THE XPERT NETWORK

Workshop on Best Practices and Tools for Computational and Data-Intensive Research

In conjunction with the International Conference on Supercomputing (ICS 2019)

Editors:

Rudi Eigenmann

Parinaz Barakhshan

University of Delaware

https://sites.udel.edu/xpert-cdi/

Supported by the National Science Foundation

FORMAT OF THIS WORKSHOP REPORT 2
SECTION I: WORKSHOP MOTIVATION & FORMAT 3
WORKSHOP MOTIVATION 3
WORKSHOP FORMAT 3
WORKSHOP AGENDA 4
WORKSHOP ATTENDEES 4
SECTION II: RAW WORKSHOP OUTCOMES 6
TOPIC 1: BEST PRACTICES FROM YOUR USE CASES 6
Topic Motivation 6
Scribe's summary (Robert Sinkovits) 8
TOPIC 2: CHAPTERS OF BEST PRACTICES GUIDE 11
Topic Motivation 11
Scribe's summary (Maythal Dahan) 12
TOPIC 3: HOW CAN WE COORDINATE OUR PROJECTS? 14
Topic Motivation 14
Scribe's Summary (Sandra Gesing) 15
TOPIC 4: INTERACTION WITH TOOL DESIGNERS 17
Topic Motivation 17
Scribe's summary (Sameer Shende) 18
TAKEAWAYS 19
SECTION III- Best Practices 21
Discovery phase 21
Design phase 22
Development phase 23
Testing Phase 25
Deployment phase 25
Maintenance Phase 26
SECTION IV- Workshop Evaluation 27
Highlights 27
REFERENCES 28

FORMAT OF THIS WORKSHOP REPORT

This report has four Sections. Section I explains the motivation and format of the workshop. Section II presents the raw workshop outcomes in essentially unedited form (except for typos and small improvements for readability), giving the reader an impression of what happened at the meeting, followed by the main take-away points that the attendees mentioned. Section III summarizes the best practices in Xpert assistance mentioned in the position papers submitted to the workshop, combined with those discussed at the workshop. Section IV provides key points of the workshop evaluation.

SECTION I: WORKSHOP MOTIVATION & FORMAT

WORKSHOP MOTIVATION

This workshop was one of a series of events that aim to bring together projects and individuals that provide support to domain scientists developing computational and data-intensive (CDI) applications.

The series is supported by the National Science Foundation under award # OAC-1833846, entitled “The Xpert Network: Synergizing National Expert-Assistance and Tool-Support Teams for Computational and Data-Intensive Science,” https://sites.udel.edu/xpert-cdi

The workshop was by invitation. Invitees were from three groups:

- Those who provide direct assistance to CDI researchers (called Xperts in this report; also called Research Software Engineers or Research Facilitators in other contexts)
- CDI domain researchers
- Designers of tools that facilitate CDI applications development

The intent of the workshop was to:

- Create synergy among Xperts, by introducing them to each other and discussing best practices, and
- Improving the toolbox for CDI application development, by bringing Xperts and CDI domain researchers together with tool designers.

WORKSHOP FORMAT

The workshop was conducted in World Café style. The World Café is a structured conversational process for knowledge sharing in which groups of people discuss a topic at several tables, with individuals switching tables periodically and getting introduced to the previous discussion at their new table by a "table host".

Key World Café principles and steps include setting the context, creating hospitable space, exploring questions that matter, encouraging everyone's contribution, connecting diverse perspectives, listening together for patterns and insights, and share collective discoveries. [1]

WORKSHOP AGENDA

This was a half-day workshop on June 26, 2019, from 8AM to 11AM. The organizers picked four topics for discussion at the workshop, documented in Section II. After an introductory presentation, the attendees split into four groups, one per table, for these discussions. The attendees were encouraged to use large "scribble sheets" to express their thoughts in an informal style. The scribble sheets are included in Section II. Every 30 minutes, the participants were asked to go to the next table, at random, for contributing to a different topic. A scribe, or table host, for each topic stayed with the table, providing continuity. Following the discussions, the scribes presented a brief summary in a plenary session. There was no preparation time for the summaries, so they represent the scribes' immediate impressions. Section II also includes these summaries. Immediately after the workshop, the participants were asked for their main take-away points from the workshop. Section II includes the responses as well.

WORKSHOP ATTENDEES

	Name	Affiliation
1	Dahan, Maytal	Texas Advanced Computing Center (TACC)
2	Shende, Sameer	University of Oregon
3	Jha, Shantenu	Rutgers University
4	Sinkovits, Robert	San Diego Supercomputer Center (SDSC)
5	Yan, Yonghong	University of South Carolina
6	Arora, Ritu	Texas Advanced Computing Center (TACC)
7	Neeman, Henry	University of Oklahoma
8	Gesing, Sandra	University of Notre Dame

9	Brazil, Maria Isabel Marisa	Purdue University
10	Sanielevici, Sergiu	Pittsburgh Supercomputing Center (PSC)
11	Liao, Chunhua	Lawrence Livermore National Laboratory
12	Olschanowsky, Cathie	Boise State University
13	Alameda, Jay	University of Illinois
14	Zhang, Chengxin	University of Michigan
15	Cheatham, Thomas E.	Center for High Performance Computing, University of Utah
16	Bockmon, Ryan Joseph	Lawrence Livermore National Laboratory
17	Bartschat, Klaus	Department of Physics and Astronomy, Drake University
18	Pistorius, Julian	BIO5 Institute & CyVerse
19	Hall, Mary	University of Utah

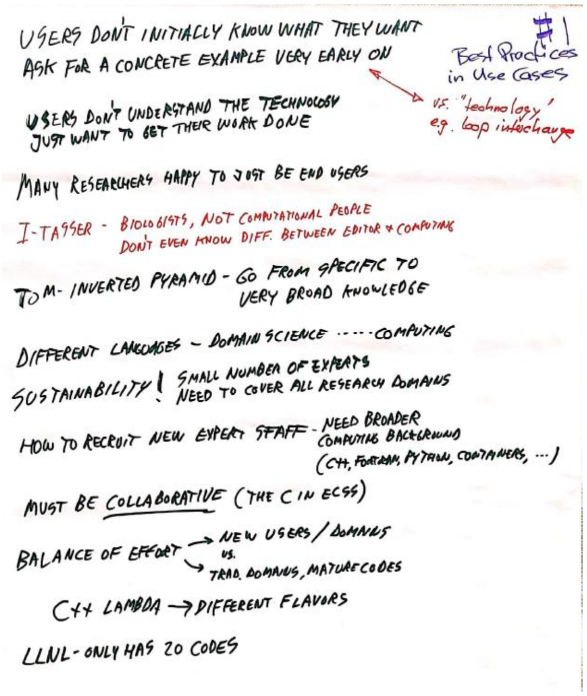
SECTION II: RAW WORKSHOP OUTCOMES

This section contains the direct outcomes of the workshop in essentially unedited form. The scribble sheets and summaries for each of the four topics are included, followed by the main take-away points that the attendees mentioned right after the workshop.

