Better (Small) Scientific Software Teams

SIAM CSE
Atlanta, GA
February 28, 2017

Tutorial slides available at: *http://bit.ly/siam-cse17-mt3*

Michael A. Heroux, Sandia National Laboratories

Acknowledgments

- Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2016-8466 C.

Outline

- Introduction
- Small Team Models, Challenges.
- Agile workflow management for small teams
  - Intro to terminology and approaches
  - Overview of Kanban
  - Free tools: Trello, GitHub.
- Hands on: Issue tracking via Kanban in GitHub.

Objectives

- Productivity – Output per unit input.
- Sustainability – The future cost of usability.
- Goals for today:
  - Learn how to improve
    - Developer productivity.
    - Software sustainability.
  - For the purposes of better scientific productivity,
  - Using tools, processes and practices.

Tradeoffs: Better, faster, cheaper

- "Better, faster, cheaper: Pick two of the three."
  - Scenario: (Today)
    You are behind in developing a sophisticated new model in your software that you want to use for results in an upcoming paper.
  - Which of these could be reasonable choices?
    - Develop a simpler model for the paper.
    - Set other work aside and spend more time on development.
    - Ask for an extension on the paper deadline.
    - Develop sophisticated model, but don't test its correctness.
    - Develop sophisticated model, but don't document it or check it in.

Improved developer productivity

"Better, faster, cheaper: Pick all three." – Near term.

Scenario: (6 months later)
After investing in **developer productivity improvements**, you are on time in developing a sophisticated new model in your software that you want to use for results in an upcoming paper.

Invest in developer tools, processes, practices.

Improved software sustainability

"Better, faster, cheaper: Pick all three." – Long term.

Scenario: (3 years later)
After investing in **software sustainability improvements**, you are on time in developing **several** sophisticated new models in your software that you want to use for results in upcoming papers.

Invest in testing, documentation, integration for long-term software usability.

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.

Large team challenges

- Composed of small teams (and all the challenges).
- Additional interaction challenges.

Small team challenges

- 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.

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.
  - https://software.sandia.gov/trilinos/developer/sqp/checklists/index.html
- Steady state: Policy-driven.
  - Example: xSDK Community policies.
  - https://xsdk.info/policies/

xSDK Mandatory Policies

Must:

- M1. Support xSDK community GNU Autoconf or CMake options [4].
- M2. Provide a comprehensive test suite.
- M3. Employ user-provided MPI communicator.
- M4. Give best effort at portability to key architectures.
- M5. Provide a documented, reliable way to contact the development team.
- M6. Respect system resources and settings made by other previously called packages.
- M7. Come with an open source license.
- M8. Provide a runtime API to return the current version number of the software.
- M9. Use a limited and well-defined symbol, macro, library, and include file name space.
- M10. Provide an accessible repository (not necessarily publicly available).
- M11. Have no hardwired print or IO statements.
- M12. Allow installing, building, and linking against an outside copy of external software.
- M13. Install headers and libraries under `<prefix>/include/` and `<prefix>/lib/`.
- M14. Be buildable using 64 bit pointers. 32 bit is optional.

xSDK Recommended Policies

Should:

- R1. Have a public repository.
- R2. Possible to run test suite under valgrind in order to test for memory corruption issues.
- R3. Adopt and document consistent system for error conditions/exceptions.
- R4. Free all system resources it has acquired as soon as they are no longer needed.
- R5. Provide a mechanism to export ordered list of library dependencies.

Your checklists & policies?

- Checklist: New team member?
- Policies: Ongoing work?
- Checklist: Before someone departs?

Collaborative Work Management

Managing with Kanban

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)
- IDEAS project:
  - Jira Agile + Confluence: Turnkey web platform (ACME too)
  - Kanban: Simplest of widely known Agile SW dev processes

Informal, less training

Formal, more training

IDEAS productivity

Kanban principles

- Limit number of "In Progress" tasks
- 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

Task: Have Eureka moment by Tuesday.

