SO, YOU WANT TO BE AGILE?

Strategies For Introducing Agility Into Your Scientific Software Project

HPC Best Practices Webinar Series

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.

Barely Sufficient Project Management

A few techniques for improving your scientific software development efforts

HPC Best Practices Webinar Series

Michael A. Heroux
Senior Scient
Scientist

Previous Webinar: September 2017

https://www.exascaleproject.org/event/barely-sufficient-project-management/

https://betterscientificsoftware.github.io/A-Team-Tools/

NNSA National Nuclear Security Administration
U.S. DEPARTMENT OF ENERGY | Office of Science
IDEAS productivity

Outline

- My Perspective
- Agile & Scientific Software: A Characterization
- Brief Recap of Kanban, Checklists, Policies – Because they are good
- Planning: An Opportunity for Improvement for Scientific SW.
- Two Practical Techniques: Stories, Lightweight Design Doc
- Strategies for Introducing New Practices & Tools
- Appealing to the Best in Each of Us: Personal Expectations
- A New Approach (AFAIK): Software Science

My Perspective

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

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

5 Mike Heroux, “So, You Want To Be Agile?” HPC Best Practices Webinar Series, May 8, 2019

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

https://agilemanifesto.org
https://agilemanifesto.org/principles.html

That is, while there is value in the items on the right, we value the items on the left more.

De facto Scientific Software Development

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

Scientific Software Engineering

"A scientist builds in order to learn;
an engineer learns in order to build."

- Fred Brooks

Scientist: Barely-sufficient building.

Engineer: Barely-sufficient learning.

Both: Insufficiency leads to poor SW.

Agile and Scientific Software

A combination worth pursuing.

Some Relevant Agile Attributes for Scientific Software Teams

- Agile methodologies: A diverse collection of approaches
  - Common: Scrum, Extreme Programming, Kanban.
  - Scrum is most popular in industry.
- Characteristics:
  - Iteration: Define, develop software via a sequence of feature sets.
  - Incrementation: Deliver a portion based on a sense of what is needed now.
  - Engage users regularly: Feature sets delivered and exercised after each iteration.
  - Hierarchical planning with refinement: Holistic big picture. A clear detailed plan will evolve.
- Agile consistent with scientific software development:
  - Iteration/incrementation: Computational results impact plans for next phase.
  - Users: Often same team, or lead scientist, regular interaction.
  - Planning: Overall scientific goal well defined. Steps along the way will change, become clear.

Characterization: Scientific SW Development Attributes

- Scientific SW developer:
  - Advanced degree (often PhD) in scientific domain: Chemistry, physics, other.
    - Substantial time engaging community: experiments, analysis, paper, presentations, literature.
    - Limited time for cultivating advanced software practices.
    - Persistent choice: Better practices or new result sooner? Both needed to compete well.
    - As SW become more inter-developed, better practices essential. Aggregate projects common.
  - Specialized expertise:
    - Usually one team member best for a particular software feature.
    - Mainstream agile:
      - Assume some skill redundancy for load balancing.
      - We can mitigate this issue somewhat by code review, good documentation.

Manifesto Anti-themes and Scientific Software

- Processes and tools:
  - Good adoption of emerging platforms – GitHub, GitLab.
  - **Talk topic: Stories for new processes, tools.**
- Comprehensive documentation:
  - Readthedocs, Doxygen, publications.
  - **Talk topic: Lightweight design documentation.**
- Contract negotiatons:
  - Funding proposals, in-house.
  - **Won't discuss today.**
- Following a plan:
  - Science plan, funded proposal statement.
  - **Talk topics: Significant opportunities for improvement.**

Brief Repeat and Update of Past Topics

Policies, checklists, Scrum, Kanban

Briefly: Checklists & Policies – See previous talk

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/
- How-to resource:
  - https://betterscientificsoftware.github.io/A-Team-Tools/

