SAND2020-7957PE

Agile Methodologies Redux

David E. Bernholdt
Oak Ridge National Laboratory

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

Better Scientific Software Tutorial

Software Productivity Track
ATPESC 2020

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & 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.

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Better Scientific Software tutorial, in RF SciDAC 2020 Workshop, Knoxville, Tennessee. DOI: 10.6084/m9.figshare.11918397**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software Tutorial…

Acknowledgements

- Additional contributors to this this tutorial include: Anshu Dubey, Mike Heroux, Alicia Klinvex, Jared O'Neal, and Katherine Riley, James M. Willenbring
- 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 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 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. SAND NO SAND2017-5474 PE

Outline

- Refining our Epic
- PSIP: Productivity and Sustainability Improvement Planning

More on Epic, Story, Task

Epic, Story, Task Review

- 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
  - “Definition of Done” – understandable to user
- Tasks are the steps necessary to complete a story
  - May not individually provide value to the customer

Definition of Done

- Simplified definition: When all acceptance criteria are met
- Acceptance criteria
  - “Conditions that a software product must satisfy to be accepted by a user, customer or stakeholder.” – Microsoft Press
  - “Pre-established standards or requirements a product or project must meet.” – Google
  - Can include functional, non-functional, and performance requirements.

Definition of Done

- Important to establish for a story before estimating or beginning a task
- Defined by the team, acceptable to customer
  - Customer language
- Should not specify an implementation unnecessarily

Refining Our Epic

- Epic: 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
    - Definition of Done
    - Task list
  - Story 2: Separate out integration function
    - Definition of Done
    - Task list

Refining Our Epic

- Story 1: Separate out utilities
  - Definition of Done
    - Unit tests pass
    - Code review completed
    - Integration/system tests pass
    - Utility performance is at least 95% of pre-separation performance
    - Utility usability demonstrated outside of heat equation application

- Story 2: Separate out integration function
  - Task 1: Add testing for integration function to protect functionality during refactor
    - Needed testing should be specified
  - Task 2: Generalize interface to allow alternative implementations
  - Task 3: Expose current integration function through the new interface & run tests

Agile Estimation

- Estimating is hard
  - Requires practice
  - With practice, it is still hard
- Stories are estimated using “story points”
  - Relative estimate
  - Many estimating techniques
  - Should NOT map to hours, days, etc
  - Definition of done needed, tasking not required
- Tasks are estimated in hours
  - Absolute estimate
- Useful for planning schedules

Key concept:
It is easier to accurately estimate many small tasks than to estimate a large epic.

Epic: Huge refactor effort

Tasks:

- Add tests
- Generalize interface
- Expose existing interface

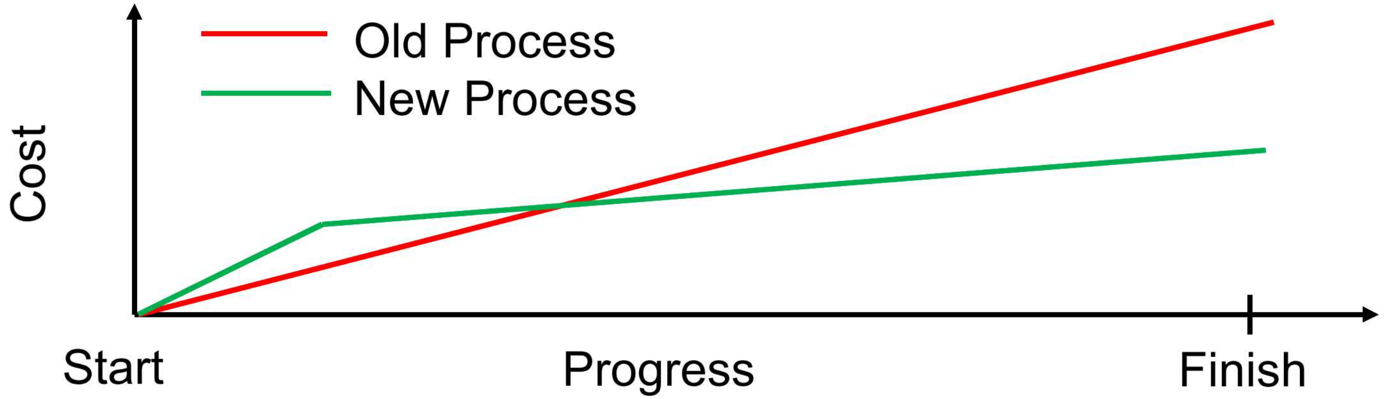
How To Get Better

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

- *Adaptation of Martin Fowler quote*

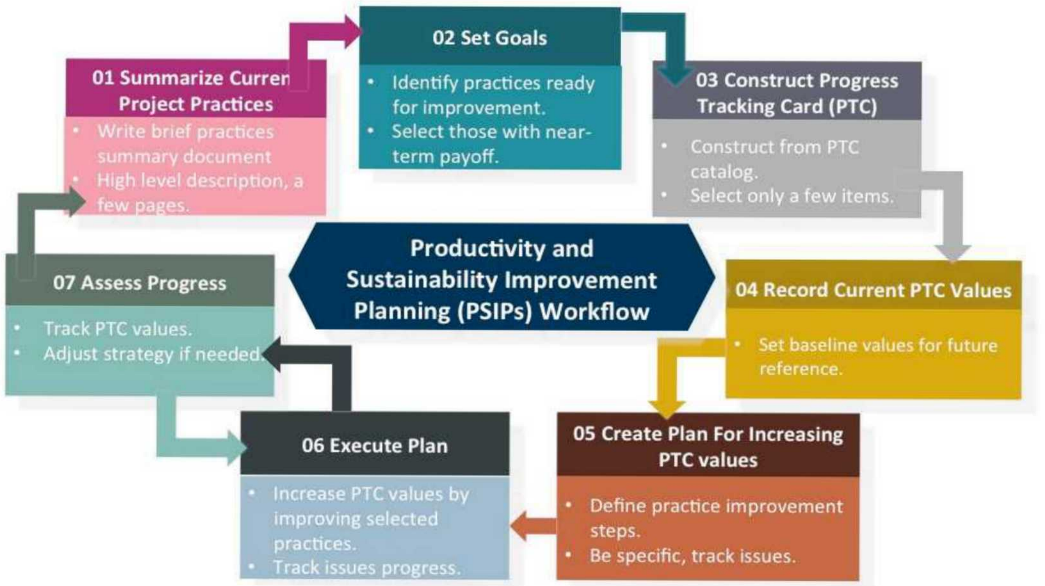
Strategy for Incremental Productivity Improvements

- 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.

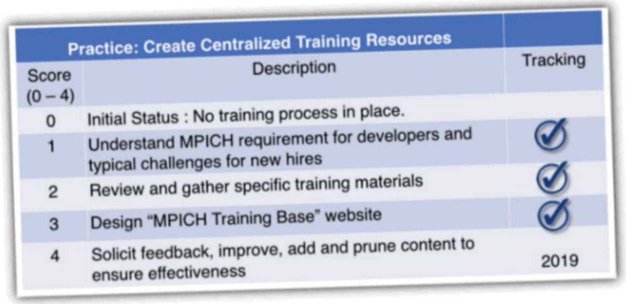
Productivity and Sustainability Improvement Planning (PSIP) Examples: EXAALT & MPICH – Add PSIP URL



Productivity and Sustainability Improvement Planning (PSIPs) Workflow
01 Summarize Curren 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.

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.

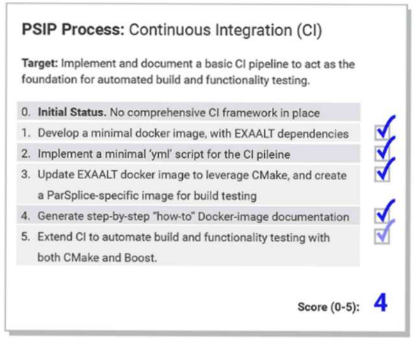
MPICH PSIP: Onboarding new team members



Practice: Create Centralized Training Resources		
Score (0 – 4)	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

EXAALT PSIP: Continuous integration (CI) testing

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

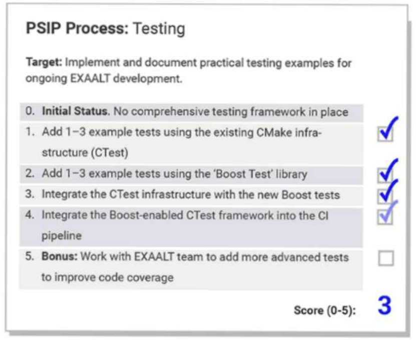


PSIP Process: Continuous Integration (CI)

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