UTILIZING ATLASSIAN JIRA FOR LARGE-SCALE SOFTWARE DEVELOPMENT MANAGEMENT*

J. Fisher, D. Koning, A.P. Ludwigsen, LLNL, Livermore, CA 94550, U.S.A.

Abstract

Used actively by the National Ignition Facility (NIF) [1] since 2004, the JIRA issue tracking system from Atlassian is now used for 63 different projects. NIF software developers and customers have created over 80,000 requests (issues) for new features and bug fixes. The largest NIF software project in JIRA is the Integrated Computer Control system (ICCS), with nearly 40,000 issues. In this paper, we'll discuss how JIRA has been customized to meet our software development process [2]. The ICCS software team developed a custom workflow in JIRA for tracking code reviews, recording both developer and quality control team test results, and managing product releases. JIRA's advanced customization capability has proven to be useful in tracking key metrics about ICCS development effort (e.g. developer workload). ICCS developers store software in a configuration management tool called AccuRev, and document all software changes in each JIRA issue. Specialized tools developed by the NIF Configuration Management (CM) team analyse each software product release, insuring that each software product release contains only the expected changes.

JIRA HISTORY

JIRA is an issue tracking system developed by Atlassian Corporation starting in 2002. It is most commonly used for software bug tracking, but thanks to its advanced customization features, is highly suitable for other types of ticketing systems (work orders, help desks, etc.), and project management.

NIF ICCS began using JIRA in 2006 for tracking software development. Prior to that, a locally developed tracking system was used. While sufficient during the early development of ICCS, limitations in functionality and development resources encouraged migration to a more advanced toolset. Fortunately, JIRA provided a number of data migration tools, so data was easily transferred. The ICCS team has successfully maintained a complete history of product software modifications since its origin.

Today the NIF employs JIRA for tracking many software development projects including the ICCS and also for other needs such as IT work orders, and high level requirements.

SOFTWARE DEVELOPMENT TRACKING

JIRA provides a mature, powerful toolset for local customizations to meet specific project needs. This includes custom fields, issue types, workflows, notifications, and user entry screens.

NIF originally adopted a custom software development workflow modelled on good industry practices. It included distinct phases for software development and quality control. In spring 2013, the workflow was modified to be more agile, allowing for the testing of software changes during earlier phases of software development.

All software changes require a JIRA issue. This includes not just bug fixes, but enhancements and new features. In fact, documentation in JIRA issues is used as a basis for all end-user release documentation. In addition, database changes (in particular configuration data), code reviews, and design reviews are tracked with JIRA.

When a JIRA issue is first entered into the system, the reporter specifies the software project and the issue type (Enhancement, Change, or Problem). From here, the initial set of JIRA fields is entered, as listed in Table 1.

Table 1: JIRA Fields Used by Reporter

Field Name	Description
Summary	A one-line description of the request
Priority	Urgent, Important, Normal, or Low. Urgent issues may be handled as patch releases
Component	The product within the project, chosen from a project-specific list
Category	Software, Operational Data, Infrastructure, Documentation.
Description	A freeform text field describing the request
Affects Versions	What software version this request relates to
Environment	Where the issue manifests (main facility, side lab, etc.)
Origin	Design Review, Coding, Developer Unit Test, Operations, Offline Tests, etc.
Reporter	Who requested the change (auto filled)
Recommend-ation	If a particular fix is needed, it can be specified here
Locos #	A ticket reference to the NIF Operations Problem Log system (not JIRA-based)
Wrap-around	A flag indicating that this was a data change that originated in the production environment.

*This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. #LLNL-ABS-632634, #LLNL-CONF-644176