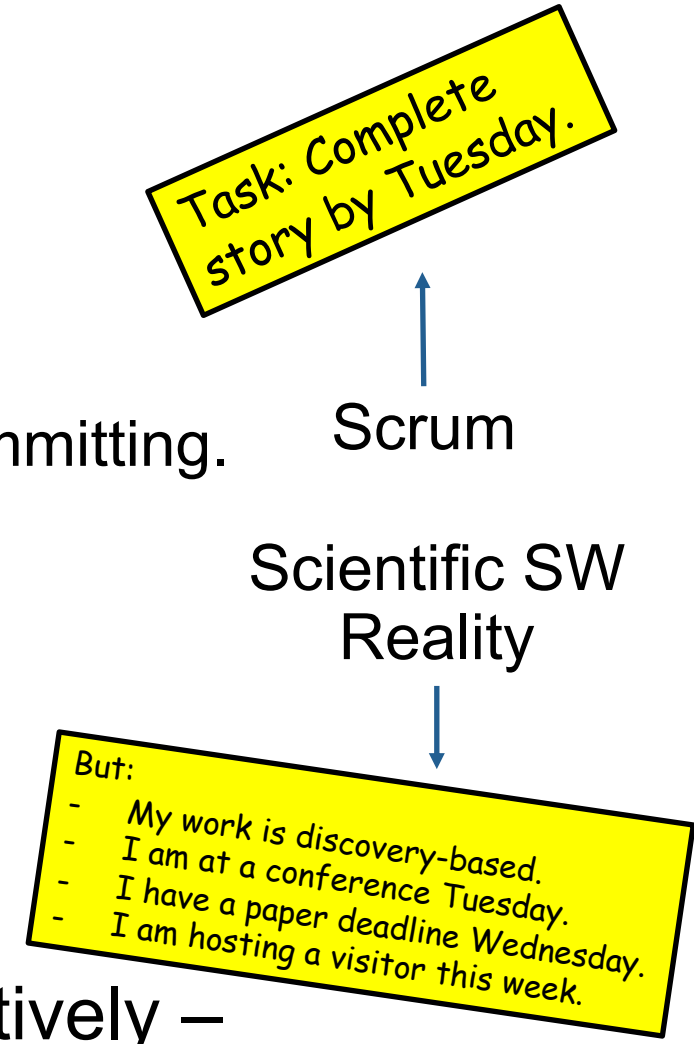
Iterative work systems: Scrum and Kanban

- Scrum: A popular process framework, widely and successfully used.
- Could it work for you? Maybe.
- Emphasis: Regular sprints, reviews, retrospectives, stories, backlog, product owner, scrum master, and more.
- Scrum adherents:
  - Much of industry: automotive, oil & gas, aerospace – Stories are inspirational.
  - Production teams at labs: Many teams whose developers are primarily 9-to-5.

- https://leankit.com/learn/kanban/kanban-vs-scrum/
- Kanban and Scrum -- Making the Most of Both, by Henrik Kniberg and Mattias Skarin
- https://www.scrumalliance.org

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.
- Scrum may be in my future, or could be applied selectively – still TBD.



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 columns: In Review, Blocked.
- Can be creative with columns:
  - Waiting on Advisor/Supervisor Confirmation.
  - Tasks I won't do.

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.

