SOFTWARE ENGINEERING BEST PRACTICES

Why, What and How

Michael A. Heroux
Senior Scientist, Sandia National Laboratories
Scientist in Residence, St. John’s University, MN

better scientific software

So your code will see the future.

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

IDEAS productivity

Acknowledgments

- 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.

My Background

- 1998 – now: Staff member at Sandia National Labs
  - Lead these projects:
    - ECP SW Technology since Nov 2017.
    - Trilinos: collection of scientific libraries – trilinos.github.io.
    - Mantevo: "Miniapps" project for HPC co-design – mantevo.github.io
    - IDEAS Productivity: Scientific Productivity and sustainability – ideas-productivity.org
    - HPCG Benchmark: Complementary benchmark for Top 500 – hpcg-benchmark.org
    - Better Scientific Software: Portal and content for productivity and sustainability – bssw.io
    - SC18 Reproducibility Chair, SC19 special role
  - Concurrent: Scientist in Residence, St. John's University, MN USA
- 1988 – 1998: Staff member at Cray Research
  - 88 – 93: Math libraries developer, sparse solvers, LAPACK, BLAS: LIBSCI
  - 93 – 95: Application analyst, computational engineering group: FIDAP, Fluent, Star-CD.
  - 95 – 98: Scalable systems applications specialist: Cray T3E "MPP".

Outline

- My perspective.
- Why we increasingly care about better software practices.
- Barely-sufficient software engineering.
- Lightweight small team management.
- Agile methodology and scientific software.
- Planning: An opportunity for improvement for scientific SW.
- Two practical techniques: Stories, lightweight design docs.
- How to introduce better practices.
- Appealing to the best in each of us: Personal expectations.

My Perspective

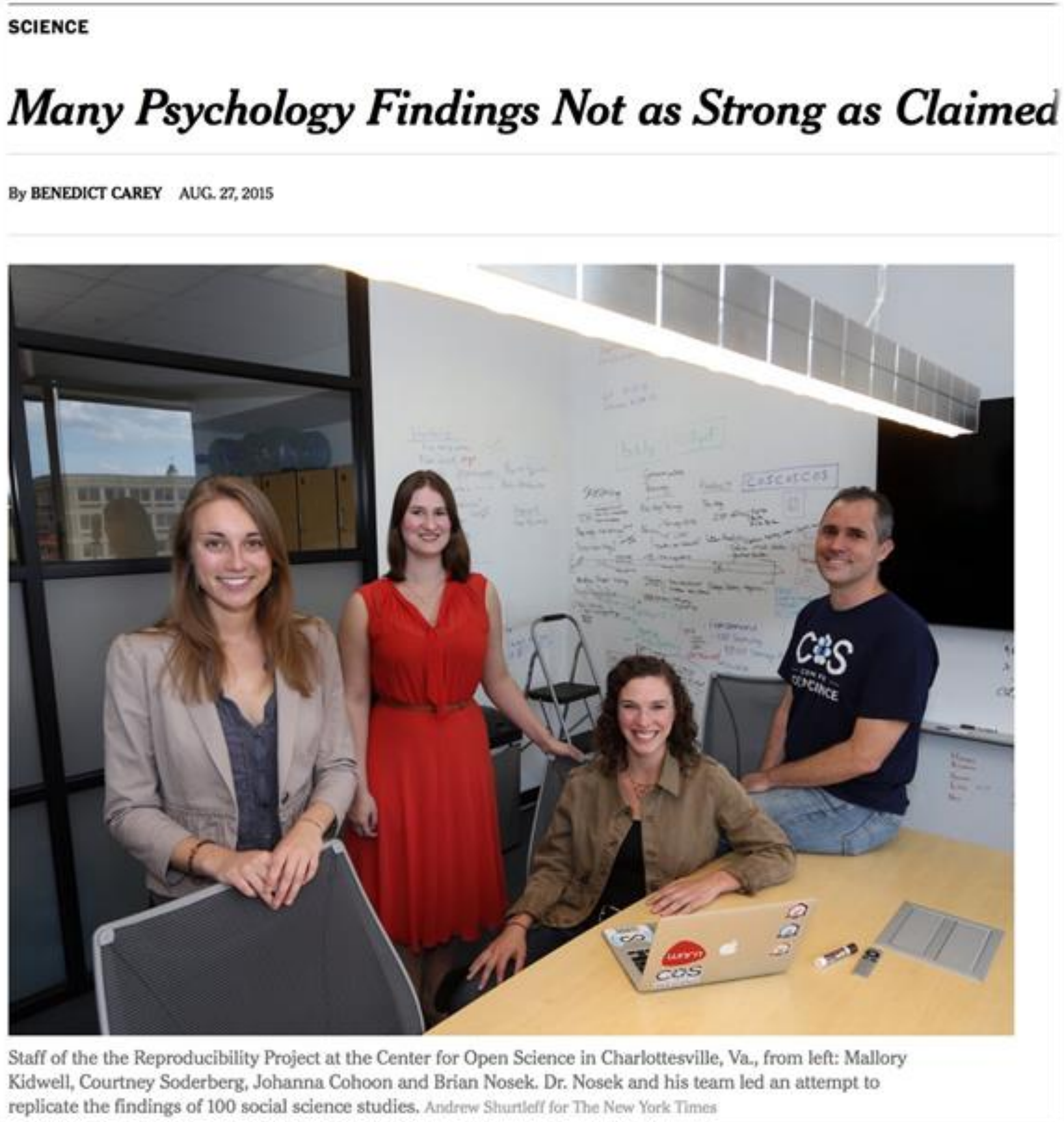
- Regarding observations on opportunities to improve:
  - *More like a scarred veteran than expert.*

- Regarding software tools, processes, practices improvements:
  - *More like a carpenter than expert.*

Transparency & Reproducibility

Why we pursue better software practices

Transparency & Reproducibility



SCIENCE

Many Psychology Findings Not as Strong as Claimed

By BENEDICT CAREY   AUG. 27, 2015

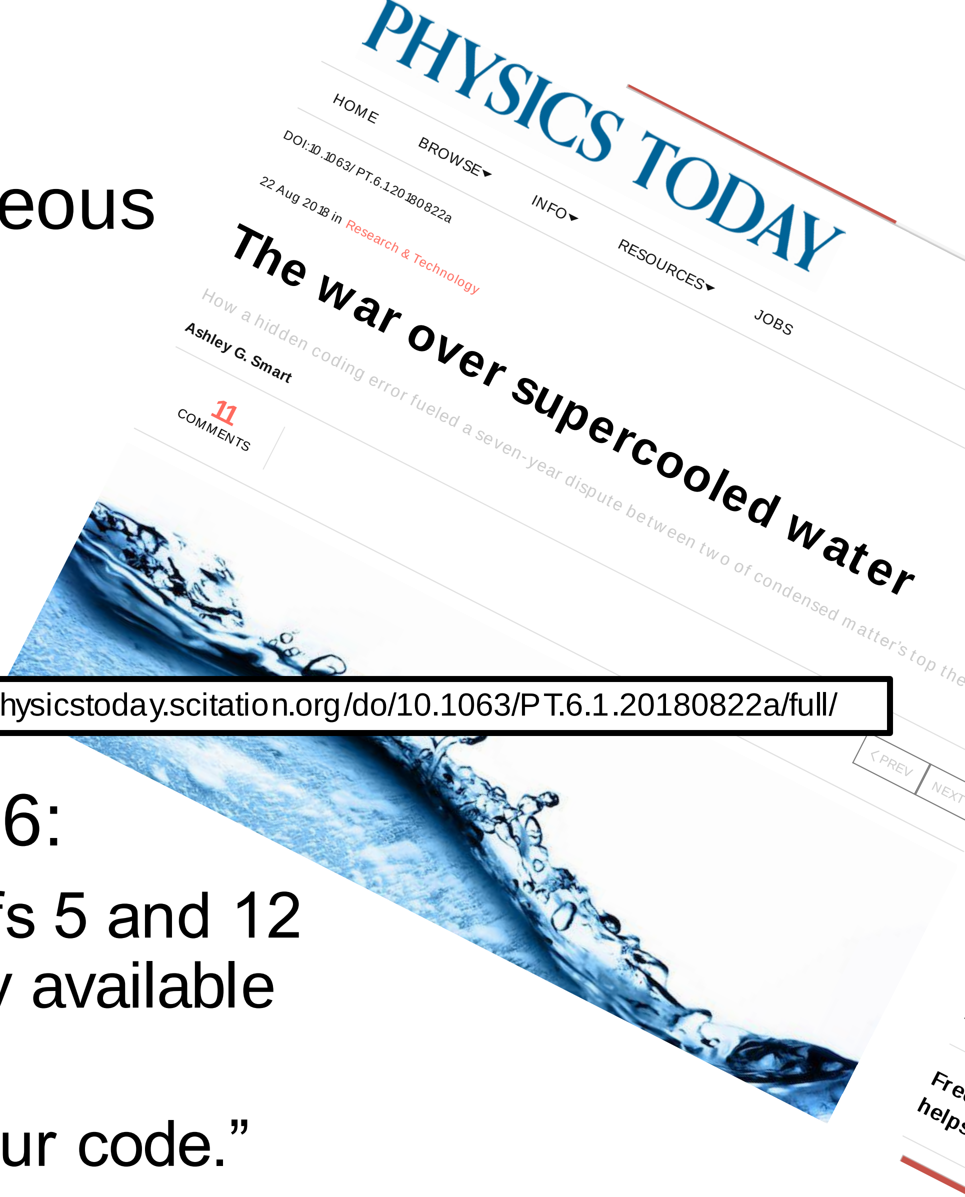
Staff of the the Reproducibility Project at the Center for Open Science in Charlottesville, Va., from left: Mallory Kidwell, Courtney Soderberg, Johanna Cohoon and Brian Nosek. Dr. Nosek and his team led an attempt to replicate the findings of 100 social science studies. Andrew Shurtleff for The New York Times

- NY Times highlights “problems”.
- Only one of many cited examples.
- Computational science ***had*** been spared this “spotlight”.

http://www.nytimes.com/2015/08/28/science/many-social-science-findings-not-as-strong-as-claimed-study-says.html

Computational Science Example

- Behavior of pure water just above homogeneous nucleation temperature (~ - 40 C/F).
- Debenedetti/Princeton (2009):
  - 2 possible phases: High or low density.
- Chandler/Berkeley (2011):
  - Only 1 phase: High density.
- No sharing of details across teams until 2016:
  - Chandler in Nature: “LAMMPS codes used in refs 5 and 12 are standard and documented, with scripts freely available upon request.”
  - Debenedetti with colleague Palmer: ”Send us your code.”
  - Received code after requests and appeal to Nature.



