The NOAO Science Archive:
Agile Development Processes: Delivering a Successful Data Management Platform Now and in the Future

Evan Deaubl, Sonya Lowry
National Optical Astronomy Observatory
Data Products Program

Why Switch to Agile Methodologies?

Developing a flexible, extensible architecture for scientific data archival and management is a monumental task under older development methodologies. Some of the problems experienced have been:

- Requirements are collected up front and frozen to control change, meaning requirement changes during development cause a renegotiation of requirements and schedules, or a significantly delay to include those changes to avoid affecting the schedule
- Results are only visible when the final product is released months or years after requirements are gathered, meaning the end result may be something that is no longer relevant, or worse, something the users don't want
- There is no focus on testing by developers, instead the development team "throws releases over the wall" to the operations team that does the thorough testing

Agile development techniques are helping our team to integrate astronomer and operator input into the development process, deliver functional software earlier, and ensure that the software is maintainable and extensible in the future.

How the Work Gets Done

The methodology we are using centers around the workflow of how feature requests become part of the evolving product. A brief summary of the process is as follows:

- Requirements are collected by a collaboration of developers, operators, and astronomers
- Requirements are split up into user stories encompassing 1-2 weeks of work, prioritized, and entered into the bug tracking system, JIRA (which we also use for project tracking and our help desk)
- User stories are assigned to developers at the weekly status and planning meeting
- The assigned developer works on a user story until these completion criteria have been satisfied:
- The code to implement the user story has been completed and builds successfully
- The code has sufficient developer-level documentation (Javadoc)
- The code has sufficient unit test coverage and passes all tests
- Acceptance tests have been written using Fitnesse and pass for all new code
- Design documentation has been updated to reflect changes
- Database update scripts have been included for any database schema changes
- User stories are reviewed for completeness at the weekly status and planning meeting
- Completed user stories are deployed at the end of a 2-3 week iteration to make progress visible to external groups

Advantages to Agile Methodologies

We've found that this process helps solve the problems inherent in older, conventional software methodologies. Some of the benefits we've seen are:

- The work necessary to ensure that the system is maintainable is built in to the schedule, and is a requirement before completion of a user story is accepted
- Development progress is visible to external groups in weeks instead of months (or years!)
- Shorter release periods reduce the turnaround for urgent feature requests, while at the same time minimizing the disruption to already scheduled work
- System integration and release is shorter because the automated build and test systems give developers the information needed to keep the system working
- Automated build and test systems provide feedback to developers at a much higher frequency (on the order of hours), allowing problems to be fixed earlier

Additional Guidelines

A methodology is more than just the process for developing features into code; other practices are needed to help guide development. Some of the ones we've adopted are:

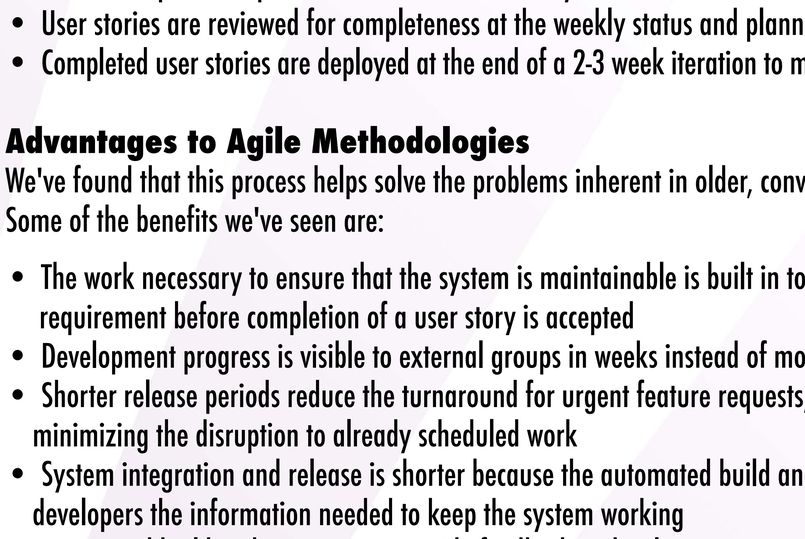
- Implementation of user stories should contain only what is necessary to satisfy the user story as written; additional unrequired features may not be wanted by the end users and are an added maintenance and documentation burden
- Communication about features of the code that needs to happen between team members should occur in a public, archived space so that design and architecture decisions arrived at as a result of these discussions are preserved, and not lost as was common with E-mail
- If one of the automated build or test systems reports problems with the current development version of the system, those problems receive highest priority until the trunk builds and tests cleanly