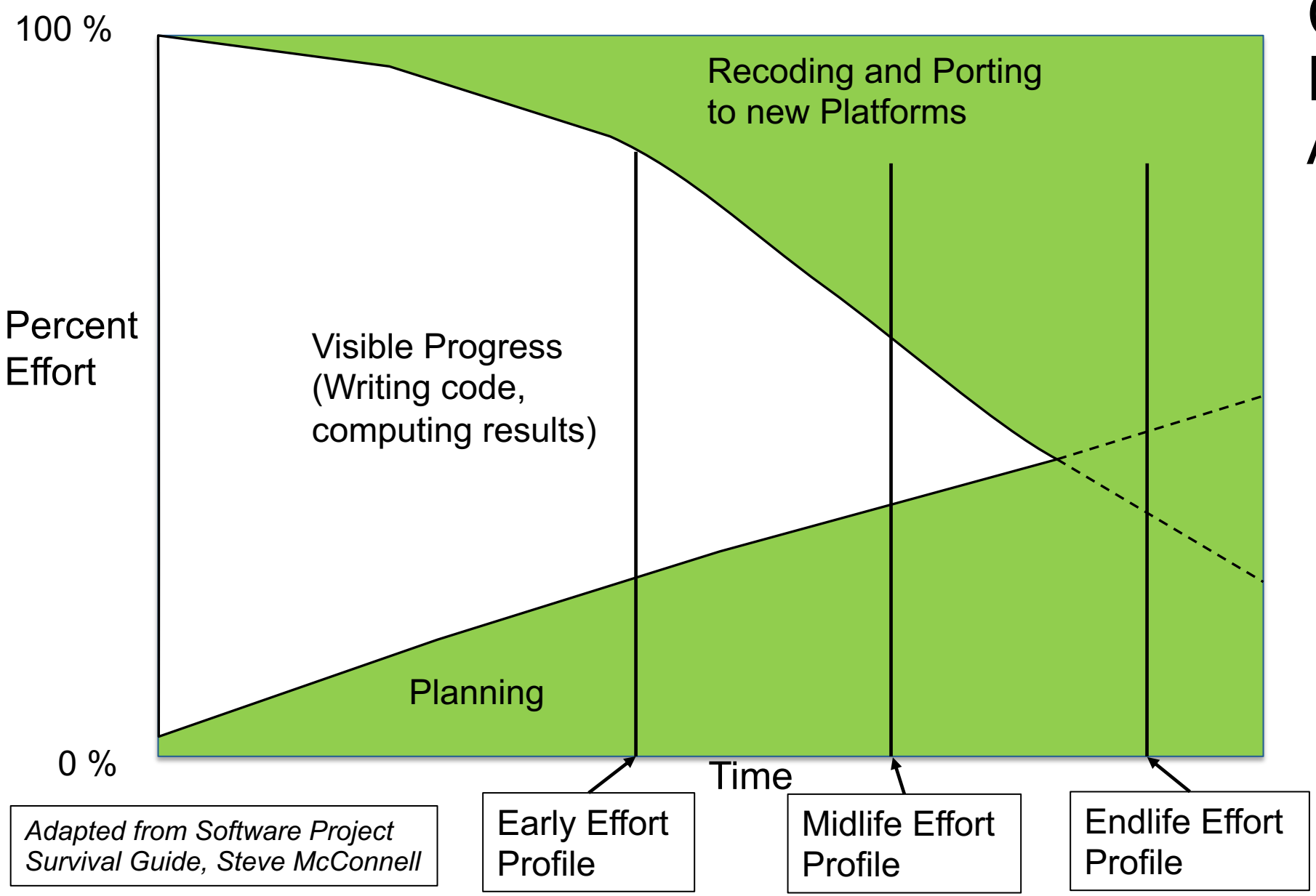
Questions, comments?

Planning

Plans are worthless, but planning is everything.

