Nerd Herding:
Practical Project Management in the Field

Matthew B. Harris, Samuel B. Fries, Sterling A. Baldwin, Dakotah S. M. Webb

***Abstract*— Project managers and developers don't always see eye-to-eye on what tools to use. So what happens when an unstoppable force meets an immovable object? Something has to give. We have had to find a balance of keeping the project management happy, while simultaneously conforming to their documentation and time-management policies. We also have had to keep our development team on task and moving forward in a way that is most efficient for the project. In this paper, we discuss the techniques, tools, and methodologies we're using to stay on track, and the compromises we've had to make in order to keep as many people happy about the process as possible.**

***Index Terms*— Agile, GitHub, Project Management, Scrum**

I. Introduction

In "No Silver Bullet", Fred Brooks wrote: "There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity."[1] Still, nearly 30 years later, we keep searching.

Our team supports the ACME (Accelerated Climate Model for Energy) [2] project, creating software to facilitate climate science research. The project under discussion is the ACME Web Dashboard; an ambitious project that binds together the many disparate services, calling out to them from a single web application. Matthew Harris started as the sole developer for this project and transitioned into the team leader as more developers were added, including a number of remote developers from other companies and facilities. His role has been to implement and enforce project policies and practices, while guiding the development direction and team member tasks.

This experience has been very insightful for our team. Our project is open source, so we host the code, issues and wiki on GitHub [2]. As a group of around eight developers, this model has worked well, but as the team grows, we must allow our project management approach to grow with it. Our first goal was to produce a demo piece of software to show as a proof of our concept. Once we hit that point in early June of 2015, it was time to get on board with the rest of the project.

We started assigning tasks through GitHub issues, with tags and milestones, and the first two sprints were very successful. At that point, we were directed to conform to the testing, documentation, and project management tools proscribed by the management of the ACME project. Along with this, we had to transfer the entire project to Atlassian's [4] Confluence [5] and JIRA [6].

Manuscript received July 9, 2015; revised July 22, 2015. This work was funded by the ACME project at Lawrence Livermore National Laboratory as a process for managing the ACME dashboard web team.

M. B. Harris is a Computer Scientist, Mathematical Programmer in the AIMS (Analytics and Informatics Management Systems) team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-423-8978 email: harris112@llnl.gov).

S. B. Fries is a Computer Scientist, Mathematical Programmer in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-422-5859 email: fries2@llnl.gov).

S. A. Baldwin is a Computer Science Intern, in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-423-8954 email: sterling16@mac.com).

D. S. M. Webb is a Computer Science Intern in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 541-761-7041 email: Dakotah.Webb@oit.edu).

II. Developer's Management Tools

A. GitHub

GitHub is one of the best places to share code with friends, co-workers, classmates, or even complete strangers. Over 9.9 million people use GitHub to build amazing things together [2]. Our team started and still uses GitHub as our primary development tool for that very reason. Some of the features (besides sharing code) are listed here:

1) Repositories

Our source code is hosted on GitHub, stored in a publicly readable git [7] repository. Each team member maintains a fork of the repository to which they make changes. When ready to merge, they submit a "Pull Request" to the main repository, owned by our organization, which will eventually be approved and merged.

2) Issues

GitHub's Issues are a developer-friendly way to manage bugs and tasks. They are tied directly to the repository, and have many features tying them to the source code. They can be automatically managed via commit messages, and can display snippets of source code inline. Each Issue can be assigned to a Milestone, which is an easy way to schedule a deadline for when work should be completed by. Issues can also be directly assigned to a specific developer, prompting them with email notifications, and can be watched by interested parties [8].

3) Documentation (Wiki)

GitHub Wikis provide a barebones documentation system that allows developers to share long-form content about their project; installation instructions, dependencies, API documentation, development

guidelines, etc. They provide a straightforward mechanism to host information about a project without having to jump through too many hoops to set up [9].

B. *Travis CI*

Travis CI [10] is an open-source, hosted, distributed continuous integration (CI) [11] service used to build and test projects hosted by a git repo. We directly integrated our GitHub repo with Travis CI using GitHub's Service Hooks feature [12], so a new build is triggered on the submission of each Pull Request to the repo. This build process can be used to execute arbitrary commands, allowing for all sorts of post-processing on the built software, including packaging, testing, and code linting.

