# Software Engineering Challenges and Best Practices for Multi-Institutional Scientific Software Development

Aug 2021 Webinar

*Best Practices for HPC Software Developers*

Keith Beattie (ksbeattie@lbl.gov)
Dan Gunter (dkgunter@lbl.gov)

*Lawrence Berkeley National Laboratory*

# Introduction: Who am I?

## Background

- CSE at LBL for ~20 years
- BA Math & MS in Comp Sci
- Worked in industry for ~5 years

## Roles

SW Developer, Facilitator, Release Engineer:

- IceCube (DAQ)
- CCSI/CCSI$^2$
- SPOT (“superfacility”)
- IDAES, NAWI, DISPATCHES, ...
- Lux Zeplin

# The Big Question:

*How can we develop scientific software that best serves our scientific mission?*

# Software Development: Industry vs Scientific

The comparison is tempting and misleading but *can be helpful*.

Industry Roles:

- Sales
  - Revenue
- Marketing
  - Competitive Analysis, Branding, Sales leads
- Product Manager
  - Roadmap, Customer Support
- Engineering
  - Implementation, research
- Quality
  - Verification, stability, reliability, usability
- Operations
  - DevOps, Hardware, Uptime

Science Roles:

- Funder / Project Manager
  - Scientific Mission
- Principal Investigator
  - Scientific Results
- Scientists
- Engineers/RSEs
- Post-docs
- Students

} Often interchangeable based on skills and experience

Who is facing what changes over time: i.e. Big Data (science only), reusable software (industry only)

What can be learned by comparing the two? What about Universities vs National Labs?

# Institute for the Design of Advanced Energy Systems (an example)

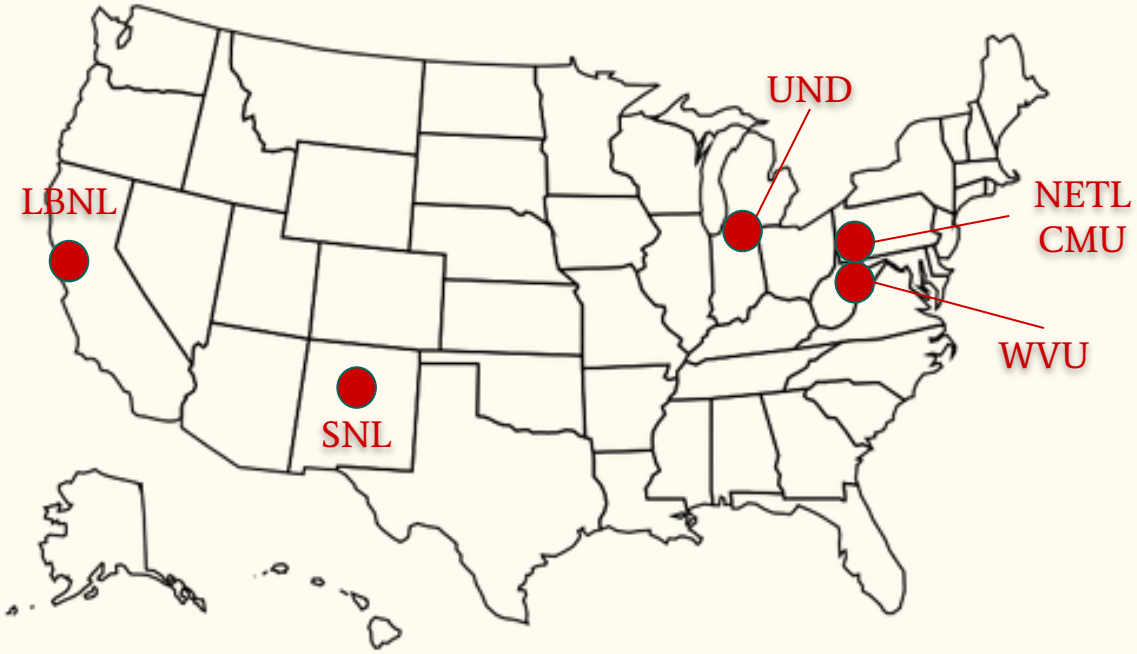
IDAES
Institute for the Design of Advanced Energy Systems

Software framework for modeling chemical processes. Original focus on power plants (DOE Fossil Energy) expanded to Process Systems Engineering (PSE)

