Agile Methodologies

David E. Bernholdt
Oak Ridge National Laboratory

Michael A. Heroux, James M. Willenbring
Sandia National Laboratories

Software Productivity Track, ATPESC 2020

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Mark C. Miller, Katherine M. Riley, and James M. Willenbring, Software Productivity Track, in Argonne Training Program for Extreme Scale Computing (ATPESC), online. DOI: 10.6084/m9.figshare.12719834**
- Individual modules may be cited as *Speaker, Module Title*, in Software Productivity Track…

Acknowledgements

- Additional contributors include: Patricia Grubel, Rinku Gupta, Mike Heroux, Alicia Klinvex, Jared O'Neal, David Rogers
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2020-8110 PE

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Outline

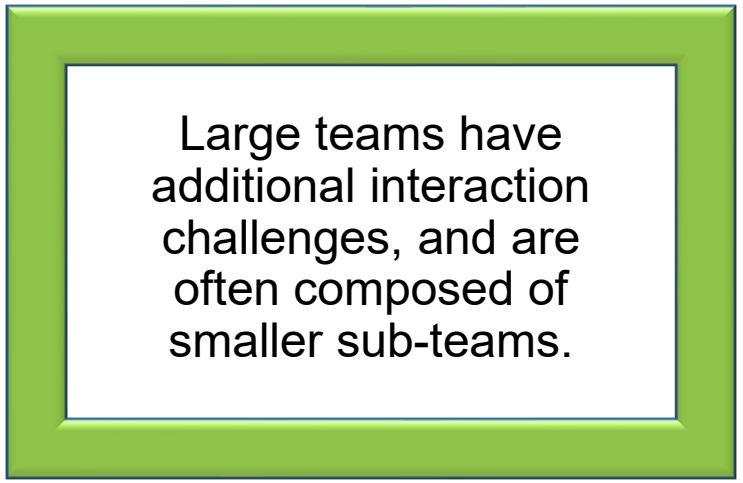
- Small Team Models, Challenges
- Agile workflow management for small teams.
  - Intro to terminology and approaches
  - Overview of Kanban
  - Building on Kanban
  - Free tools: Trello, GitHub

Small Teams

Ideas for managing transitions and steady work.

Small team interaction model

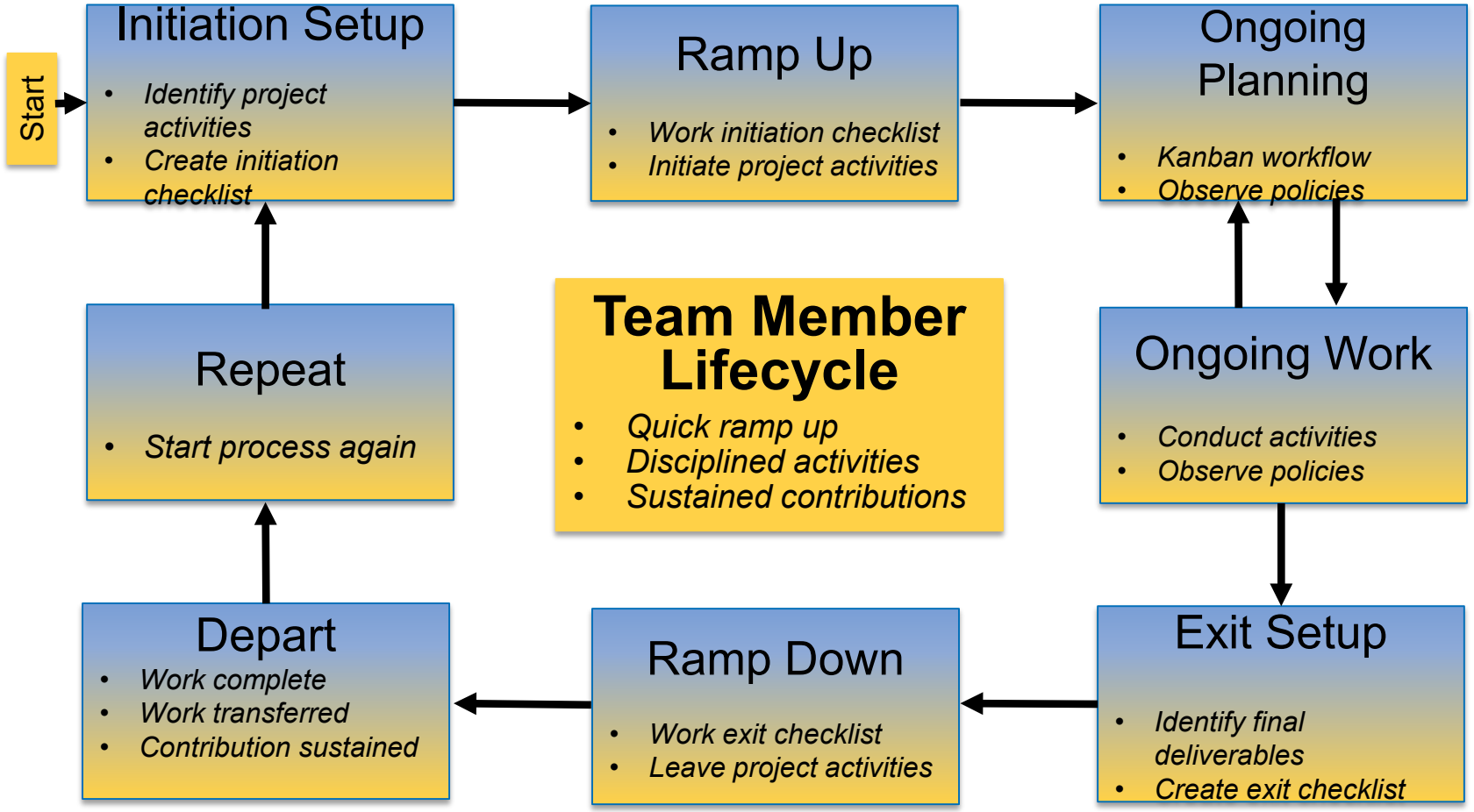
- Team composition:
  - Senior staff, faculty:
    - Stable presence, in charge of science questions, experiments.
    - Know the conceptual models well.
    - Spend less time writing code, fuzzy on details.
  - Junior staff, students:
    - Transient, dual focus (science results, next position).
    - Staged experience: New, experienced, departing.
    - Learning conceptual models.
    - Write most code, know details.