TOPIC 1: BEST PRACTICES FROM YOUR USE CASES

Topic Motivation

The primary outcome of the Xpert project, of which this workshop is one element, is the creation of a Best Practices Guide for the community of Xperts. The discussion on Topic I contributes to this outcome in a direct way.


#1
Best Practices in Use Cases

USERS DON'T INITIALLY KNOW WHAT THEY WANT
ASK FOR A CONCRETE EXAMPLE VERY EARLY ON

vs. "technology" e.g. loop interchange

USERS DON'T UNDERSTAND THE TECHNOLOGY
JUST WANT TO GET THEIR WORK DONE

MANY RESEARCHERS HAPPY TO JUST BE END USERS

I-TASSER - BIOLOGISTS, NOT COMPUTATIONAL PEOPLE
DON'T EVEN KNOW DIFF. BETWEEN EDITOR & COMPUTING

TOM - INVERTED PYRAMID - GO FROM SPECIFIC TO VERY BROAD KNOWLEDGE

DIFFERENT LANGUAGES - DOMAIN SCIENCE ····· COMPUTING

SUSTAINABILITY! SMALL NUMBER OF EXPERTS
NEED TO COVER ALL RESEARCH DOMAINS

HOW TO RECRUIT NEW EXPERT STAFF - NEED BROADER COMPUTING BACKGROUND
(C++, FORTRAN, PYTHON, CONTAINERS, ...)

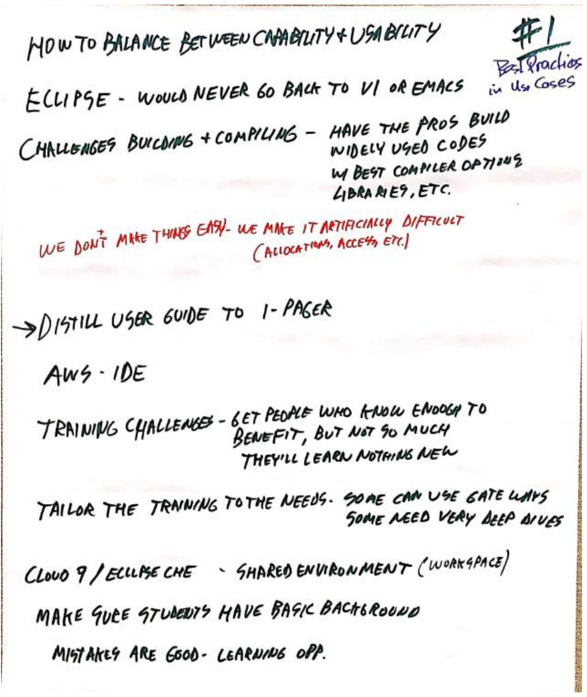
MUST BE COLLABORATIVE (THE C IN ECSS)

BALANCE OF EFFORT → NEW USERS / DOMAINS
vs.
→ TRAD. DOMAINS, MATURE CODES

C++ LAMBDA → DIFFERENT FLAVORS

LLNL - ONLY HAS 20 CODES

Figure 1


#1
Best Practices in Use Cases

HOW TO BALANCE BETWEEN CAPABILITY & USABILITY

ECLIPSE - WOULD NEVER GO BACK TO VI OR EMACS

CHALLENGES BUILDING + COMPILING - HAVE THE PROS BUILD WIDELY USED CODES
W/ BEST COMPILER OPTIONS
LIBRARIES, ETC.

WE DON'T MAKE THINGS EASY - WE MAKE IT ARTIFICIALLY DIFFICULT
(ALLOCATIONS, ACCESS, ETC.)

→ DISTILL USER GUIDE TO 1-PAGER

AWS - IDE

TRAINING CHALLENGES - GET PEOPLE WHO KNOW ENOUGH TO BENEFIT, BUT NOT SO MUCH
THEY'LL LEARN NOTHING NEW

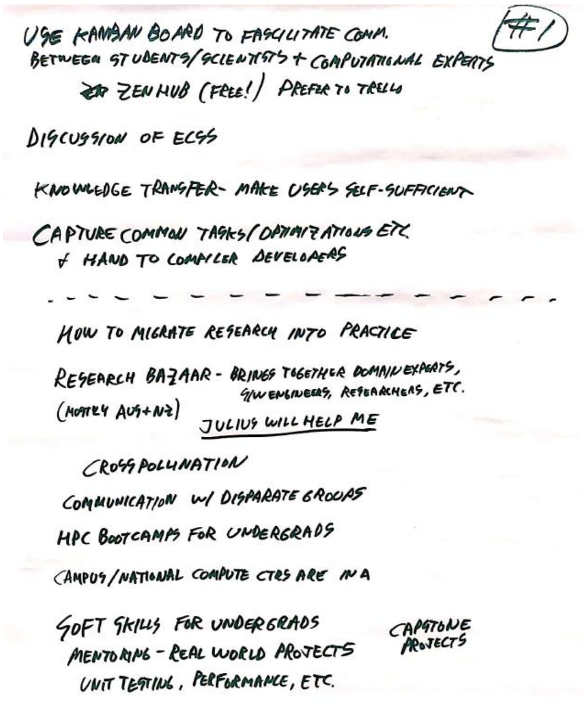
TAILOR THE TRAINING TO THE NEEDS - SOME CAN USE GATEWAYS
SOME NEED VERY DEEP DIVES

CLOUD 9 / ECLIPSE CHE - SHARED ENVIRONMENT (WORKSPACE)

MAKE SURE STUDENTS HAVE BASIC BACKGROUND

MISTAKES ARE GOOD - LEARNING OPP.

Figure 2


#1

USE KANBAN BOARD TO FASCILITATE COMM.
BETWEEN STUDENTS/SCIENTISTS + COMPUTATIONAL EXPERTS
~~ZH~~ ZENHUB (FREE!) PREFER TO TRELLO

DISCUSSION OF ECSS

KNOWLEDGE TRANSFER - MAKE USERS SELF-SUFFICIENT

CAPTURE COMMON TASKS / OPTIMIZATIONS ETC.
& HAND TO COMPILER DEVELOPERS

- - - - - - - - - - - - - - - -

HOW TO MIGRATE RESEARCH INTO PRACTICE

RESEARCH BAZAAR - BRINGS TOGETHER DOMAIN EXPERTS,
S/W ENGINEERS, RESEARCHERS, ETC.
(MOSTLY AUS + NZ)

JULIUS WILL HELP ME

CROSS POLLINATION

COMMUNICATION W/ DISPARATE GROUPS

HPC BOOTCAMPS FOR UNDERGRADS

CAMPUS / NATIONAL COMPUTE CTRS ARE IN A

SOFT SKILLS FOR UNDERGRADS
MENTORING - REAL WORLD PROJECTS
UNIT TESTING, PERFORMANCE, ETC.