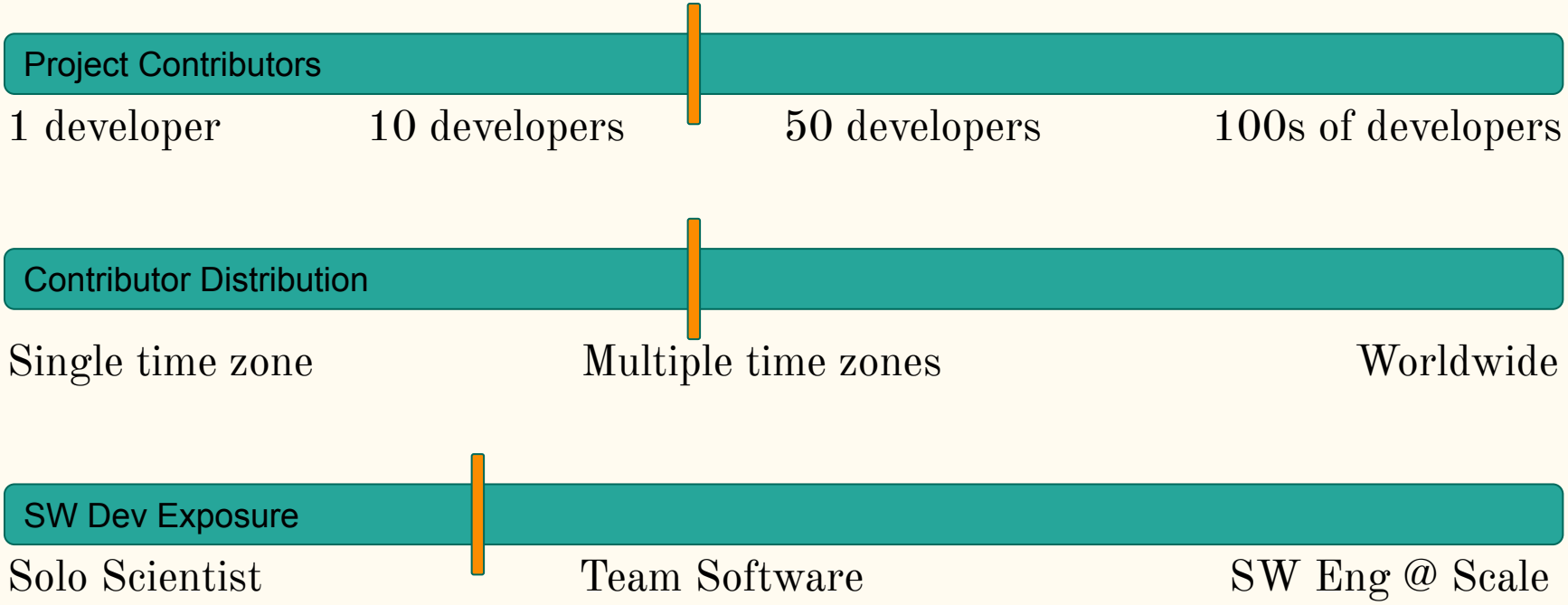
~ 40 contributors (mostly part-time)
~30 chemical or process engineers
~5 are computer scientists
~5 are chemists / material scientists

Used by several other projects:

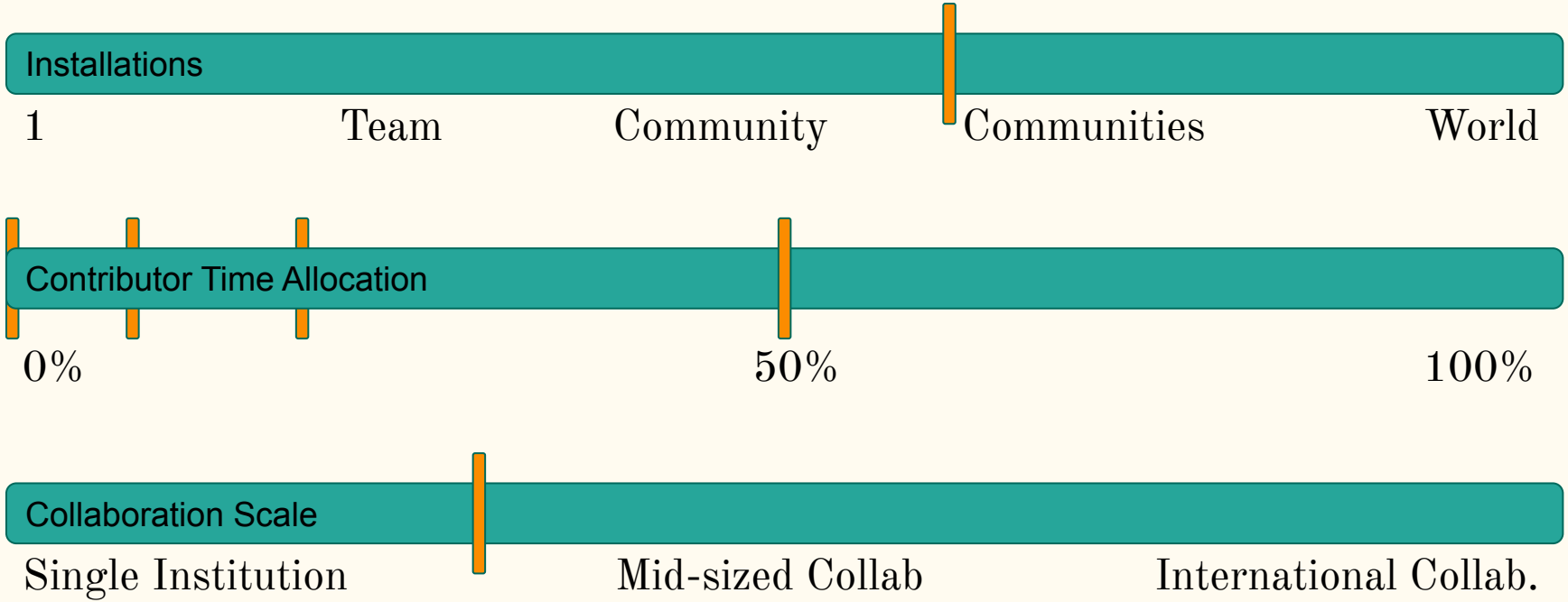
- NAWI (RO/Desal)
- DISPATCHES (Power Grid)
- PARETO (Produced Water)



