SAND2019-5829C

EXASCALE COMPUTING PROJECT

IDEAS productivity

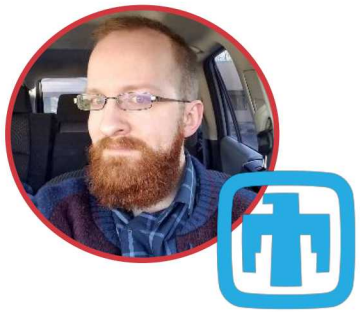
better scientific software

So your code will see the future.

Position Paper: Towards Usability as a First-Class Quality of HPC Scientific Software

Reed Milewicz
Sandia National Laboratories
rmilewi@sandia.gov

Paige Rodeghero
Clemson University
prodegh@clemson.edu



@rmmilewi

May 28th, 2019

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.

Overview of This Talk

Position Paper: Towards Usability as a First-Class Quality of HPC Scientific Software

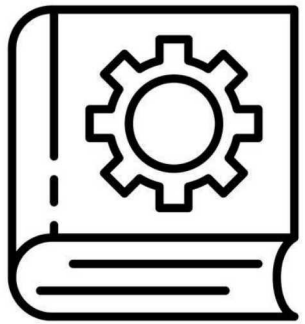
- The goal of a position paper is to convince you that the opinion presented is valid and worth listening to.
- **The Position**: Usability must become a first class quality of HPC scientific software.

@rmmilewi

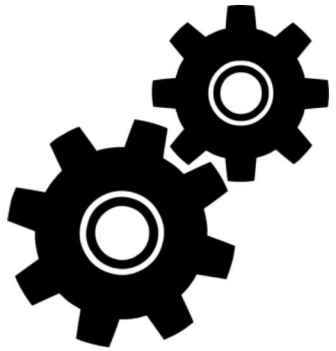
What is Usability?

Usability: The capability of the software product to be understood, learned, used and attractive to the use, when used under specified conditions.

ISO 9126:2001



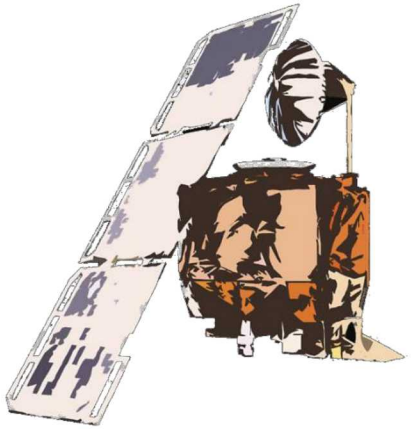
Learnability



Efficiency

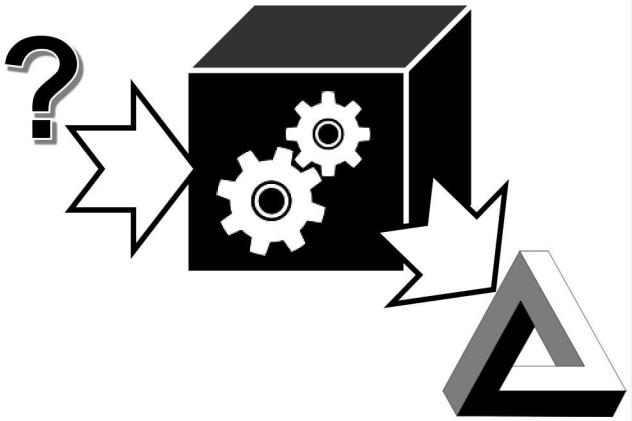
Memorability

Many of Us Have Been Talking About Usability



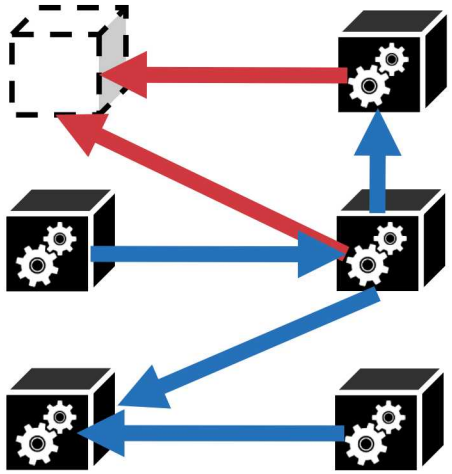
(e.g. Smith; Danish)

What is needed to understand code can be missing, inconsistent, or scattered.