PHYSICS TODAY

HOME BROWSE▾ INFO▾ RESOURCES▾ JOBS

DOI:10.1063/PT.6.1.20180822a

22 Aug 2018 in Research & Technology

The war over supercooled water

How a hidden coding error fueled a seven-year dispute between two of condensed matter's top the

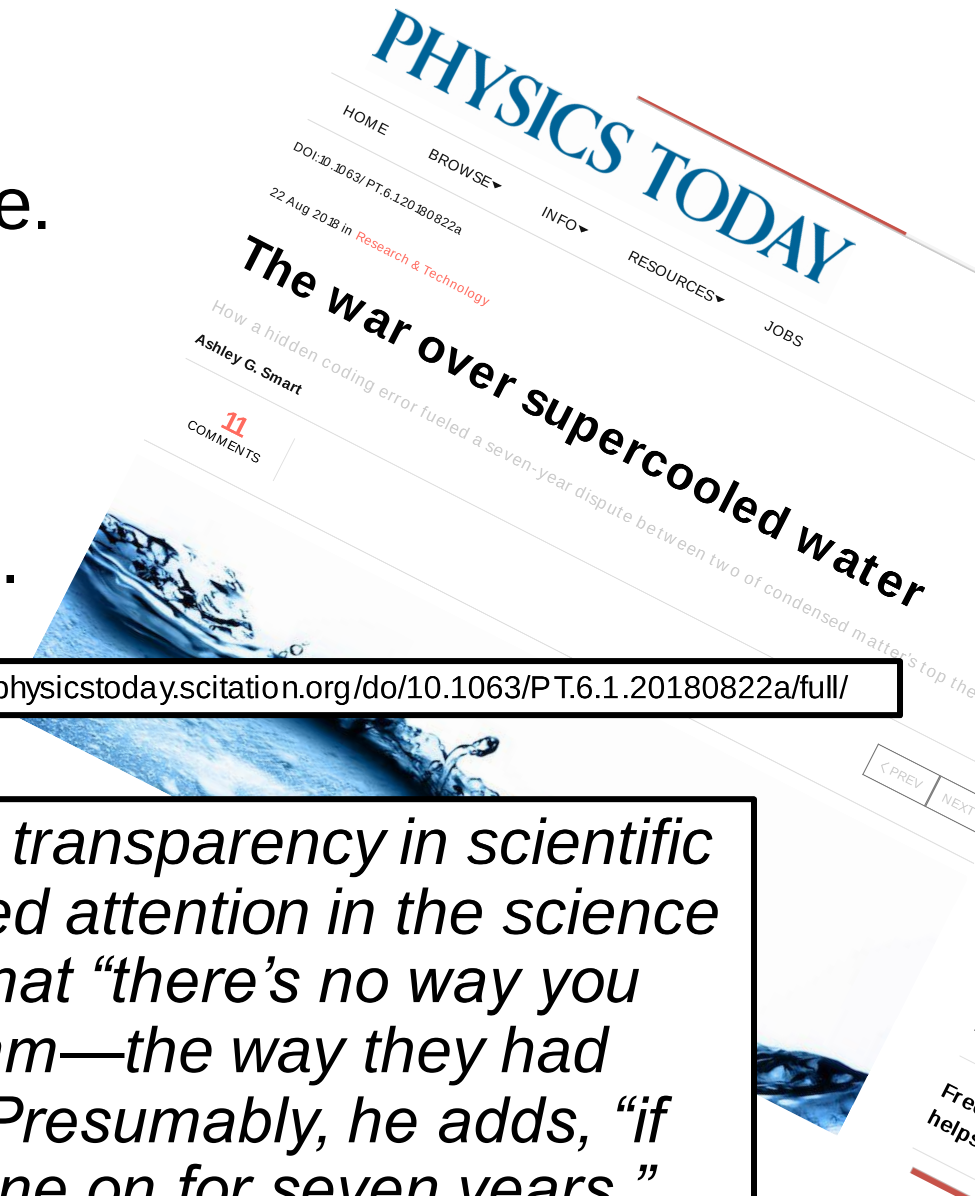
Ashley G. Smart

11 COMMENTS

Source: https://physicstoday.scitation.org/do/10.1063/PT.6.1.20180822a/full/

Computational Science Example

- Palmer located bug/feature in Berkeley code.
- Used to speed up LAMMPS execution.
- Replaced with more standard approach.
- Obtained result similar to Debenedetti 2009.
- Resolution took 7 years.



PHYSICS TODAY

HOME BROWSE▾ INFO▾ RESOURCES▾ JOBS

DOI:10.1063/PT.6.1.20180822a

22 Aug 2018 in Research & Technology

The war over supercooled water

How a hidden coding error fueled a seven-year dispute between two of condensed matter's top the

Ashley G. Smart

11 COMMENTS

‹ PREV NEXT

Fre

helps

Source: https://physicstoday.scitation.org/do/10.1063/PT.6.1.20180822a/full/

For Palmer, the ordeal exemplifies the importance of transparency in scientific research, an issue that has recently drawn heightened attention in the science community. “One of the real travesties,” he says, is that “there’s no way you could have reproduced [the Berkeley team’s] algorithm—the way they had implemented their code—from reading their paper.” Presumably, he adds, “if this had been disclosed, this saga might not have gone on for seven years.”

Publication Trends

Increased Emphasis on Transparency & Reproducibility

ACM TOMS Reproducible Computational Results (RCR)

- Submission: Optional RCR option.
- Standard reviewer assignment: Nothing changes.
- RCR reviewer assignment:
  - Concurrent with standard reviews.
  - As early as possible in review process.
  - Known to and works with authors during the RCR process.
- RCR process:
  - Multi-faceted approach, Bottom line: Trust the reviewer.
- Publication:
  - Reproducible Computational Results Designation.
  - The RCR referee acknowledged.
  - Review report appears with published manuscript.

SC18 Reproducibility Initiative

- Two appendices:
  - – Artifact description (AD).
    - Blue print for setting up your computational experiment.
    - Makes it easier to rerun computations in future.
    - AD appendix will be mandatory for SC19 paper submissions.
  - – Artifact Evaluation (AE).
    - Targets ”boutique” environments.
    - Improves trustworthiness when re-running hard, impossible.
- Details:
  - – https://sc19.supercomputing.org/submit/reproducibility-initiative/

Improving Trustworthiness at Scale

What if we can’t re-run a computational experiment?

Reproducibility and Supercomputing

Scenario:
You compute a “hero” calculation using 5M core-hours on Mira and submit your results for publication. During the review process, a referee questions the validity of your results. What options are feasible:

- The reviewer re-runs your code on a laptop or cluster.
- The reviewer re-runs your code on Mira.
- You re-run your code on Mira.
- Your results are rejected.
- Your results are accepted, but with risk.

Sources for meta-computations

- Synthetic operators with known:
  - Spectrum (Huge diagonals).
  - Rank (by constructions).
- Invariant subspaces:
  - Example: Positional/rotational invariance (structures).
- Conservation principles:
  - Example: Flux through a finite volume.
- General:
  - Pre-conditions, post-conditions, invariants.

Can you think of something for your problems?

Example: HPCG Benchmark

- Symmetry:
  - For any linear operator A, $x^TAy = y^TA^Tx$.
  - If A symmetric $A = A^T$, so $x^TAy = y^TAx$.
  - And $\boldsymbol{x^TAy - y^TAx = 0}$.
- HPCG computes the above expression for:
  - User matrix and the preconditioner.
  - Numerical detail: Need to scale by vector & matrix norms.