Small team challenges

- Heavy processes are often neither necessary nor appropriate
  - Adopt only those processes that add value
- Ramping up new junior members:
  - Background.
  - Conceptual models.
  - Software practices, processes, tools.
- Preparing for departure of experienced juniors.
  - Doing today those things needed for retaining work value.
  - Managing dual focus.

Research Team Member Lifecycle



Checklists & Policies

Team Member Phase		
New Team Member	Steady Contributor	Departing Member
Checklist	Policies	Checklist

- New, departing team member checklists:
  - Example: Trilinos New Developer Checklist.
  - Simple – prevents omissions
  - https://github.com/trilinos/Trilinos/wiki/New-Trilinos-Developers
- Steady state: Policy-driven.
  - Example: xSDK Community policies.
  - https://xsdk.info/policies/

New developer checklist snippet

_x_ Verify familiarity with and configure git. Each machine requires base configuration:
https://github.com/trilinos/Trilinos/wiki/VC-%7C-Initial-Git-Setup
Introductory material available at:
https://github.com/trilinos/Trilinos/wiki/Tools--%7C-Git
Date completed:

_x_ Learn about the Trilinos develop / master branch workflow:
https://github.com/trilinos/Trilinos/wiki/VC-(VERSION-CONTROL)
https://github.com/trilinos/Trilinos/wiki/VC-%7C-'develop'-'master'-workflow
Date completed:

_x_ Become familiar with the Trilinos Policies page and review relevant policies:
https://github.com/trilinos/Trilinos/wiki/POLICIES
Date completed:

_x_ Complete a GitHub pull request with a mentor:
+ Fork Trilinos and issue a pull request from a branch on your fork.
+ Remember that all pushes to the Trilinos repository and modifications to Trilinos webpages are world-wide releases of information, so institution-specific copyright, review, approval and other appropriate policies must be followed.
+ Make any necessary changes to GitHub Issues (also after the next day's test harness results, if appropriate).
Date completed:

Agile Methodologies

Why Agile?

- Fits the research experience better than heavier-weight approaches
  - Aligns more naturally with how scientific progress is made
- Well-suited for scientific software efforts (when tailored correctly)
  - Works well for small teams
  - Provides meaningful, beneficial structure that promotes
    - Productivity
    - Productization
    - Sustainability
    - Flexibility in requirements
    - Communication

What is Agile?

- Agile is not a software development lifecycle model
- I've seen Agile informally defined as
  - I don't write documentation
  - I don't do formal requirements, design, or really test…
  - Agile is not an excuse to do sloppy work
- Some people consider agile to be synonymous with Scrum
  - From Atlassian: Scrum is a framework that helps teams work together
  - Scrum is Agile, Agile is not (only) Scrum
  - A square is a rectangle, not all rectangles are squares
  - Agile is not Kanban either

What is Agile?

http://agilemanifesto.org/

Principles behind the Agile Manifesto

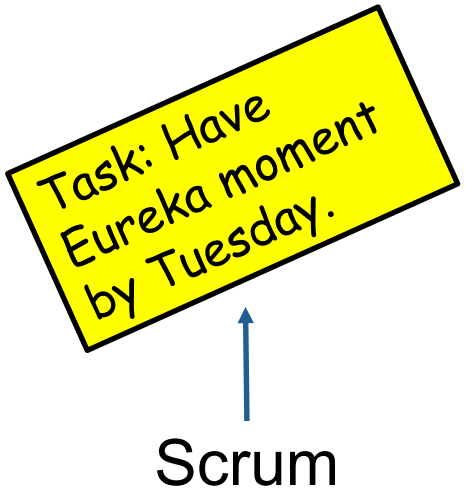
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Principles behind the Agile Manifesto

- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done- is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Getting Started with Agile

- Agile principles are not hard and fast rules
- Try adopting a few Agile practices
  - Following a rigid, ill-fit framework usually leads to failure



- Kanban is a good starting framework
  - Follow basic principles, add practices when advantageous
  - Better than removing elements from Scrum

Kanban principles

- Limit number of “In Progress” tasks
  - Must be tuned by each team
  - Common convention: 2n-1 tasks where n = # team members
- Productivity improvement:
  - Optimize “flexibility vs swap overhead” balance. No overcommitting.
  - Productivity weakness exposed as bottleneck. Team must identify and fix the bottleneck.
  - Effective in R&D setting. Avoids a deadline-based approach. Deadlines are dealt with in a different way.
- Provides a board for viewing and managing issues