(e.g. Kempf/Bastian)

Libraries can be powerful, correct, performant! But they can also have horrendous learning curves.



(e.g. Bartlett)

The ecosystem is not yet mature, and the ground shifts under users' feet.

(e.g. Badreddin)

The incentive structures aren't well-aligned with the need to invest resources to make code usable.

@rmmilewi

UUR

“The specialized domain knowledge embodied in the software codes and algorithms become inaccessible due to **the growing arbitrary complexities**.”

“they are not often used to develop new “widgets” because they are **hard to understand**.”

“**[we want] software that is quicker to write, easier to understand**, and containing fewer bugs than ever before.”

“However, these frameworks have been **struggling with the learning curve experience** of new users […]”

But How Should We Frame This?

@rmmilewi

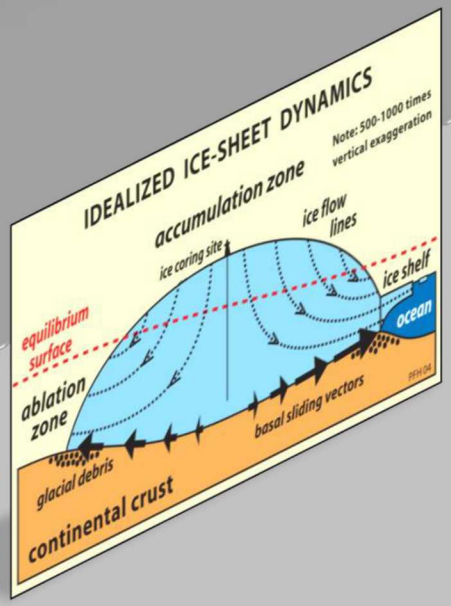
UUR

```
PHX::MDField<const
ScalarT,Cell,Side,Node>
beta; //[kPa yr/m]
```

A glaciologist looking to run a melt simulation installs Albany. She reads the above line of code from `LandIce_BasalMeltRate.hpp`.

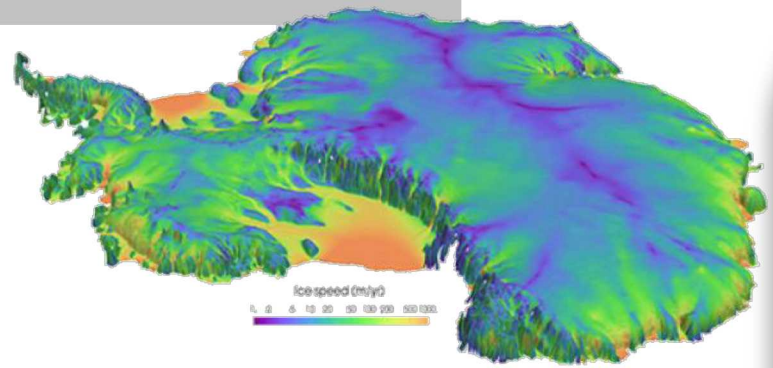
But what is she **actually** reading?

```
PHX::MDField<const
ScalarT,Cell,Side,Node>
beta; //[kPa yr/m]
```



@rmmilewi

- Loop or output visualization
- Simulate basal sliding
- Effects of subglacial hydrology
- Calculate the gravitational force on the ice.



