also, don’t create duplicate statuses that also mean “Closed”. The default “Closed” or the business friendly “Done” status is just fine! You don’t also need to create a “Completed” status!
* Don’t create statuses like "Abandoned" or "Rejected". Instead, use the "Resolution" field to indicate the "how" or "why" an issue is in the "Closed" status.
* Don’t create "temporary" or "dead" statuses where issues are likely to sit for an indefinite amount of time.
	+ One example I see often is the "On Hold” status. A status like this is only useful if someone regularly reviews issues in this state.
* Don’t create ultra-specific status names or use the wrong word tense.
	+ For example, don’t use "Pending Review by Marketing" or "Pending Review by John." These are too specific and make workflows hard to maintain and share between projects.
	+ Another bad example is a status named "Reviewed." A word in past tense is a "dead end" and doesn't tell the user what needs to happen next.
* Finally, avoid creating illogical workflows. Consider the example pictured.
	+ At one company, their first status was "In Progress." This communicates to the reporter that the minute their issue is created, someone is working on it.
	+ Of course that wasn’t really true. Normally, issues need to be triaged, prioritized, approved, or even reviewed for understanding first.
	+ Instead, the first status should be "Open,” "To Do," or something similar.
	+ The first status should signify that the issue was received, but no action has been taken yet.

Please join me for the next lesson, which is all about custom workflow planning. We’ll interview a business team member so we can understand their process and think about the best ways to represent it in Jira.

1. Custom Workflow Planning

Let’s talk about the planning you should do before you create a custom workflow in the Jira application. That’s right, log out of Jira because building a custom workflow should always start on paper!

It's certainly possible to capture every little step in your work process and build that into a complex and long Jira workflow.  An alternative however, is a phased approach.  Simply break your process into phases that represent a collection of smaller steps.  The phases represent key decision points. An issue can’t be moved to another phase until the requirements of that phase have been satisfied.

That way each *phase* represents a status in Jira and all the small steps in a phase don’t need to be also be statuses.

For example, imagine your company is signing a partnership agreement with anther company.

The process requires a review of the contract by both parties and potential edits before final execution. It’s a predictable process requiring a short workflow like:

*Open > In Review > In Execution > Closed*

Even though it’s a simple workflow, the Legal team is doing many things in the background that don’t need to be reflected in the workflow. For example:

* In the “In Review” phase, the Legal team is reviewing the contract, researching legal topics, communicating with internal teams, and negotiating terms with the external company.

	+ **TIP:** A generically named status like “In Review” is better than a legal-specific name like “In Contract Review”. This way, other Jira projects can use the generic status regardless of what type of *thing* needs review. You want to share assets and schemes between projects as much as possible.
* In the “In Execution” status, the CEO is finding his favorite signing pen, both companies are trading paperwork, and your Legal team is entering the final documents into their contracts database.

So what do you think? Is it useful to create a status for every step that occurs in the contracts process? Do you need to track how many times the contract was modified during the review process? Do you need to track which parties have signed the agreement so far? If the answer is “no” a phased approach may be more useful.

If you do need to track signatures a custom field might be smarter than a status. But only create that custom field if you’re going to report on that piece of information.

**TIP:** If you're not going to query for "all issues in a certain status", that’s a clue that the status may not be necessary or useful.

**Interview**

Before creating a new custom workflow, have a team member explain their real-life process to you.

This is Chris, the owner of a business strategy company. He’d like to start using Jira to track his consulting process. Hi Chris, tell me about your process.

|  |
| --- |
| **Consulting Process Narrative***“We start off with a search for candidates and identify ideal clients we’d like to work with. We acquire a lead through various lead generation methods. We do company research prior to meeting with the candidate. We book a meeting. At the meeting, we use our “Fact Finding Template” and “Performance Checklist.” After the meeting we develop a solution, plan, or resources for how the consultant can help the company. For ongoing contracts, we’ll have weekly, quarterly, semi-annual, and annual meetings for their quarterly and long-term goals.”* |

Thanks Chris. I’ll be back in touch with a draft workflow.