Basic Kanban

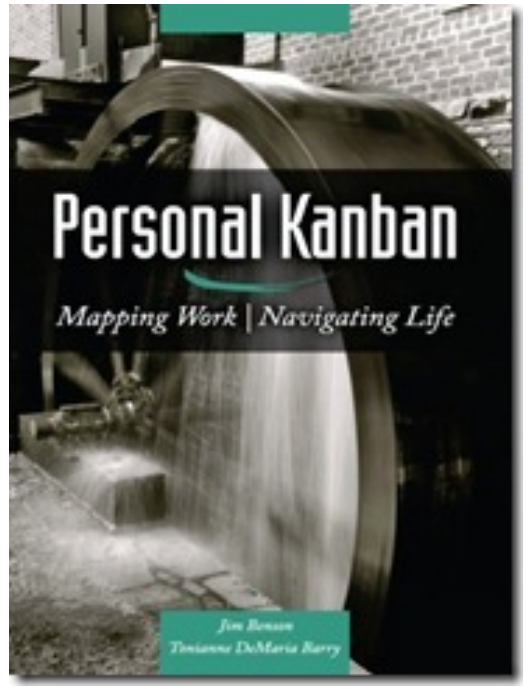
Backlog	Ready	In Progress	Done
• Any task idea • Trim occasionally • Source for other columns	• Task + description of how to do it. • Could be pulled when slot opens. • Typically comes from backlog.	• Task you are working on *right now.* • **The only Kanban rule: Can have only so many "In Progress" tasks.** • Limit is based on experience, calibration. • **Key: Work is *pulled*. You are in charge!**	• Completed tasks. • Record of your life activities. • Rate of completion is your "velocity".

Notes:

- Ready column is not strictly required, sometimes called "Selected for development".
- Other common column: In Review
- Can be creative with columns:
  - Waiting on Advisor Confirmation.
  - Blocked

Personal Kanban

- Personal Kanban: Kanban applied to one person.
  - Apply Kanban principles to your life.
  - Fully adaptable.
- Personal Kanban: Commercial book/website.
  - Useful, but not necessary.



https://bssw.io/items/using-personal-kanban-for-productivity

http://www.personalkanban.com

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Kanban tools

- Wall, whiteboard, blackboard: Basic approach.
- Software, cloud-based:
  - Trello, JIRA, GitHub Issues & Project Board.
  - Many more.
- I use Trello (browser, Android, iPhone, iPad).
  - Can add, view, update, anytime, anywhere.
  - Different boards for different contexts
    - Effective when people are split on multiple projects

Big question: How many tasks?

- No single answer. Choose something and adjust from there.
- Personal Kanban approach: Start with 2 or 3.
- Teams: Consider 2n-1, where n=number of team members.
- Use a freeway traffic analogy:
  - Does traffic flow best when fully packed? No.
  - Same thing with your effectiveness.
- Spend time consulting board regularly.
  - Brings focus.
  - Enables reflection, retrospection.
  - Use slack time effectively.
  - When you get out of the habit, start up again.
  - Steers towards previously started tasks

Importance of “In Progress” concept for you

- Junior community members:
  - Less control over tasks.
  - Given by supervisor.
- In Progress column: Protects you.
  - If asked to take on another task, respond:
    - Is this important enough to
      - back-burner a, b, and c?
      - become less efficient?
    - Sometimes it is.

Building on Kanban

- Focus: Solve issues!
  - (not add process)
- 15 minute stand-ups
  - Maybe not daily
- Planning meetings
- Retrospectives
- Scrum Master
- Product Owner
- **Epic, story, task**
- Definition of Done

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Building on Kanban

- Epic, Story, Task
  - Formal or informal
  - Start with high-level requirements
  - Break down and refine when and as needed
    - Close to when the work will be done
    - Only for work that will take place
    - Can be valuable for estimating
    - There is no “correct” level of granularity
  - Epics are very high level objectives
  - Stories should represent an increment of value to the customer
    - ”Done” criteria – understandable to user
  - Tasks are the steps necessary to complete a story
    - May not individually provide value to the customer