Coming to Your World Soon: Reproducibility Requirements

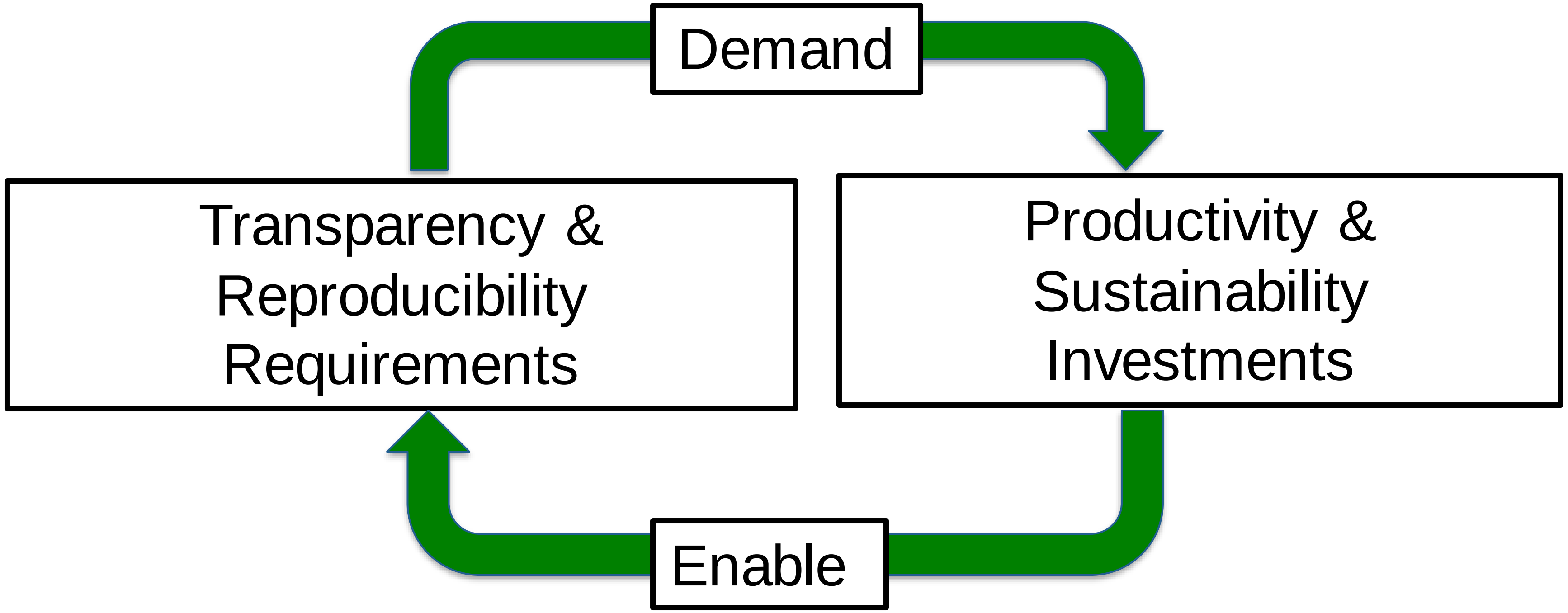
- These conferences have artifact evaluation appendices:
  - CGO, PPoPP, PACT, RTSS and SC.
  - http://fursin.net/reproducibility.html
- ACM ~~Replicated~~ Reproducible Computational Results (RCR).
  - ACM TOMS, TOMACS.
  - http://toms.acm.org/replicated-computational-results.cfm
- ACM Badging.
  - https://www.acm.org/publications/policies/artifact-review-badging
- NISO Committee on Reproducibility and Badging.
  - https://www.niso.org/niso-io/2019/01/new-niso-project-badging-scheme-reproducibility-computational-and-computing
  - Publishers: ACM, IEEE, figshare, STM, Reed Elsevier, Springer Nature

Questions, comments?

Better Productivity and Sustainability

Essential for affordable reproducibility

Incentives Demand Investments, Enabled by Investments


Demand
Transparency & Reproducibility Requirements
Productivity & Sustainability Investments
Enable

Common statement: “I would love to do a better job on my software, but I need to:

- Get this paper submitted.
- Complete this project task.
- Do something my employer values more.

Goal: Change incentives to include value of better software, better science.

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.

Which of These Enhance Reproducibility?

- Code written by first-year, untrained grad student.
- Tuning for high performance.
- Dynamic parallelism of modern processors.
- Better software testing.
- Source code and versioning management.
- Investing in developer productivity.
- Investing in software sustainability.

Software Engineering Basics

Barely-sufficient Software Engineering

Barely-sufficient Software Engineering: Updated

1. Manage Source: GitHub, GitLab – Use GitHub Desktop Client.
2. Use Issue Tracking: GitHub Issues, Jira – Kanban Board.
3. Set up communication: Slack, Confluence, mail lists.
4. Set up, use checklists for repeated activities.
5. Create source-centric documentation: Readthedocs, Doxygen.
6. Use configuration management tools: CMake.
7. Write tests first, use for integration testing (via pull requests).
8. Make selective use of pair programming.
9. Do formal releases using semantic versioning: X.Y.Z.
10. Perform continual process improvement.

Michael A. Heroux and James M. Willenbring. 2009. Barely sufficient software engineering: 10 practices to improve your CSE software. In Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering (SECSE '09). IEEE Computer Society, Washington, DC, USA, 15-21. DOI=http://dx.doi.org/10.1109/SECSE.2009.5069157

https://dl.acm.org/citation.cfm?id=1556930

Small Teams

Ideas for managing transitions and ongoing 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.

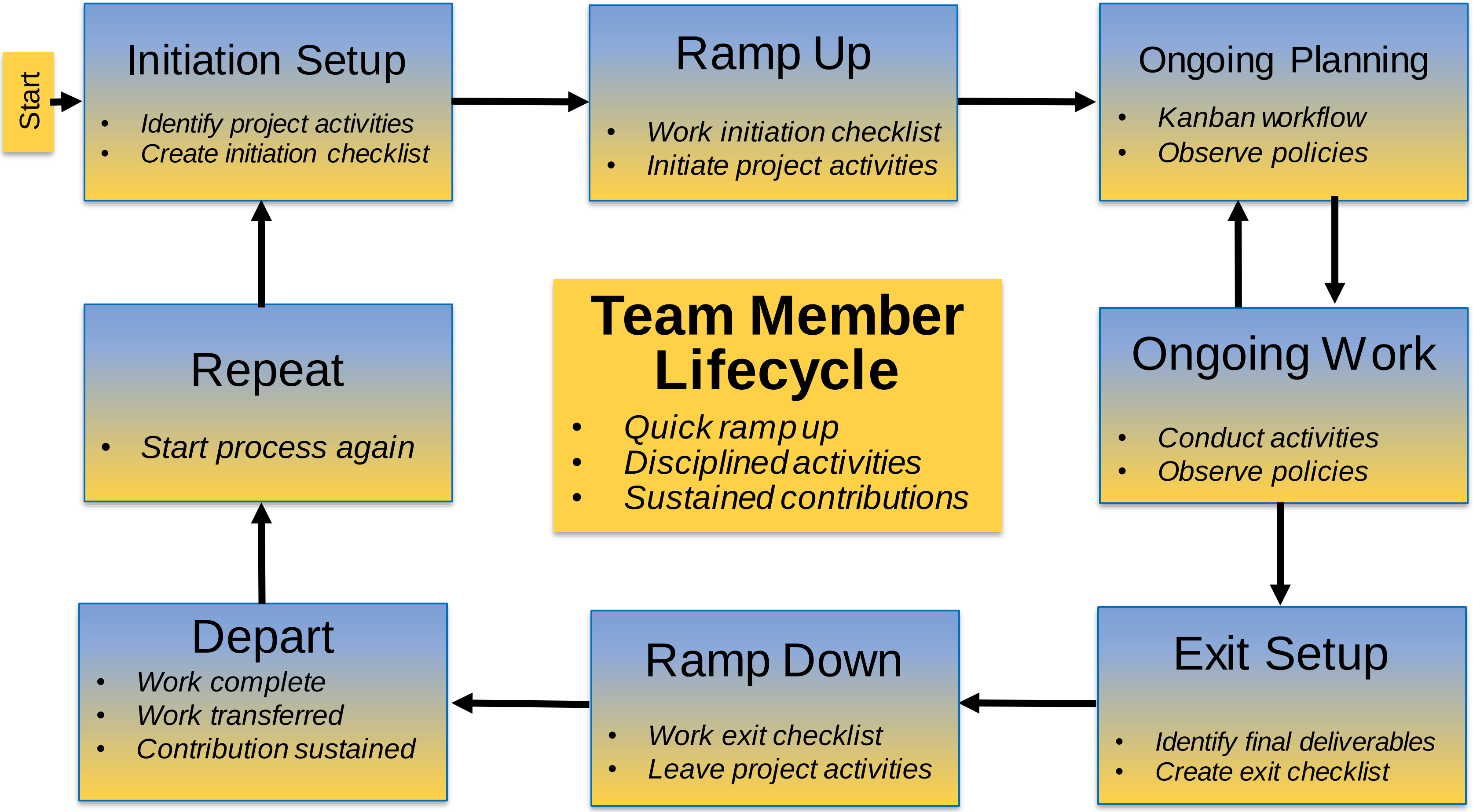
Large team challenges

- Composed of small teams (and all the challenges).
- Additional interaction challenges.
- Policies, regularly cultural exchanges important.

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.

Research Team Member Lifecycle



Start

Initiation Setup
- *Identify project activities*
- *Create initiation checklist*

Ramp Up
- *Work initiation checklist*
- *Initiate project activities*

Ongoing Planning
- *Kanban workflow*
- *Observe policies*

Ongoing Work
- *Conduct activities*
- *Observe policies*

Exit Setup
- *Identify final deliverables*
- *Create exit checklist*

Ramp Down
- *Work exit checklist*
- *Leave project activities*

Depart
- *Work complete*
- *Work transferred*
- *Contribution sustained*

Repeat
- *Start process again*

Team Member Lifecycle
- *Quick ramp up*
- *Disciplined activities*
- *Sustained contributions*