UR

Phalanx

LandIce

Albany

TRILINOS

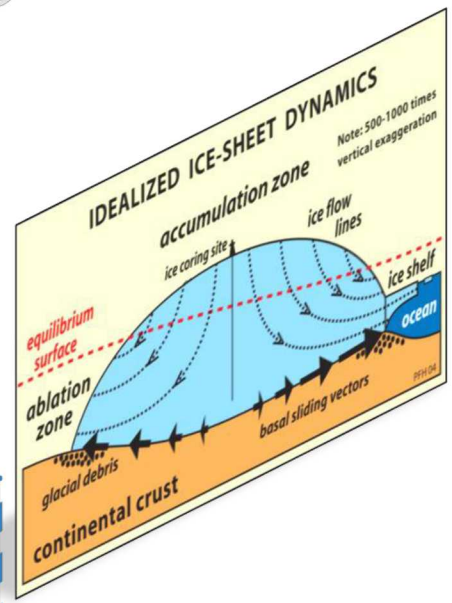
```
PHX::MDField<const
ScalarT,Cell,Side,Node>
beta; //[kPa yr/m]
```


```
PHX::MDField<const
ScalarT,Cell,Side,Node>
beta; //[kPa yr/m]
```

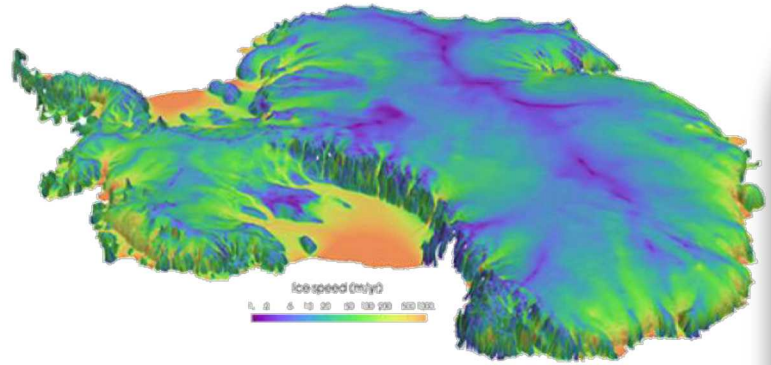
```
PHX::MDField<const
ScalarT,Cell,Side,Node>
beta; //[kPa yr/m]
```


¿ Cell ?

- ¿Triangle ?
- ¿Quadrilateral?
- ¿Tetrahedron?

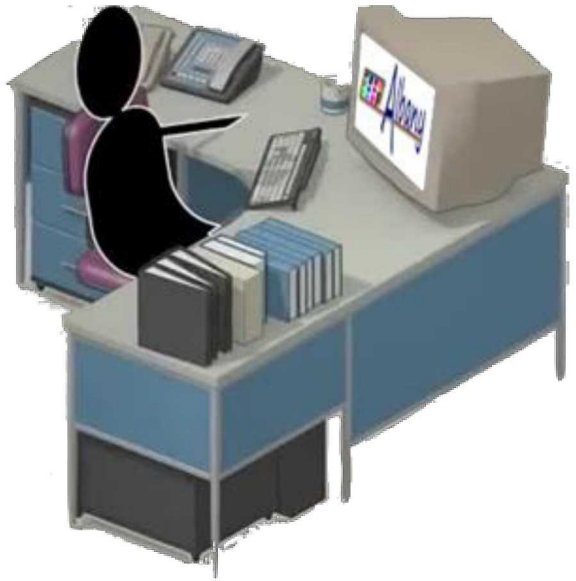


- Loop or output visualization
- Simulate basal sliding
- Effects of subglacial hydrology
- Calculate the gravitational force on the ice.