CAPSTONE PROJECTS

Figure 3

Scribe’s summary (Robert Sinkovits)

For the topic of best practices from your use cases, we had a bunch of great discussions. We ended up being all over the place, which I think ultimately was very productive.

We started off with what I think is one of the best pieces of advice: users do not initially know what they want. Therefore, we need to ask for a concrete example early on. Doing this groundwork to get the project going makes a big difference. Sometimes the user would come to you with very vague needs. Like, my code is too slow. So, what we will do is try to dive in and figure out what exactly it is that they want and then we try to strip things down to test cases and problems that we can work with.

What one of my own home examples was, I had collaborators on the LSST (Large Synoptic Survey Telescope) [2] team about eight years ago. And the software stack was enormous, but he was working on just one small piece of it. So, I said try to trim that down to a tractable problem that we can work with. We found that users do not understand the technology. They just want to get their work done. So, we need to keep that in mind as we are working with our users.

We got into a lot of discussions with all the groups that there are different needs from different users. Some of them do not need to go very deep. An extreme example would be an ecologist who needs to construct phylogenetic trees [3]. They can just go to the CIPRES Science Gateway[4], upload sequences and they are done.

But one of the folks in our discussion said that he tries to create one-page User Guides for some of his users. Now, not everybody can get away with the one pager but sometimes if you distill things down to the key items, such as “here is how you log in, these are the file systems, here is your home, your scratch folder in your project's storage, this is how you compile or this is how you load a module, etc.” that could be everything they need.

We had a big discussion about balancing between capability and usability. And one of the best practices when it comes to training and working with users is that we take what we have, and we tailor it accordingly.

There are some folks who will need to go very deep in building and maintaining complex software stacks. They need to know every detail regarding the network performance, the processor architecture, all the compiler flags, how to make sure that all the correct libraries are linked in. This gets into the training challenges and how we can tailor things. You cannot make it perfect for everybody. So sometimes you are not going deep enough; sometimes you are going too deep. But if we could do a little bit of triage, it makes sure that our less advanced users are brought up to speed so that they can benefit from the training. Maybe for our workshops and summer institutes we could take the folks who are already working at a very advanced level who could almost be teaching what we are covering and try to nudge them out of our workshops.

There was a great comment on using Kanban boards [5] to facilitate communications between groups of collaborators between students, researchers and computational experts. I think this is a great idea. This is something that we have discussed doing in ECSS[6]. We have a much heavier-weight solution where you can use Jira [7] to maintain all our project. But Jane and I had discussed that we cannot really open up Jira to the research teams without getting into all sorts of access control issues, but maybe we can use something lighter weight that lives outside of Jira to allow our teams to communicate.

A lot of talk was about knowledge transfer. When we collaborate with users, there is the expression “throwing the code over the fence” - let me know when you are done with it, hand it back. We all agree that it does not work. But we do need to focus on the knowledge transfer.

We had a collaboration with researchers from the University of Arizona and I wish every one of our ECSS projects was like this. It was a successful project but also this particular PI went from knowing nothing about computing, like he did not know how to log into the system; he did not know what a batch scheduler’s work was; he did not know what a workflow was. By the time he was done, he was completely self-sufficient. So, part of our best practice must be whenever possible, we transfer the knowledge.

Finally, we had a great discussion about education, working with undergraduates and by bringing disparate groups together. Something called a research Bazaar [8] that I had not heard of before, which I think is a phenomenal idea. I am going to try to set one of these up in San Diego where it brings together domain experts, software engineers, researchers and so on because we all realize that the world has become so complicated that you just cannot know it all anymore. But sometimes, something that you may be struggling with or what somebody else is struggling with would be so simple. I give the example of working with James Cuff at Harvard years ago. He talked about a researcher who came to him and was so thrilled that they taught him how to use SCP [9]. This is the secure copy. Until then, the researcher gets to use the thumb drive and go back and forth. So, this cross-pollination, getting disparate groups together is the best practice.

Finally, we talked about working with undergraduates and I think we can do a lot, especially those of us that are at the campus and the national supercomputing centers. The students can get to work on the skills that they are not necessarily getting through the computer science curriculum. This could be anything from working on a run on real world projects, like some projects where they do the whole thing from planning to execution, until final report to working on soft skills within the communications and writing, including all of these other things that are really essential to go along with your technical skills.

TOPIC 2: CHAPTERS OF BEST PRACTICES GUIDE

Topic Motivation

What is a good structure of a Best Practices Guide? What chapters need to be there? The participants were asked to come up with a list of chapter topics. They were encouraged to think beyond technical issues (such as, we need to learn about MPI) and also consider topics addressing the terminology gap between Xperts and domain scientists, the issue of how to pick from among many CDI application projects that may need the Xpert's help, and how to find outside expertise if needed (e.g., need someone who knows AI to see if my application can be improved that way).

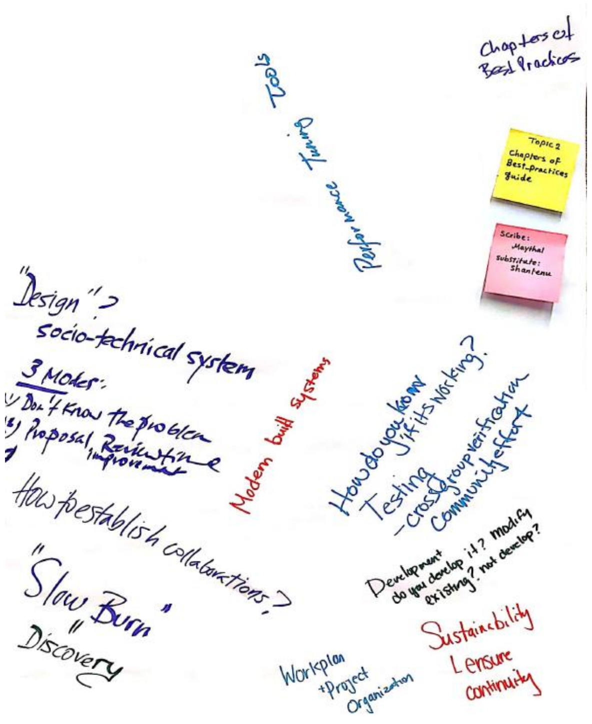

Chapters of Best Practices
Tools
Performance Tuning
Topic 2 Chapters of Best-practices guide
Scribe: Maythal Substitute: Shantenu
"Design"?
socio-technical system
3 Modes:
1) Don't know the problem
2) Proposal, Review time
improvement
How to establish collaborations?
"Slow Burn"
"Discovery"
Modern built systems
How do you know if its working?
Testing
-cross group verification
Community effort
Development do you develop it? modify existing? not develop?
Sustainability
L ensure continuity
Workplan +Project Organization