III. Leadership's Management Tools

A. *Atlassian*

1) *JIRA*

JIRA is an issue tracker. It provides a diverse array of features, including Project Planning, Time Tracking, and Reporting Tools. It allows organization to create customized workflows, require specific pieces of information, and provides an astronomical amount of metadata for issues [6].

2) *Confluence*

Confluence is a wiki, also by Atlassian. It allows users to create detailed pages containing documentation, project plans, and more. It allows documents to be organized and centralized, gives the ability for users to discuss pages, is searchable, and has a simple integration with JIRA that allows for directly embedded JIRA issues. [5].

B. *Citrix GoToMeeting*

Citrix GoToMeeting is digital conference software. It allows users to view screens remotely. It lets us demo new features to one another, and handles conference calls as well as recording the entire presentation to share with anyone who did not attend the meeting. It also allows for remote debugging sessions when corporate firewalls prevent access to development servers [13].

IV. Management Methodologies

A. *Agile Development*

Agile development is a methodology that espouses 4 core tenets: Individuals and Interactions over Processes and Tools, Working Software over Comprehensive Documentation, Customer Collaboration over Contract Negotiation, and Responding to Change over Following a Plan [14]. These are backed up by 12 principles, which follow themes of having software always be in a buildable state, working directly with stakeholders, and generally remaining as flexible about the development as possible [15].

B. *Scrum*

The Agile manifesto doesn't go into any specifics as to how to run a project; instead, it's just a series of general guidelines as to how to make decisions. Scrum is an Agile methodology for incremental product development that uses small teams that manage independent parts of a project. [16].

Scrum uses fixed-length iterations, called "sprints" that are typically 1-2 weeks long, but are almost never more than 30 days. Scrum teams attempt to build a potentially shippable (properly tested) product increment after every sprint.

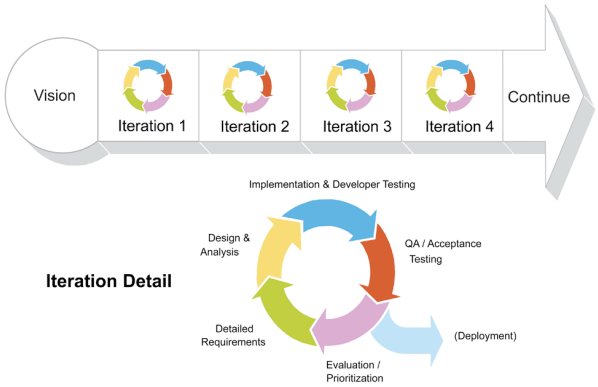

Fig. 1. An example of four sprints and shows the details of iterations [16].

Scrum is the most popular way of introducing Agility, due to its simplicity and flexibility. Because of this popularity, many organizations claim to be "using Scrum" but aren't actually applying anything close to Scrum's actual definition. Scrum emphasizes empirical feedback, team self-management, and strives to build properly tested product increments within short iterations [17].

V. ACME Dashboard Team Implementation

A. *Overview of Practices*

We are following the Agile Methodology and implementing some of the core concepts from Scrum. We try to follow the principles of the Agile Methodology to the best of our ability. Due to the realities of the project and the organization, we have adopted some loose guidelines from Scrum and tried to work with the Agile Manifesto as a guiding light for our project planning processes. Our process as a whole has continuously evolved as time goes on.

At the start of the project, when there was only one developer dedicated to the task, there was no real structure; he implemented the mock-up for the project, demoed it to management, and was eventually given the green light to start a team. The initial team was distributed across the country, and meetings were geared towards requirements gathering as the scope of the project was assessed.

As more developers were brought on, development began in earnest. Due to the nature of the project (an overarching frontend to a variety of existing services), work was broken up quite naturally by service. One person started work on building up the frontend to actually integrate in all of the external services, while the other developers worked on creating APIs from those external services to integrate with. Unfortunately, poor communication led to wasted development, and a stricter project management scheme was needed.

