Agile in Science

National Centers for Environmental Information

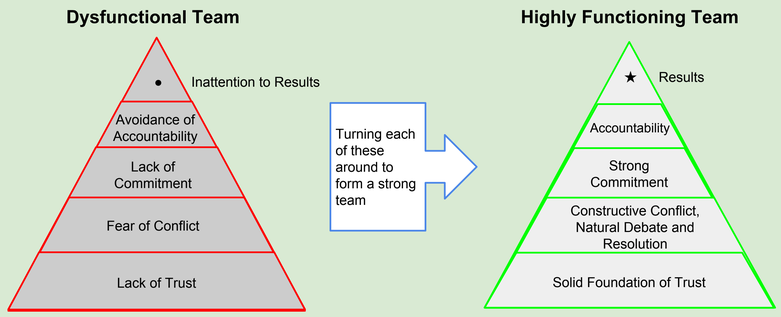
Brendan Billingsley, Aaron Caldwell, Danielle Harper, Waverly Hinton, Evan McQuinn, Ken Tanaka, David Neufeld: NCEI, CIRES, University of Colorado

Teaming

Productive teams have 3 to 7 developers, plus a Product Owner and a ScrumMaster. The team is allowed to self organize to best solve problems and do their work in an environment that supports focus. The team is supported by Management with minimal interruptions and clear tasking, they have access to subject matter experts and opportunities for learning.

We emphasize building highly productive, gelled teams. We use the Patrick Lencioni model for creating a highly functioning team. We use our retrospectives to discuss both our process and reflect on our team and what changes we can make to be a higher functioning team. We emphasize healthy conflict and creating an environment for productive conflict. We make commitments and hold each other accountable for those commitments. This builds to producing great results.

Patrick Lencioni described *The Five Dysfunctions of a Team*, often summarized with a pyramid diagram:



Development Practices

Shared Space

We work in close collaboration by sitting in a shared space. We use this to our advantage with frequent communication between team members while problem solving. This allows all developers to overhear conversations and decide if they will join in or not.

Pair Programming

We use pair programming as a tool for cross training, solving difficult problems and improved focus. Developers decide when and if they want to pair while working on a feature and we frequently use the pairing approach when a developer gets stuck. Overall we pair program roughly 10% of the time.

Evolutionary Design

We create and modify the design of our systems as they are developed. We start with the simplest design that can possibly work (as opposed to no design) and revisit the design constantly through the development of the code. We place emphasis on simple designs that enable agility in the architecture even when they are hard to create and invest in these designs up front.

Collective Code Ownership

The team owns the system, no single developer can ever claim any part of the system as their own. This makes our teams robust and prevents single point of failure problems.

Sustainable Pace

We work at a sustainable pace throughout each sprint and across sprints. We benefit by having consistent high performance.

Automated Testing

We write automated unit tests and end-to-end tests. These tests are developed as part of every story. They are automatically executed by the build server and provide quick feedback when features are broken and confirmation that new features work. Some members use test-driven development (TDD) to develop tests, while others write the tests after developing the code, our requirement is that tests be written concurrently with the feature to be tested.

Continuous Integration

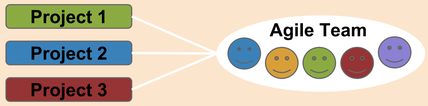
We have automated builds for continuous integration of the code as it is developed. All check-ins trigger an automated build, and developers are encouraged to commit to the master branch frequently.

Continuous Deployment

Every check-in results in a build with code linting and unit tests. Following successful passage of these steps, code is deployed to an integration environment and end-to-end tests are run. Code which passes end-to-end test is marked as a potentially deliverable artifact and deployed to our acceptance environment where stakeholders can review the product. We have found deployment to be complex, so automation of integration, testing and delivery is worth the investment of the team's time. Every sprint cycle we typically include some improvements to our infrastructure.

Science Challenges

Many Diverse Projects



Science organizations often have many small software development projects. Most agile literature focuses on situations where one big project is developed by one or more teams. Meeting the requirements of science organizations requires working on many different projects, often switching between projects in the weeks to months time frame. We address this challenge in multiple ways.

First, we find common needs between projects and design larger, enterprise systems. This allows us to maintain focus on a single system while still engaging multiple stakeholders and solving multiple problems at once.