Checklists & Policies

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

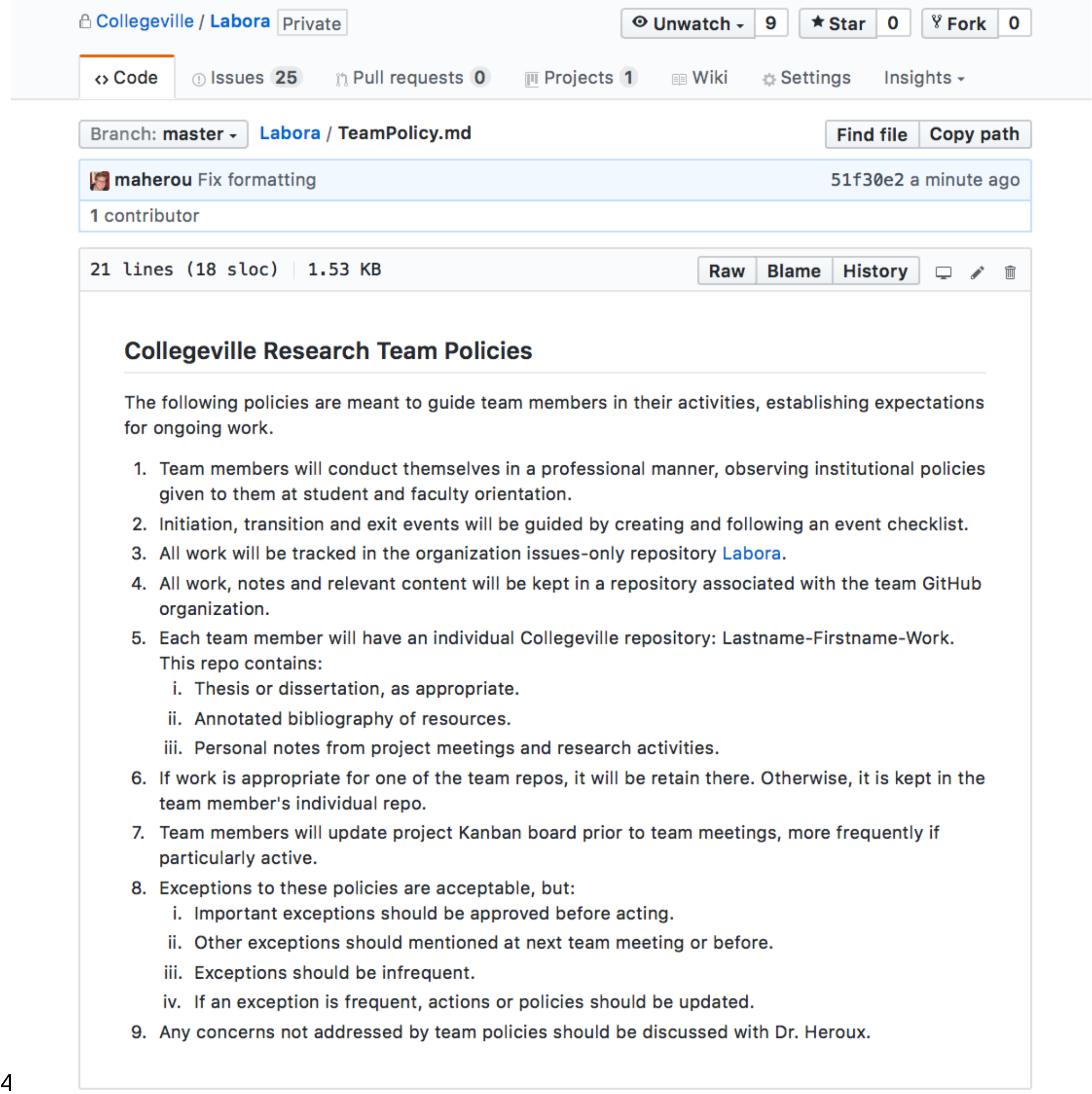
- New, departing team member checklists:
  - Example: Trilinos New Developer Checklist.
  - https://software.sandia.gov/trilinos/developer/sqp/checklists/index.html
- Steady state: Policy-driven.
  - Example: xSDK Community policies.
  - https://xsdk.info/policies/

Your checklists & policies?

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

- Discuss in your local group, type in the Google Doc.

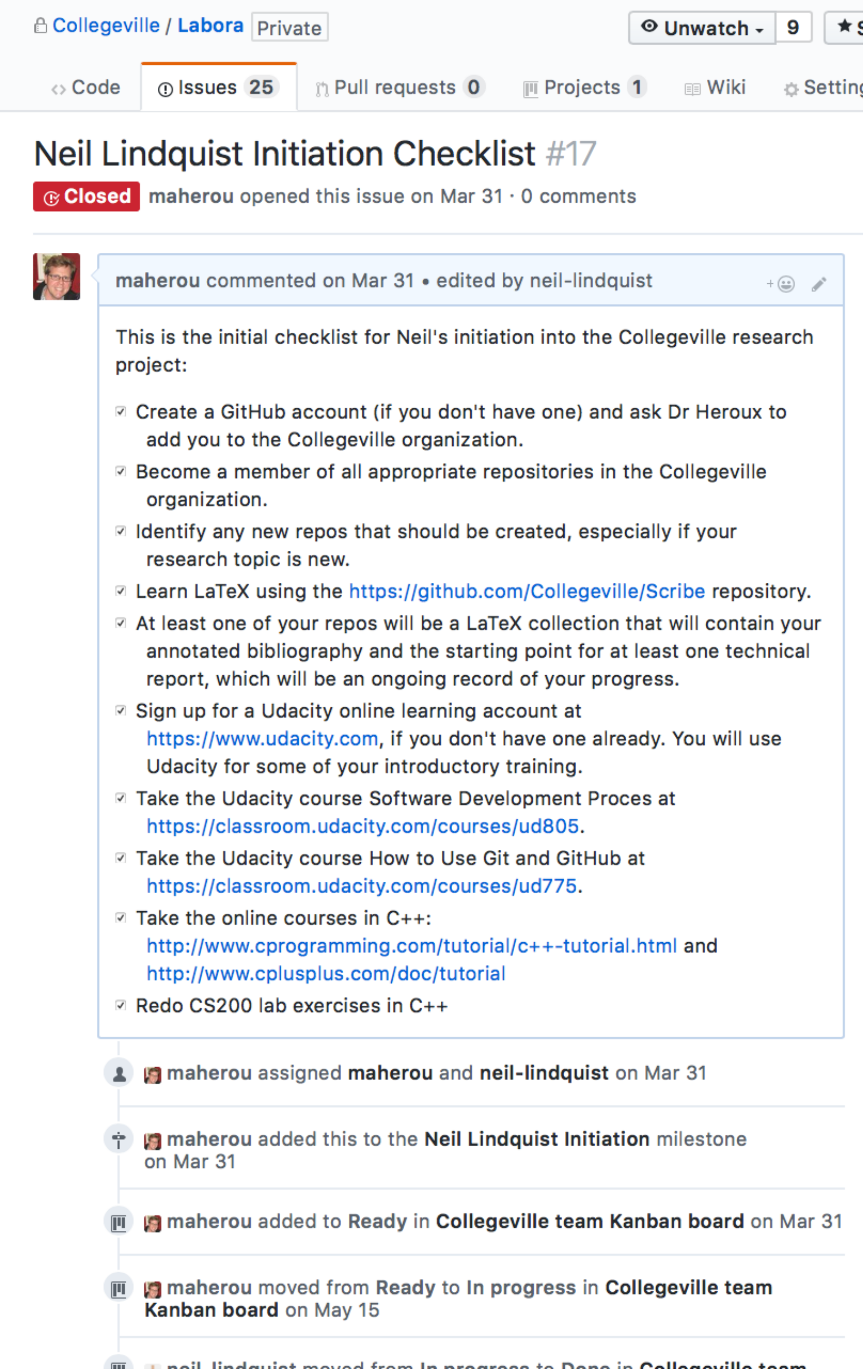
Samples from Collegeville Org: Policies, Initiation Checklist



Collegeville Research Team Policies

The following policies are meant to guide team members in their activities, establishing expectations for ongoing work.

1. Team members will conduct themselves in a professional manner, observing institutional policies given to them at student and faculty orientation.
2. Initiation, transition and exit events will be guided by creating and following an event checklist.
3. All work will be tracked in the organization issues-only repository Labora.
4. All work, notes and relevant content will be kept in a repository associated with the team GitHub organization.
5. Each team member will have an individual Collegeville repository: Lastname-Firstname-Work. This repo contains:
   i. Thesis or dissertation, as appropriate.
   ii. Annotated bibliography of resources.
   iii. Personal notes from project meetings and research activities.
6. If work is appropriate for one of the team repos, it will be retain there. Otherwise, it is kept in the team member's individual repo.
7. Team members will update project Kanban board prior to team meetings, more frequently if particularly active.
8. Exceptions to these policies are acceptable, but:
   i. Important exceptions should be approved before acting.
   ii. Other exceptions should mentioned at next team meeting or before.
   iii. Exceptions should be infrequent.
   iv. If an exception is frequent, actions or policies should be updated.
9. Any concerns not addressed by team policies should be discussed with Dr. Heroux.



Neil Lindquist Initiation Checklist #17

Closed maherou opened this issue on Mar 31 · 0 comments

maherou commented on Mar 31 • edited by neil-lindquist

This is the initial checklist for Neil's initiation into the Collegeville research project:

- ☑ Create a GitHub account (if you don't have one) and ask Dr Heroux to add you to the Collegeville organization.
- ☑ Become a member of all appropriate repositories in the Collegeville organization.
- ☑ Identify any new repos that should be created, especially if your research topic is new.
- ☑ Learn LaTeX using the https://github.com/Collegeville/Scribe repository.
- ☑ At least one of your repos will be a LaTeX collection that will contain your annotated bibliography and the starting point for at least one technical report, which will be an ongoing record of your progress.
- ☑ Sign up for a Udacity online learning account at https://www.udacity.com, if you don't have one already. You will use Udacity for some of your introductory training.
- ☑ Take the Udacity course Software Development Proces at https://classroom.udacity.com/courses/ud805.
- ☑ Take the Udacity course How to Use Git and GitHub at https://classroom.udacity.com/courses/ud775.
- ☑ Take the online courses in C++: http://www.cprogramming.com/tutorial/c++-tutorial.html and http://www.cplusplus.com/doc/tutorial
- ☑ Redo CS200 lab exercises in C++

maherou assigned maherou and neil-lindquist on Mar 31

maherou added this to the Neil Lindquist Initiation milestone on Mar 31

maherou added to Ready in Collegeville team Kanban board on Mar 31

maherou moved from Ready to In progress in Collegeville team Kanban board on May 15

neil-lindquist moved from In progress to Done in Collegeville team

Questions, comments?

Agile and Scientific Software

A combination worth pursuing.

What is Agile?

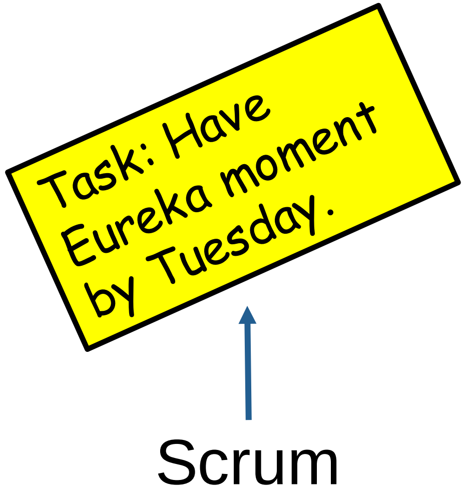
- Agile is not a software development lifecycle model
- I’ve seen Agile informally defined as
  - I don’t write documentation
  - I don’t do formal requirements, design, or really test…
  - Agile is not an excuse to do sloppy work
- Some people consider agile to be synonymous with Scrum
  - From Atlassian: 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

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



Task: Have Eureka moment by Tuesday.