- Dwight D. Eisenhower

Code-and-Fix Development Approach

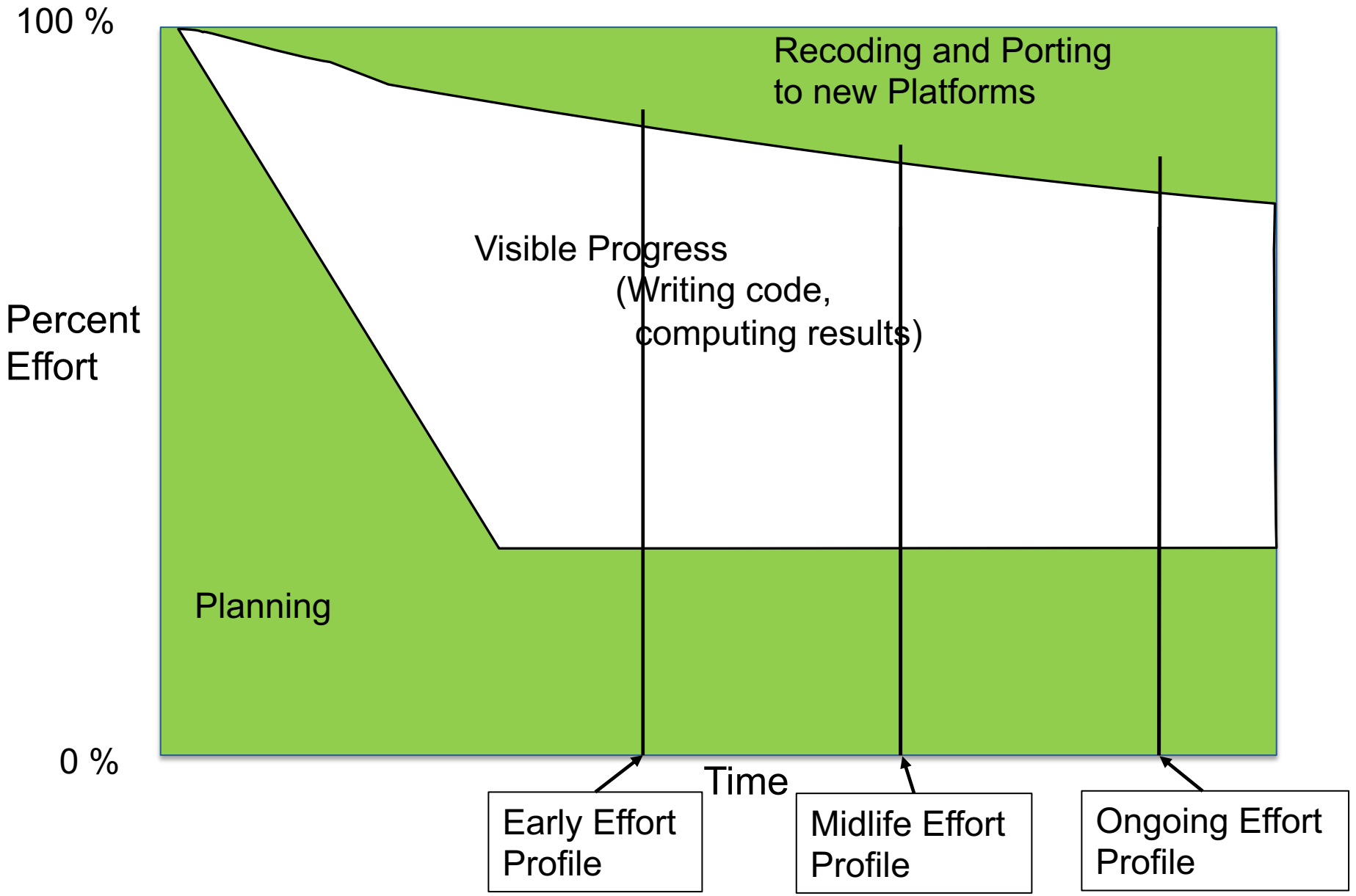


Adapted from Software Project Survival Guide, Steve McConnell

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Simple Planned Development Approach



IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

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 in the category 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