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".

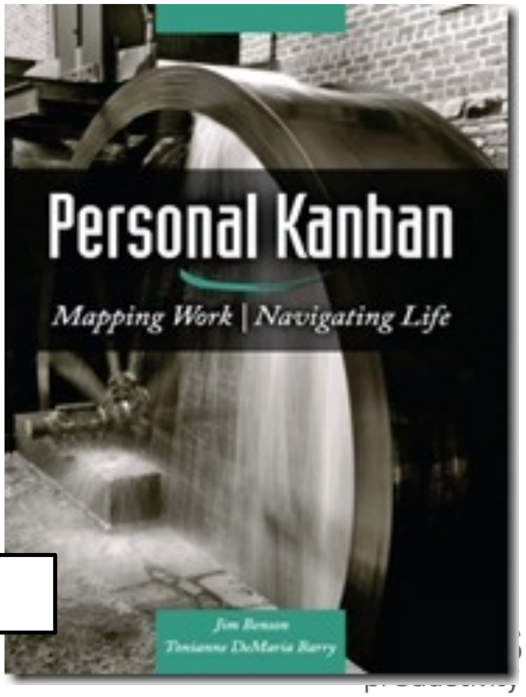
Notes:

- 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.
  - Tasks I won't do.

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, iPhone, iPad).
  - Can add, view, update, anytime, anywhere.

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.

Importance of “In Progress” concept for you

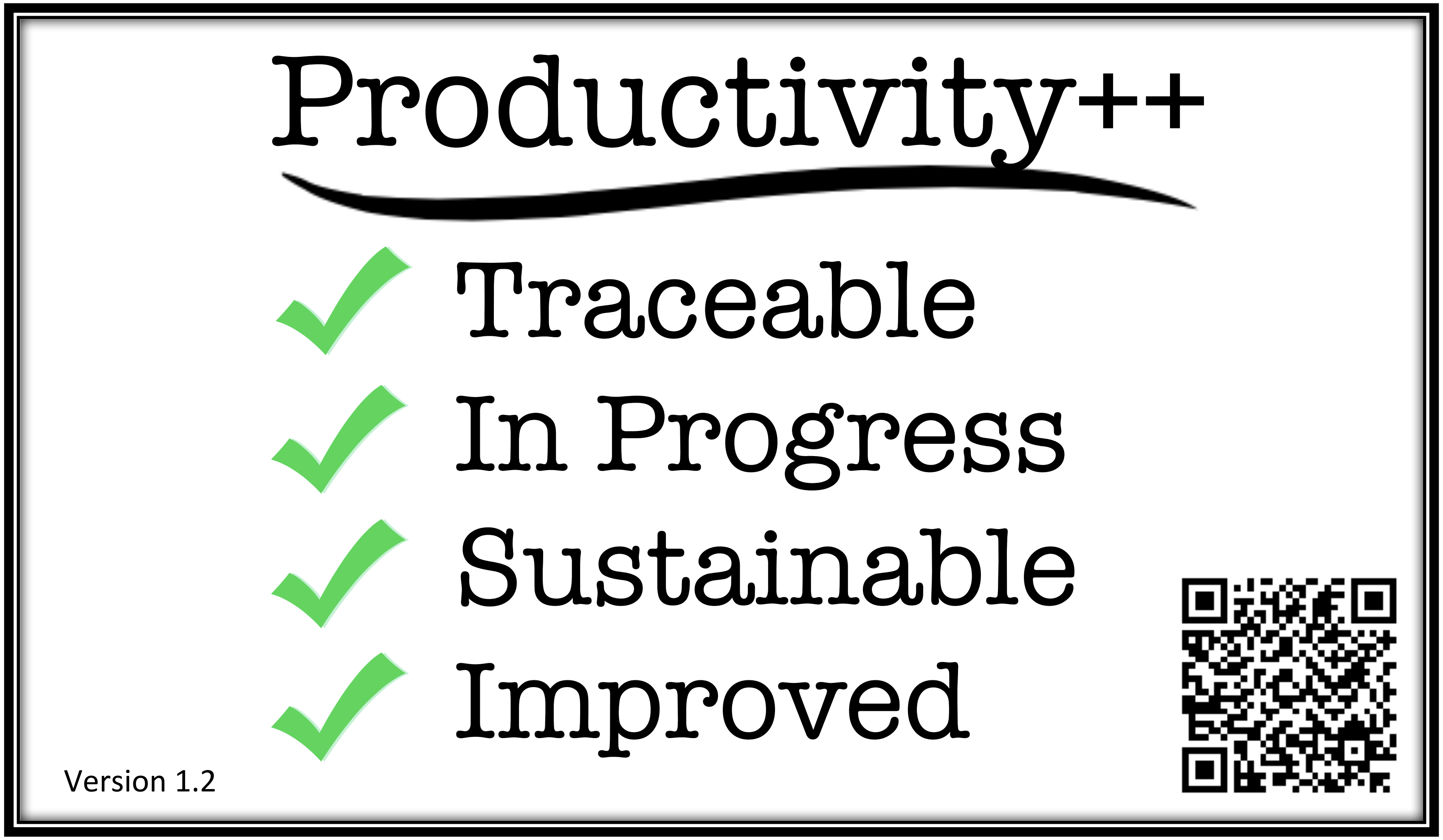
- Junior community members:
  - Less control over task.
  - Given by supervisor.
- In Progress column: Protects you.
  - If asked to take on another task, respond:
    - Is this important enough to become less efficient?
    - Sometimes it is.