UR

A Two-Way Process



Code, APIs, GUIs,
documentation,
tutorials, user support

Needs to identify, use, debug,
or extend the software

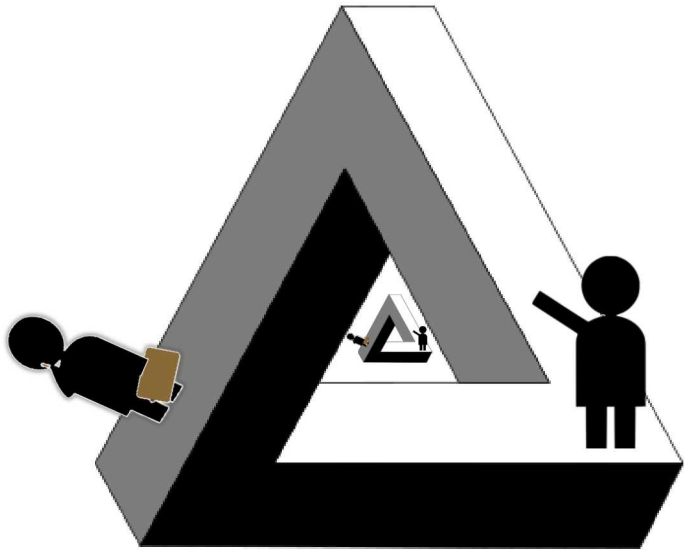
Scientific software made without the users in mind can lead to situations where the software is “not perceived as being usable and so [is] not used.”
(Segal 2011)

But...

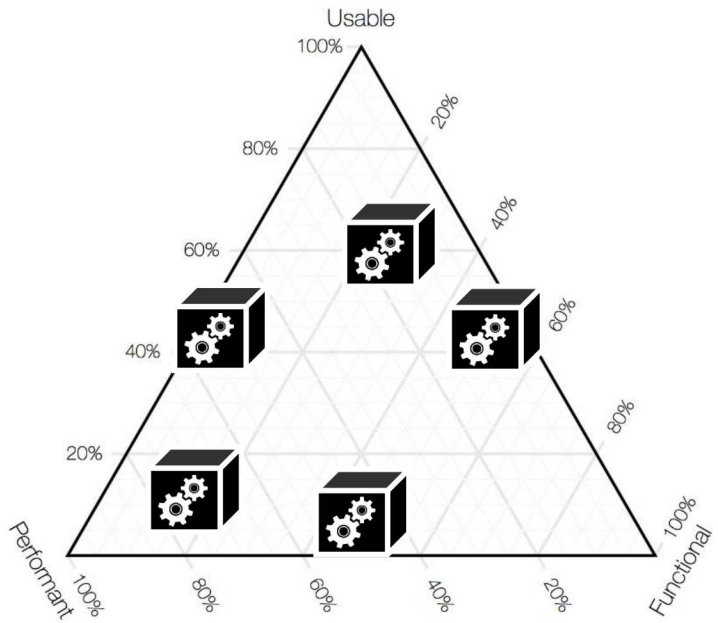
Usability is **not** an absolute good.