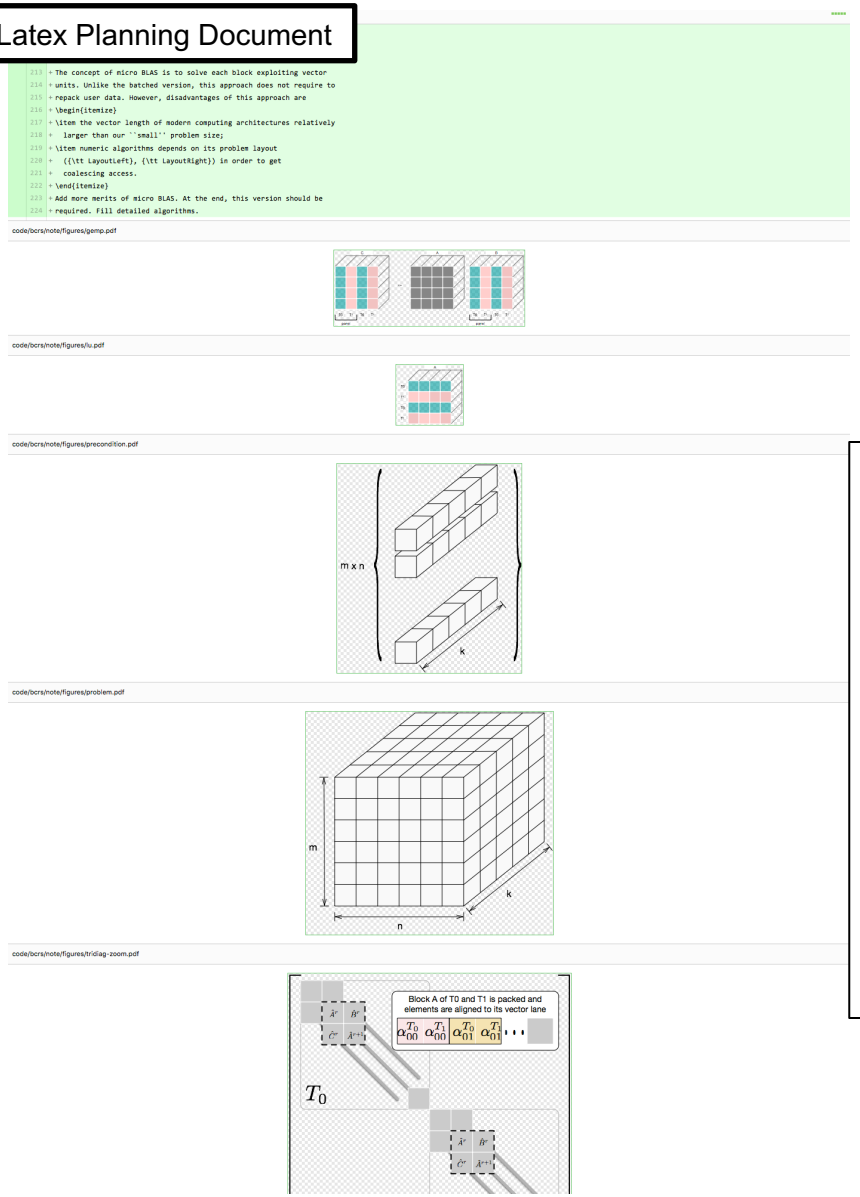
- Tool selection: Select team IDE - Cloud9 AWS – GoogleDocs of IDEs.
- Team policy: Behaviors, practices expected from my team.
- Group reorg: What each team member needs - Guided ECP ST reorg.

Documenting Design

Fix mistakes before you write the software.