# Sci SW Dev Contexts & Challenges: 6 scales (1-3)



# Sci SW Dev Contexts & Challenges: 6 scales (4-6)



# The Scrum answer…

At the end of a presentation at LBL on the Scrum software development methodology, when the presenter was asked:

> *How can we, in a research and scientific environment where our collaborators are spread across both multiple unrelated projects and time zones, best apply the Scrum methodology?*

After a long pause, his answer was to….

"*find another job*".

# Agile: Scrum or Kanban or ?

- Agile is a philosophy, Scrum & Kanban are methodologies
  - "*We value X over Y. While Y is important, X is moreso.*"
  - https://agilemanifesto.org/
  - Mike Heroux (SNL) 2019 Webinar: https://www.exascaleproject.org/event/agile/
- Scrum makes some (unsafe, for us) assumptions
  - Common location, Full-time participation, Single authority
- What parts do still apply?
  - All of the Agile Manifesto
- Kanban
  - More flexible, more visual ("card carrying") approach, more about continuous delivery
  - Still not a perfect fit, but better.
- I'm working a modified Kanban approach...

# My Approach: Practices & Roles

## Practices

- Scheduled Meetings
- Scheduled Releases
- Iterative, incremental improvements
  - To the process / practices themselves
- Education
  - Internal and external

## Roles

Facilitator

- The person driving the process and managing the process.
- Domain expertise not needed but SW Dev experience is

Contributors

- Sr developers
- Jr Developers
- Users

Stakeholders

- PIs, PMs, Industry Advisory

# Scheduled Meetings, Scheduled Releases

## Weekly telecons with tech team

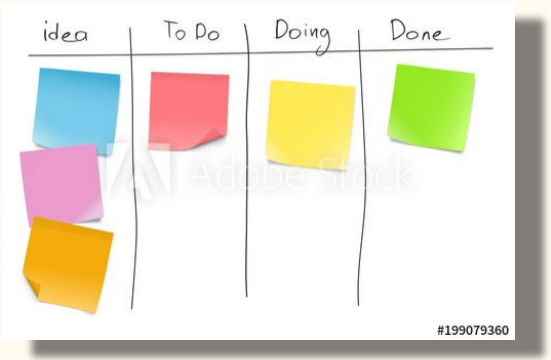
- Daily stand-ups impractical, weekly call usually possible
- Screen share, video on, builds team cohesion

## Date-driven over feature-driven releases

- "*If you miss this bus, there will be another one coming along soon.*"
- Subtle but effective motivation to meet dates

## Kanban Project Boards

- Priority board: All issues and PRs (backlog)
- Release board: Issues and PRs per release




## Results

- Open forum for technical discussions
- Testing, training, project milestones
- Establish supportive, productive culture

# Best Practices: Process, Tools and Team Culture

## Issues, Pull Request / Code Reviews

- Prioritize & target a release project board
- Issues: bugs, features, discussions
- PR / Code reviews:
  - Approved reviewers
  - CI testing, coverage, coding standards/style guide, static analysis, documentation
  - Education, supportive, culture building

