Introduction to software sustainability

Sustainable Software in HEP Workshop
https://indico.cern.ch/event/930127
22 July 2020

Daniel S. Katz
d.katz@ieee.org, @danielskatz

ILLINOIS
NCSA | National Center for Supercomputing Applications

Workshop background & goal

- HL-LHC and other facilities of the 2020s will be relevant through at least the 2030s
- Related software efforts need to consider sustainability to enable their adaptability to new challenges, longevity, and efficiency
- Helps ensure software will be easier to develop and maintain, remains available in the future on new platforms, meets new needs, and is as reusable as possible
- Sustainability practices could lead to new collaborations, including elements of HEP software being of direct use outside the field, and HEP developers contributing to software developed outside the field rather than reinventing it, as has happened more frequently in recent years
- Give HEP software developers an important skill that is essential to careers in the realm of software, inside or outside HEP

- Workshop goal:
  - Bring together experts from HEP as well as those from outside to share their experiences and practices
  - Articulate a vision that helps IRIS-HEP in creating a work plan to implement elements of software sustainability

Software collapse

- Software stops working eventually if is not actively maintained
- Structure of computational science software stacks:
  1. Project-specific software (developed by researchers): software to do a computation using building blocks from the lower levels: scripts, workflows, computational notebooks, small special-purpose libraries & utilities
  2. Discipline-specific software (developed by developers & researchers): tools & libraries that implement disciplinary models & methods
  3. Scientific infrastructure (developed by developers): libraries & utilities used for research in many disciplines
  4. Non-scientific infrastructure (developed by developers): operating systems, compilers, and support code for I/O, user interfaces, etc.
- Software builds & depends on software in all layers below it; any change below may cause collapse

K. Hinsen, "Dealing With Software Collapse," 2019.
https://doi.org/10.1109/MCSE.2019.2900945

Defining research software sustainability

- Research software sustainability is the process of developing and maintaining software that continues to meet its purpose over time, which includes that the software adds new capabilities as needed by its users, responds to bugs and other problems that are discovered, and is ported to work with new versions of the underlying layers, including software as well as new hardware

- In order to sustain research software, we can
  - Do things that reduce the amount of work needed
  - Do things that increase the available resources
  - Do things that do both reduce the amount of work needed and increase the available resources

Methods to sustain research software (1)

- To reduce the amount of work needed
  - Train its developers, which involves finding or developing training material
  - Use best practices, which involves finding or developing best practices

- To increase the available resources
  - Create incentives so that people want to work on the software
    - Citations that help in existing career paths
    - Adjusted existing career paths that they reward software work
    - New career paths
  - Increase available funding by first making the role of software in research clear to research funders, and then by clearly making the case for them to increase funding for new software, and to provide funding for software maintenance
  - Seek institutional resources if the software is considered sufficiently important to the institution, operationally or reputationally

Methods to sustain research software (2)

- To both reduce work and bring in new resources, encourage collaboration
  - Using the work of others rather than reimplementing a function or package reduces what a software team (or its developers) needs to do themselves, even without assuming that the collaborators contribute to the software, which also may happen
  - Similarly, if others use a team's software and contribute to maintaining it, the team has less they need to do
  - To make this work, software has to be designed from the start to be modular and reusable, and it must also be clearly documented and explained to potential users, even those in fields other than the developer's
  - And the team has to put effort into engaging and working with the potential user and contributor community

Volunteers & incentives (1)

- Why do volunteers or collaborators choose to put effort into our software project?
- How we can engage them?

- In the context of community activities and organizing, Porcelli defines

  Engagement = intrinsic motivation + extrinsic motivation + support – friction

- Intrinsic motivation = self-fulfillment, altruism, satisfaction, accomplishment, pleasure of sharing, curiosity, real contribution to science
- Extrinsic motivation = job, rewards, recognition, influence, knowledge, relationships, community membership
- Support = ease, relevance, timeliness, value
- Friction = technology, time, access, knowledge

