Agile Methodologies & Useful GitHub Tools

Presented at
ATPESC 2019

James M. Willenbring
Senior Member of R&D Technical Staff
Sandia National Laboratories

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

IDEAS productivity

License, citation, and acknowledgments

License and Citation

- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
- Requested citation: James M. Willenbring, Agile Methodologies & Useful GitHub Tools, tutorial, in Argonne Training Program on Extreme-Scale Computing (ATPESC) 2019. DOI: TBD.

Acknowledgements

- 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.
- Sandia National Laboratories is a multimission 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. SAND2019-9328 TR.
- 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.

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

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

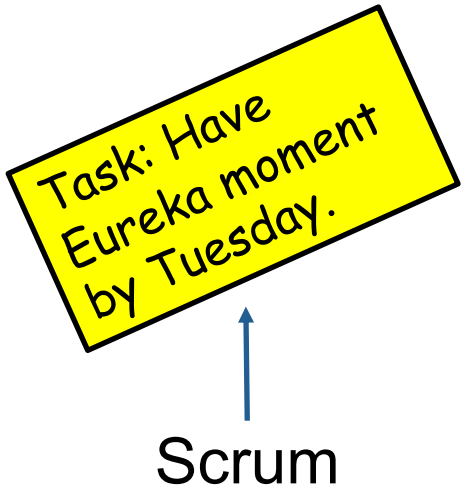
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art 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.

Why Agile?

- Well adept to scientific software efforts (when tailored correctly)
  - Lighter-weight than “traditional” approaches
  - Provides meaningful structure that promotes
    - Productivity
    - Productization
    - Sustainability
    - Flexibility in requirements
    - Communication

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".

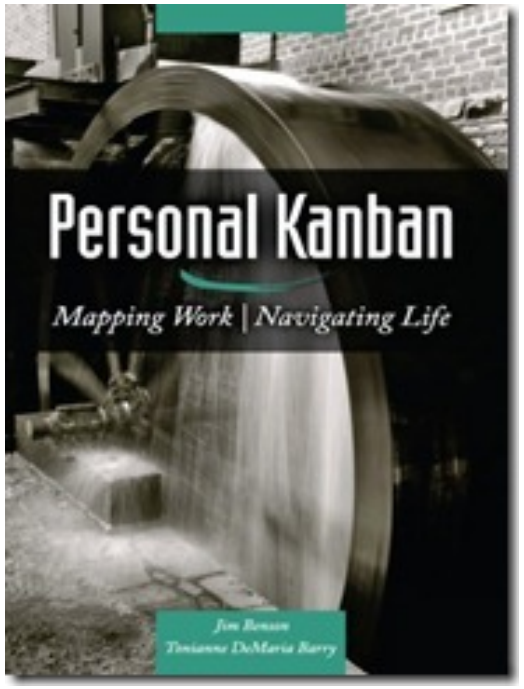
Notes:

- 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.



http://www.personalkanban.com

Kanban tools

- Wall, whiteboard, blackboard: Basic approach.
- Software, cloud-based:
  - Trello, JIRA, GitHub Issues.
  - 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?

- Personal question.
- Approach: Start with 2 or 3. See how it goes.
- 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)
- 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 as needed

- Example:
  - Add a validation test suite → Add test harness, add test A, B, etc.