Second, our resource management team (RMT) coordinates stakeholder requirements and agile team input in order to make hard decisions and prioritize the projects we take on.

Project Prioritization

3/5	ECS Proj	4/2	PAD Proj	5/14	GA Proj	6/4	GA Proj
3/19	PAD Proj	4/16	CSB Proj	5/28	GA Proj	6/18	GA Proj
		4/30	CSB Proj				

In many agile environments, projects are authorized, funded and prioritized by upper management in a centralized manner. In many science organizations PIs get funding for projects and there is not strong centralized decision making.

Our resource management team (RMT) prioritizes projects and schedules them into two week team sprints. This is done by maintaining a calendar with projects placed into each two week sprint. This feeds into a prioritized epic backlog with team estimates for which epics can be accomplished in the allotted time. The backlog and calendar are updated at least every two weeks.

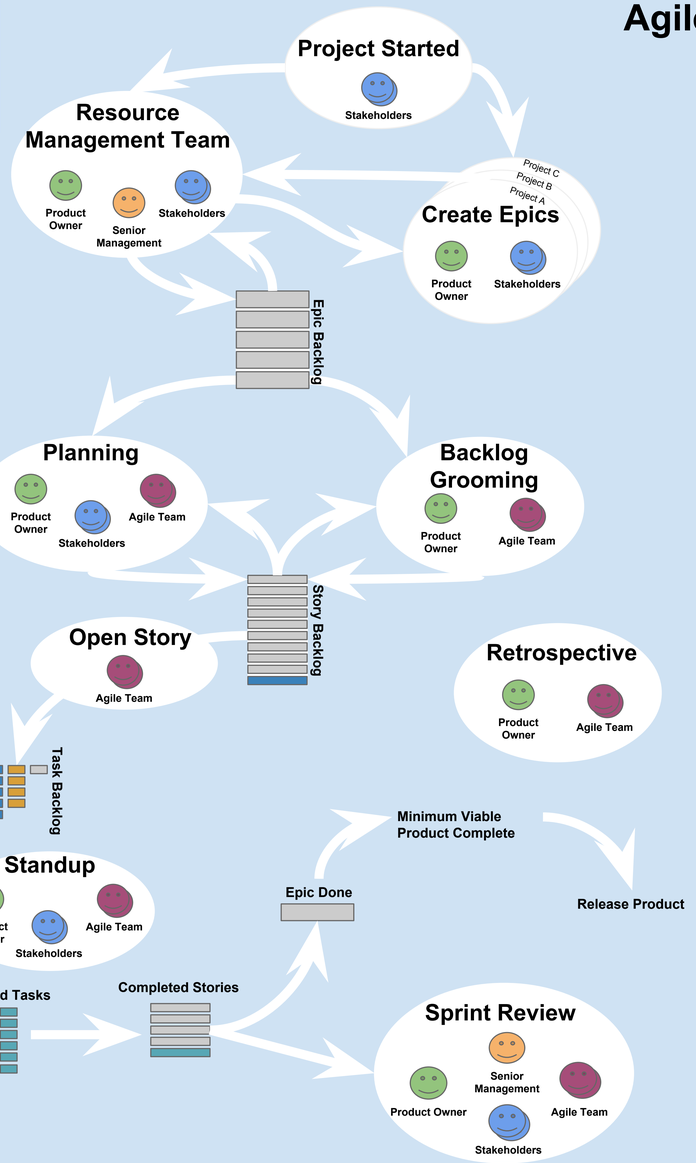
Estimates are frequently incorrect, the RMT is responsible for adjusting estimates to reflect reality as it unfolds. When features are completed earlier than expected, the RMT decides if the project gets more epics or the next project is started early. When estimates are wrong or projects have unexpected changes, the RMT decides if the organization will double-down on that project by extending the number of sprints allotted, recover by reducing scope for that project or pivot and deliver different work than originally planned.

Siloed Developers and Legacy Software

Many science organizations have a history of individual developers assigned to specific projects. In the transition to a team approach, it is challenging to incorporate these legacy projects into the team workflow. It is also difficult to have the organization adjust to individual projects without a fully committed full-time developer to direct.