Personal Expectations

Calling out the best in team members

(Personal) Productivity++ Initiative

Ask: *Is My Work ________ ?*



Productivity++

- ✓ Traceable
- ✓ In Progress
- ✓ Sustainable
- ✓ Improved

Version 1.2

https://github.com/trilinos/Trilinos/wiki/Productivity---Initiative

Project: Atlanta

- 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

Hands on issue tracking: Go to Github

- Goal: Learn how to set up communication in GitHub:
  - Pre-steps: Set up a repository, communication paths.
  - Create:
    - Issues – Any task you want to accomplish.
    - Labels – Categories for grouping issues by type.
    - Milestones – Groups of issues for tracking progress.
    - Projects – Kanban board for tracking progress.

Hands on issue tracking: Go to Github

- https://github.com/
- Create new (public) repository called atlanta.
- Add collaborators (pick your neighbor).
  - Settings -> Collaborators
  - Type Github ID (not email address).
- Set up a Google Groups email address.
  - groups.google.com
  - Email address: project-name@googlegroups.com
- Add email notification:
  - Settings -> Integrations & Services -> Add service -> Email
  - Type in address, no Secret needed, uncheck ”Send from author”

Other resources

The Agile Samurai: How Agile Masters Deliver Great Software (Pragmatic Programmers), Jonathan Rasmusson. Excellent, readable book on Agile methodologies. https://www.amazon.com/Agile-Samurai-Software-Pragmatic-Programmers/dp/1934356581

Code Complete, Steve McConnell. Excellent testing advice. His description of Structure Basis Testing is good, and it is a simple concept: Write one test for each logic path through your code.

Outline

Part I: 9:10-10:50 am

- [10 min] **Background, introductions, objectives, setup**
- [15 min] **Why effective software practices are essential for CSE projects**
- [25 min] **Software licensing**
- [50 min] **Effective models, tools, processes, and practices for small teams, including agile workflow management**
  - Interactive exercises

Part II: 1:30-3:10 pm

- [25 min] **Reproducibility**
- [75 min] **Scientific software testing**
  - Automated testing and continuous integration
  - Interactive exercises for code coverage
    - Access to Linux environment with Git and GNU compiler suite