## Developer Onboarding

- How to set up a dev environment
  - Run tests and generate docs locally
- How to set up a user env
  - Reproduce issues, test features
  - Supported OSs: Conda, Docker, VMs
- Copyright, license and IP issues
  - Open or closed, keep the wrong things out of the repo (even a branch or fork)

## Offboarding

- When someone leaves the project what needs to be done?
- Permissions removed, ownership hand-off

# Best Practices: Tools

*The specific tool you use is less important than that you actually use one*

## Version Control, Issues & Code Reviews

- git, github, gitlab

## Continuous Integration

- Automatically run tests & code analysis
- Unit, integration, style guide, static analysis
- With each PR & each commit in each PR
- Jenkins, github actions, gitlab runners
- Locally runnable in developer environment

## Continuous Deployment

- Nightly, Weekly
- Installs & long running tests

## Testing

- Test Plan: Governing document
- Test Levels: smoke, system, integration
- Coverage metrics
  - Not perfect
  - Linters, Black, Coverity, Coverage, Code-Checker
- Testing approaches:
  - Failures
  - Random/Fuzzy input
  - Performance
  - Security
  - Regression & Backward compatibility
  - many more...

# Best Practices: Documentation

- Automatically generated as part of CI
  - ReadTheDocs, Sphinx, Javadocs, Doxygen
  - With each PR and locally for developers
- “Executable” documentation
- Jupyter Notebooks

https://diataxis.fr/ “*The Grand Unified Theory of Documentation*”

- Tutorials
- How-Tos
- Background/Concepts
- Reference

# Iterate and SoapBox

- Iteration is educational, forgiving and forceful
  - Can and should be done at all scales
  - Seek feedback from everywhere
- Tech level: CI Testing
  - Start early and small, with simple examples, build incrementally
  - Add test coverage enforcement, linting, style guide enforcement
- Process level: Evangelize the Process
  - Technical team
  - Project Management
  - Improve the processes itself
- Future level: Perception of Scientific Software
  - Engage with Professional Organizations
  - Funding Sources

# Pearls of Wisdom...

The “Software is like cooking” analogy

Not all tech problems have tech solutions

- Person-to-person communication, coordination, compromise is sometimes required

Not all social problems have social solutions

- Tools can help: pull-request reviews, linters, coding standards, etc.

“*Culture eats strategy for breakfast*” - Peter Drucker(?)

# The Future: Scientific Software Stewardship

"Find another job"

*Rather than modify the process to fit the environment, what can we modify in the environment to fit the process?*

The emerging role of the RSE:

Research Software Engineer

- Identified career path
- Scientist or Engineer?
- Competition with industry
- Opportunity to recruit and retain under-represented groups

Orgs:

- US-RSE Association (https://us-rse.org/)
- Society of RSEs (UK) (https://society-rse.org/)
- Better Scientific Software (https://bssw.io/)

Funding Software Stewardship

- "Transitioning ASCR after ECP" report
  - Beyond HPC and ASCR?
- Sustainable software
  - Many dimensions: Karlskrona Manifesto
  - **Individual, Social, Economic, Environmental, Technical**
- Funding models
  - Hard, Soft, Mix?
- Return on Investment
  - Minimize churn of both software and people

# Summary and Conclusion

- Challenges of our environment
  - Distributed, Multi-disciplinary, Time-sliced developers
  - Example: IDAES project
- Neither Industry nor Scrum are the answers but both have much to teach us
- Proposed Approach
  - Scheduled facilitator-led meetings
  - Scheduled releases
  - Continuous, Iterative improvement
  - Soapboxing: Evangelize from the bottom up
- First-class citizens of Science Mission
  - Scientific Software Stewardship
  - Scientific Software Careers
- What else?
  - Technical and social challenges will remain
  - Project Culture is the foundation for technical and social solutions
  - Address our Science/Research challenges in addition to accommodating them

# References


Mike Heroux's Agile webinar (2019):

- https://www.exascaleproject.org/event/agile/

"Transitioning ASCR after ECP" Report (2020):

- https://science.osti.gov/-/media/ascr/ascac/pdf/meetings/202004/Transition_Report_202004-ASCAC.pdf

Karlskrona Manifesto

- https://www.sustainabilitydesign.org/karlskrona-manifesto/

RSE Orgs, Conference & Workshops

- US Research Software Engineer Association (US-RSE): https://us-rse.org/
- Society of Research Software Engineers (UK-RSE): https://society-rse.org/
- Better Scientific Software (BSSw): https://bssw.io/
- Collegeville Workshops: https://collegeville.github.io/
- Body of Knowledge for Software Sustainability (BoKSS): https://bokss.github.io/