Figure 4

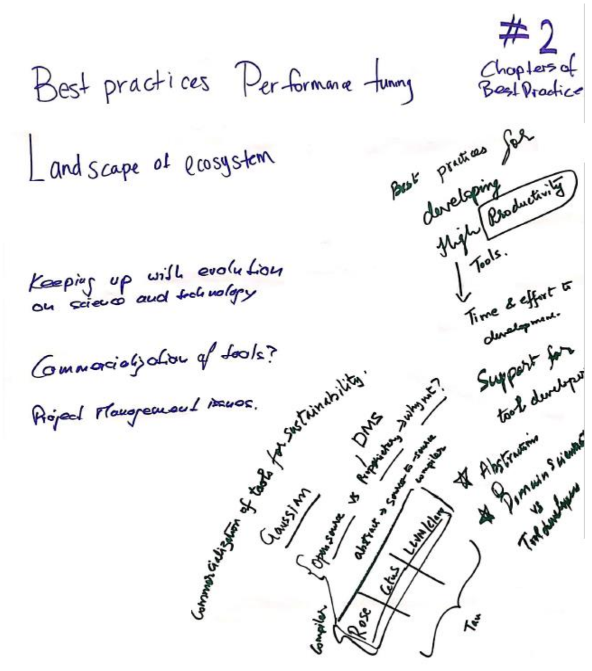
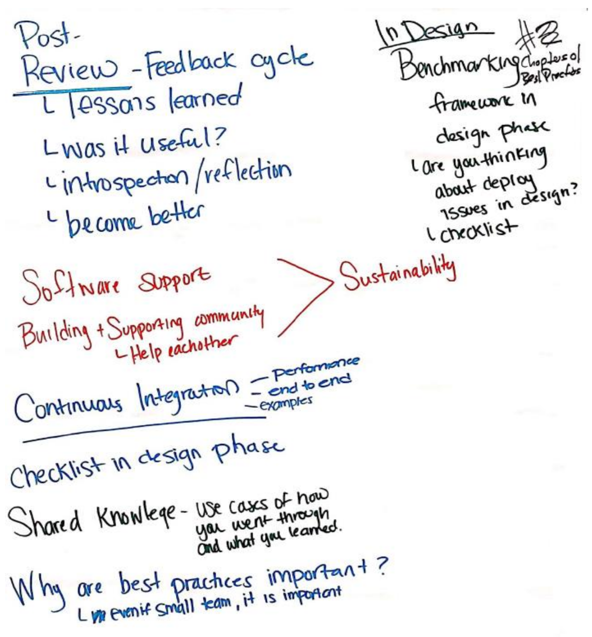

#2 Chapters of Best Practice
Best practices Performance tuning
Landscape of ecosystem
Keeping up with evolution on science and technology
Commercialization of tools?
Project Management issues.
Best practices for developing High Productivity Tools.
Time & effort to development.
Support for tool developers
Commercialization of tools for sustainability.
Gaussian
DMS
Open source vs Proprietary why not?
abstract ⇒ source-to-source compilers
Compiler
Rose
Cetus
LLVM/clang
Tau
Abstraction
Domain Scientists vs Tool developers

Figure 5



Post-Review - Feedback cycle
- Lessons learned
- was it useful?
- introspection/reflection
- become better

In Design #3
Benchmarking Chapters of Best Practices
framework in design phase
- are you thinking about deploy issues in design?
- checklist

Software Support
Building + Supporting community
- Help eachother

Sustainability

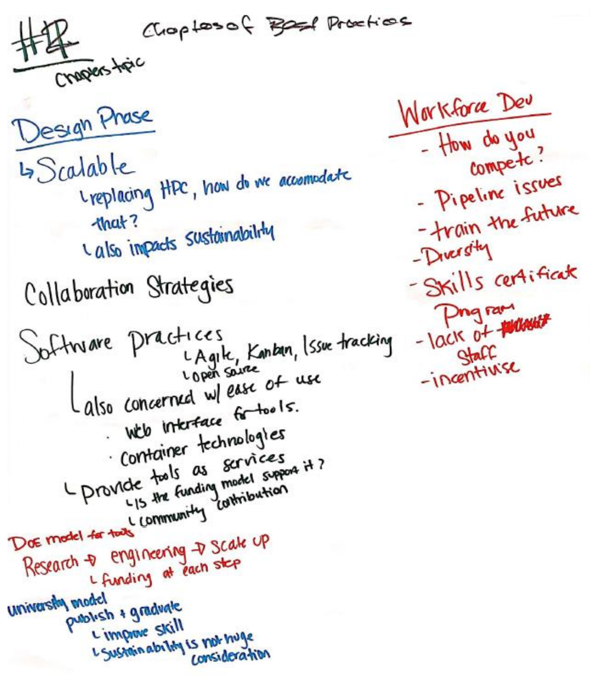
Continuous Integration
- Performance
- end to end
- examples

Checklist in design phase

Shared Knowlege - Use cases of how you went through and what you learned.

Why are best practices important?
- even if small team, it is important

Figure 6



#4 Chapters of Best Practices
Chapters topic

Design Phase
- Scalable
  - replacing HPC, how do we accomodate that?
  - also impacts sustainability

Collaboration Strategies

Software practices
- Agile, Kanban, Issue tracking
- open source
- also concerned w/ ease of use
  - web interface for tools.
  - container technologies
- provide tools as services
  - is the funding model support it?
  - community contribution

DOE model for tools
Research → engineering → scale up
- funding at each step

university model
publish + graduate
- improve skill
- Sustainability is not huge consideration

Workforce Dev
- How do you compete?
- Pipeline issues
- train the future
- Diversity
- Skills certificat Program
- lack of Staff
- incentivise

Figure 7

Scribe's summary (Maythal Dahan)

How are you interacting with a domain researcher that you are trying to help? Are you taking the time to really explain and transfer knowledge instead of coming up and saying I can fix this for you? Are you taking the time to explain your science to me, explain your research to me, before you even get down to the code development?

How do you establish collaboration?

Also, the different modes of how you are working with the domain researchers. So, some may come to you saying, “I don't know what my problem is”. Others might come to you saying, “here are the areas that I need help fixing”. So how are you interacting with them appropriately?

In the design phase, what is beyond those technical details, what the system should do, what are the properties?

A big issue that came across on all these sessions was sustainability and how are you ensuring continuity.

How are you balancing software engineering practices such as readability, understandability of your code, and documentation with the staff and the time constraints that you have and a lot of that really depends on the kind of environment that you are in?

Also, testing was something that came across well. So, in the design phase, are you thinking about how you are going to test this code?

One of the things that came about in this session was continuous integration. So, in the design phase, can there be a checklist of “here is all the things you are going to need to think about throughout the whole process”? How are you going to handle continuous integration? What is your end environment going to be like, and are you thinking about benchmarking issues? Are you thinking about deployment issues? This is like a checklist for designing software. For example, think of sustainability with regards to software support. Or, what will happen if your product or software is going to be used at a large scale? Then how do you build a community to support and help each other?

One of the things to consider is “post review”. So, when you are done, have you gone back and asked what are the lessons learned? What was useful and do some reflection and have that as part of your lifecycle?

One of the things that came up in the discussion was software practices. Are you being productive in how you are developing software? Are you thinking about how your users are going to use your software most productively, with ease of use?