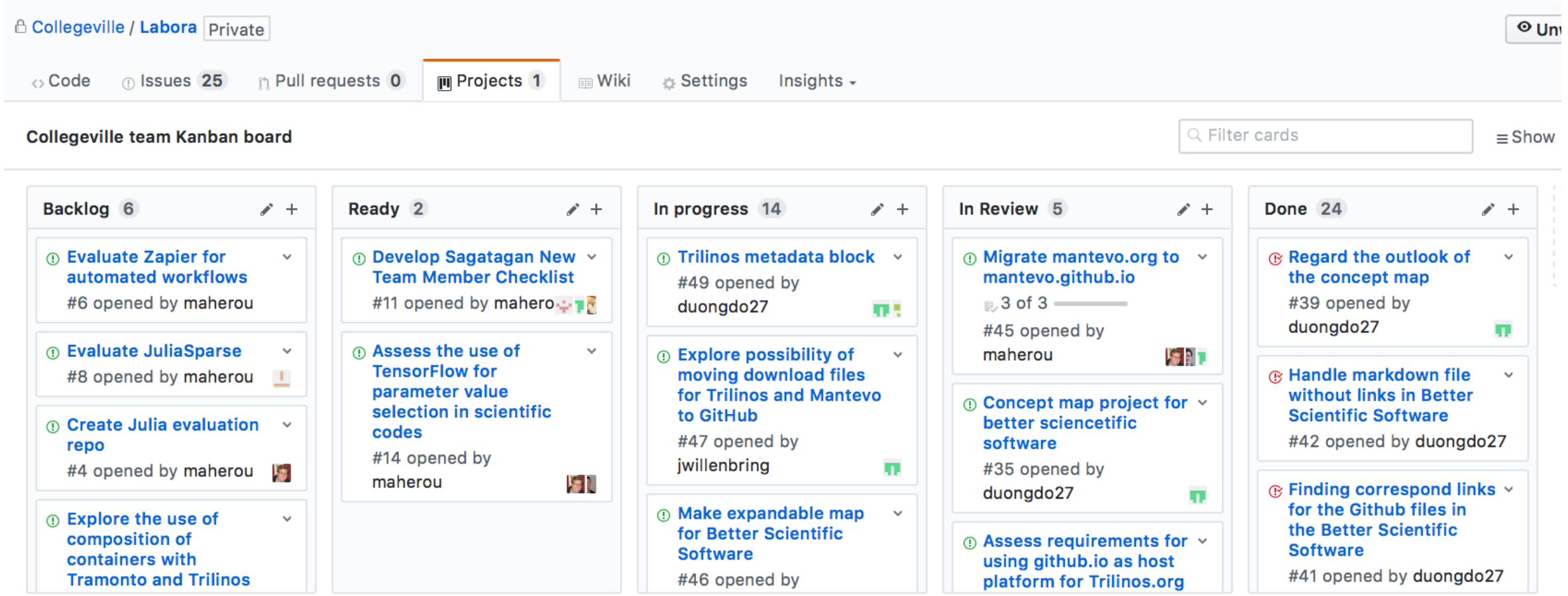
Building on Kanban

- 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

Samples from Collegeville Org: Kanban Board



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Kanban in GitHub

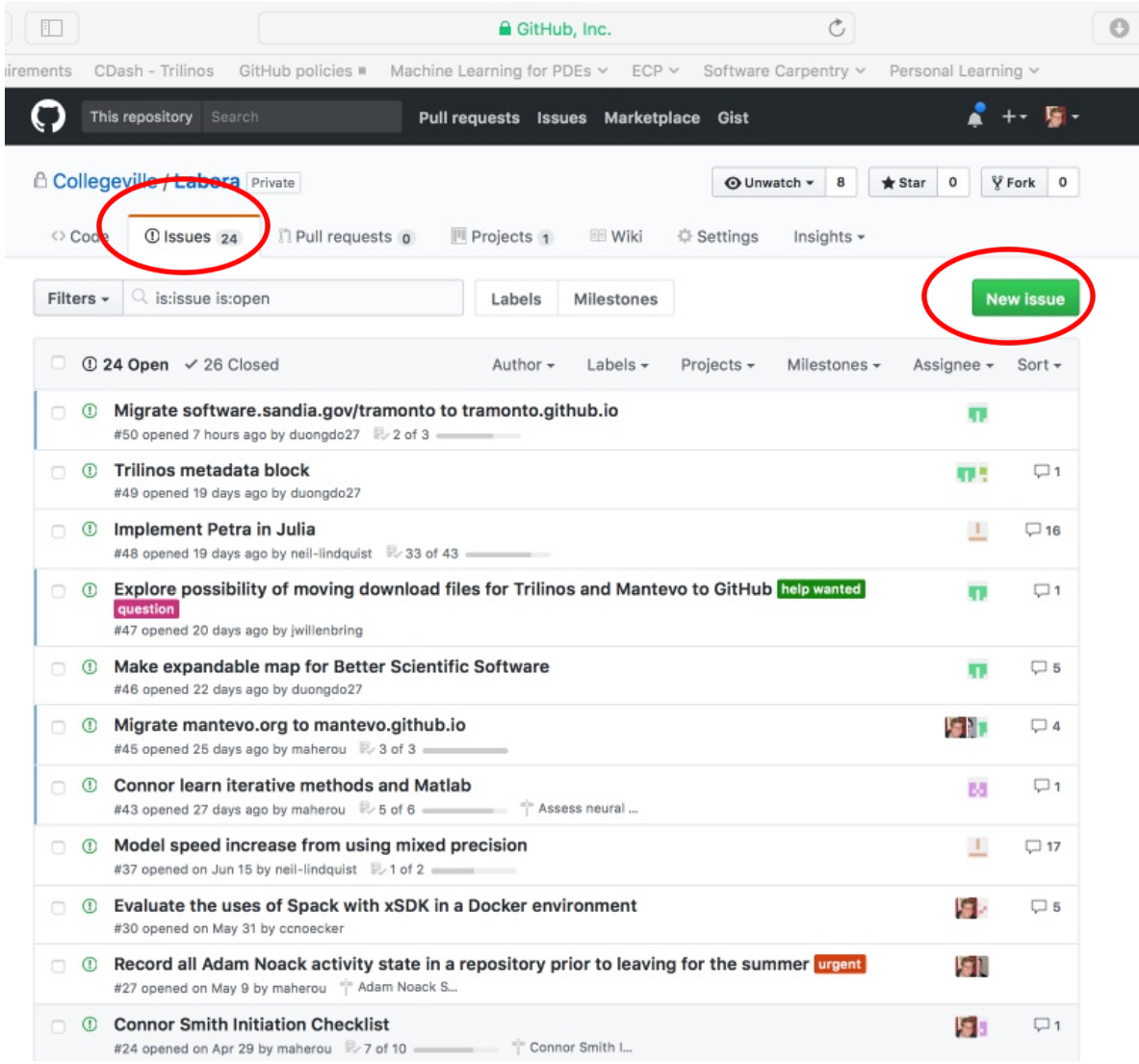
- GitHub supports basic Agile development workflows
  - Filing issues
    - @mention
  - Kanban board
  - Projects
- GitHub lacks more advanced features
  - Dependencies between issues
    - You can reference one issue in another
  - Advanced notification schemes
  - Custom fields
    - You can create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