Custom Workflow Process

Let’s use the information we collected in the previous interview, to represent the process as a Jira workflow.

First, write the process out in words. This can uncover additional needs you may have neglected to consider.

Second, split the process narrative into logical phases to determine the statuses. Identify the steps or high-level phases an issue must go through in its lifecycle.

In the example, I’ve split the process into 4 phases and given each a short status name: Open, Planning, Proposal, and Active. We also need a “Closed” status for when a lead becomes inactive.

Next, draw the statuses in a flow chart. In the example:

A lead is acquired and on creation is in the “Open” status. Research and meetings occur in the “Planning” status. Solutions are determined and pitched in the “Proposal” status. If the proposal is accepted, the issue moves to the “Active” status where all ongoing consulting occurs. When consulting is complete, the issues moves to the final “Closed” status.

Next, determine the forward transitions and add them to the flow chart. In the example:

A lead is acquired and on creation is in the “Open” status. The user clicks the “Start Planning” transition button to move to the “Planning” status where all research and meetings occur. The user clicks the “Start Proposal” button to determine solutions and pitch them to the customer. If the proposal is accepted, the user clicks the “Start Consulting” button to move to the “Active” status where all ongoing consulting occurs. When consulting is complete, the user clicks the “Close” button to move to the final “Closed” status.

After, determine the needed backwards and alternate transitions. In the example:

A lead is acquired and on creation is in the “Open” status. The user clicks the “Start Planning” transition button to move to the “Planning” status. If no planning is needed, the user clicks “Skip to Proposal” to skip to the “Proposal” status. In the “Planning” status, the user clicks “Start Proposal” to move forward to the “Proposal” status or clicks “Back to Open” to move backwards to the “Open” status. In the “Proposal” status, the user clicks “Start Consulting” if the proposal is accepted. If the proposal is declined, the user clicks the “Close” button to move to the final “Closed” status. In the “Active” status, the user clicks the “Close” button when consulting is complete. In the “Closed” status, a user clicks the “Reopen” button to reopen the issue.

I’ve noted that the “Close” transition is global, with an asterisk.

Finally, determine and document the needed transition behaviors.

After we’ve determined everything needed for our custom workflow, it’s time to log in to Jira as an administrator and build it!

First, I’ll need to create any new statuses. Our workflow needs three: Planning, Proposal, and Active. The “Open” and “Closed” statuses already exist. Add new statuses sparingly and only when the existing statuses can’t be used.

Next, I’ll create a new workflow. I’ll name it “Leads” for the process it supports. I’ll use the description field to record the Jira request ID for this customization. That way I can reference requirements and implementation notes later.

Next, I’ll add all the needed steps and link them to statuses.

After, I’ll add all the needed forward, backward, skip, and global transitions.

I’ll need to enter “Diagram” mode to add the global “Closed” transition.

As you can see, a global “Closed” transition was added to every step. I don’t like the default name of “Closed” so I’ll edit it to read “Close” instead. Then, I’ll associate this action with a transition screen.

Now, I’m consulting my planning documentation to see what other transition behaviors to add. Here I’m adding a new post function and modifying an existing one.

And that’s it! Our custom workflow is built and ready for testing.

Stick around for the next lesson, to learn more about the workflow concepts in the previous demo.

1. Workflow Concepts

In this lesson, we’ll talk about important concepts that apply to any type of workflow.

**Tip:** If you’re already a certified Jira administrator, you can extend your certification by taking the ACB-110 “Advanced Jira Workflows” exam. Use the info in this lesson as additional study material.

Read more: [atlassian.com/university/certification/badges/exam-acb-110](https://www.atlassian.com/university/certification/badges/exam-acb-110)

There are two views for creating and editing workflows. On the left is a workflow shown in “Diagram Mode” and on the right is the same workflow in “Text Mode.” Each mode has pros and cons.

