PharmaSUG 2018 - Paper LD02

Customized Project Tracking with SAS® and Jira™

Nancy Brucken, Syneos Health™

ABSTRACT

As programmers and statisticians, most of us are far better at programming tasks than we are at project management. Jira is a powerful and inexpensive commercially-available application designed to help programming teams track their progress on projects. It is built on top of a PostgreSQL database, which can be queried from SAS to generate a variety of custom reports.

INTRODUCTION

One of the hardest tasks that we as programmers face in our daily work is project management- planning out the work of the programming team, estimating timelines, determining the number of full-time equivalents (FTEs) required to meet those timelines, and answering questions from our management on whether the project is on schedule. Many of us are far happier programming a complicated dataset than trying to determine whether we can complete the remaining project work before the deadline with the FTEs currently assigned to the project.

Jira, from Atlassian, is a project management application built for use by programming teams. It allows you to customize a workflow to fit your work processes, automatically notifies you via email when a task has been assigned to you, and keeps the entire history of your program development and validation in a single location. It comes bundled with a package of default reports, but since the underlying data is stored in a PostgreSQL database, it is possible to retrieve that information using SAS, and develop customized reports to meet your company needs.

AGILE DEVELOPMENT

A full description of agile software development is outside the scope of this paper. Please see the references at the end for more details. However, some basic concepts are helpful for describing how the process works, and how Jira can be implemented to support the project work. One of the hallmarks of agile development is the division of work into increments, or *sprints*, which are short timeframes, generally no longer than a 1-2 weeks, during which time the programming team commits to completing a certain amount of work. Sprints consist of *issues*, which are tasks, such as programming an ADSL dataset or writing the code to generate a set of demographics tables. The *user story* is a description of what the task is expected to produce, and the *story points* represent an estimate of the amount of time that will be required to complete the task.

PROJECT WORKFLOW

The first step in setting up a project in Jira is to define the types of issues that will be tracked, and the workflow for each of those issues. For example, a typical program would be pulled from the to-do stack, created by the original programmer, sent to the validator, passed from there to statistical QC, and then on to a senior reviewer, before the output generated is delivered to the sponsor.

A diagram of this workflow might look something like this:

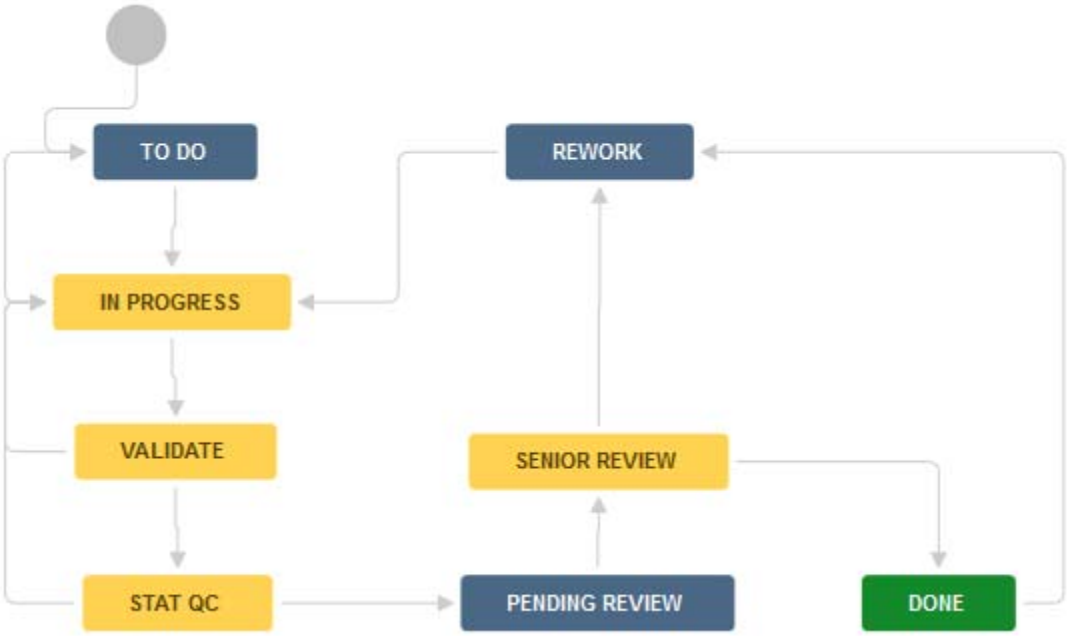

Figure 1. Sample Workflow

At each step, work is performed on the issue, which is then either completed and sent on to the next stage, or sent back to a previous step for additional changes. Each time the issue is moved between stages, an email notification is automatically sent to the person it is then assigned to, along with anyone else on the project team who is monitoring that issue.