ISBN 978-3-95450-139-7

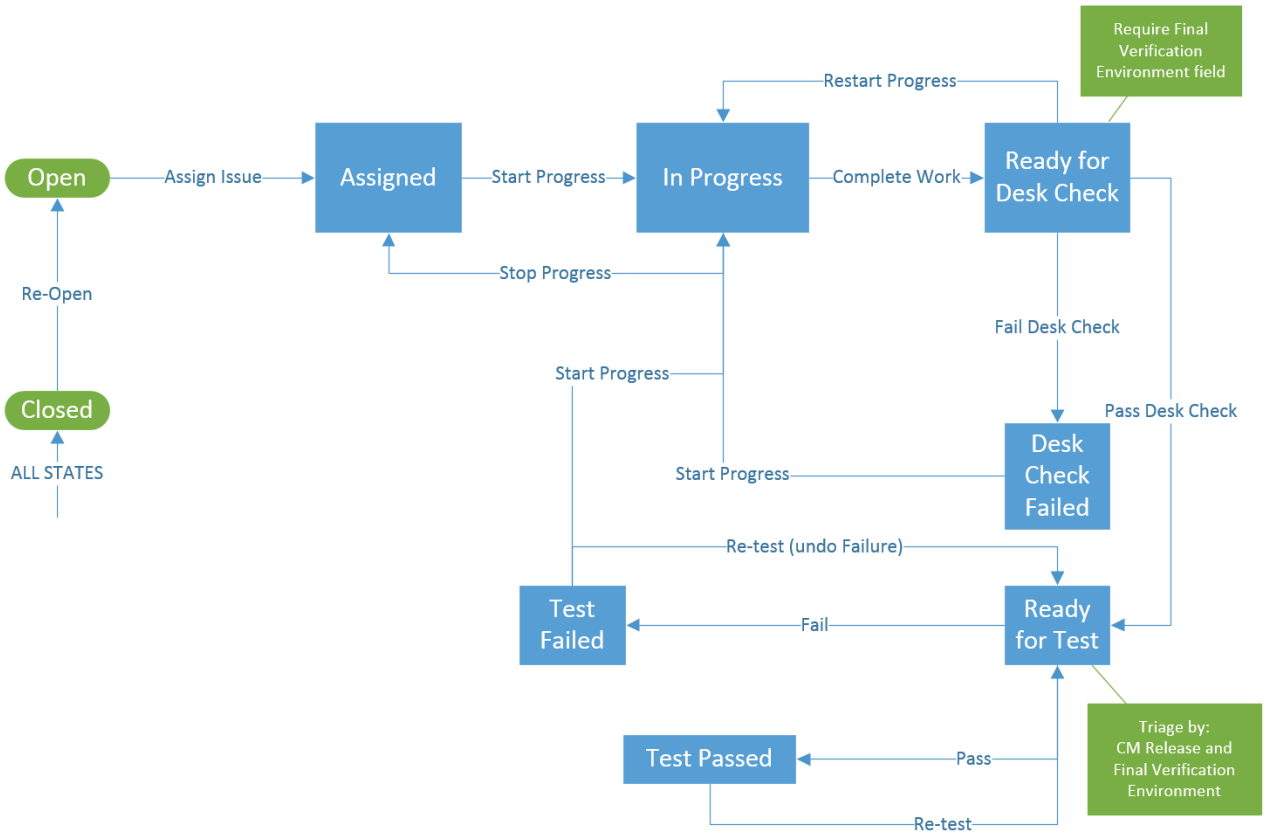

Figure 1: The NIF Software Engineering JIRA workflow, used by over 60 different software projects.

When an issue is submitted, the team lead associated with the specified component automatically receives a notification email, and becomes the default assignee. Figure 1 depicts the workflow for all software issues, where the boxes are states and the arrows are transitions.

When each transition occurs, the Reporter (and designated watchers) will receive a notification email. The team lead has the responsibility of triaging these incoming JIRA issues with the Assign Issue transition (which includes an assignment to a developer). The assignee will receive an email notification of the assignment. The Start Progress transition is used once work begins. When work is completed (including unit testing), via the Complete Work transition, the Quality Control phases can begin.

QUALITY ASSURANCE

Several quality assurance phases are employed for all JIRA issues:

- Design Reviews
- Code Reviews
- Desk Checks
- Quality Control

Whether (and when) each of these steps take place depends on the nature of the JIRA issue, as described below.

Design and Code Reviews

Review teams perform design and code reviews for significantly complex software efforts. Design reviews of course should be done prior to the start of coding.

Reviews are expected for:

- Fundamentally new software designs (e.g. a new framework or use of new Commercial Off-the-Shelf (COTS)
- Fundamentally new functionality (e.g. a new subsystem or complex algorithm)
- Software with significant risk or impact
- Existing software that has encountered a significant failure

The code reviewers will typically include those with strong technical background in the area. Design reviewers will additionally include customers, and software testers. One or two software developers who are *not* familiar with the technical area are also encouraged as reviewers, since design and code reviews are an excellent means of cross training.

The need for a formal review is assessed by the component lead, or by higher level management. Distinct JIRA issues are created for these reviews (linked to the JIRA issue for the actual coding) and used to store the review results. These JIRA issues are associated with a software release (and go through Figure 1 workflow), allowing project managers to insure that all appropriate formal reviews have been done.

ISBN 978-3-95450-139-7

Desk Checks

Every JIRA issue requires a desk check by secondary developer. The goal of a desk check is to make certain the work was performed correctly, including the following:

- The JIRA issue Release Notes contains a list of all affected files and their version numbers
- The JIRA issue Resolution Notes contains appropriate test criteria, and the results of the unit test as done by the developer
- The JIRA issue End User Notes and Processes to Restart fields are filled in properly
- The code implements the requirements and design as specified in the description
- The code conforms to existing architecture, framework, and coding standards
- The code is robust, with appropriate concurrency, exception handling, and commenting
- The impacted software deliverables have been correctly constructed and placed under the configuration management

It the desk checker verifies the criteria (listed above) are met, then the desk check can pass. For substantially complex or large code changes, a formal code review may be requested. In some cases, if minor, non-critical issues are found, the desk checker may pass the desk check and create an additional JIRA issue to resolve these at a later time.