Building on Kanban

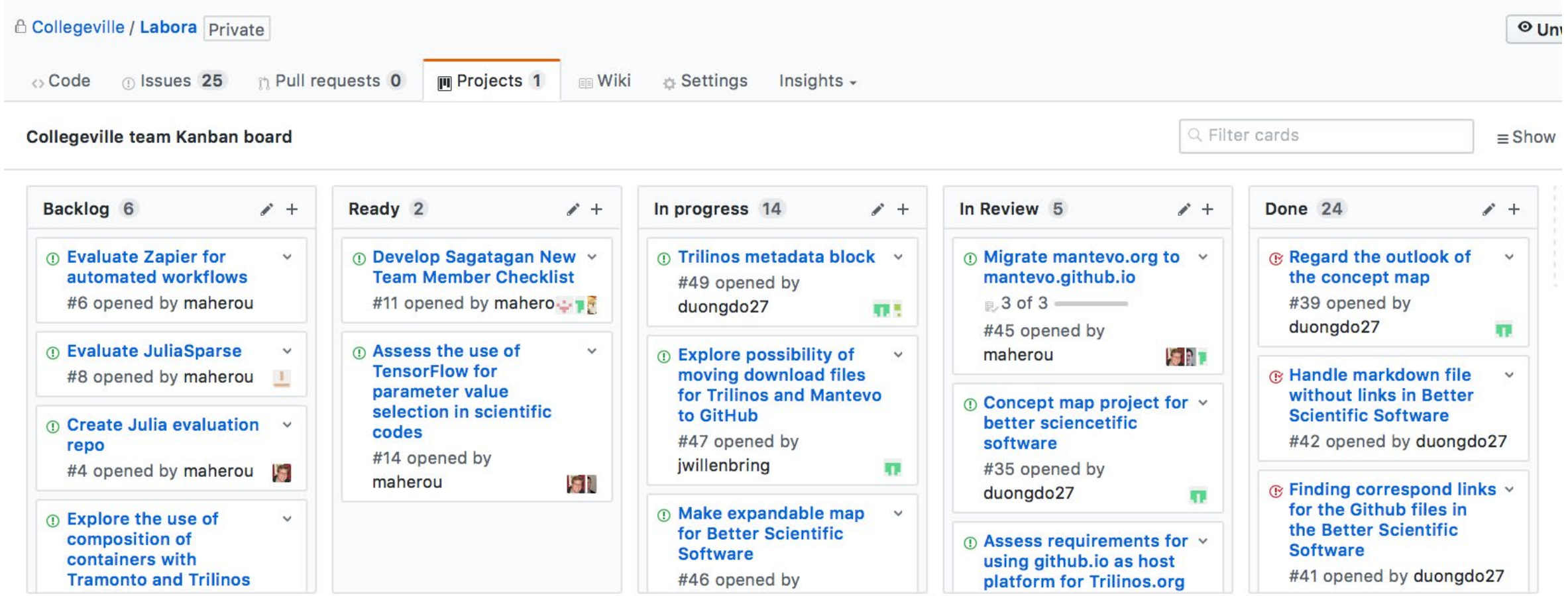
- User stories (optional)
  - Form: As a <stakeholder>, I want <describe what is needed> so that <why do you want this?>
  - Can be useful to improve communication and requirements elicitation
- In heat example:
  - User stories collected
    - As a developer, I want to modularize the heat equation utilities so that I can more easily make use of the utilities for other projects.
    - As a developer, I want to be able to use multiple integration functions easily so that I can utilize the function best suited for the problem I am solving.

Building on Kanban

- Epic (derived from user stories): Refactor code for enhanced modularity
  - Description: The heat equation code needs refactoring to improve modularity. Specifically, there are utilities that could be generalized and used with for other applications. Also, the integration function is currently hard-coded. In the future, we want to use alternative integration functions, so we should generalize the interface for this function.
    - Story 1: Separate out utilities
    - Story 2: Separate out integration function

- This idea needs to be socialized with stakeholders
- No staffing/funding currently available

Samples from Collegeville Org: Kanban Board



Kanban in GitHub

- GitHub supports basic Agile development workflows
  - Filing issues
    - @mention
  - Kanban board
  - Projects
- GitHub lacks more advanced features
  - Dependencies between issues
    - You can reference one issue in another
  - Advanced notification schemes
  - Custom fields
    - You can create custom labels

Building on Kanban

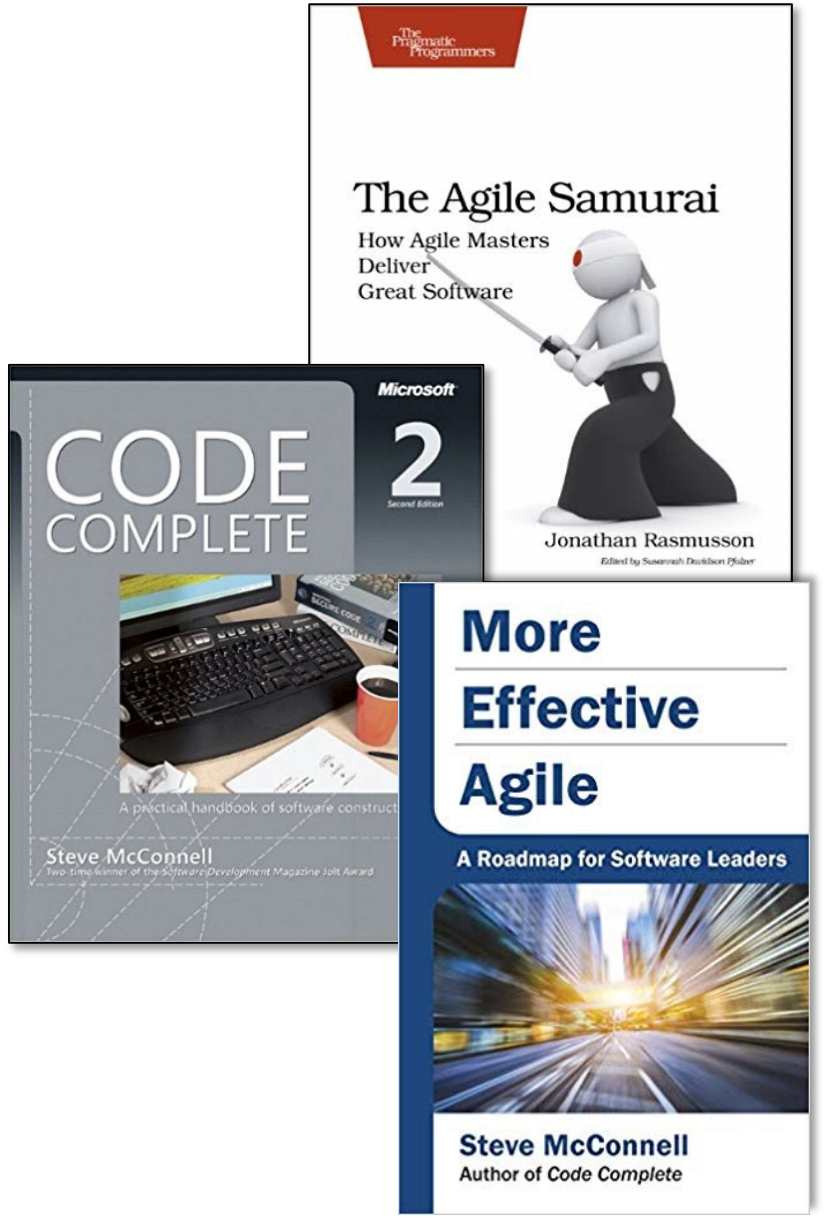
- A-Team Tools: A collection of resources for understanding and applying lightweight agile practices to your scientific SW project
  - Especially useful for
    - Small teams
    - Teams of teams
    - Teams that frequently have members come and go
  - https://betterscientificsoftware.github.io/A-Team-Tools/