To help formalize our project, we began breaking tasks down into milestones of seven two-week sprints. Due to the

distributed nature of the team, rather than tie everyone up with extra phone meetings, we used a weekly meeting to make sure nobody was blocked on their tasks. Developers that are co-located interact more frequently, helping each other with blocking issues.

B. *Tools Used*

Our organization relies heavily on GitHub for code hosting, which meant that we defaulted to using GitHub Issues and the Github Wiki for project planning and documentation. This became a pain point, as our organization uses Atlassian products for planning overall progress, and our project management wanted us to track our time using JIRA, as well as our development progress in Confluence. Initially, tasks were double-entered; once in GitHub for the developers, and eventually in JIRA for the project management team to gather data for reports.

This became a huge time-sink. To address this issue, we decided to dedicate some developer time. A subtask of our project was to create a dedicated site for users of our many disparate services to report bugs. To avoid having them register GitHub accounts and hunt down all of the correct repos for the exact issue they have, we created a single form with a series of Yes/No questions that helps identify what the general category of their problem is, and pass the issue to the appropriate location. It was easy enough to add a small web-hook [18] implementation to this site, and automate the duplication of issues from GitHub to JIRA. This allowed our developers to file issues on only one platform, and kept project management's reports full of the data that they are looking for.

We also integrated Travis CI to make sure our frequent pull requests would not break the build. As part of the build process, we run a code linter that enforces a style guide against our code. Since we write all of our backend code in Python, we chose to use the PEP8 standard [19] (with the line-length requirement removed). All code is homogenous, and easily readable by all developers.

Before a build can be marked as "Successful", it has to pass all of the tests in our test suite. We wrote a small wrapper around Robot [20], which runs all of our Selenium [21] tests on the frontend. The wrapper integrates the frontend tests with our web framework's testing module, so our whole test suite can be run with a single command.

With the automated testing and a strict "no new features without tests" philosophy, we are able to refactor existing code without any concerns of breaking things; an example is a recent upgrade of our web framework, which turned out to be a very straightforward upgrade with only a few imports to tweak.

C. *What differs between your system and Agile or Scrum?*

Our process is less customer-integrated than Agile aims to be. We currently demo to our principal stakeholder no more than monthly; moving forward, we'd like to have a live and user-visible server as we reach the point of having genuinely useful features. Also, due to the requirements of our organization, we've had to focus more on "Processes and Tools" than on "Individuals and Interactions", though that's improving. We definitely respond well to change; new feature requirements, like the GitHub Webhook, were easy to work in, and the various services we're integrating with are still being defined, so we focus work on whichever ones are in a stable state.

As for Scrum, our milestones are underdeveloped and our middle range goals are a little fuzzy. We do have short-term and long-term goals; the end product is defined, though loosely enough for us to fill in the gaps as we come to them. Our backlog isn't fully fleshed out, we don't really have daily scrums, and our Sprint Planning / Review / Retrospective meetings are generally collapsed into a single meeting, in which we discuss what we've accomplished and what we're going to be working on for the next sprint.

Currently, strict Scrum is not the fit we need, but sticking to a loose Agile for its streamlined approach and quick iterations seems to be working well. We still like and use a generalized form of the meeting structure in Scrum's handbook; however, we find it easier to adhere to a much looser structure than any one methodology.

VI. Conclusion

The title of this report has a purpose greater than amusement; it is a metaphor for the position of project managers. We are the shepherds of a group of amazing and talented developers, who are tasked with getting our project delivered on time—budgeted and properly documented per the customer's demands. Getting organizational requirements and developer practices to meet in the middle is challenging, but ultimately accomplishable. Adhering to a strict methodology proved to be fruitless for our team, though using a loose set of guidelines with a barebones structure as the foundation has led to a harmonious development process.

The Agile methodology provides bright, clear guidelines. It doesn't spell out the exact way of doing things; it actually decries that quite specifically (Individuals and Interactions over Process and Tools). There are many ways to implement them, and many strict methodologies that have sprung up around them over the years. Picking and choosing the parts that fit with our process has proven to be a very effective strategy for our group, though it may not for all teams. The lesson learned here should be that if you trust the team, and give them the enough power to control the tools they use, you can still find a comfortable place between the unstoppable force and the immovable object.