Scrum

- 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".

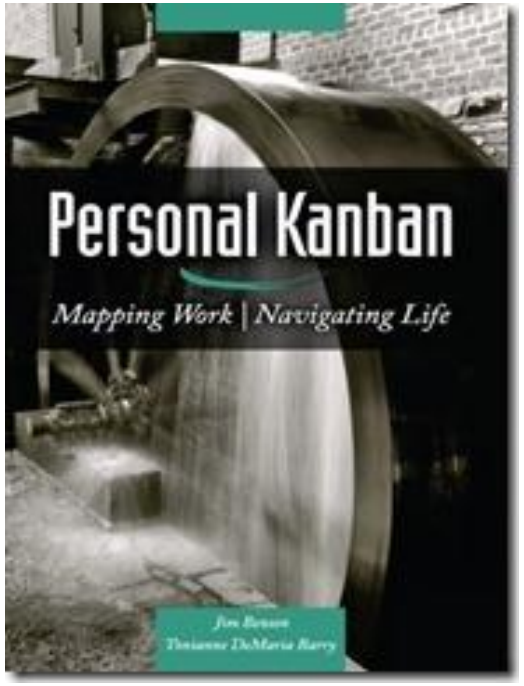
Notes:

- 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

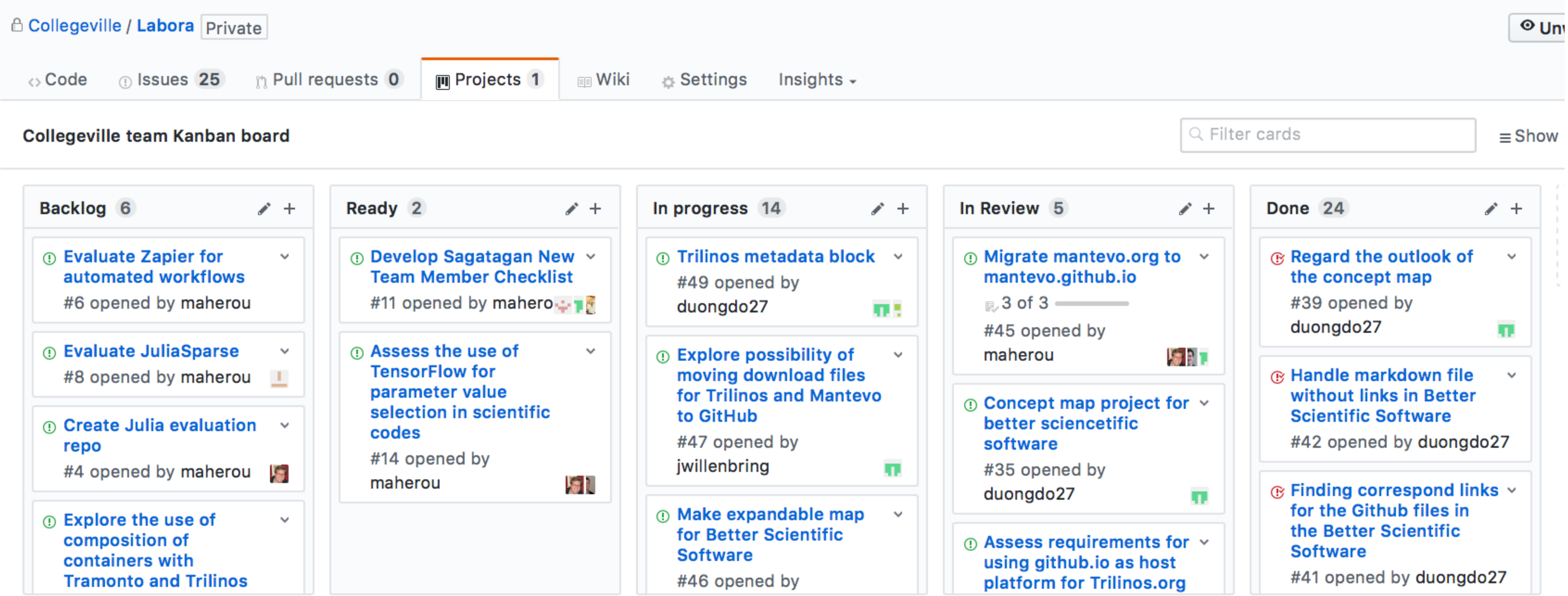
Importance of “In Progress” concept for you

- 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

- 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/

Samples from Collegeville Org: Kanban Board



Collegeville team Kanban board

Backlog 6

- Evaluate Zapier for automated workflows — #6 opened by maherou
- Evaluate JuliaSparse — #8 opened by maherou
- Create Julia evaluation repo — #4 opened by maherou
- Explore the use of composition of containers with Tramonto and Trilinos

Ready 2

- Develop Sagatagan New Team Member Checklist — #11 opened by maherou
- Assess the use of TensorFlow for parameter value selection in scientific codes — #14 opened by maherou

In progress 14

- Trilinos metadata block — #49 opened by duongdo27
- Explore possibility of moving download files for Trilinos and Mantevo to GitHub — #47 opened by jwillenbring
- Make expandable map for Better Scientific Software — #46 opened by

In Review 5

- Migrate mantevo.org to mantevo.github.io — 3 of 3 — #45 opened by maherou
- Concept map project for better sciencetific software — #35 opened by duongdo27
- Assess requirements for using github.io as host platform for Trilinos.org

Done 24

- Regard the outlook of the concept map — #39 opened by duongdo27
- Handle markdown file without links in Better Scientific Software — #42 opened by duongdo27
- Finding correspond links for the Github files in the Better Scientific Software — #41 opened by duongdo27

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
- Jira: Full-featured issue tracker.

My Experience with Kanban

- Can be applied to any existing project immediately, partially:
  - Start collecting and organizing your tasks in columns.
  - GitHub, GitLab, Jira all support column layout of work tasks.
  - Instant improvement: The dashboard alone has immediate value.
- Capture entire context of a task (required input, due date) in one place.
  - I use this approach to organize ECP Software Technology leadership work.
- Make “In Progress” column empty before vacation.
  - Complete or delegate task.
- Run meeting using shared dashboard.
- When you fall out of the habit of using your board, start again.

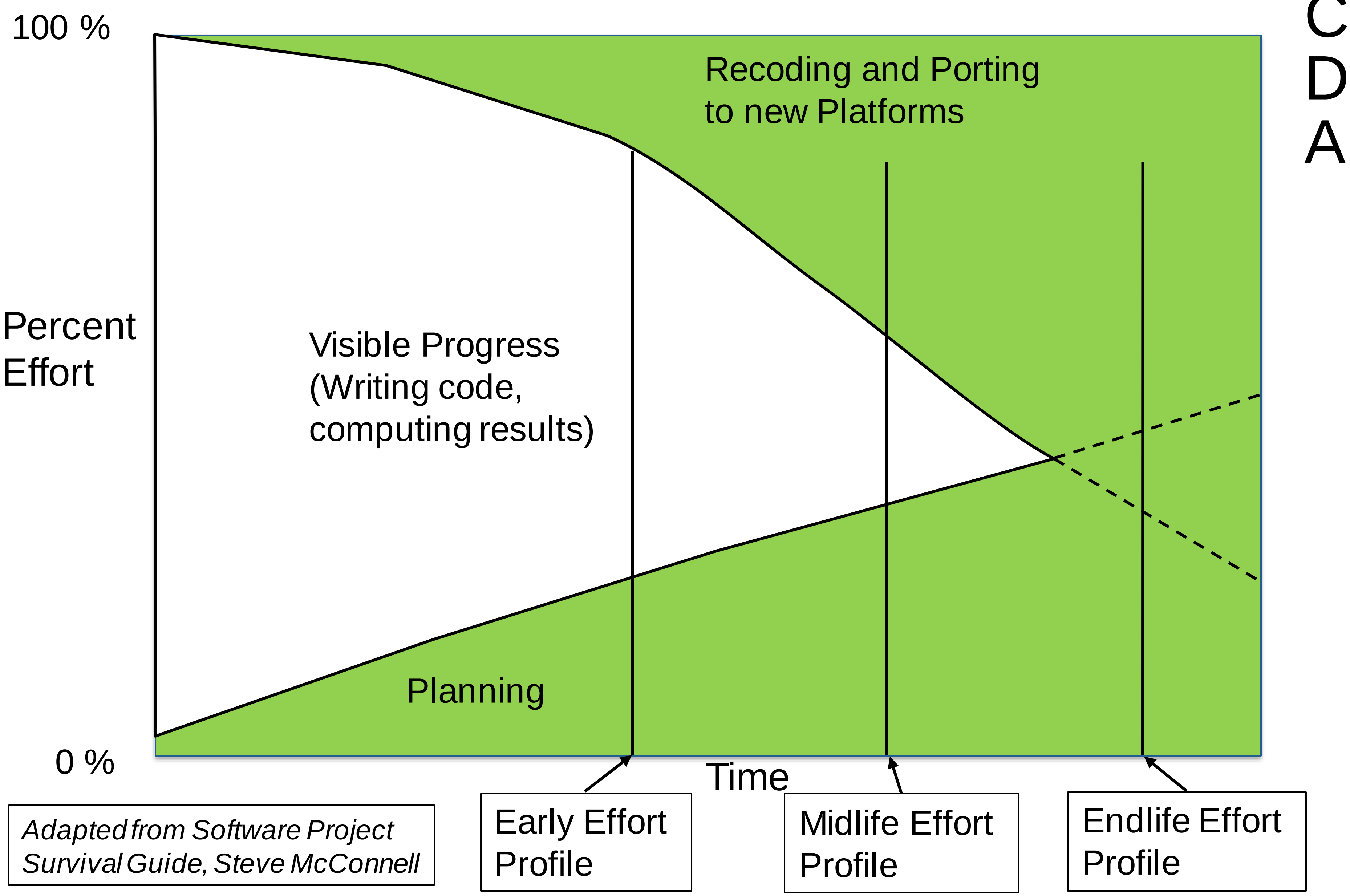
Questions, comments?