Other questions are: How are you sharing your knowledge with your community? How are you sharing what you have learned with all others?

The landscape of the ecosystem would be useful to explain. It includes the possible funding models and the metrics of success for your tools.

One of the things we did not discuss was project management. How are you going to manage your project effectively? I think software engineers are pretty good at some of the software management tools but do not really think about the project management. And then do you know how you deal with the soft skills? How is your team going to work together? How are you going to communicate effectively?

TOPIC 3: HOW CAN WE COORDINATE OUR PROJECTS?

Topic Motivation

Coordination of the projects represented by the participants will have benefits, such as better exchange of ideas and reduced overlap. How can such coordination be done with little extra time and effort spent? For example, could existing project meetings be used to periodically invite participants from other projects?

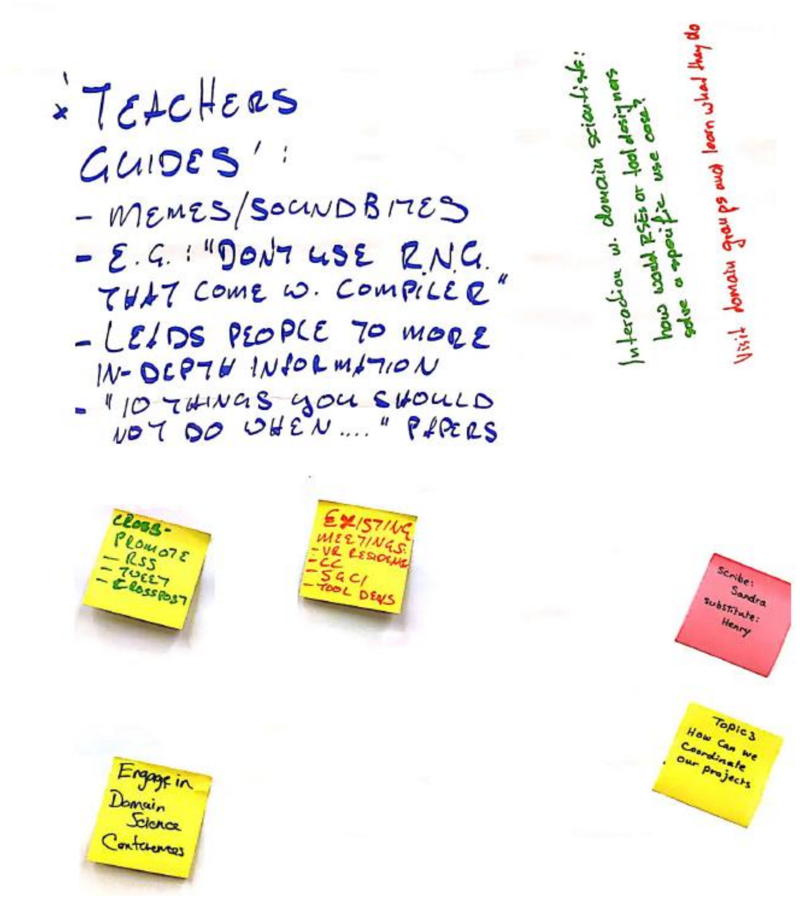

"TEACHERS GUIDES":
- MEMES/SOUNDBITES
- E.G.: "DON'T USE R.N.G. THAT COME W. COMPILER"
- LEADS PEOPLE TO MORE IN-DEPTH INFORMATION
- "10 THINGS YOU SHOULD NOT DO WHEN...." PAPERS
Interaction w. domain scientists:
how would RSEs or tool designers solve a specific use case?
Visit domain groups and learn what they do
CROSS-PROMOTE
- RSS
- TWEET
- CROSSPOST
EXISTING MEETINGS:
- VR RESIDENCE
- CC
- SGCI
- TOOL DEVS
Scribe: Sandra
Substitute: Henry
Engage in Domain Science Conferences
Topic 3 How Can we Coordinate our projects

Figure 8

Scribe's Summary (Sandra Gesing)

There are all the different organizations we know about, from Campus Champions[11] to software Sustainability Institute[12] to ACI-REF[13], and virtual residency[14]. How can we work the best together and how to get involved as a domain researcher?

One idea was that tool developers could talk at events like virtual residency, and webinars which already exist, about new tools. Doing so could avoid duplication because tool developers must go to each of the communities themselves. That is not scalable. There should be recommendations and examples for using or not using certain tools. It could also help guide users in the use of online material for the tools.

We know that facilitators and RSEs often have totally different backgrounds, so we need some material for them to work with domain researchers. One idea is to have something like a central repository. But the disadvantage of a central repository is always how to maintain it by all these different organizations. Who is responsible and who knows the status of all these projects?

So, one idea is to cross-advertise events of other organizations. For example Campus Champions would point to science gateways webinars[15], Science gateway people point to Campus Champions, to the virtual residency and the PEARC conference. So, for example, the virtual residency would not directly advertise a gateway catalogue. There would be a link to the science gateway institute's advertisements.

When somebody is new to the community, what would be the starting point for information? One idea is that it does not really matter with which group people start. As long as we all point to each other's communities, they will find out which is the best fit for them.

Another idea was to have a discussion forum, so that people can put topics out there. It is community-driven, and nobody has to maintain it. But maybe someone must edit it, to maintain appropriate content.

The other idea is to have the basic outreach via papers and Interviews. A central registry would be hard to maintain, because the domains are different. Having everything openly accessible could be a starting point.

Everything that is out there should be openly accessible through GitHub [16]. One of the ideas was also to go to domain conferences. To directly be in contact with domain researchers and to motivate them to work with positive experiences are really success stories. Sometimes there are financial limitations that keep Domain researchers from working with us. Including research software engineers in domain proposal takes of course some money away from the domain research. There must be a big motivation to shift funding in this way. Champions on the domain side could talk about their positive experiences and help create the motivation.

Presentations to the facilitators and at conferences like RDA [17] and Research Data Alliances [18], which has already a lot of domain scientists, would help in passing the knowledge.

One of the challenges for facilitators and research software engineers to support newcomers is to learn domain-specific skills. We need online tutorials and realistic environments, with prototypes and exercises. here are distribution channels via the different organizations. Software Carpentry [19] for example capitalizes the science of learning.

TOPIC 4: INTERACTION WITH TOOL DESIGNERS

Topic Motivation

Even though good tools can help tremendously in developing CDI applications, many tools are underused. By bringing together tool designers with Xperts and domain scientists, the Xpert project aims to address this issue. How can these communities best interact?