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Other Resources

- **The Agile Samurai: How Agile Masters Deliver Great Software (Pragmatic Programmers),** Jonathan Rasmusson.
  - http://a.co/eUGIe95
  - Excellent, readable book on Agile methodologies.
  - *Also available on Audible.*
- **Code Complete: A Practical Handbook of Software Construction,** Steve McConnell.
  - http://a.co/eEgWvKj
  - Great text on software.
  - *Construx website has large collection of content.*
- **More Effective Agile: A Roadmap for Software Leaders,** Steve McConnell.
  - http://a.co/22EPvt6
  - New: A realistic view of Agile effectiveness with great advice for project leaders.



IDEAS productivity

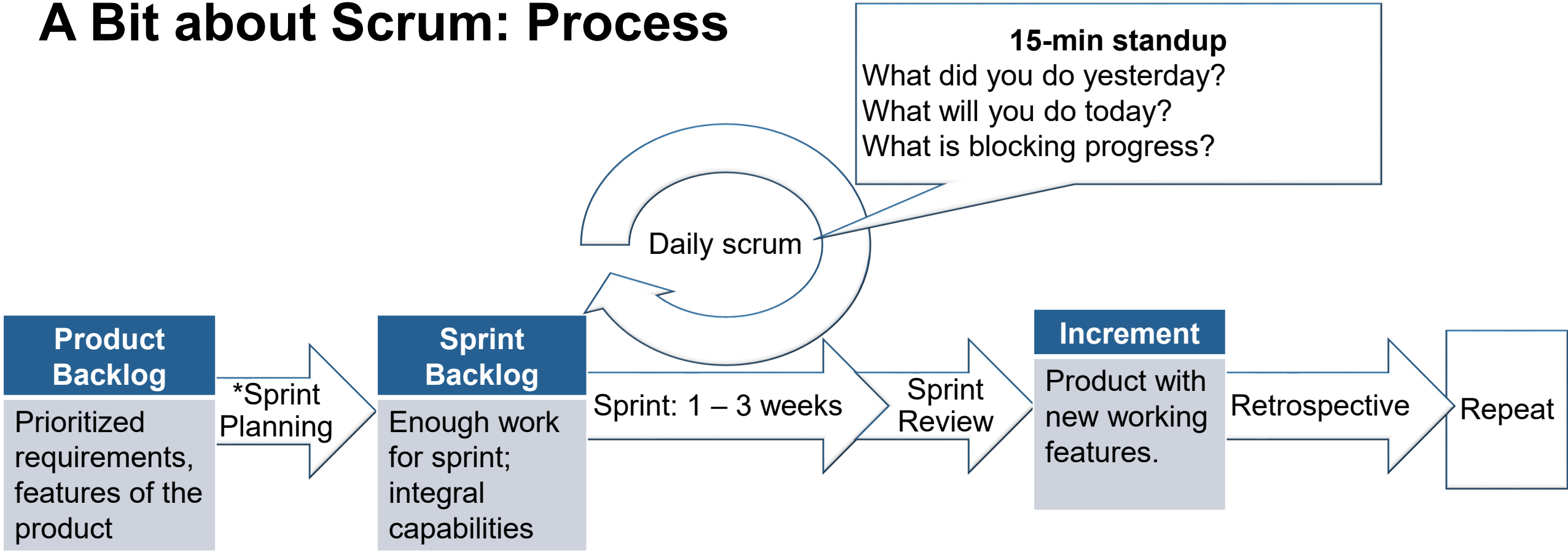
ECP EXASCALE COMPUTING PROJECT

A Bit about Scrum: Roles

Scrum team

Product Owner	Scrum Master	Development Team
• Interface between development team and stakeholders. • Responsible for defining and managing work backlog. • Needs good domain knowledge. • Needs adequate time to do job well.	• Leads and coaches development team. • Assures scrum processes followed. • Needs good Scrum knowledge and discipline. • Can be a developer if sufficient time.	• Cross-functional group of 3 – 9 that develops product. • Completes all work necessary to be done-done. • Collectively need design, development, testing, documentation skills. • Works in collaboration with product owner, scrum master.