ISSUES

An issue is simply a unit of work to be tracked. Issues can represent the creation of SDTM domain or analysis dataset specifications, programs creating a dataset or TLFs, or a task such as developing analysis results metadata for a series of TLFs. Creation of an issue in Jira involves populating the requisite information on the issue screen:

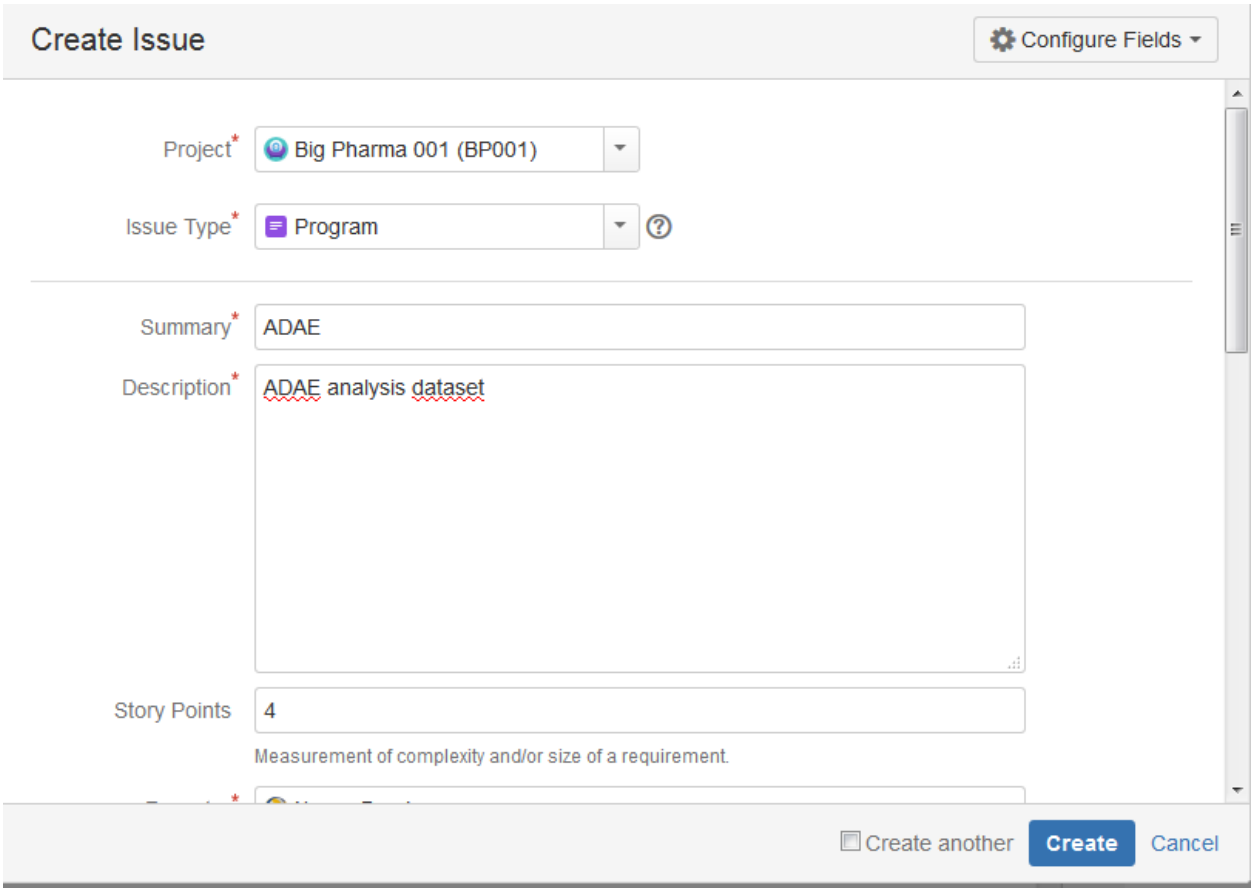

Figure 2. Sample Issue Screen

The issue screen is customizable, so you can add additional fields, such as the name of the programmer to whom this issue will initially be assigned, or the names of the deliverables that include the issue. All of this information is stored in a single PostgreSQL table, JiraIssue.