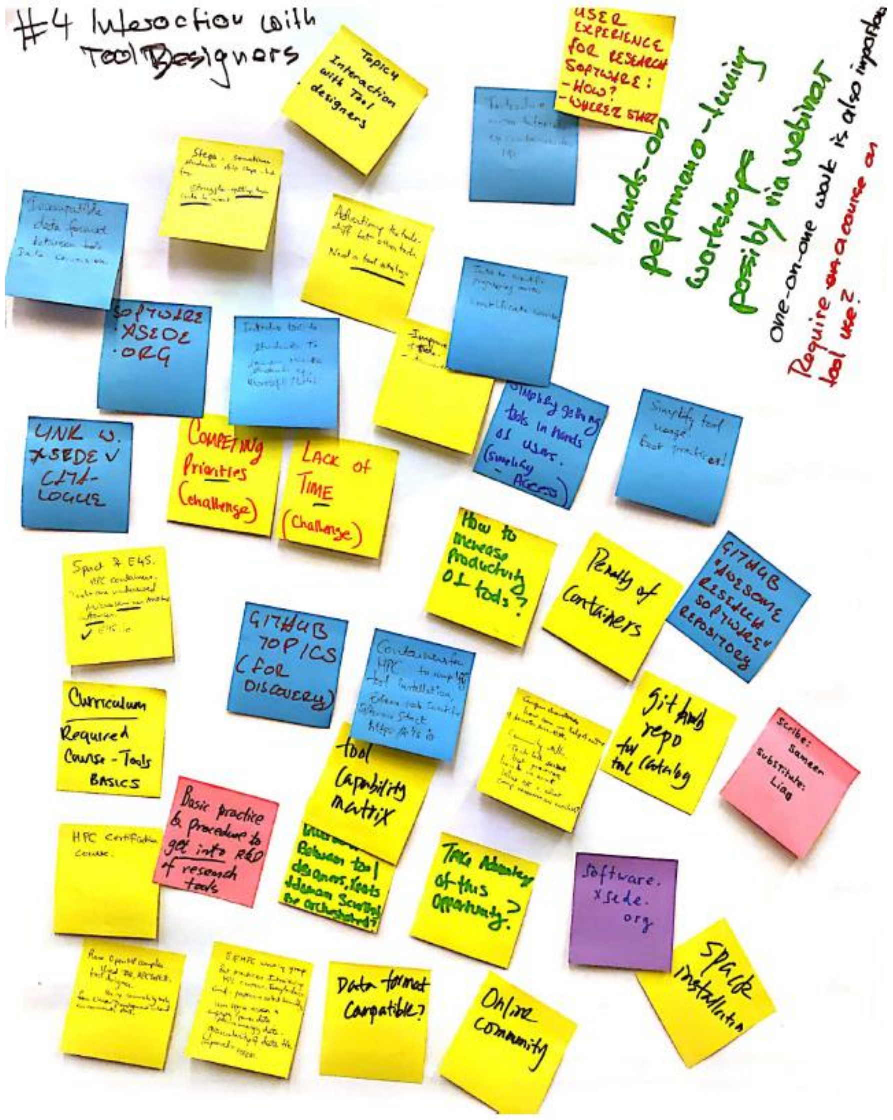

#4 Interaction with Tool Designers
Topic 4 Interaction with Tool designers
USER EXPERIENCE FOR RESEARCH SOFTWARE: - HOW? - WHERE? STARt?
hands-on performance-tuning workshops possibly via webinar
One-on-one work is also important
Require ~~as a~~ course on tool use?
SOFTWARE .XSEDE .ORG
LINK W. XSEDE V CATALOGUE
Competing Priorities (challenge)
LACK OF TIME (Challenge)
How to increase Productivity of tools?
Reality of Containers
GITHUB "AWESOME RESEARCH SOFTWARE" REPOSITORY
GITHUB TOPICS (FOR DISCOVERY)
Curriculum Required Course - Tools Basics
HPC Certification Course
Basic practice & Procedure to get into R&D of research tools
tool capability matrix
Git hub repo for tool catalog
Scribe: Sameer Substitute: Liang
Take Advantage of this Opportunity?
Software. xsede. org
Spack installation
Data format Compatible?
Online Community
Simplify tool usage Best practices!
Simplify getting tools in hands of users (Simplify Access)
Incompatible data format between tools

Figure 9

Scribe’s summary (Sameer Shende)

We're talking about the interactions between tool developers and how we increase the productivity of tools. For example, we can have one-page short descriptions of the tools, and have a catalog of tools (e.g. the research software portal provided by XSEDE), and a tool capability matrix. There can be an actual listing of the repositories of the various tools on Github, like there is in the extreme skill project.

We also look at the other big problem facing the tool community. HPC tools are getting more and more complex and very hard to install. How do we address that issue? Because the dependency matrix can sometimes be 40 deep, or can be a hundred deep, installing all the dependencies before the tools can be deployed correctly becomes a challenge. This is solved in one way by the Spack platform[20] that the Department of Energy's (DOE) Exascale Computing Project (ECP) is funding from Lawrence Livermore National Laboratory (LLNL). And, by using these containers from the Extreme-scale Scientific Software Stack(E4S)[21]. So, you can have containerized distributions, where all the tools are pre-installed with Spack that can assist in some of these things.

But there are other issues like the basic practices and procedures to get R&D into research tools. There are issues such as curriculum development, training, and certification. Leveraging online learning management systems and sandboxes may facilitate the training. Both grad students and domain scientists should be taught the usage of those tools and basic skills. There are also discussions about common data formats of tool outputs to enhance interoperability among tools.

And there are also some other practices like hands-on performance tuning workshops that can be valuable, and also Webcasts. But the one-on-one work is also important. And integrating all these concepts in a curriculum or a course can help.

TAKEAWAYS

Immediately after the workshop, the participants were asked for their main take-away points. These are the responses.

“There's an opportunity to discover whether software tool developers can get more scalable outcomes by working with the research computing facilitator community. In particular, if developers teach facilitators about their software tools, then facilitators will be well positioned to introduce those tools to researchers who could get value from them, but might otherwise not encounter them.”, ***Henry Neeman***

"The ecosystem around facilitators and research software engineers (RSEs) is complex and hard to define. Interaction between different projects concerned with improving the situation for facilitators and RSEs is to point to each other’s actions and websites and maybe a central repository - the challenge is to keep the latter up to date. A key component is to involve domain researchers, work on online tutorials, realistic environments, prototypes and spread the word about positive experiences by domain researchers to reach more domain researchers.”, **Sandra *Gesing***

“1) There is a lot of stuff out there that could be useful to me (and my group), but I really don’t (didn’t) know about them. One particular example that stuck with me concerned the “containers” that Sameer mentioned. They seem to be useful when installing packages — we spend a lot of time hunting down all the dependencies, finding where somebody might have installed the package(s) that we need, etc. Having it all in one place seems very useful.

2) I don’t think we came up with a perfect solution (if we had, the problem would be trivial anyway), but it was good to brainstorm how to improve the communication between tool developers and tool users. We really need to keep working on that issue.