Software Testing

As shown in Figure 1, all JIRA issues reach a "Ready for Test" status. Based on the nature of the software changes, JIRA issues may be tested with various levels of rigor, as specified by the Final Verification Environment (FVE) field. The FVE field choices include Development, Integration, Formal Test, and Production. The JIRA field QA Verifier specifies the person who will perform the software test, and changes the JIRA issue status to Test Passed or Test Failed.

Very few JIRA issues have an FVE of Development. This is mostly for changes that can only be verified by code inspection (such as an improvement in code commenting).

Each major ICCS release contains a variety of new features and bug fixes across many different subsystems. During the early development stages, individual developers are expected to perform appropriate unit testing. As a release gets close to completion, the repository is locked, and the testing emphasis switches to how all the various code changes work together as an ensemble.

Interface changes, database schema changes, and most importantly the ICCS Experiment Automation System (EAS) all need to be verified during integration testing. The developers themselves perform this step for all JIRA issues with an FVE of Integration, led by an Integration Lead. A formal software release is not actually created by the CM team at this point; rather, developers can (with Integration Lead approval) promote code into the development code tree to address problems found during the integration testing.

JIRA issues that are substantially impactful, or require specialized hardware are given FVE values of Formal Test or Production. Once all Integration testing has been performed, the CM team generates a software release, and delivers the code to the Formal Test environment. A dedicated team of Software Quality Control experts, in a dedicated lab, then verify those software changes. For certain changes, the NIF itself ("Production") is the only appropriate place to validate a software change. Dedicated on-line regression testing and integrated shot testing is performed for all software releases to the NIF.

CONFIGURATION MANAGEMENT

The ICCS project uses AccuRev (developed by a company of the same name) as a software version control system. AccuRev's stream-based architecture provides automatic merging and inheriting between code streams; it has significantly improved the efficiency of the development and CM teams, over CVS, the previous version control system used.

The CM team has developed mechanisms for connecting AccuRev with JIRA, to allow easier documentation of code changes. After completing a code change, the software developer enters the JIRA issue ID into the AccuRev software commit comments dialog (along with other details). Through a web page the developer can then input the JIRA issue ID, and generate a complete list of all code changes corresponding to that JIRA issue. This list is then pasted into the JIRA issue Release Notes, as shown in Figure 2. The desk checker uses the field as a guide when reviewing an issue.

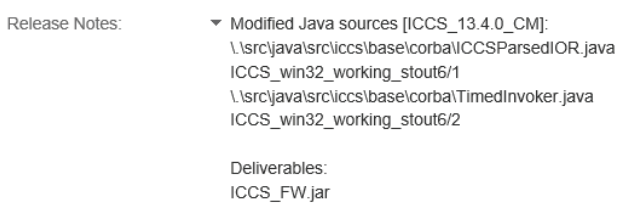

Figure 2: An example of specific software changes made in AccuRev, as documented in the JIRA Release Notes.

LIMITATIONS AND WORKAROUNDS

Over time, ICCS has confronted some limitations of JIRA, as described below.

Field-level Permissions

JIRA has no ability to control editing of specific fields within a JIRA issue. If a field is available on an entry screen, then it can be edited by any user with permission to edit that JIRA issue. Some of the custom fields in a JIRA issue (such as those used for tracking purposes by the CM team, or Integration Leads) should only be edited by specific users or groups of users.

ISBN 978-3-95450-139-7

Fortunately, every change made to a JIRA issue is documented through a change history, so should a field value be inappropriately set, details of how and why can be inspected.

Workload and Release Tracking

ICCS puts out about five major releases per year, each comprising on average 200 software change requests. Patches to deployed releases occur on a more frequent basis to fix urgent issues. Given the significant level of coordination with the NIF to schedule releases, timely delivery and detailed tracking of development and testing is critical.

To aid in a more detailed tracking of software releases, ICCS leverages external tools to mine the JIRA database.

Figure 3 illustrates release tracking through Microsoft Excel. Twice a week, the Integration Leads gather and plot the current issue counts for each release (resolved, desk checked, integration tested, and QA tested). While JIRA provides great tools for "current" status, Excel is required to plot historical trends and identify completion rate problems.