The Summary and Description text are searchable. We have included appendix numbers (when known) and TLF titles in the descriptions of issues representing TLF programs, which helps to identify which TLF program/issue is associated with a given table or listing output.

Notice the Story Points entry at the bottom of the screen. Story points are used to represent the amount of time required to complete an issue. In the case of a dataset program, that would include programming, validation and statistical QC of the dataset. In the example above, it was felt that the ADAE program would take less than 2 hours each for programming and validation, leaving the balance for QC.

RELEASES

Once all of the issues for a project have been created, the next step is to group them into releases. We have defined a release to represent a deliverable. Examples of a release for a study might be DMC #1, Dry Run, Blinded Data Review, Draft Topline, Draft Non-Topline, etc. This approach allows us to easily track our progress towards completion of all of the issues for that deliverable.

Defining a release is simply a matter of filling in a few boxes on the Release screen:

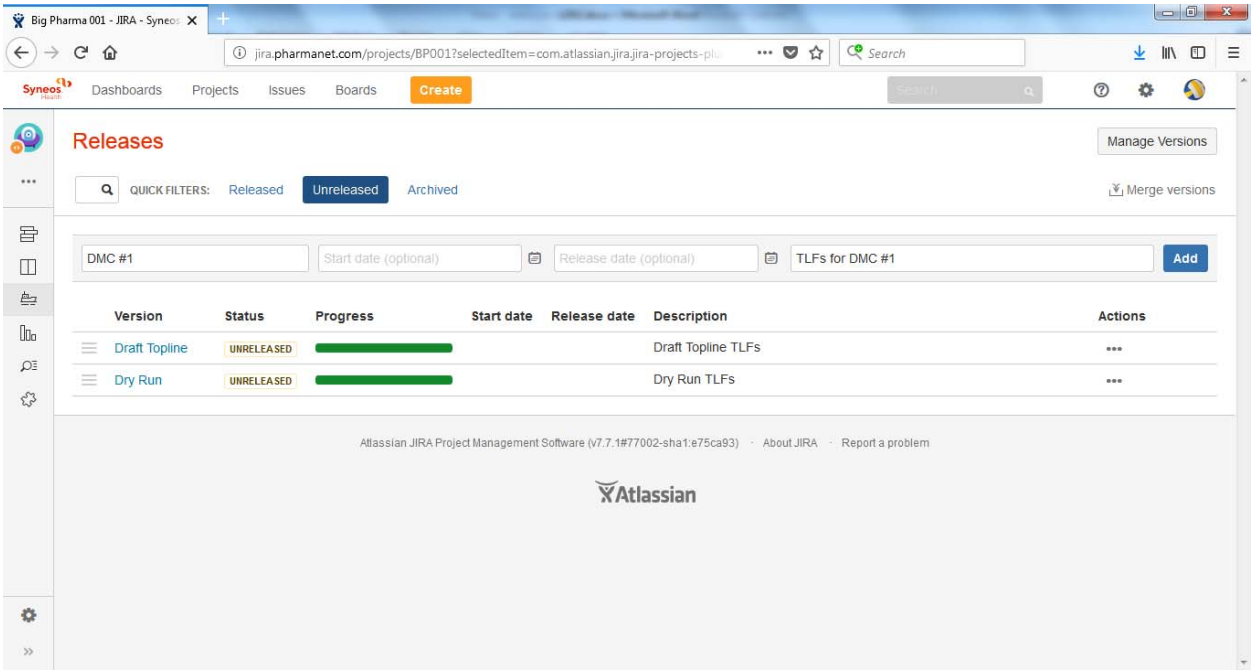

Figure 3. Sample Release Screen

Once the releases are defined, the issues can be associated with their corresponding releases. Note that an issue can be assigned to multiple releases. For example, a program can be assigned to a DMC, a Dry Run, and a Topline delivery, and will be tracked with each release that it is associated with.