Tools

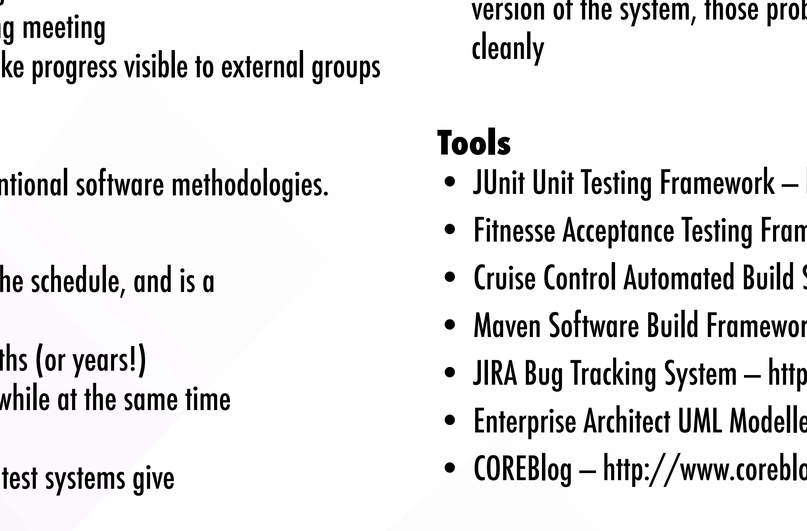
- JUnit Unit Testing Framework – http://www.junit.org
- Fitnesse Acceptance Testing Framework – http://www.fitnesse.org
- Cruise Control Automated Build Server – http://cruisecontrol.sf.net
- Maven Software Build Framework – http://maven.apache.org
- JIRA Bug Tracking System – http://www.atlassian.com/software/jira/
- Enterprise Architect UML Modeller – http://www.sparxsystems.com
- COREBlog – http://www.coreblog.org

References

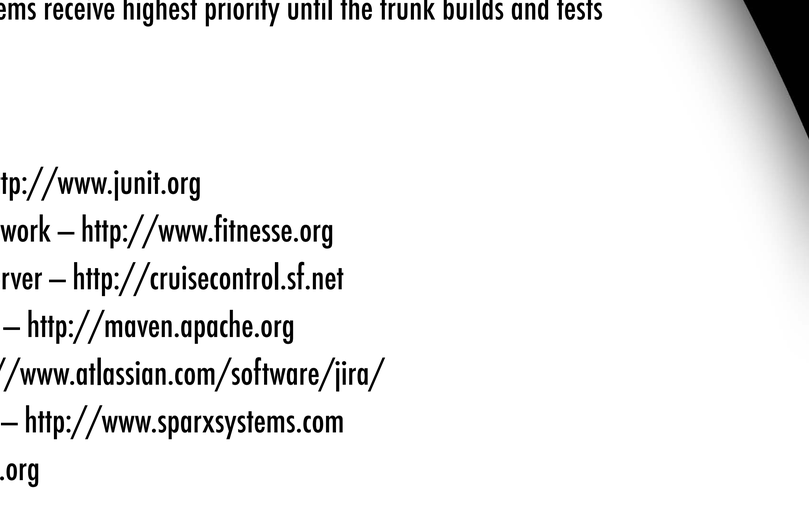
- Agile Alliance – http://www.agilealliance.org



Eclipse running JUnit test



JIRA Bug Tracking System



Fitnesse Acceptance Testing

The National Optical Astronomy Observatory (NOAO) is operated by the Association of Universities for Research in Astronomy (AURA), Inc. under cooperative agreement with the National Science Foundation.