**Diagram Mode**

* Diagram mode is the simplest and easiest view for your users.
* This mode includes a “Workflow Designer” which lets you:
	+ add a status or transition,
	+ drag a status or transition around visually,
	+ edit transition properties and behaviors, and
	+ add global transitions.
		- A global transition allows any status to transition to a different status.
		- For example: I often create a global status called “Close” so an issue can skip straight to the “Closed” status from anywhere in the workflow.
		- You can only create global transitions in diagram mode.

**Text Mode**

* Text mode is considered more advanced.
* You can edit statuses and transitions plus work directly with the steps.
	+ For example, if I wanted the workflow to populate a field on the create action, I can do that by clicking the step name. Steps are only visible in text mode.
* I prefer text mode. I think the transitions are much easier to understand in table format.

Once a workflow is created, there are different states to be aware of: Active, Draft, and Inactive.

**Active**

An active workflow is currently used by one or more projects. You can’t delete an active workflow. To delete, you must make it *inactive* by reassigning any projects using it.

When you try to edit an active workflow, a draft is created. That way, there’s no impact to a project’s issues until the draft is published. When you publish, Jira will help you migrate existing issues to new statuses, if needed.

The screenshot shows I’m editing a draft of an active workflow.

**Draft**

There are some limitations to editing drafts though.

* First, you can’t change the workflow’s name.
	+ You can edit the description however.
* You can’t remove a status.
	+ You can set your transitions to point to a different status however. That means the status is still there but an issue won’t go through it during the workflow.
* You can’t add outgoing transitions to a status with no existing outgoing transitions.

**Tip:** If you’re planning to make major workflow adjustments, make a copy of the workflow and make changes to the copy instead. Then you’ll have no editing limitations.

**Tip:** Also, build or modify your workflow in a test environment first. Then, after you've verified all your changes, export the workflow and import it into your production environment. This prevents users getting spammed with notifications while you’re testing. Alternatively, you could temporarily assign your workflow to a production project built specifically for workflow testing.

**Tip:** Changing or renaming a status will break user filters and board mappings. This will impact user dashboards, boards, and reports.

**Inactive**

The last state is inactive. Inactive workflows aren’t in use by any projects and therefore have no limitations. You can edit them in any way or delete them.

Documentation: [jirastrategy.com/link/jira-workflow](http://www.jirastrategy.com/link/jira-workflow)

Next, I’d like to tell you about two incorrect workflow assumptions I made.

See video.

Now let’s discuss transitions, which are the buttons users click to move between statuses.

Users often confuse the standard issue buttons with the workflow transition buttons. This screenshot shows both.

* Because workflow transition buttons differ between projects and issue types, users aren't sure which button to click.
* Sometimes users mistakenly think the first workflow transition button is the issue's current status.
* Sometimes users don’t know that there can be extra transition options under the “Workflow” button.

Hopefully a little bit of user education can help avoid these issues.

**Tip:** Don't name a workflow transition button with the same name as a standard issue button. Otherwise, you'll have two "Assign" buttons or two "Comment" buttons, which will really confuse your users.

Now let’s talk about transition types.

They are: forward, backward, global, skip, and administrative. Let’s recall our sample legal workflow of:

*Open > In Review > In Execution > Closed*.

**Forward**

A forward transition is just like it sounds. It moves an issue forward in the workflow. I like to name forward transitions so users have an idea of what the next status is. For example, a good transition name between “Open” and “In Review” would be “Start Progress” or “Ready for Review.”

**Backward**

A backward transition moves an issue to a previous status in the workflow. I like to make this movement clear in the transition’s name. For example, to move backwards from the “In Review” to “Open”, I’d name the transition “Back to Open.”

**Global**

We talked about global transitions earlier. Use these to allow all statuses to transition to a different status. In the sample workflow, I could create one global transition to easily close an issue from anywhere in the workflow. Without this feature, I’d need to create three individual transitions to achieve the same effect.