Planning

Plans are worthless, but planning is everything.

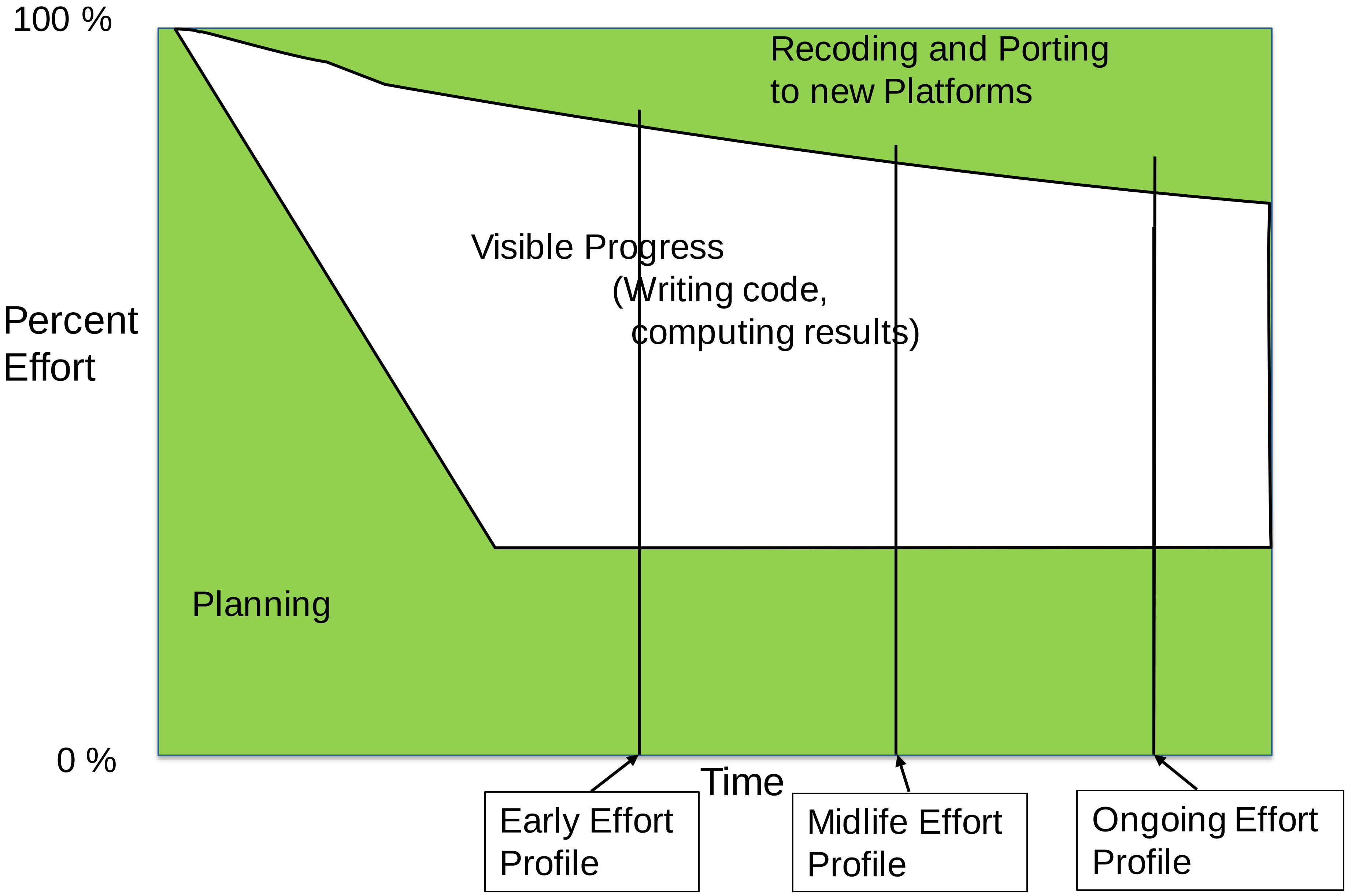
- Dwight D. Eisenhower

Code-and-Fix Development Approach



Adapted from Software Project Survival Guide, Steve McConnell

Simple Planned Development Approach



Stories

Gathering and refining requirements

User Stories: “As a < >, I want < >, so that < >.”

- **User Stories:** Lightweight method for defining, refining, prioritizing requirements.
- **Resource:** Many sources. We list just a few:
  - User Stories – Atlassian: Makers of Confluence, Jira.
  - User Stories: An Agile Introduction: Agile Modeling
  - User Stories: Commercial site (Mountain Goat Software), but good basics.
  - How to write good user stories: A YouTube video from CA Technologies that provides some tips for useful stories.
  - Getting Started with Agile : Epics, Features, and User Stories - YouTube: A nice overview of agile requirements.
  - “As a, I want, So that” Considered Harmful: Does not dismiss value of user stories, reminds the reader that the format is not magic, and that variations can be useful for encoding requirements more effectively. More generally, Crisp's Blog (note: many articles are in Swedish) is a good resource for topics in modern software development.
  - Replacing the User Story with the Job Story: “When < >, I want to < >, so I can < >.”

Phase 1: Generate Stories

- Brainstorm: Don’t worry about phrasing, size, details.
- Collect in a table (shared document is best):

#	Title	Reported by	Story	Stakeholders	Resources (URLs)
1			As < >, I want < >, so that < >.		
			When < >, I want to < >, so I can < >.		

Phase 2: Scope, refine, prioritize

- **Discussion:** Discuss each story for scope, understanding and right-sizing.
  - **Out of scope:** Identify stories that are out of scope for our class.
  - **Clarify and right-size:** Clarify the story, perhaps split into more than one, or combine with another, such that the stories are roughly the same "size" and scope.
- **Prioritization and choosing:** Order top (not all) stories based on importance and ability to execute.

Stories Useful in Many Settings

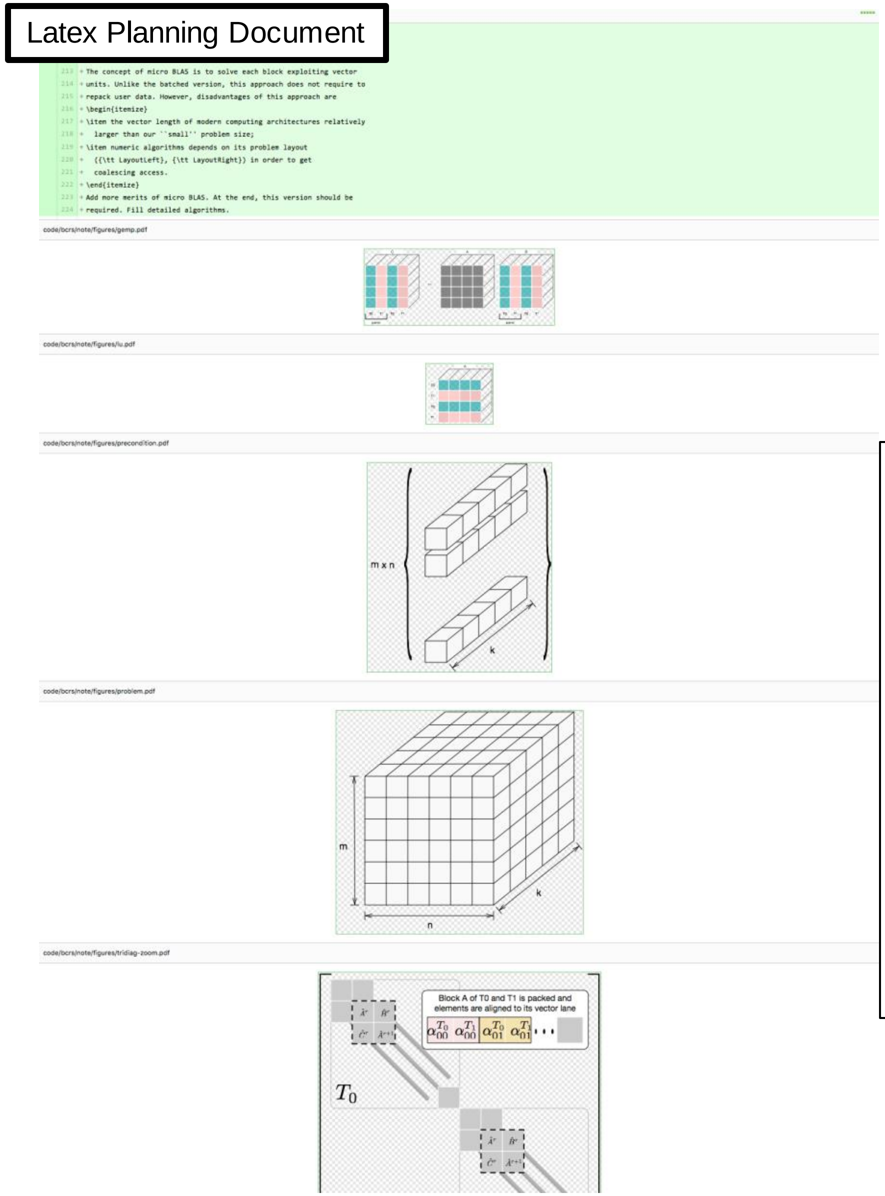
- Tool selection: Select team IDE - Cloud9 AWS – GoogleDocs of IDEs.
- Team policy: Behaviors, practices expected from my team.
- Group reorg: What each team member needs - Guided ECP ST reorg.

Documenting Design

Fix mistakes before you write the software.