@rmmilewi

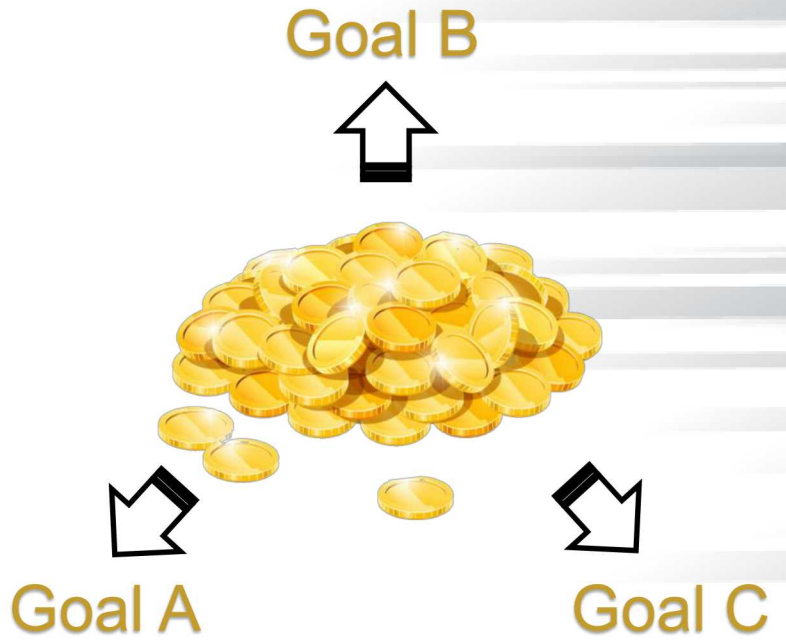
Usability is Not An Absolute Good



Sometimes scientific codes are necessarily **complicated** because the problems they solve are **also complicated**



A more usable code may also be **less** capable, **less** performant, or **less** portable.



Deciding to improve a non-functional quality requires an investment of **time and money**, and those resources may be spent more effectively elsewhere.

@rmmilewi

UUR

Asking Questions

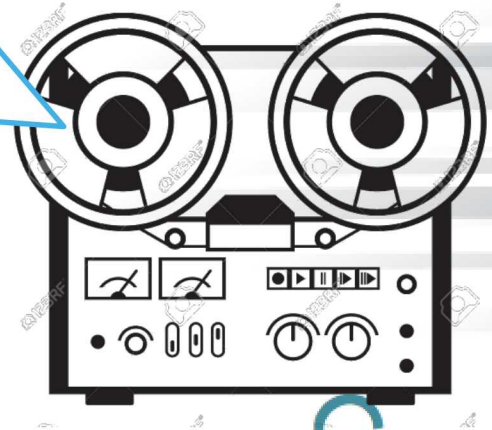
1. How does a developer know that she has made a usable product?
2. When is a change to software of consequence to usability?
3. Perhaps most importantly, how does she find funding for usability work unless she has well-defined methods, clear goals, and measurable outcomes?

Our Claim

- “First-class” → Implementable, measurable, and incorporated into the state of practice.
- We don’t need to reinvent the wheel. Usability engineering is a mature field in conventional industry.
- Translating U(x)E to the CSE domain presents many open research questions that must be addressed.
- Examples include…

User Testing: Evaluators observe participants interacting with software in order to identify usability issues.

Partcipant: So, I need to call the simulation function, but I don't understand the calling syntax. I'm going to pull up the Doxygen entry…