Although this transition is difficult, it is critical for the success of the team and the agile process. Transferring ownership of legacy applications to an agile team requires discipline. The team needs to prioritize cross-training over short term speed when working on these projects. Stakeholders are required to go through the product owner and the resource management team in order to get project time scheduled.

Agile Process



Roles

Stakeholder: The people who enable, and at a high level direct, the project. Often the funding source for the project. Usually the stakeholders include the project lead for the bigger project that the specific product is a part of. They provide the domain expertise and are involved in reviewing the product.

Group Point of Contact: The point of contact for a group within the organization. This individual helps coordinate between the product owner and the stakeholders, acting as a product owner assistant. Responsibilities include working with project leads (stakeholders) to develop epics, meeting with the product owner frequently to discuss epics and prioritization concerns and communicating group concerns with the product owner.

Product Owner (PO): The member of the agile team responsible for maintaining the Epic and story backlogs. They meet with the stakeholders and management to gather requirements and translate the agile process. They work with the team to direct the product development. They work with the RMT to provide input on project scheduling and efficiency opportunities.

Team Member: A member of the cross-functional software development team responsible for delivering the Epics. The team self organizes on tasks and research, and cross-mentors members to advance skills.

ScrumMaster: The member of the agile team responsible scheduling and maintaining the ceremonies. Identifies and coordinates the removal of impediments. Advocates and coaches the agile framework, keeps the team metrics up to date, and directs external work requests through proper channels.

Artifacts

Epic Backlog: A prioritized list of epics. Epics are large bodies of work with a description of requirements, scope of work and definition of done defined by the product owner, stakeholders and optionally a couple agile team members.

Story Backlog: A prioritized list of stories that are ready to be worked by the agile team. Stories describe a small piece of functionality to be worked on by the agile team and a definition of acceptance criteria that defines when the story is done created by the product owner with help from the team.

Task Backlog: Tasks that are required to complete each story. The tasks are organized on a table with columns for each step in the process of completing a task (e.x. todo, in progress, ready for review, done). Agile team members pull the next highest task from the todo column when they need work to do.

Minimum Viable Product: A set of epics that is the minimum required for the stakeholder to release the product, can be a single epic.

Agile Metrics: Collection of statistics on team performance by story points completed toward epics in the current project. Used to monitor productivity and improve estimates of the time to complete requested features. Common products of these metrics include the burn-up chart of features, and reports on the percentage of on-task and external work for the Sprint Review.

Ceremonies

Resource Management Team (RMT): Team that owns prioritization of projects and scheduling into the sprint calendar.

Epic Creation Meeting: Epics are created by the stakeholders, reviewed by the PO and then turned in to meetings.

Planning: The product owner holds this meeting every two weeks to select what work to do next. This includes refining the current story backlog, adding and prioritizing any new stories that have come up since the last meeting and breaking down the next epic into stories. The goal is to have a prioritized story backlog deep enough to last at least one week (until the grooming meeting).

Backlog Grooming: Similar to the planning meeting but with a greater emphasis on refining the backlog and looking for missing stories, bug stories, story refining and reprioritization. As with the planning meeting, the outcome of this meeting is to have a prioritized backlog that is at least deep enough for one week of work. This also identifies stories that need additional information before work can be started, significantly reducing the time needed for the subsequent planning meeting.

Sprint Review: The sprint review is held every two weeks to demo the work the team has done in the last two weeks, provide transparency into how the work is progressing and discuss with stakeholders and management what problems have come up and what will be done next. The reviews are alternated between whole center reviews and smaller stakeholder reviews.

Retrospective: The ScrumMaster holds this meeting every two weeks to reflect on the last two weeks of work and iterate on the process. This meeting also reflects on the teaming process and team dynamics. The meeting produces up to three actions to focus on during the next sprint.

Standup: Daily meeting for the agile team to report to each other. Each member briefly describes what they have done since the last meeting, what they plan to do next and what impediments they are encountering. Topics that need discussion are noted in a parking lot and scheduled for discussion later. Stakeholders are welcome to observe and participate in any follow on meetings.

Opening Story: When a team member needs to pull a task and none exist they let the team know they are opening the next story and some members of the team join them to discuss the story and create the tasks needed for completing the story.

May 1st, 2015