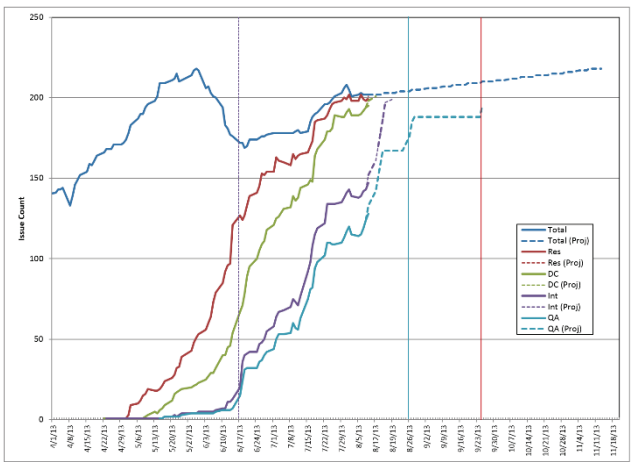

Figure 3: Tracking completion of the ICCS 13.4.0 software release, using Excel.

For a more detailed analysis of a software release, based on actual time estimates, an interactive dashboard is used as shown in Figure 4, using SQL queries and the Splunk big data analysis tool. The figure depicts a user's drilldown from the top graph to workload for the 13.4.0.1 software release, then to the workload for a specific developer. This ability to navigate through multiple graphs into specific workload details is beyond the current capabilities of JIRA.

FUTURE PLANS

The following improvements are planned:

- Upgrade from JIRA 4.1.1 to 6.0
- Migrate JIRA database from MySQL to Oracle
- Migrate JIRA from a bare metal server to a Virtual Machine Operating System (VMOS).
- Upgrade AccuRev's AccuSync JIRA server and AccuRev (to 6.0) to simplify and improve AccuRev and JIRA integration.
- Automate frequent JIRA workflow tasks, such as CM release field updates using JIRA REST APIs.

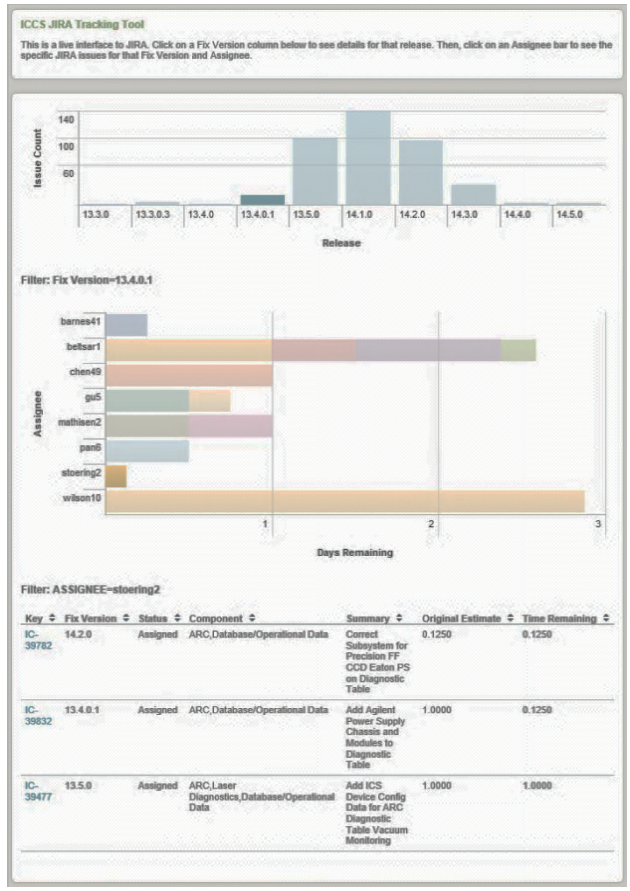

Figure 4: Splunk is used to provide drill-down dashboard capabilities beyond what can be done with JIRA's built-in dashboards.

CONCLUSION

With nearly eight years' experience with JIRA, ICCS has highly leveraged its many capabilities. Filtering and dashboards are used daily by all team members to track personal workload. The powerful search engine and change history allows quick access to what and when changes occurred in the ICCS software codebase. Where limitations have been encountered in JIRA, its open data interfaces have made it relatively straightforward to leverage external analysis tools.

REFERENCES

[1] P. Van Arsdall, et al, "National Ignition Facility Project Completion and Control System Status," ICALEPCS'09, Kobe, Japan, Oct 2009, TUP078, p. 260 (2009); http://www.JACoW.org.

[2] A. P. Ludwigsen, et al, "Software Engineering Processes Used to Develop the NIF Integrated Computer Control System," ICALEPCS'07, Knoxville, Tennessee, USA, Oct. 2007, ROAB01, p. 500 (2007); http://www.JACoW.org

ISBN 978-3-95450-139-7