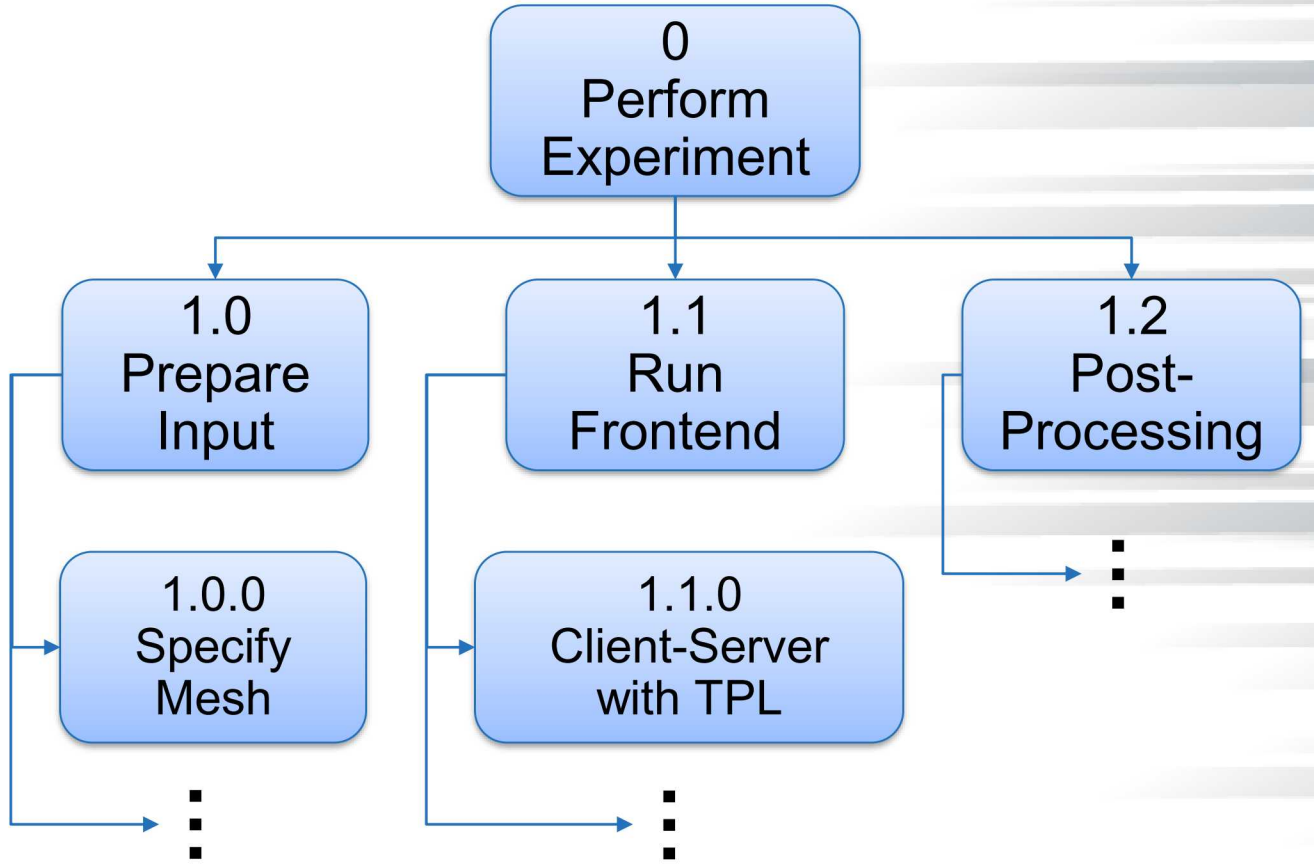
Coaching
Shadowing
Co-discovery
Question-Asking
Retrospective_Testing
Log_File_Analysis
Remote_Testing
Think-Aloud
Teaching

@rmmilewi

While user testing is a tried-and-true method, it is also an expensive and time-consuming process, and it isn’t yet clear about how user testing fits into the scientific software workflow.

How and when should user testing be employed, and under what conditions will user testing be most effective?

Task-Environment Analysis: Evaluators create an analytical model that maps users' goals to interactions with the software system.



1.0.0 The user can introduce errors in the input file.

Our input parser displays informative error messages.

But…

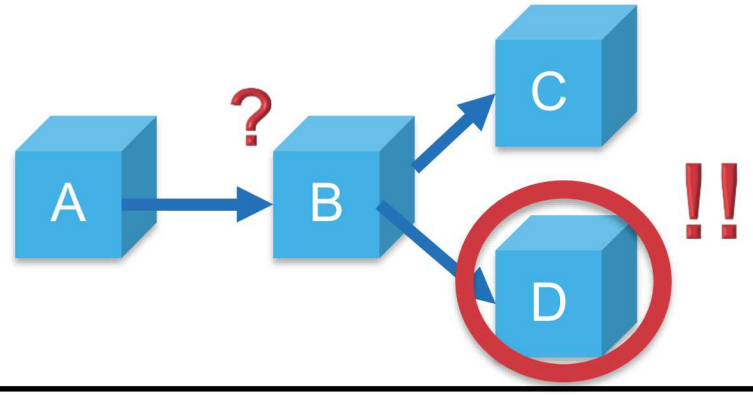
1.1.0 The workflow requires communication with a TPL.

We provide tutorials for how to handle the communications between the libraries.

UUR

Task analysis can provide a cross-cutting perspective that informs many other activities, such as keeping existing use cases well-supported and identifying emerging use cases.