A Bit about Scrum: Process



* Sprint planning happens during previous sprint

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Team Management Example

Team Policy

Checklists

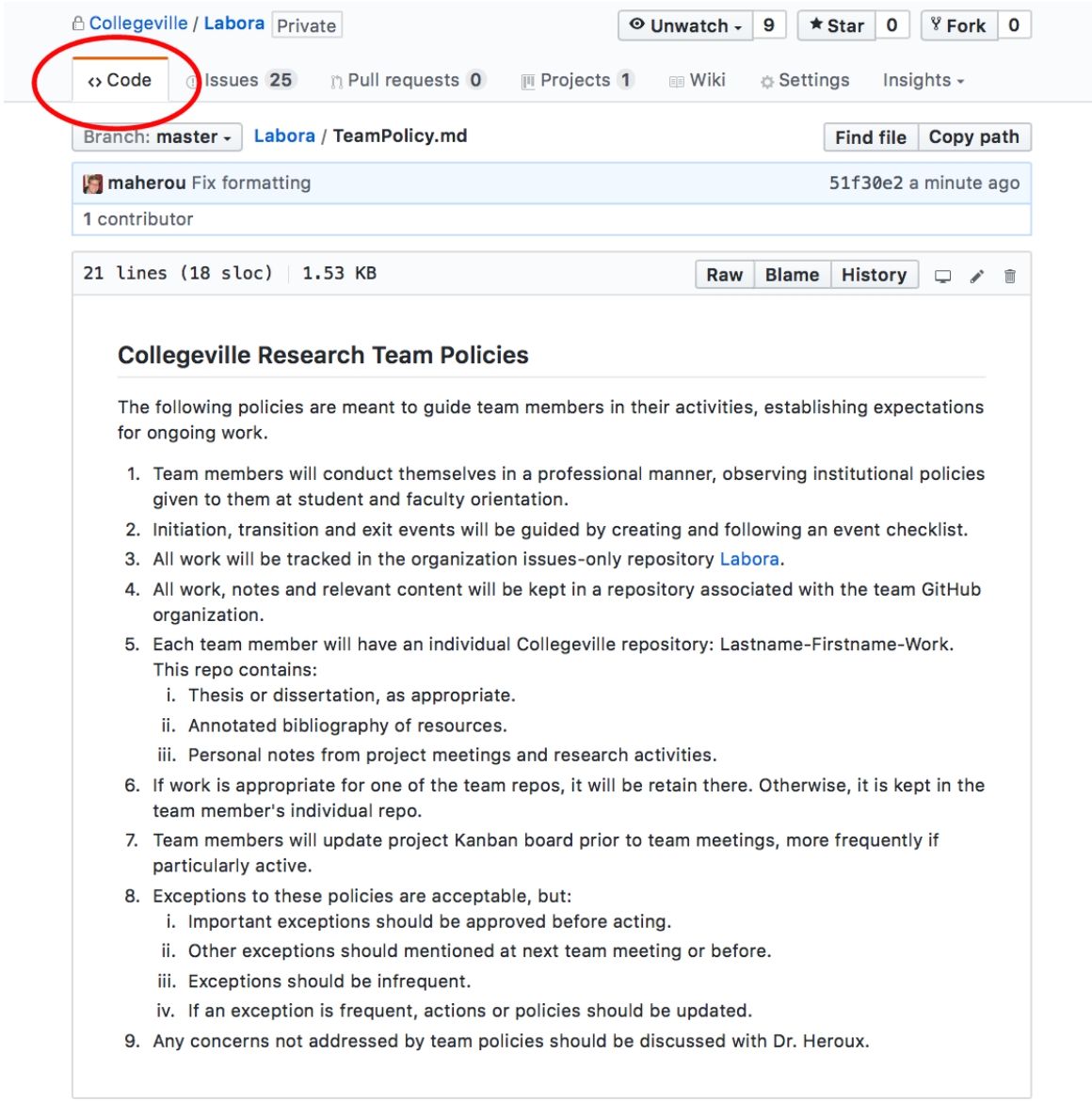
Kanban Board

Step 1: Create Issues-only GitHub repo

- Go to https://github.com/username
  - Example: https://github.com/maherou
- Create new repo:
  - Click on “+” (upper right).
  - Select New repository…
  - Give repo a name, e.g., **Issues**
  - Select Public. In real life, this repo is often private (requires $ or special status)
  - Init with README.
  - Don’t add .gitignore or license.
  - Click Create Repository.