3) I also found it useful to talk about optimizing the ECSS support through XSEDE. Even though it seems to be working reasonably well, it’s always good to investigate if/how the program could be improved.”, ***Klaus Bartschat***

“Training domain experts from basic research for best practices and popular tools in CDI is important, but hard to achieve due to general lack of interest from the audience. CDI training for such audience would be more approachable if it is coupled with usage example closely associated with domain experts' research.”*,* ***Chengxin Zhang***

“I was impressed and encouraged by the assembled participants' willingness to collaborate in the creation of an overarching system for the professional development and sustained interaction of people who help researchers make efficient use of advanced computing and data analysis infrastructure. The challenge is how to organize such a system, who should do it, and with what funding.”, **Sergiu *Sanielevici***

“The main takeaway I got from the workshop was that there is a need to have a website or collection of all the different tools available; what they do, how are they different, how to use them, etc. The other takeaway is that there is a need for classes to teach how to use these different tools.”*, **Ryan Joseph Bockmon***

“There is a need to invest time and effort in creating high-productivity tools that can help not only the domain-scientists but also the tool developers who help in provisioning such high-productivity tools. ‘’, ***Ritu Arora***

“it is extremely valuable to regularly attend inter-agency (e.g. DOE and NSF) information exchange meetings for tool development. I learned a lot from XSEDE and they are doing impressive work.”***, Chunhua Liao***

“The workshop was a wonderful opportunity to meet a diverse set of attendees interested in Computational and Data Intensive tool development. These diverse backgrounds offered well-balanced and innovative conversation in order to come out of the workshop with interesting ideas and outcomes. The format of the workshop created a productive mechanism for which to gather and share ideas on each topic area. I look forward to seeing the outcome of the workshop and participating in future events. I also left the workshop meeting attendees that I plan to connect to in the future for potential collaborations and opportunities.”***, Maytal Dahan***

“Usability of HPC resources are still challenging for average HPC users. Enabling access from cloud or web interface via cloud, IDE or Jupyter Notebook is an option that we could give a try.”***, Yonghong Yan***

“This was a proof-of-concept of what can happen when you put experts together in a room to exchange best practices. As it turns out this was a very valuable (not surprisingly). Ideas flowed, creative solutions emerged, and connections were made.”***, Julian Pistorius***

SECTION III- Best Practices

This section extracts the best practices in Xpert assistance based on what is described in the position papers sent by the participants before the workshop and combines them with those discussed at the workshop.

A list of Best Practices can be organized in many ways. This workshop report uses the five stages of the software lifecycle: Discovery, Design, Development, Test Deployment and Maintenance phases for organization.

Discovery phase

1. We write with our users two types of documents: use case specification and software requirements specification. Use case documents describe deployment and execution scenarios alongside the abstract workflow users want to implement. Software requirement documents describe in progressive detail the software system we are going to co-design and then develop to satisfy one or more use case documents. Both documents use templates tailored to serve cross-domain research and are constantly updated in collaboration with the users. This accommodates the progressive process of mutual understanding between developers and domain-scientists and enables rapid prototyping of software solutions.
2. Initially, we simplify use cases, requirements, and capabilities to deliver simple, baseline results in the very first phases of the project. This approach helps to reduce feature creep and gives the opportunity to the user to develop an immediate understanding of the relation between design properties and scientific results. During the project, we progressively and carefully increase complexity, at a pace dictated by the users.
3. The time spent eliciting requirements and that taken to deliver solutions needs to be minimized. We manage this tension by engaging users across software development activities and by adopting iterative processes with relatively short cycles.
4. Two types of meetings are arranged for coordinating all management and technical stuff. Management meetings have a bi-weekly cadence while we found that a weekly schedule is better for technical discussions.
5. Communication is key when developing and improving CDI applications. It leads to getting the right requirements from the user.
6. Set up regular in-person meetings or online video calls to keep communication with the user personally.
7. Empathy. If you care for your users, they will care for your product.

8. Users don't initially know what they want. Therefore, we need to ask for a concrete example early on. Doing this groundwork to get the project going makes a big difference.
9. When the user comes to you with very vague needs, what we will do is trying to dive in and figuring out what exactly they want, and then we try to strip things down to test cases and problems that we can work with.
10. Most users do not understand the technology. They just want to get their work done. So, we need to keep that in mind as we are working with our users.

Design phase

1. User interface design in high performance computing (HPC) has largely languished with traditional command line interfaces. An increasing majority of individual is no longer familiar with command line interfaces.
2. Make the design integrable so that researchers can keep working in their chosen computational environment and can receive additional features instead of having to switch to a different piece of software.
3. Provide a user-centric view to support research more effectively by considering usability, scalability, and interoperability.
4. Usable design and execution efficiency are simultaneously achieved when code design includes a specific kind of separation of concerns. Separation of concerns generally refers to breaking an application into sections that address specific concerns (modular programming). The concerns we propose separating are the primary computation, the execution schedule, and the storage mapping. The primary computation is the algorithm or mathematics being performed; the order the expressions composing the primary computation are executed is the execution schedule, and the storage mapping is how the values required and written by the computation are stored in memory and to disk.
5. Developing Tools with less steep learning curves will make it easier for us to educate our PIs and their teams in the use of the tools.
6. Be open to the idea of working with other organizations to expand the scope beyond your organization and create a broader repository that can be used as a starting point for others who need to carry out similar work.
7. As the hardware platforms on which the CDI applications typically run have a short lifespan of 4-5 years, it is prudent to “future-proof” the applications so that they continue to be usable and scalable on the future hardware platforms and hence enjoy a long life.
8. At the software design stage, one can consider adopting the software engineering principles such as the Open-Close Principle (OCP), as per which, the software is designed such that it is closed for modification but open for extension. For building, deploying

and distributing self-contained applications that can run on different but comparable hardware architectures, one can consider containerization. For scalability, the applications can be designed and developed to work in distributed computing environments.

9. Minimize the steps it takes to complete a task.
10. Keep the interface clean. Having a busy or noisy interface can cause frustration when trying to learn a new application.

Development phase

1. Successful approaches are characterized by being technology agnostic, using APIs and standard web technologies or delivering a complete solution for serving a community efficiently.
2. Scientific applications often use embedded domain-specific languages (eDSLS) that achieve separation.
3. Disseminate the information that was gained through the projects. It gives the staff an opportunity to publish findings and results that are not necessarily a good fit for journal publications or conference proceedings. For example, these reports may go into the low-level details illustrating how the code was modified to achieve higher performance or improved scalability. These technical reports can capture valuable information that might otherwise have been lost.
4. Having a strong connection with your users will provide you with some great feedback.
5. Adopting cloud-based development environments, such as Eclipse Che and Amazon Cloud9, for collaborative software development, documentation, debugging and testing. Tools for continuous integration, such as Travis CI, also become cloud-based such that testing is triggered on cloud servers when commits are pushed to the source code repository. Those tools significantly improve development productivity and reduce the amount of human involvement for routine setup and maintenance tasks. Due to the collaborative nature, software tools and applications for scientific research should consider cloud-based development, documentation, and testing. There would be some benefits for it but also some disadvantages. The benefits are tremendous, such as:

    I. Enabling real-time development on the same project, reproducing results, and debugging of a problem from the same session;
    II. Enabling access to a development environment using a web browser from anywhere;
    III. Improving productivity by using and sharing complex scientific software or hardware stack needed to develop tools and applications;
    IV. Facilitating training of students and new developers;

V. Enabling quick access to special hardware (such as latest GPUs) without paying the cost of buying the hardware;
VI. Enabling a seamless connection with other cloud-based services such as source code, hosting, continuous integration, and deployment of tools as cloud-based service.