How in-depth should these analyses be? What is the “shelf life” of an analytical task model?



Cognitive Walkthroughs: Evaluators take on the role of a user and stepping through interactions with the product.

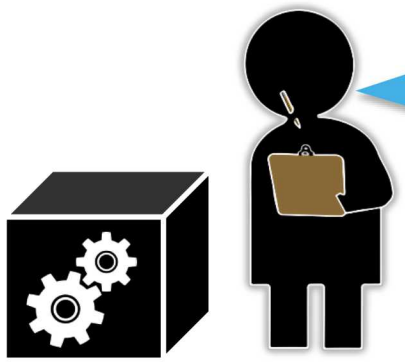
Would **Steve** be familiar with the way the API is set up?

Is the output in a format that would be easy for **Steve** to share with his collaborators?

@rmmilewi

CWs can be applied at any stage of the life cycle and do not demand extensive user participation, but they do require accurate and informative user models.

How do we take what we know about scientific software developers and turn it into an actionable model? Are these models generalizable?



To be fair, we probably could make those status updates available on the panel…

Heuristic Evaluation: Evaluators apply a checklist of statements about the desirable properties of the system.

Feedback: A Grid Computing application should keep users informed on the jobs' progress, indicating both the global and the detailed state of the system. The application should deliver appropriate feedback on users' actions.

- Example 1…
- Example 2…
- Example 3…

Comments…

@rmmilewi

Heuristic evaluation strategies allow developers to obtain feedback early in the design process. Applying the correct heuristic can help suggest the best corrective measures to designers.

Checklists are low-hanging fruit in that they are easy to apply, but the contents have yet to be determined.

Towards Usability as a First-Class Quality of HPC Scientific Software

- The value of scientific software is intimately tied to its usability, the ability to pick it up and put it to work answering a scientific question.
- This is an area ripe for exploration!
- If this interests you, check out the TL;DR card!

@rmmilewi

TL;DR: Towards Usability as a First-Class Quality of HPC Scientific Software

Our Position: Usability must become a first class quality of HPC scientific software.

Contact

Reed Milewicz
rmilewi@sandia.gov

Paige Rodeghero
prodegh@clemson.edu

Questions Include…

- How and when should **user testing** be employed, and under what conditions will user testing be most effective?
- How in-depth should a **task analysis** be? What is the "shelf life" of an analytical task model?
- How do we take what we know about scientific software developers and turn it into **personas**? Are they generalizable?
- What would a **heuristic evaluation checklist** for scientific HPC software look like?

@rmmilewi

(paper; slides)
rmmilewi.github.io

Citations

- Contrastin, Mistral and Rice, Andrew and Danish, Matthew and Orchard, Dominic A. (2015) Units-of-Measure Correctness in Fortran Programs. (1). pp. 102-107. ISSN 1521-9615.
- Szymczak, Dan, Spencer Smith, and Jacques Carette. "Position paper: A knowledge-based approach to scientific software development." *2016 IEEE/ACM International Workshop on Software Engineering for Science (SE4Science)*. IEEE, 2016.
- Badreddin, Omar. "Powering software sustainability with blockchain." *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. IBM Corp., 2018.
- Kempf, Dominic, et al. "Automatic Code Generation for High-Performance Discontinuous Galerkin Methods on Modern Architectures." *arXiv preprint arXiv:1812.08075* (2018).
- Bartlett, Roscoe A., and M. D. Rockville. "A Roadmap for Sustainable Ecosystems of CSE Software." (2015).
- Segal, Judith, and Chris Morris. "Scientific end-user developers and barriers to user/customer engagement." *Journal of Organizational and End User Computing (JOEUC)*23.4 (2011): 51-63.
- Das, Deya, et al. "Effect of Crystallinity on Li Adsorption in Polyethylene Oxide." *Chemistry of Materials* 30.24 (2018): 8804-8810.