Acknowledgment

We would like to thank each member of our team for all their hard work and dedication to the project and its goals: Dean Williams, Renata McCoy, Matthew Harris, Jonathan Beezeley, John Harney, Samuel Fries, Bibi Raju, Brian Smith, Sterling Baldwin, Dakotah Webb and every member of the ACME project.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DEAC52-07NA27344.
LLNL-CONF-674398

REFERENCES

[1] Brooks, F.P., Jr., "No Silver Bullet Essence and Accident in Software Engineering." *Computer*, vol.20, no.4, pp.10,19, April 1987 <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1663532&isnumber=34828>.

[2] "Build Software Better, Together." GitHub. N.p., n.d. Web. 18 June 2015. <https://github.com/>.

[3] "ACME." Enterprise. N.p., n.d. Web. 07 July 2015. <http://aims.llnl.gov/acme.html>.

[4] "Software Development and Collaboration Tools | Atlassian" N.p., n.d. Web. 18 June 2015. <https://www.atlassian.com>

[5] "Where Work Becomes Teamwork." Confluence. N.p., n.d. Web. 18 June 2015. <https://www.atlassian.com/software/confluence/>.

[6] "JIRA." Atlassian. N.p., n.d. Web. 18 June 2015. <https://www.atlassian.com/software/jira/>.

[7] "Git." Git. N.p., n.d. Web. 18 June 2015. <http://www.git-scm.com/>.

[8] "Issues 2.0: The Next Generation." GitHub. N.p., 09 Apr. 2011. Web. 18 June 2015. <https://github.com/blog/831-issues-2-0-the-next-generation>.

[9] "GitHub Help." About GitHub Wikis. N.p., n.d. Web. 18 June 2015. <https://help.github.com/articles/about-github-wikis/>.

[10] "Travis CI - Test and Deploy Your Code with Confidence." Travis CI - Test and Deploy Your Code with Confidence. N.p., n.d. Web. 07 July 2015. <https://travis-ci.org/>.

[11] "Thoughtworks." Continuous Integration. N.p., n.d. Web. 31 June 2015. <http://thoughtworks.com/continuous-integration>.

[12] “GitHub Services.” Official GitHub Services Integration. N.p., n.d. Web. 07 July 2015. <https://github.com/github/github-services>

[13] "GoToMeeting - Online Meetings and HD Vdeo Conferencing." Citrix.com. N.p., n.d. Web. 18 June 2015. <http://www.citrix.com/products/gotomeeting/overview.html>.

[14] "Manifesto for Agile Software Development." Manifesto for Agile Software Development. N.p., 2001. Web. 30 June 2015. <http://agilemanifesto.org/>.

[15] "Principles behind the Agile Manifesto." Principles behind the Agile Manifesto. N.p., 2001. Web. 30 June 2015. <http://agilemanifesto.org/principles.html>.

[16] "Scrum Reference Card." Scrum Reference Card. N.p., n.d. Web. 18 June 2015. <http://scrumreferencecard.com/>.

[17] “A glance at Agile Scrum Methodology.” A glance at Agile Scrum Methodology. N.p., 08 August 2014. Web. 30 June 2015. < http://www.360logica.com/blog/2014/08/glance-agile-scrum-methodology.html>

[18] “Webhooks level up.” Webhooks level up. N.p., 02 February 2011. Web. 30 June 2015 <https://github.com/blog/1778-webhooks-level-up>

[19] "PEP 8 -- Style Guide for Python Code." PEP 8 -- Style Guide for Python Code. N.p., n.d. Web. 18 June 2015. <http://www.python.org/dev/peps/pep-0008/>.

[20] "ROBOT FRAMEWORK." Robot Framework. N.p., n.d. Web. 18 June 2015. <http://robotframework.org/>.

[21] "Browser Automation." Selenium Blog Posts. N.p., n.d. Web. 07 July 2015. <http://docs.seleniumhq.org/>.