6. Use libraries that have been developed by others when possible.
7. Try to write code that is reusable.
8. Documentation. If a code is not well documented, it becomes useless as time goes on. If one gets into the habit of adding comments directly in the code, this may be very helpful to other developers.
9. Accessibility Tests: Know your users. Find out if your users have any disabilities that could impair their use of your application.
10. Usability Test: making sure that the application is easy to use and easy to adapt.
11. Improve the application without having to change the look and feel of it. Nothing is more frustrating than having to relearn an application after a major update.
12. Each system has a set of explicitly defined: (1) entities;(2) functionalities that operate on these entities; and (3) states, events and errors for each entity. These systems become building blocks because they can be integrated with minimal code writing both among themselves and with systems independently developed by third-party developers. While this approach tends to be 'design-heavy' and an unstructured and rapid development approach
13. Basing our development and user support on GitHub. We use tailored branch and release models, pull requests, unit and integration tests, continuous integration, and style guide enforcement. We use documentation generators for API and internals, read the docs and GitHub wiki sites. User support is managed and performed via GitHub tickets.
14. Enforcing a taxonomy for labeling all the tickets and pull requests (mainly for the bug fixing), allowing for a statistical understanding of our development process and user support effort. For example, by mining our labels we can understand ticket distribution across core developers, third-party developers and users per year and software system; ticket response and lifetime; the correlation between bugs and portion of the code or specific topics; the correlation between topics feature requests or documentation request.
15. The uptake of containerization approaches such as Docker or Singularity allows for providing the full environment for a computational method addressing the portability between different hardware architecture. However, the performance difference between containerized program and the same program running natively without containerization must be carefully benchmarked. Occasionally, containerization

solutions such as Docker could lead to significant slower running speed (~20%) due to, for example, unintentional kernel calls. While there is not always an observable performance differences caused by containerization, it is still important to test the program speed if a containerization solution is chosen.

16. Containerization addresses reproducibility of science and preservation of software environments. The long-term aspect of preservation of software for over 15-20 years is probably not well addressed via containerization - immanent in dependencies on container versions, operating systems, and existing research infrastructures - but delivers an intermediate solution.
17. When it comes to training and working with users, we take what we have, and we tailor it accordingly. There are some folks who will need to know every detail regarding the network performance, the processor architecture, all the compiler flags, how to make sure that all the correct libraries are linked in. This gets into the training challenges and how we can tailor things. You cannot make it perfect for everybody. So sometimes you are not going deep enough, sometimes you are going too deep. But if we could do a little bit of triage, that makes sure that our less advanced users are brought up to speed so that they can benefit from the training.

Testing Phase

1. The testing phase needs to be thought of in advance and part of the planning phase.
2. Develop benchmark dataset and regression test cases that can systematically and sensitively test the performance and potential bugs of the software.

Deployment phase

1. Define a common reporting format to facilitate communication among tools. Similarly, tools should expose various APIs so that developers and other tools can freely leverage tools functionality and query their internal status.
2. The cloud-based access to tools makes it extremely convenient for users to quickly get results by using only a browser. However, there are still many open challenges, including the cost of running cloud-based services, the security of the servers, the privacy of clients, and the complexity of the application building/execution process.
3. Work closely with infrastructure support teams, scientific support teams, system administrators and development communities of the third-party software tools we use in our middleware. We develop trust and long-term relationships by contributing testing code for the issues we find, replicators and extensive testing documentation.

4. Further, we adopt a strictly user-driven support policy, deprioritizing every consideration that is not directly related to enabling users to perform science on their target resources. We engage with third-party tools developers by sharing our use cases and describing in detail our implementation approach. Also, in this case, we contribute testing effort and, when possible, contribute code adhering to open source community best practices.
5. Strive to shorten the time between the point a user identifies issues and how long it takes to fix them.
6. Try to create one-page User Guides for some of the users. Not everybody can get away with the one pager but sometimes if you distill things down to the key items, such as “here is how you log in, these are the file systems, here is your home, your scratch folder in your project's storage, this is how you compile or this is how you load a module, etc.” that could be everything they need.
7. Part of our best practice is to transfer the knowledge whenever possible
8. After each deployment phase, go back and figure out how it could have been improved, what could be done better, how the development of tasks and deployment of features could have been better, more efficient, more reliable, etc.

Maintenance Phase

1. Use Kanban boards to facilitate communications between groups of collaborators between students, researchers and computational experts and to maintain all your projects without getting into all sorts of access control issues.

SECTION IV- Workshop Evaluation

Highlights

Participants liked:

- The Topics presented,
- The Interactions made,
- The engaging format of the Workshop (small groups of people)
- The diversity of the Participants (different experts of diverse backgrounds)
- The encouraging, and open format of the discussions

Participants Disliked:

- The shortness of the workshop
- How participants were switching the tables. Some suggested a rotation schedule for the participants is required so that the number of people on each table is approximately the same.

Most participants were asking for more workshops like this.

REFERENCES

1. A Quick Reference Guide for Hosting World Café : http://www.theworldcafe.com/wp-content/uploads/2015/07/Cafe-To-Go-Revised.pdf
2. https://www.lsst.org
3. https://en.wikipedia.org/wiki/Phylogenetic_tree
4. https://www.phylo.org/
5. https://www.atlassian.com/agile/kanban/boards
6. https://www.xsede.org/for-users/ecss
7. https://www.atlassian.com/software/jira?&aceid=&adposition=1t1&adgroup=56999361780&campaign=1439934479&creative=338489531915&device=c&keyword=jira&matchtype=e&network=g&placement=&ds_kids=p34164036185&ds_e=GOOGLE&ds_eid=700000001558501&ds_e1=GOOGLE&gclid=Cj0KCQjwvdXpBRCoARIsAMJSKqKoiMPlPUmCh-EI3CouazdgVO1ycyzinN3yFYWckVkW8h6L2TekEYoaAjSsEALw_wcB&gclsrc=aw.ds
8. https://software-carpentry.org/blog/2015/12/resbaz-feb2016.html
9. https://en.wikipedia.org/wiki/Secure_copy
10. https://www.turing.ac.uk/research/research-projects/data-science-tools-high-performance-computing
11. https://www.xsede.org/community-engagement/campus-champions
12. http://urssi.us/
13. https://aciref.org/
14. http://www.oscer.ou.edu/virtualresidency2019.php
15. https://sciencegateways.org/engage/webinars
16. https://github.com
17. https://www.rd-alliance.org/
18. https://us.rd-alliance.org/
19. https://software-carpentry.org/
20. https://spack.io/
21. https://e4s.io