SPRINTS

Once all of the issues for a project have been assigned to releases, the next step is to group them into sprints associated with each release/deliverable. A sprint represents the work that the project team commits to completing within a certain timeframe, generally no more than 1-2 weeks. Thus, a given sprint often consists of a mixture of specifications, dataset programs and TLF programs, since that represents the work to be done within that time period in order to keep the project on track.

The process of assigning issues to sprints serves as an initial indicator of whether a project can realistically be completed within the required amount of time using the programming and statistical resources initially identified. Jira tracks the total number of story points assigned to each sprint, so given the length of the sprint and the number of available FTEs, it is possible to determine when a project is doomed from the beginning.

The end of each sprint also serves as a project checkpoint, since when a sprint is closed with unfinished issues remaining, Jira prompts for a decision on whether to send those issues back to the project backlog, or cascade them into the next scheduled sprint. If sent to the backlog, they will eventually need to be dealt with; if moved into the next sprint, they will affect the scope of that sprint, and may cause ripple effects throughout the rest of the project.

SPRINT TRACKING

Daily project activities and progress tracking are conducted on the sprint board. The board displays all of the issues assigned to the sprint, along with the current status of each of those issues, and who is assigned to work on each one. A predefined filter allows you to view only the activities assigned to you, and additional customized filters can be added. Here is an example of a sprint board for a project:

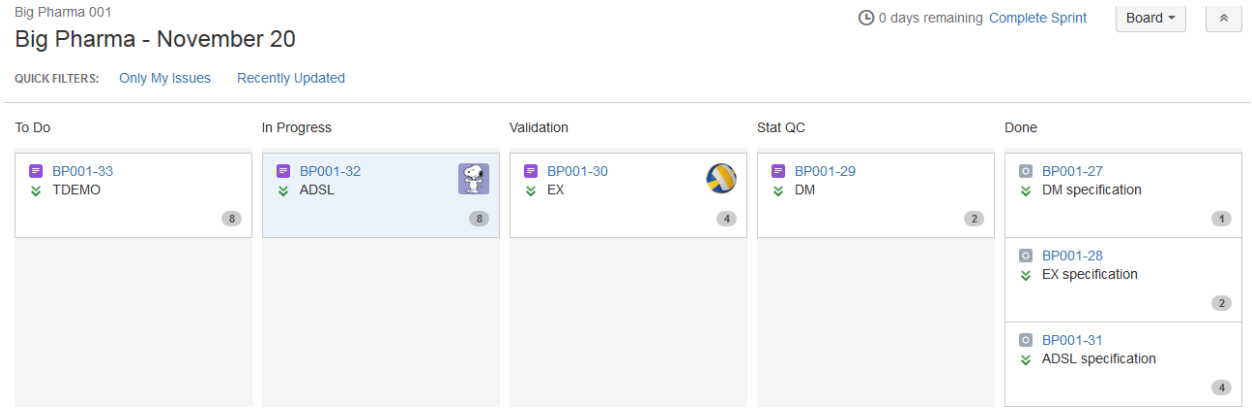

Figure 4. Sample Sprint Board

In this example, 3 issues have been completed, the DM program is awaiting assignment in Stat QC, EX is being validated, ADSL is being programmed, and the TDEMO program is ready for someone on the project team to pick up and start working on.

As the work progresses, the issue screen serves as the repository for storing all communication regarding that issue. This screen is used for assigning responsibility for the issue, logging any comments regarding the issue, including validation and QC findings, along with requested specifications changes, and moving the issue between stages. Here is an example of the comments that were recorded for the DM programming issue:

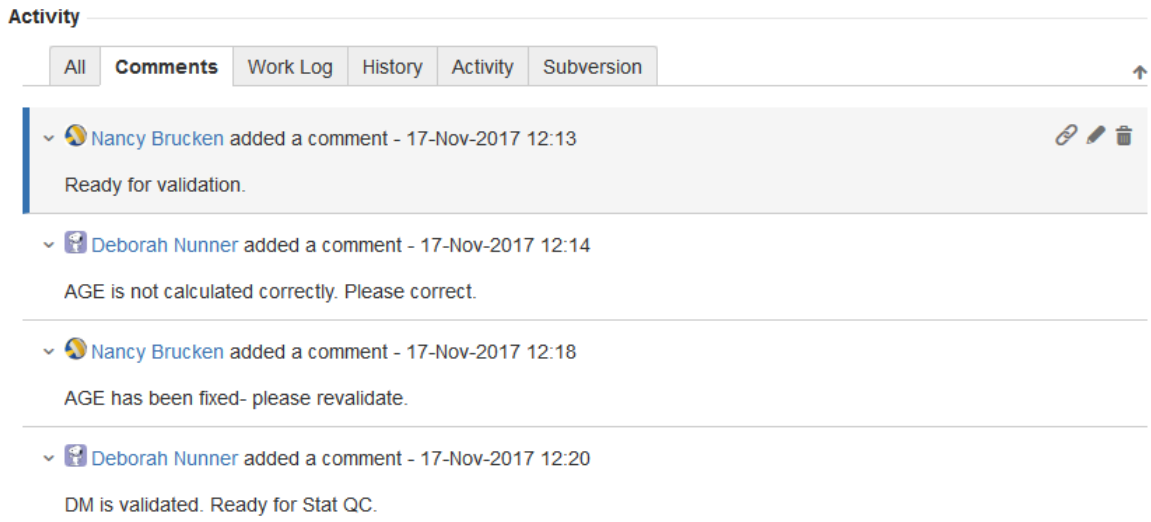

Figure 5. Sample issue comments

Jira also allows screenshots to be pasted into the comments, as well as for the uploading of associated files. The use of screenshots can improve communication between primary and validation programmers by clearly indicating where discrepancies lie, and the underlying data that have led to the results being generated.

Each exchange of comments, as well as a change in the status of an issue, triggers an email to everyone who has worked on that issue. This allows for automatic notification of the validator, for example, once a validation finding has been corrected and the issue returned. In addition, all of the status changes, assignments and comments are automatically logged, and stored in the underlying database ChangeGroup, ChangeItem and JiraAction tables. The presence of this audit trail has enabled us to do away with our validation comment tracking spreadsheets.

PROGRESS REPORTS

Jira comes bundled with a variety of pre-defined reports. A burndown chart shows graphically how much work still remains for each sprint. A velocity chart shows how much has been completed for each sprint, and several other types of reports display the amount of time it has taken to complete different types of issues. In addition, since we have the ability to access the underlying PostgreSQL database with SAS, we can develop customized reports.

One such report that we have created, a cycle report, displays the number of times that an issue has cycled between programming and validation, as well as between statistical QC and programming, along with a list of the programmers and statisticians who have worked on that issue. That has enabled management to focus quickly on the types of issues causing delays on a project, and determine strategies for resolving them now, as well as for preventing them on future projects. Here is a sample cycle report:

Syneos Health

Programming Cycles by Project and Program

		Cycles		
Project	**Program**	**In Progress to Validation**	**Stat QC to In Progress**	**All Prog/Stat**
Big Pharma 01	ADAE	4	2	Pam Programmer, Vic Validator
Big Pharma 01	ADEFF	2	0	Pam Programmer, Vic Validator
Big Pharma 01	ADJRQLQ	5	2	Pam Programmer, Vic Validator
Big Pharma 01	ADSL	10	6	Pam Programmer, Vic Validator
Big Pharma 01	ADTTE	6	3	Pam Programmer, Vic Validator
Big Pharma 01	GCIDISC	1	0	Pam Programmer, Vic Validator, Sam Statistician
Big Pharma 01	GCIPLOT	2	1	Pam Programmer, Pete Programmer, Vic Validator

Figure 6. Sample Cycle Report

In this case, ADAE went back and forth between the programmer and validator 4 times, and was sent back from QC twice. This cycling may indicate a lack of understanding on the part of the programmer and validator over what it was supposed to contain, ambiguous or incomplete specifications, or simply changes in the TLFs which required adjustments to ADSL after the initial delivery and sponsor review. The latter may trigger creation of a change order.