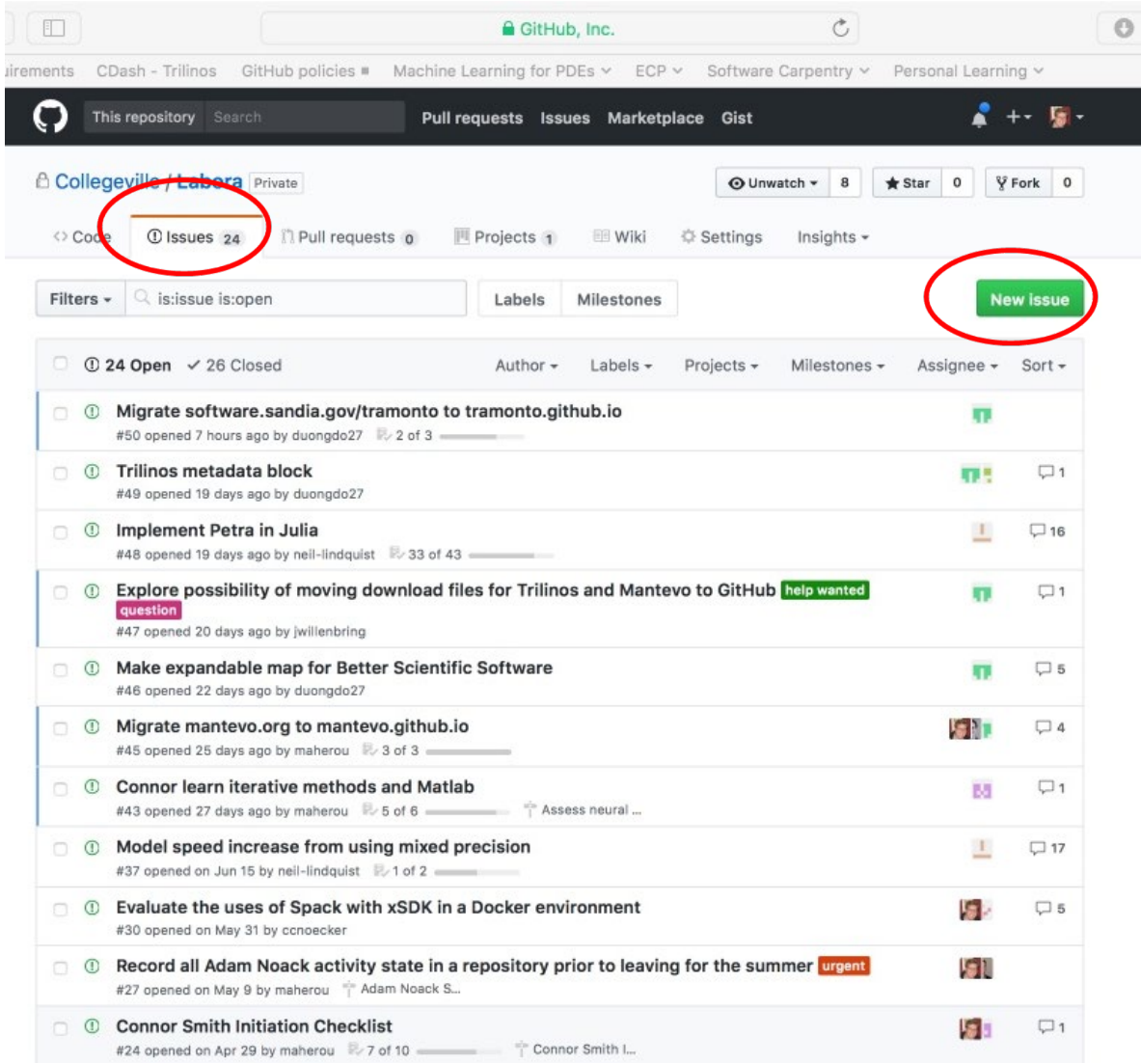
Step 2: Define Team Policy

- Create file:
  - Go to new repo: Issues.
  - Select <> Code tab.
  - Select Create new file TeamPolicy.md
- Questions to address:
  - How members support team?
  - How team supports members?
- Community version:
  - http://contributor-covenant.org
- Policy is living document:
  - Informal good practices added.
  - Avoidable bad situations addressed.



Step 3a: Create Issues

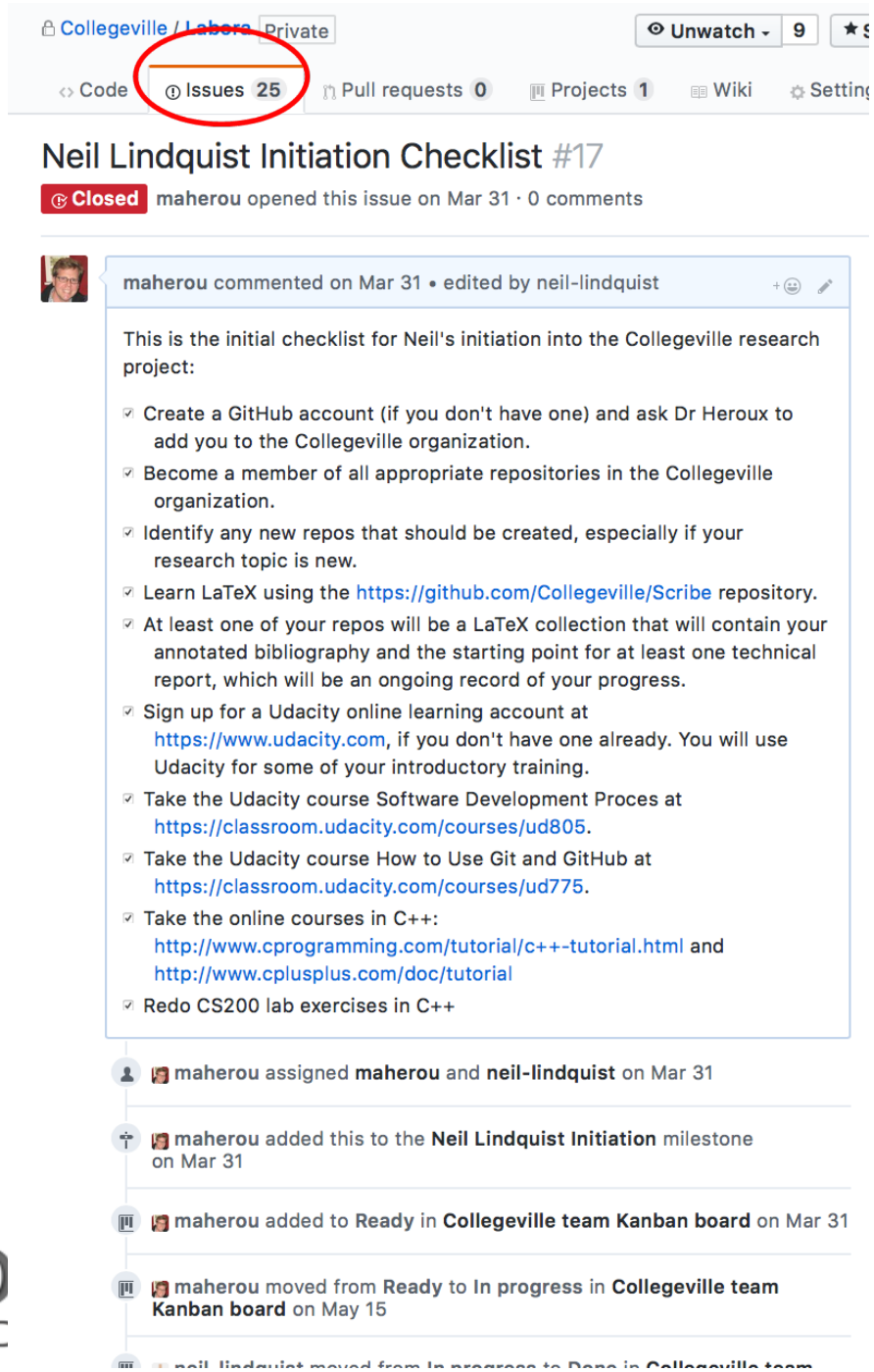
- Select the Issues tab.
- Click on New Issue.
- Type in task statement 1 (from list).
  - Type in title only.
- Click Submit new issue
- Repeat.