**Skip**

You don’t just have to move forward or backwards in the workflow, you can also skip around if needed. In the sample, I could create a transition to move from the “Open” status directly to “In Execution.” I like to make this movement clear in the transition’s name as well. For example, I’d name the button “Skip to Execution.”

**Administrative**

Finally, you can also create transitions just for fixes or administrative use. There are a million possibilities, but here’s one example: you could create a transition that updates a custom field. Then you could use bulk tools to make the same update to many issues at once. The transition doesn’t even need to change the status.

If you don’t want general users clicking these special transitions, you can hide them with a transition behavior.

Speaking of transition behaviors, there are four types: triggers, conditions, validators, and post functions.

**Triggers**

Triggers help keep Jira issue data in sync with dev tools like: FishEye and Bitbucket. We won’t cover triggers in this course however.

Documentation: [jirastrategy.com/link/triggers](http://www.jirastrategy.com/link/triggers)

**Conditions**

A Condition checks whether a transition can be performed by a user.  For example, use a condition to only allow the reporter or current assignee to see the button. If a Condition is false, the transition button is hidden. This means a user may encounter an issue with no transition buttons available to them.

**Validators**

A Validator checks whether certain data exists before the transition occurs.  If a Validator is true, the transition succeeds.  If a Validator is false, the issue does not transition until the data is updated or returns true.

For example, use a validator to make sure a required field has a value or that a user has a certain permission.

**Post Functions**

A Post Function is an additional rule or action that occurs after the transition.  These only execute if the transition is successful.

For example, use a post function to automatically assign an issue to the reporter when moving to a “more info needed” type of status. Or, use a post function to clear the resolution field when an issue is reopened. You can see that post function labeled #1 in the screenshot.

Workflow behaviors are just the beginning! You can further extend your capabilities with plugins and add-ons. Join me in the next section for some personal recommendations.

Documentation: [jirastrategy.com/link/advanced-workflow-config](http://www.jirastrategy.com/link/advanced-workflow-config)

1. Plugins and Add-Ons

Many add-ons provide additional workflow functions. Some add-ons even come pre-installed in Cloud environments. You may need to enable them from your "Manage add-ons" admin page.

Visit [marketplace.atlassian.com](https://marketplace.atlassian.com/) to explore the possibilities. Most plugins come with a free trial that you can install from: *Admin > Add-ons > Find new add-ons*.

There are so many great plugins! Here are some of my favorites:

* I use “ScriptRunner for Jira” to maintain Jira as a whole, script transition behaviors, and modify behaviors of fields.
* I use “Create on Transition” and “Update on Transition” to reduce manual work, like creating the same child tasks or completing the same filed over and over again.
* I use “Jira Misc Workflow Extensions” to help with things users sometimes forget: like start progress on a parent issue when the child issue starts progress, or make to sure all child issues are closed before a parent issue is closed.
* I use the “Jira Toolkit Plugin” to show custom user instruction messages on transition screens.
* And finally, I use “Suite Utilities for Jira” to restrict transitions to multiple groups and roles, require a comment on transition, and update custom fields.

There are so many possibilities! I encourage you to test out the standard workflow behaviors and the additional behaviors provided by add-ons.

Don’t miss the last lesson where I’ll share an activity, a quiz to test your skills, and links to downloads.

1. Resources

I hope you enjoyed this workflows course as much as I enjoyed building it! By now you should have a good understanding of the different workflow types, how to design and build custom workflows, common mistakes to avoid, and how to enhance workflows with behaviors and add-ons.

Next, test what you learned with the included activity. You’ll design and build a custom workflow based on a description of a business team’s process. You can also take the course quiz!

Finally, you can download additional materials and continue the conversation using the links shown.

Remember, it’s always best to start out with a simple workflow and build more as it’s needed. Don’t over-complicate your workflow with steps and statuses you don’t really need. Your Jira application will be cleaner and your end users will thank you for it.

I’m Rachel Wright, author of the Jira Strategy Admin Workbook. Happy workflow building!

Need help cleaning-up or maintaining your Jira instance? Learn more at: jirastrategy.com