Planning tools: Use what you know

Latex Planning Document



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

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

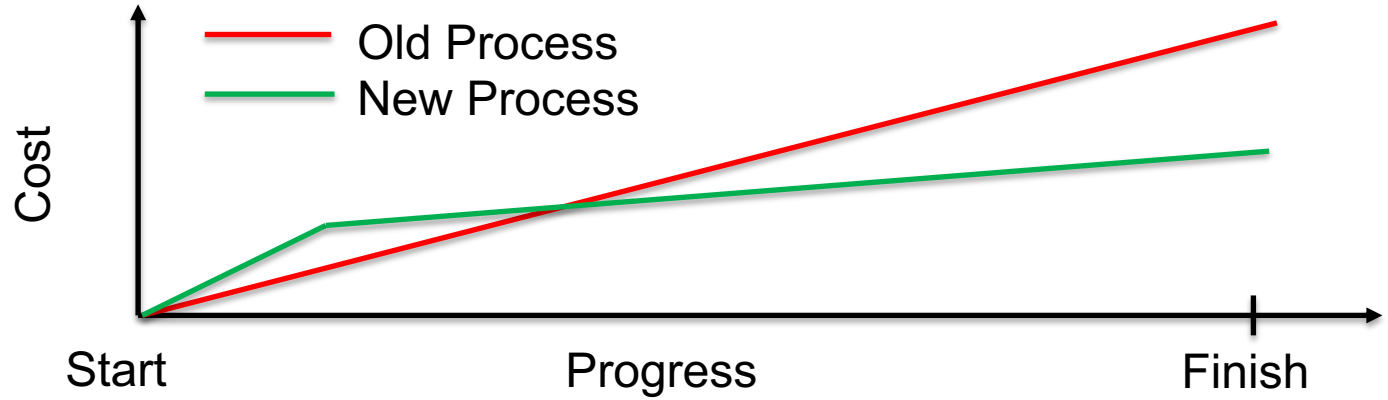
Strategies for Introducing Change

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

- *Adaption of Martin Fowler quote*

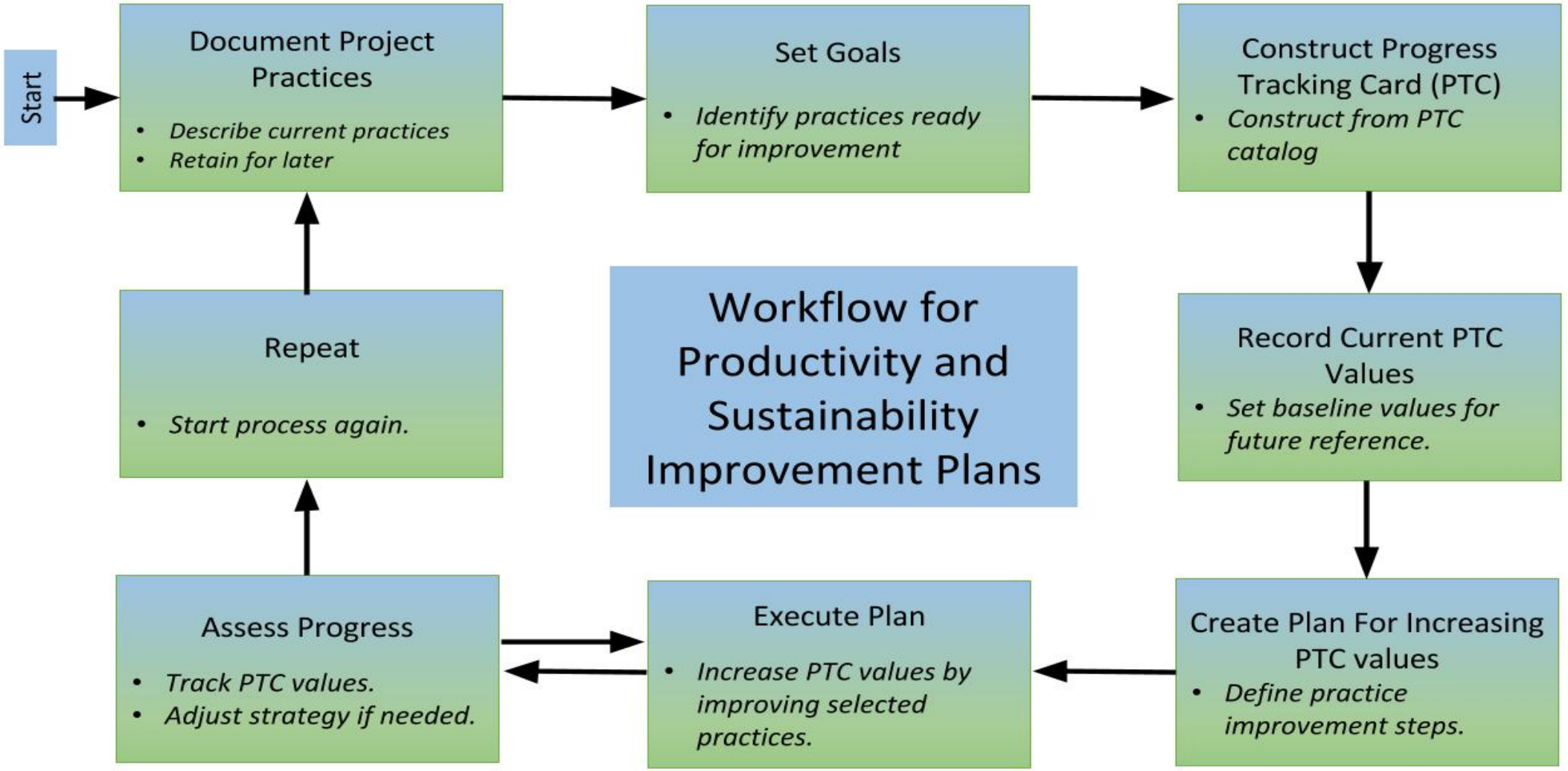
Basic Strategy for Introducing Things We Will Talk About