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 and enter 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.
- Note: You can add collaborators
  - Settings -> Collaborators
  - Type GitHub ID (not email address).

Step 2: Create Issues

- Select the Issues tab.
- Click on New Issue.
- Type in task statement
  - Type in title only.
  - Also useful: Labels
    - Categories for grouping issues by type.
- Click Submit new issue
- Repeat.

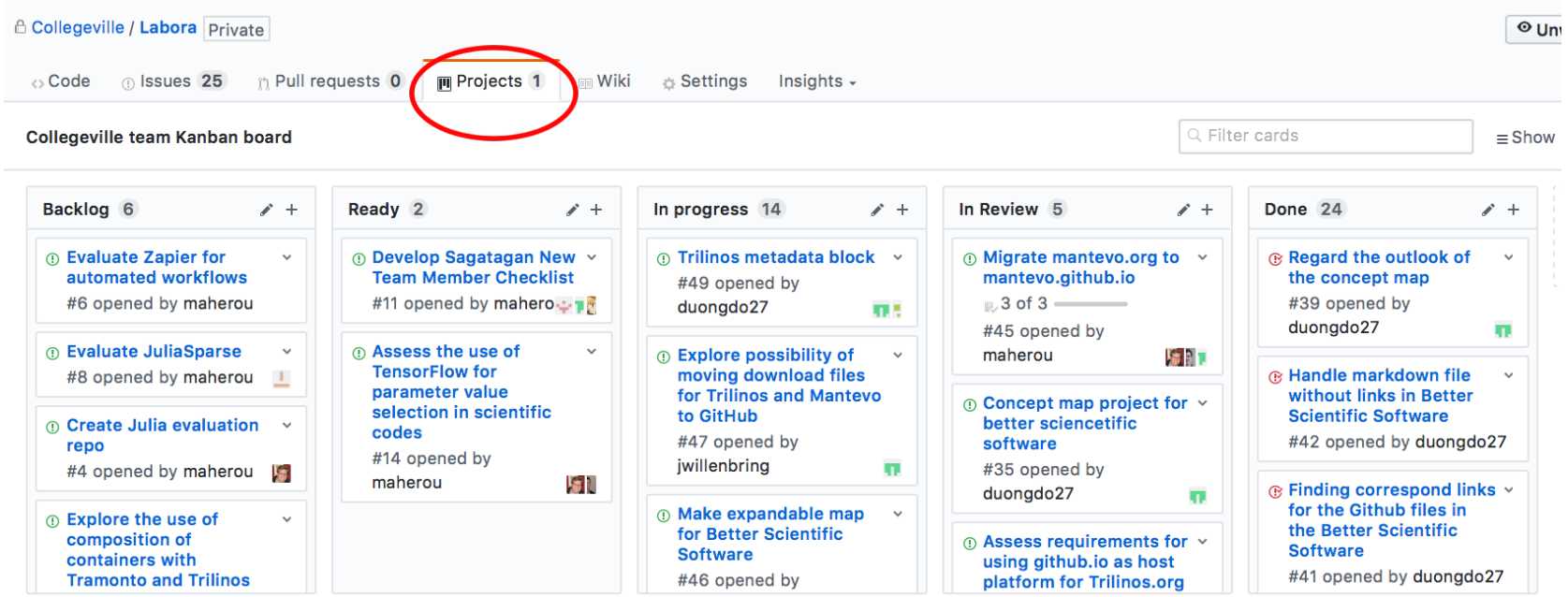


IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Step 3: 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



