Creating Workflows in JIRA from a practical perspective.

Working as a consultant on JIRA and JIRA workflow, I have done some observations that I would like to share - hopefully brining some success into Your project and implementation.

Face the music (real life)

No workflow can be made on a piece of paper and be expected to work

No workflow can be made in JIRA (possibly from a piece of paper and be expected to work)

The only way to success in to get the hands dirty, sit down with the participants or "stockholders" - people with involvement and the daily participation in the processes and **communicate**

Train, test, change/ implement and repeat (A analoge to the The Cookbook principle for Changes and SOPs

Be the "Devils Advocate"

Ask questions, be opposite (in a nice maner) and ask again - "are this really needed or the way You actually do it?" No to be arrogant, but sometimes the thought of "what we do" and "what the hands (and other people) do" - are not equal.

KISS - Divide and Conquer

Break Workflow down into "pieces"

Keep Your workflows relative simple, and break them into sub-workflow handled by sub-issues or linked issues if possible. This has several advantages:

A. The overall administration and overview of the installation is simpler

B. Specialities, as different Workflow for same issuetype in 2 projects are easier to maintain.

C. Parallel processing can be done on sub- or linked tasks by several other Assignee's - whereas one large workflow is serial with one Assignee at the time.

Tasks or subtasks?

In JIRA, Parent and Childs have different Issuetypes, after my opinion rather stupid, it could just be a setting like - this Issuetype can be: 1) Parent Only 2) Parent And Child or 3) Child only....

For example; at my work we have Changes and (yes) - SubChanges - hence 2 IssueTypes to maintain...

Learning more and more, I tend to prefer linking over child (sub) issue types, as they are more useable for every purpose, and the binding is as "strong" as in Task/SubTask relations.

With the ScriptRunner for JIRA plugin and several of the JIRA Workflow Plugins - conditions, validations and post-functions are able to handle links to...

Have a loose view on restrictions and conditions

Also, take a loose a perspective as possible on restrictions and conditions, setting the workflow up with tight restrictions breaks agility in the real world, for example under illness and leave, issues can be stuck.

Remember, all changes are tracked, just don't let people have deletion rights.

The human factor

This is to bring up a question:

What "drives" a normal JIRA user to make a transition?

A. Assigning the Issue to another user (out of sight)

B. Close an issue (getting it done)

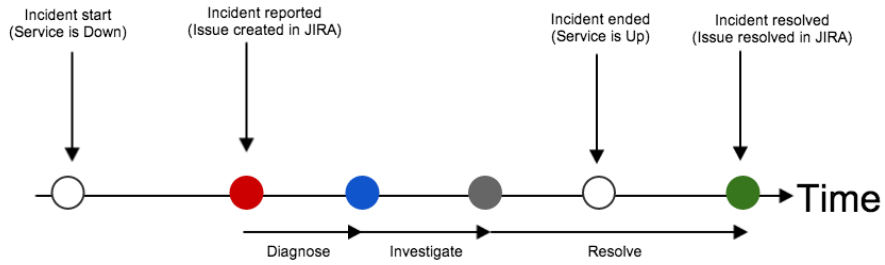
C. Sharing the Issue with another user (can be making a subtask for fulfillment)

D. Money (Bonus)

E. Serious goodwill

Most of the time, "A" and "B" is the common (and powerful) reason that drives the normal JIRA user.

One of the most common things I encounter, is the diversity between some one's reporting "needs" versus the number of steps in a workflow, lets take the good old "Incident Management" process (a small version), that could be:



Now, Support Managers want to log all times in all of the JIRA "phases" - hence 3 times:

- Diagnose done (Status going from Open ->**Under investigation**)
- Investigation Done (Status going from **Under investigation** -> **Under Resolution**)
- Resolved (Status going from **Under Resolution** -> **Resolved**)

In real life, if the JIRA User handling the Incident is that same, he will typically make the 2 last transitions (e.g. clicking the buttons) right after each other, or perhaps if this is late and he is tired, the next morning.

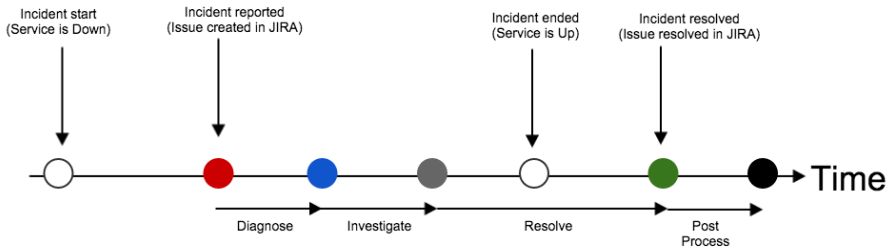
I have seen this all the time in real-life systems:

Users does not transist Issue unless one of the reasons above is the motivator for making the transition.

The solution is (in this case) 2 parts:

A. Only to make steps that triggers the drives/motivators mentioned above in the box.

B. Make a "post" function that collects/enter the correct times in additional fields. In the case above, the Service is down before the Incident is created, and up before the user resolves the issue, there times can be entered manually.



Hence, at least we need custom fields for "Incident start", "Incident End" fields for the white dot times, as the times on the Issue "Created" is the red dot, and "Resolved" is the green dot - these does not represent the real times.

The simple solution is to boil down a very ambitioned workflow with a lot of transitions to the needed ones, often identified by the fact that the assignee changes with each transition.

The use of Plugins and custom code

Workflow can be extended heavily with Plugins and custom code - but be aware this requires extra maintenance on the JIRA installation, and the risk of breaks are higher, getting raised for every extra plugin or code.

Is don't say, "don't use" - but "use with caution and keep everything updated".

If You are using plugins and custom code, do keep an eye on the catalina.out and atlassian-jira.log for stuff as:

- Java Stacktraces
- Jelly Script errors
- Groovy errror
- Custom Field errors

Using a Log tool such as splunk or similar

Documentation

This should go without saying; - documentation and training are always needed beside the "technical" part...