Planning tools: Use what you know

Latex Planning Document



```
+ The concept of micro BLAS is to solve each block exploiting vector
+ units. Unlike the batched version, this approach does not require to
+ repack user data. However, disadvantages of this approach are
+ \begin{itemize}
+ \item the vector length of modern computing architectures relatively
+   larger than our ``small'' problem size;
+ \item numeric algorithms depends on its problem layout
+   ({\tt LayoutLeft}, {\tt LayoutRight}) in order to get
+   coalescing access.
+ \end{itemize}
+ Add more merits of micro BLAS. At the end, this version should be
+ required. Fill detailed algorithms.
```

code/bcrs/note/figures/gemp.pdf

code/bcrs/note/figures/lu.pdf

code/bcrs/note/figures/precondition.pdf

code/bcrs/note/figures/problem.pdf

code/bcrs/note/figures/tridiag-zoom.pdf

Block A of T0 and T1 is packed and elements are aligned to its vector lane
T_0

Commit log messages

KokkosKernels - add design note for discussion.
This design note is very informal and working note for discussion.
…

For typeset, "make"

KokkosKernels - add more algorithm variants.
In this algorithm design, I have a few assumptions.
…

KokkosKernels:
Micro & Batched BLAS Design Document

- 6 weeks: Design by LaTeX.
  - Review by diverse experts.
  - Significant design changes: In text only.
- 2 weeks: Write code.

Message: *Use the tools you know.*

Courtesy: KokkosKernels Development Team

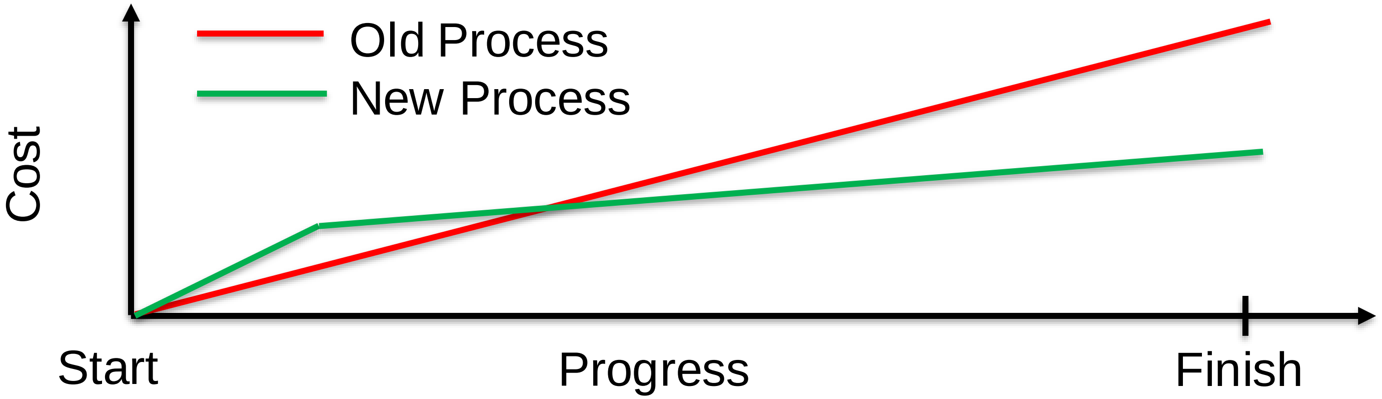
How To Get Better

“Use iteration and incrementation only for projects you want to succeed.”

- *Adaption of Martin Fowler quote*

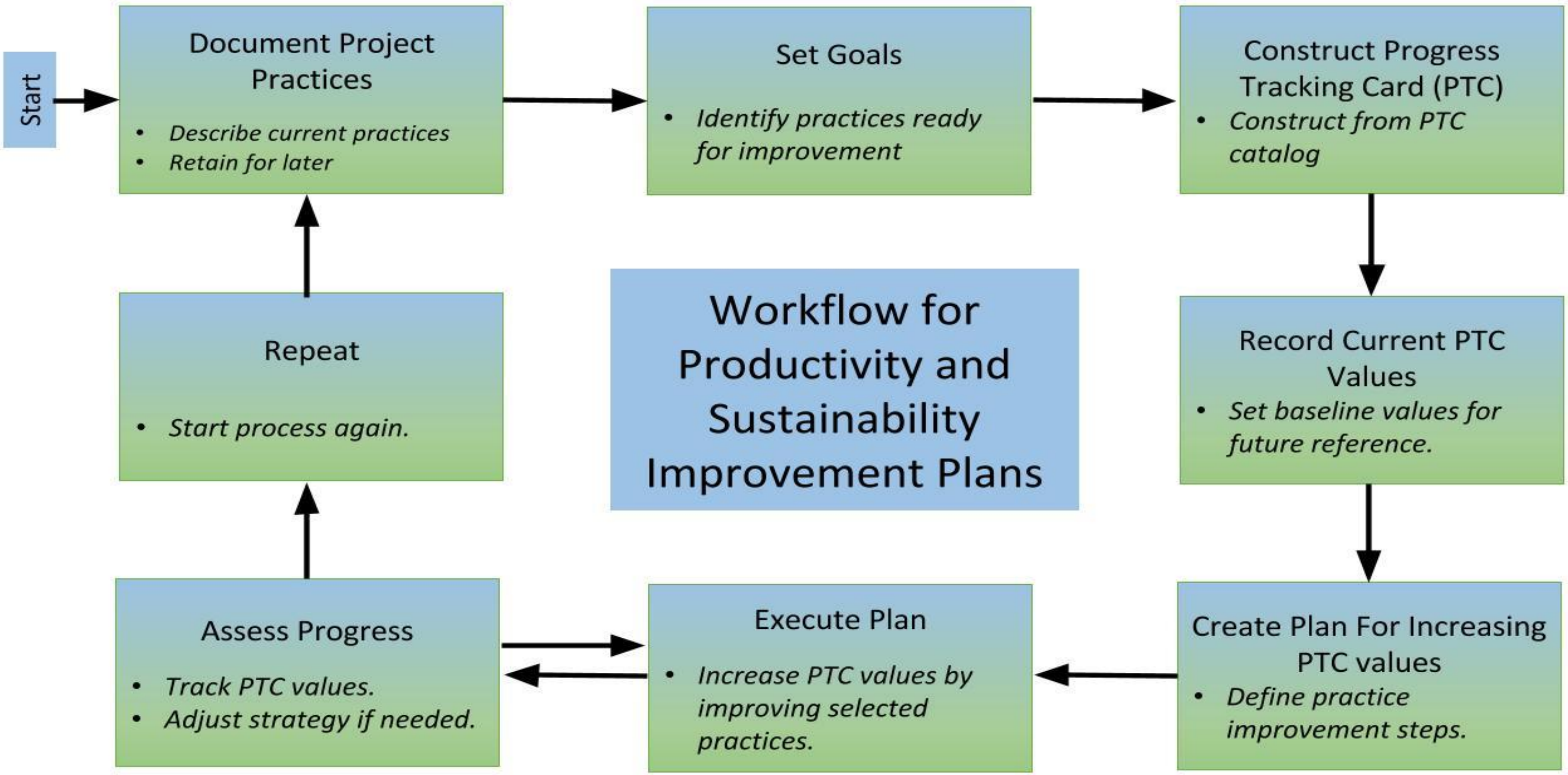
Basic Strategy for Introducing Things We Will Talk About

- Identify, analyze, prototype, test, revise, deploy. Repeat.
- Realistic: There is a cost.
  - Startup: Overhead
  - Payoff: Best if soon, clear



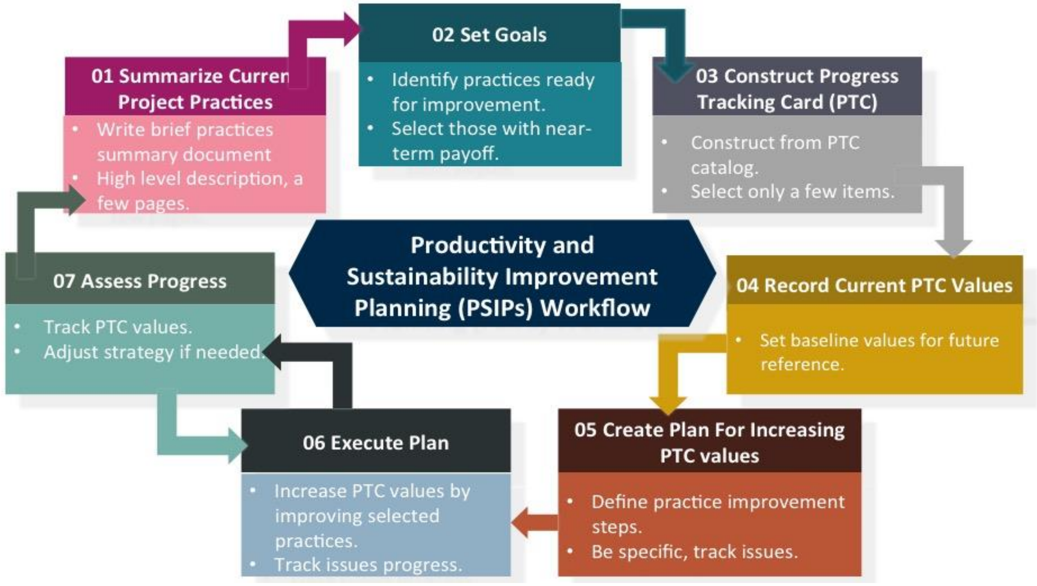
- Working model:
  - Reserve acceptable time/effort for improvement.
  - ***Improve how you do your work on the way to getting it done.***
  - Repeat.