IDEAS productivity

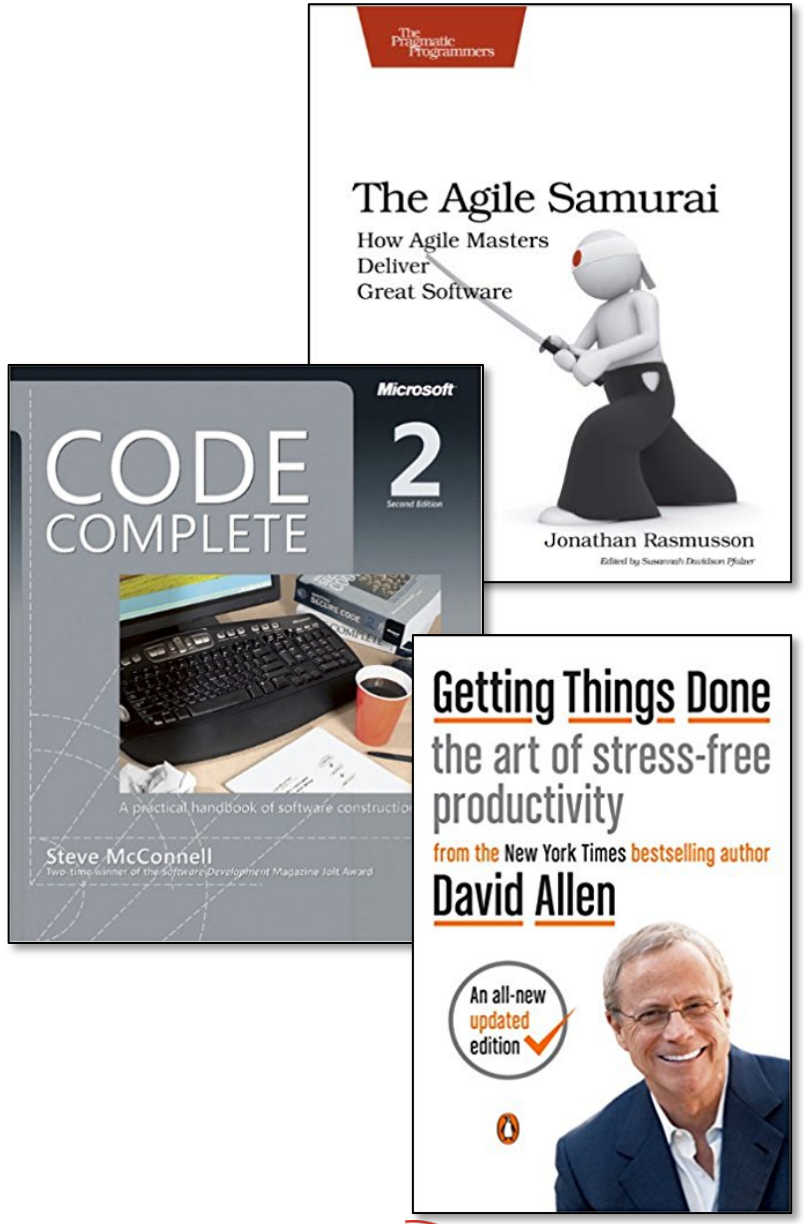
ECP EXASCALE COMPUTING PROJECT

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.

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.*
- **Getting Things Done: The Art of Stress-Free Productivity,** David Allen
  - http://a.co/22EPvt6
  - A classic in the personal productivity literature



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Managing issues: Fundamental software process

Continual improvement

- Issue: Bug report, feature request
- Approaches:
  - Short-term memory, office notepad
  - ToDo.txt on computer desktop (1 person)
  - Issues.txt in repository root (small co-located team)
  - …
  - Web-based tool + Kanban (distributed, larger team)
  - Web-based tool + Scrum (full-time dev team)

Informal, less training

Formal, more training

Key Team Management Elements

- **Checklists:**
  - Initiation, Transition, Exit
- **Policies:**
  - How team conducts its work
- **Issue tracking system:**
  - All work tracked, visible to team
  - Milestones: Aggregate related issues.
  - Kanban board
  - Regular meetings, updates

Project: ATPESC Eigen

- Four tasks:
  - Define requirements.
  - Develop design document.
  - Write test driver.
  - Write source code to make test pass.
- Notes:
  - You will have many tasks in a real project.
  - Tasks are called issues in GitHub.
  - Good reference: The Agile Samurai