Accessing the underlying PostgreSQL tables requires the SAS/ACCESS to ODBC engine. The LIBNAME statement looks like:

```
LIBNAME jira ODBC DATASRC = '<ODBC identifier for PostgreSQL database>'
                  SCHEMA = public PRESERVE_TAB_NAMES=yes;
```

Several of the PostgreSQL tables contributed to the cycle report. The following PROC SQL query was used to identify the specific project in our implementation of Jira:

```
create table projcat as
    select p.id, p.pname, p.pkey, pc.cname as category
      from jira.project p, jira.nodeassociation na, jira.projectcategory pc
      where p.id=na.source_node_id
        and na.sink_node_id=pc.id
        and pc.cname='Biostats' and p.pkey="&project_key";
```

In the code above, the PROJECT table contains the type, name and identifier of each project created in Jira. The PROJECTCATEGORY table contains the project categories, and allows us to subset to Biostats projects only. The NODEASSOCIATION table serves as a link between the two tables.

Once we had the internal project identifier for the project, we used it to generate a list of all of the issues and status changes for the project:

```
create table projchg as
    select pjc.id, pjc.pname, pjc.pkey, ji.summary, ji.description
         , cg.issueid, cg.created
         , ci.field, ci.oldstring, ci.newstring
      from projcat pjc, jira.jiraissue ji
         , jira.changegroup cg, jira.changeitem ci
      where pjc.id=ji.project and ji.id=cg.issueid and cg.id=ci.groupid and
            ji.issuetype ne '10000'
        and ((ci.field='status' and oldstring in ('In Progress', Validate',
                                                  'Stat QC') and
              not (ci.oldstring='In Progress' and ci.newstring='To Do')) or
            (ci.field='assignee'))
      order by pjc.pname, pjc.id, ji.summary, cg.created;
```

In the query above, the JIRAISSUE table contains all of the issues created for a project, including their summary, description, current assignee, and all of the other issue-level information shown on the issue screen. The CHANGEITEM table contains a record for every time a property of the issues has changed- so a change in status will generate one record, and a change in assignee will generate a separate record. Finally, the CHANGEGROUP table contains the key linking the issues and changes. Once the filters have been applied, the resulting dataset consists of 2 records per issue status change between “In Progress”, “Validate” and “Stat QC”- 1 record showing the old and new status values, and 1 record showing the old and new assignees.

issueid	created	field	oldstring	newstring
31826	17JUL2017:19:52:26.856000	assignee	Nancy Brucken	Jiefeng Chen
31826	17JUL2017:19:53:21.075000	status	In Progress	Validate
31826	24JUL2017:20:26:33.262000	status	Validate	In Progress
31826	24JUL2017:20:26:42.434000	assignee	Jiefeng Chen	Nancy Brucken

Figure 7 PROJCHG dataset

From there, it is easy to count the number of times that an issue moves from “Validate” back to “In Progress”, and from “Stat QC” back to “In Progress” using the STATUS records. In addition, we can generate the list of all of the people who have worked on that issue by concatenating the values of the OLDSTRING and NEWSTRING variables on the ASSIGNEE records.

CONCLUSION

We have found Jira to be an extremely useful tool for tracking the progress and validation status of programs for a project. It is easy to set up projects initially, once the desired workflow has been defined. Creating all of the issues for a typical project can be completed in under an hour, and it is possible to export an issue list as a CSV file from one study, and upload it to a similar study; issues for a study can also be created by importing a CSV file containing a list of the TLFs. Jira is very intuitive to use for issue tracking, so it does not place an undue burden on the project team members, and with the addition of SAS programs reading the underlying PostgreSQL database and generating custom reports, it becomes a very powerful application for project management and validation documentation.

REFERENCES

Atlassian. “Jira Software.” Accessed December 10, 2017. https://www.atlassian.com/software/jira.

Wells, Don. "Agile Software: A Gentle Introduction". Accessed December 10, 2017. http://www.agile-process.org/.

ACKNOWLEDGMENTS

Many thanks to Paul Slagle, Brian Wulff and Karen Zieker for their assistance in getting Jira up and running, and providing instruction and feedback on Jira reporting capabilities.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Nancy Brucken
Syneos Health
Nancy.Brucken@syneoshealth.com