Productivity & Sustainability Improvement Planning (PSIP)


Start
Document Project Practices
• Describe current practices
• Retain for later
Set Goals
• Identify practices ready for improvement
Construct Progress Tracking Card (PTC)
• Construct from PTC catalog
Record Current PTC Values
• Set baseline values for future reference.
Create Plan For Increasing PTC values
• Define practice improvement steps.
Execute Plan
• Increase PTC values by improving selected practices.
Assess Progress
• Track PTC values.
• Adjust strategy if needed.
Repeat
• Start process again.
Workflow for Productivity and Sustainability Improvement Plans

https://betterscientificsoftware.github.io/PSIP-Tools/

Productivity and Sustainability Improvement Planning (PSIP) Examples: EXAALT & MPICH


Productivity and Sustainability Improvement Planning (PSIPs) Workflow

01 Summarize Current Project Practices
- Write brief practices summary document
- High level description, a few pages.

02 Set Goals
- Identify practices ready for improvement.
- Select those with near-term payoff.

03 Construct Progress Tracking Card (PTC)
- Construct from PTC catalog.
- Select only a few items.

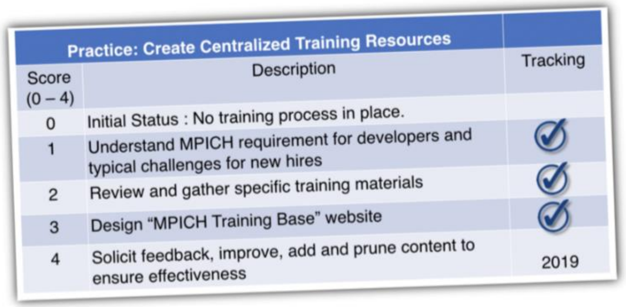
04 Record Current PTC Values
- Set baseline values for future reference.

05 Create Plan For Increasing PTC values
- Define practice improvement steps.
- Be specific, track issues.

06 Execute Plan
- Increase PTC values by improving selected practices.
- Track issues progress.

07 Assess Progress
- Track PTC values.
- Adjust strategy if needed.

MPICH PSIP: Onboarding new team members

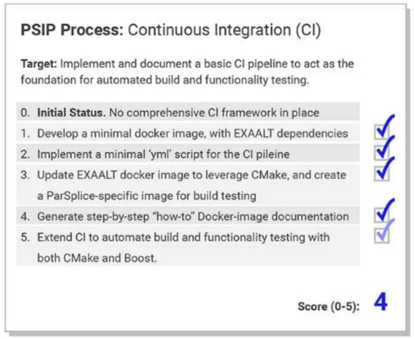


Score (0 – 4)	Practice: Create Centralized Training Resources Description	Tracking
0	Initial Status : No training process in place.	
1	Understand MPICH requirement for developers and typical challenges for new hires	✓
2	Review and gather specific training materials	✓
3	Design "MPICH Training Base" website	✓
4	Solicit feedback, improve, add and prune content to ensure effectiveness	2019

PSIP workflow helps a team create user stories, identify areas for improvement, select a specific area and topic for a single improvement cycle, and then develop those improvements with specific metrics for success.

EXAALT PSIP: Continuous integration (CI) testing

BSSw blog article: Adopting Continuous Integration for Long Timescale Materials Simulation, Rick Zamora (Sept 2018)

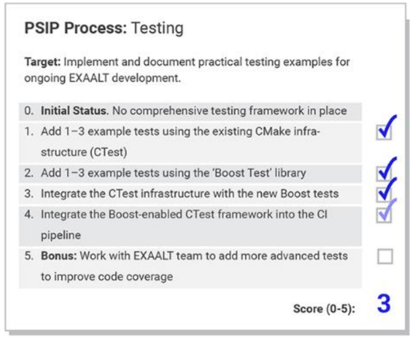


PSIP Process: Continuous Integration (CI)

Target: Implement and document a basic CI pipeline to act as the foundation for automated build and functionality testing.

0. **Initial Status.** No comprehensive CI framework in place
1. Develop a minimal docker image, with EXAALT dependencies ☑
2. Implement a minimal 'yml' script for the CI pileine ☑
3. Update EXAALT docker image to leverage CMake, and create a ParSplice-specific image for build testing ☑
4. Generate step-by-step "how-to" Docker-image documentation ☑
5. Extend CI to automate build and functionality testing with both CMake and Boost. ☑

Score (0-5): 4



PSIP Process: Testing

Target: Implement and document practical testing examples for ongoing EXAALT development.

0. **Initial Status.** No comprehensive testing framework in place
1. Add 1–3 example tests using the existing CMake infrastructure (CTest) ☑
2. Add 1–3 example tests using the 'Boost Test' library ☑
3. Integrate the CTest infrastructure with the new Boost tests ☑
4. Integrate the Boost-enabled CTest framework into the CI pipeline ☑
5. **Bonus:** Work with EXAALT team to add more advanced tests to improve code coverage □

Score (0-5): 3

Personal Expectations

Calling out the best in team members

Final Thoughts: Commitment to Quality

Canadian engineers' oath (taken from Rudyard Kipling):

https://www.egbc.ca/Member-Programs/Students/Iron-Ring

My Time I will not refuse;
my Thought I will not grudge;
my Care I will not deny
toward the honour, use,
stability and perfection of
any works to which I may be
called to set my hand.

A Few Concrete Recommendations

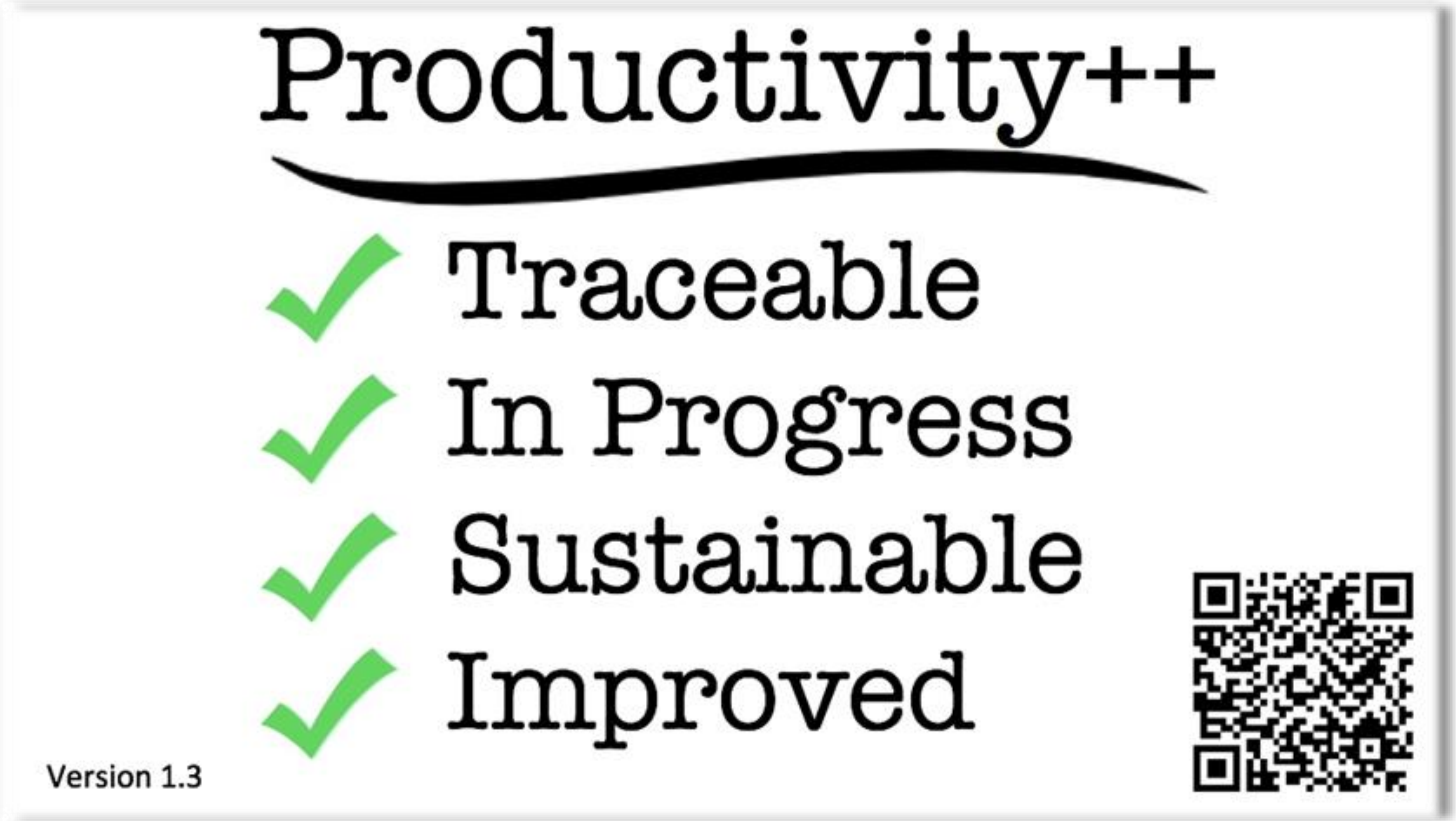
> *Show me the person making the most commits on an undisciplined software project and I will show you the person who is injecting the most technical debt.*

- GitHub stats: Easy to find who made the most commits.
  - Some people: Pride in their high ranking.

- Instead, be the person who ranks high in these ways:
  - Writes up requirements, analysis and design, even if simple.
  - Writes good GitHub issues, tracks their progress to completion.
  - Comments on, tests and accepts pull requests.
  - Provide good wiki, gh-pages content, responses to user issues.

(Personal) Productivity++ Initiative

Ask: *Is My Work ________ ?*



Productivity++

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

Version 1.3

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

Wrap Up

- Better software engineering needed: Makes transparency & reproducibility affordable.
- Barely-sufficient SW engineering is turnkey with GitHub & related platforms.
- Agile for Scientific Software can make sense:
  - Careful introduction of formality, continued adoption of tools and platforms.
- Lightweight, structured planning seems highly impactful:
  - Kanban, User/Job stories, basic design document.
- Iteration and incrementation effective for continual improvement:
  - PSIP or similar approach can be used.
  - https://betterscientificsoftware.github.io/PSIP-Tools/
- Calling out the best in team members:
  - Articulate the mission, inspire members to individual improvement.

Questions, comments?

Thank You.