- 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)



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

Progress Tracking Card Example

Practice: Test Coverage	Score (0 – 5):	
Score Descriptions		
0	Little or no independent testing. Functional testing via users.	
1	Independent functional testing of primary capabilities.	
2	Primary functional testing, some unit test coverage.	
3	Comprehensive unit testing, primary functional testing.	
4	Comprehensive unit testing, functional testing for documented use cases.	
5	Comprehensive unit, use case functional testing; test coverage commitment.	

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

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 infra-structure (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):

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.

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

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 _____ ?*

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

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Software Science

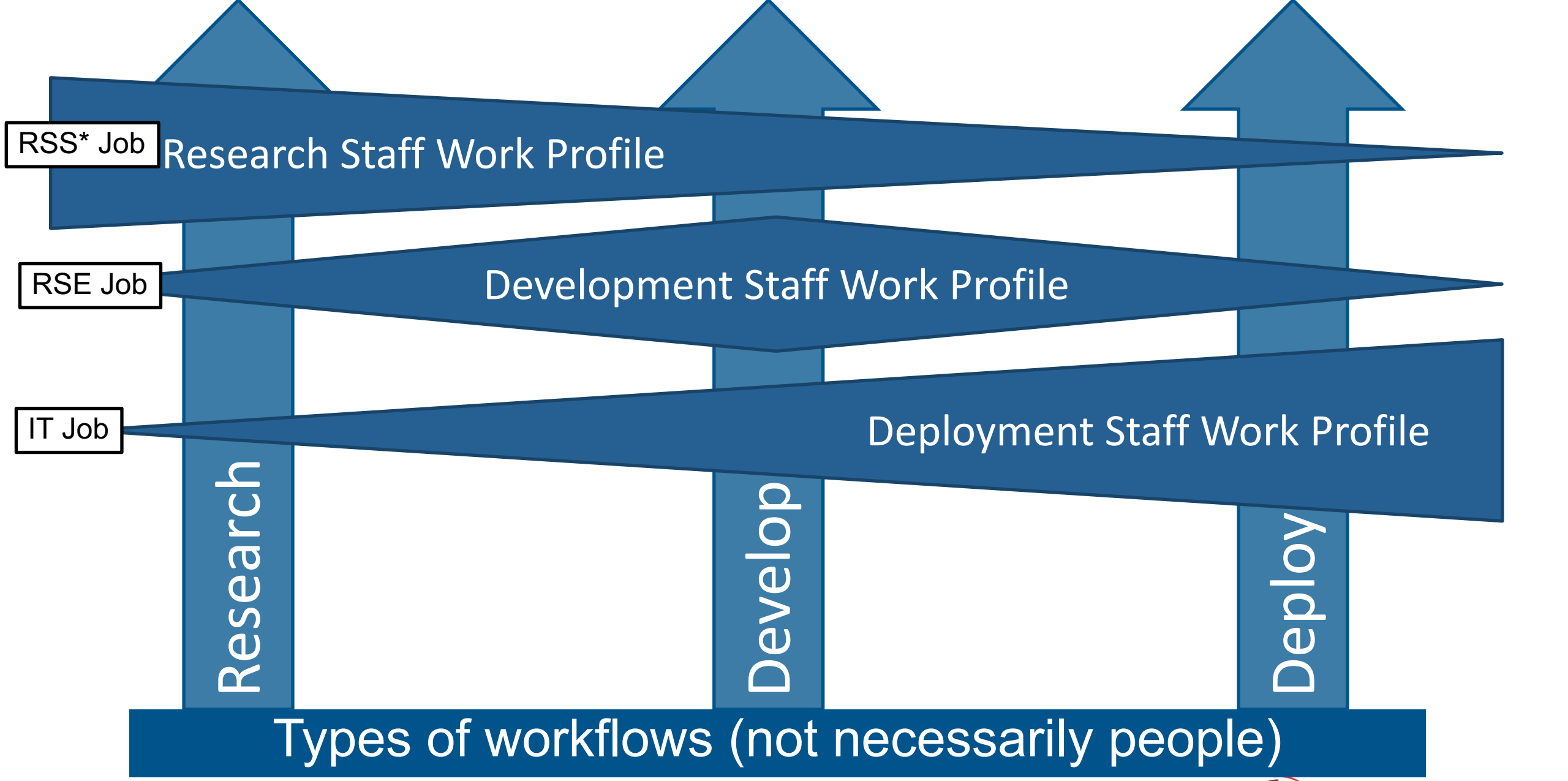
Toward Scientific Study of Research Software Development