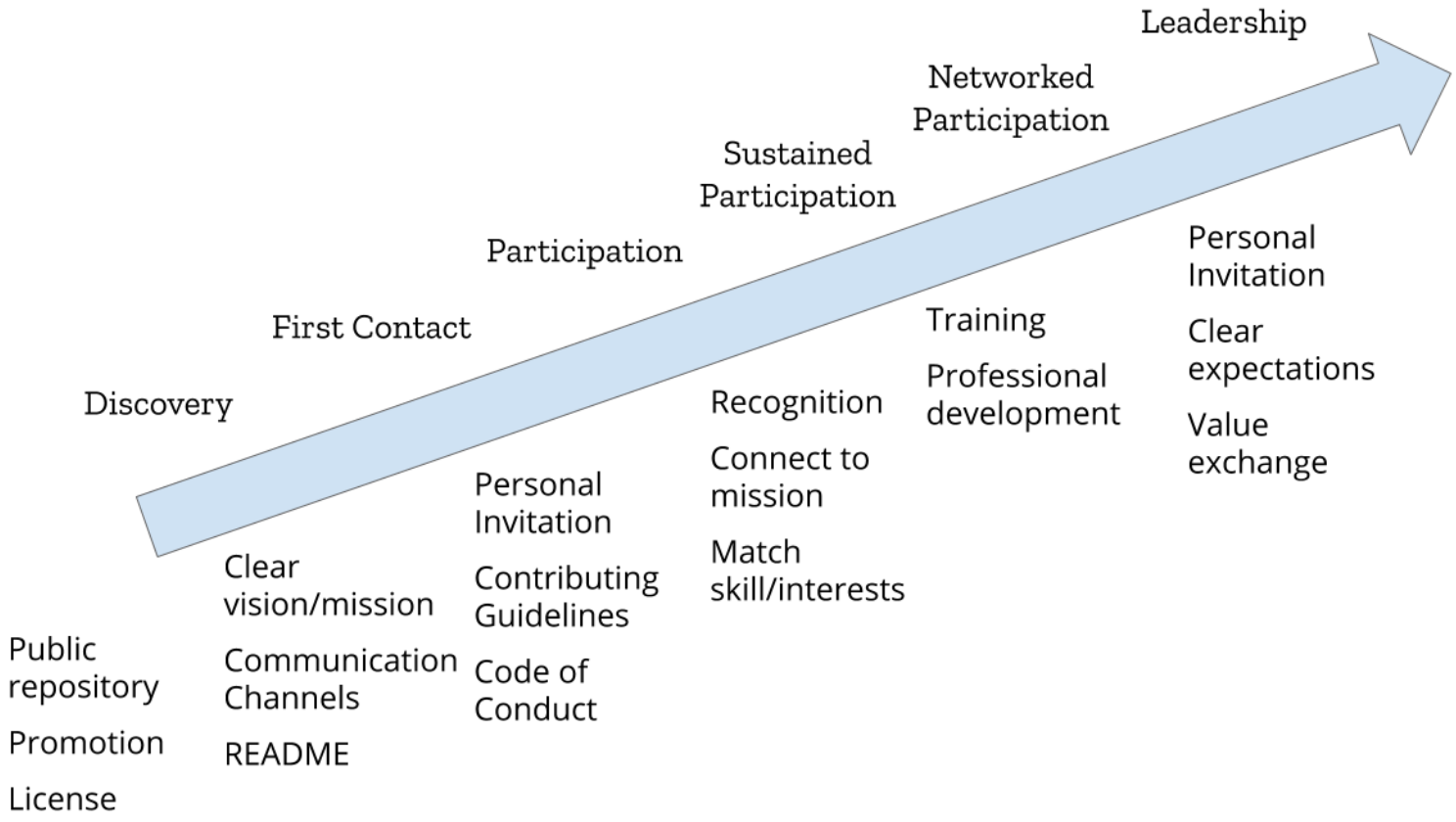
Joseph Porcelli, "How to grow users into active community members and get your community more engaged", 2013 Open Source Software Summit

Volunteers & incentives (2)

- Examples of things we can do:
  - Use GitHub for development
    - Reduce friction by using a known technology
  - Provide templates for issues and guidelines for good pull requests
    - Reduce friction by providing knowledge of how to work with our project
    - Increase support by easing the means of doing so
  - Provide a code of conduct and a welcoming and encouraging environment
    - Increase extrinsic motivation by helping develop relationships and sense of community
  - Add contributors to a list of authors who are cited when the software is used
    - Increase both intrinsic motivation and extrinsic motivation through recognizing accomplishments
  - Highlight examples of how the software is used
    - Increase intrinsic motivation by demonstrating the contribution to science

Volunteers & incentives (3)

- Plan for a progression of types of engagements
- How a project can encourage the potential contributor to move to from level to another



Organizations & communities

- Software Sustainability
  - Software Sustainability Institute (SSI)
  - US Research Software Sustainability Institute (URSSI) Conceptualization
- Research Software Alliance (ReSA)
- Research Software Engineering (RSE)
  - Society of Research Software Engineering
  - US-RSE Association
  - Groups in other countries/regions (DE, NL, Nordic, BE, AUS/NZ)
- The Carpentries

Agenda 1: Introductions & HEP experiences

CST	CEST	Title	Speaker
9:00	16:00	IRIS-HEP Blueprint process	Mark Neubauer
9:05	16:05	Introduction to software sustainability	Daniel S. Katz
9:20	16:20	Experiment experiences	Edward Moyse
9:35	16:35	Experiment experiences II	Danilo Piparo
9:50	16:50	HSF: HEP Software Foundation	Graeme A. Stewart
10:05	17:05	Community Software Successes & Failures	Elizabeth Sexton-Kennedy
10:15	17:15	Break (10 minutes)	

Agenda 2: Software, training, careers

CST	CEST	Title	Speaker
10:25	17:25	Software Sustainability Institute (SSI)	Neil Chue Hong
10:35	17:35	Astropy	Adrian Price-Whelan
10:45	17:45	The Carpentries	Erin Becker
10:55	17:55	HSF training	Samuel Ross Meehan
11:05	18:05	Research Software Engineers (RSEs)	Ian Cosden
11:15	18:15	Software maintenance and capacity building in HEP	Ketevi Adikle Assamagan

Agenda 3: Discussion & planning

CST	CEST	Title	
11:25	18:25	Breakout discussions (45 min)	• Participants assigned to groups randomly • All groups choose leader, scribe, reporter • All discuss same topic: software sustainability & HEP • **Output: 3 specific actions that the HEP community could take to make software more sustainable. At least 2 should be things that could be accomplished within 2 years, or at least that would make significant progress that would lead to a measurable difference within 2 years.** • Use google docs for notes • See detailed instructions in indico
12:10	19:10	Break (5 min)	
12:15	19:15	Reports from breakouts	Reporter from each group provides outputs in 2-3 minutes
12:30	19:30	Planning next steps	How should community (IRIS-HEP, HSF, …) move forward
13:00	20:00	Adjourn	