Step 3b: Create Initiation Checklist

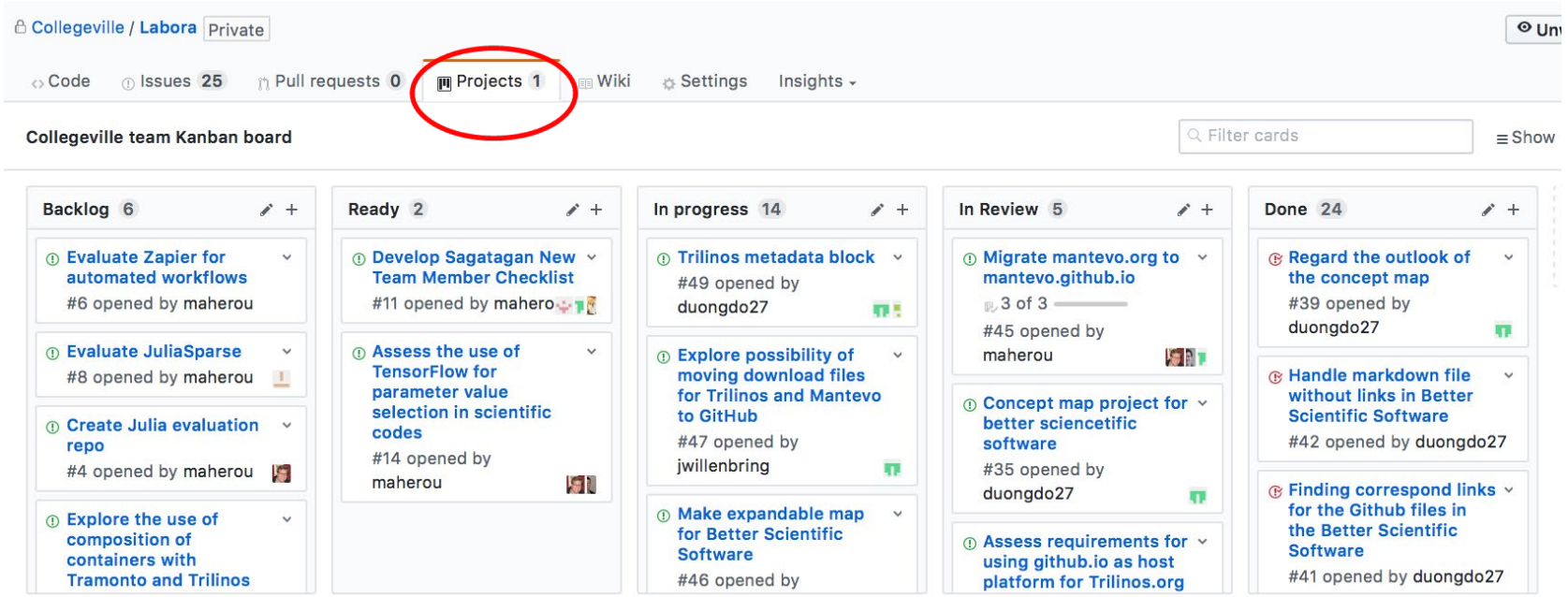
- Select the Issues tab.
- Click on New Issue.
- Select a classmate.
- Type in title: Pat Evans Initiation Checklist
- Add checklist items:
  - Use syntax:
    - [ ] Description

Spaces required



Step 4: Create Kanban Board

- Select Projects tab
- Click New Project
- Use title
  - Team Kanban board
- Add these columns:
  - Backlog, Ready, In progress, In review, Done.
- Click on +Add cards (upper right).
  - Move each issue to the proper Kanban column



Next Steps: Real Life

- Create a GitHub Org and set of repos for your team:
  - Each team member has an individual repo.
  - Each project has a repo.
  - One special repo for issues.
- Track all work:
  - Use checklists for initiation, exit, any big new effort.
  - Create Kanban board. Keep it current.
  - Aggregate related issues using milestones.
- Drive meetings using Kanban board.
- Adapt this approach to meet your needs.
- When you start to get sloppy, get back on track.