Sandia SW Engineering and Research (SEAR) Department

- New department focused on making CSE Research SW better:
  - Focus on scientific & engineering research software.
  - Improve developer productivity and software sustainability.
  - Bring a critical mass of existing staff into a department.
  - Attract new talent by making SW Eng & Research first-class citizen.
  - Build a community presence visible to DOE and external community.
  - Build on Sandia's native engineering culture.
- Three primary department workflows:
  - **Research:** Participate in research community to understand and create new knowledge for improving CSE research software.
  - **Develop:** Identify, cultivate SW best practices prioritized for CSE research software development.
  - **Deploy:** Provide effective SW tools and environments adapted to CSE research software teams.

New workflow

Existing workflows



*RSS – Research Software Scientist (new job type)

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Why A Software Science Focus now: The “No CS” Scenario

Scenario: Suppose our research centers had no formally trained computer scientists and CS work had to be done by people who learned it on their own, or just happened to study a bit of CS as part of their other formal training. This situation is undesirable in three ways:

1. We have non-experts doing CS work, making them less available in their expertise.
2. CS work takes a long time to complete compared to other work.
3. We get suboptimal results and pay high ongoing maintenance cost.

Replace ”CS” with “Software” in this scenario and the situation describes scientific software today.

Why focus on software science now:

- The role of software has become central to much of our work and the knowledge base is too sophisticated to rely only on non-experts.
- Scientific software success depends on producing high-quality, sustainable software products.
- Investing in software as a first class pursuit improves the whole scientific ecosystem.

Applying Social Science to Software Teams

- Reed Milewicz – my postdoc.
- Elaine Raybourn – Sandia social scientist recruited to my team.
- New scientific tools to study and improve developer productivity, software sustainability.
- Correlation: Happiness and connectedness.
- Next: Design experiments to detect cause and effect.

Talk to Me: A Case Study on Coordinating Expertise in Large-Scale Scientific Software Projects

Reed Milewicz and Elaine M. Raybourn
Sandia National Laboratories, 1611 Innovation Pkwy SE, Albuquerque, New Mexico 87123

.SE] 17 Sep 2018

Abstract—Large-scale collaborative scientific software projects require more knowledge than any one person typically possesses. This makes coordination and communication of knowledge and expertise a key factor in creating and safeguarding software quality, without which we cannot have sustainable software. However, as researchers attempt to scale up the production of software, they are confronted by problems of awareness and understanding. This presents an opportunity to develop better practices and tools that directly address these challenges and that end, we conducted a case study of developers of [illegible] project. We surveyed the software development [illegible] dressed and show how those problems are [illegible] they know and how they communicat[illegible] provide a series of practicable rec[illegible] path forward for future research.

Source: https://arxiv.org/pdf/1809.06317.pdf

New Professional Role: *Research Software Scientist*

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Wrap Up

- 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.
- Time for a new role: Software Scientist.

Questions, comments?

Thank You.