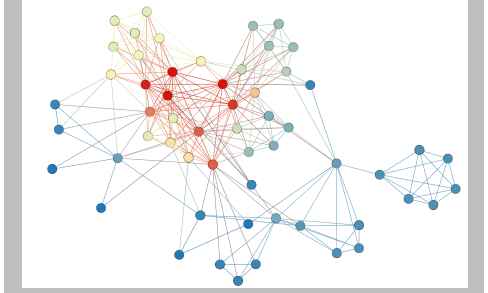
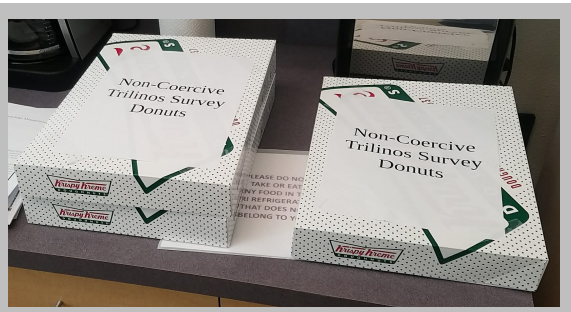
Exceptional service in the national interest

Sandia National Laboratories





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

Reed Milewicz and Elaine Raybourn

rmilewi@sandia.gov; emraybo@sandia.gov

SAND2018-12054 C
Unclassified Unlimited Release

TUG'18 (25 Oct 2018); WSSSPE 6.1 (29 Oct 2018)

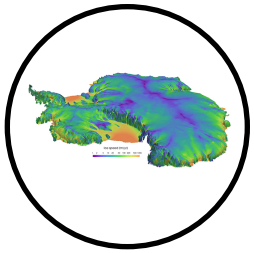
Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Overview

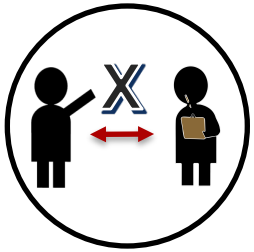
Sandia National Laboratories

- TL;DR
- Who am I?
- Defining “Large-Scale” in Scientific Software Development
- Motivating Example: The Trilinos Project
- Research Questions
- Methodology
- Conclusions; Future Work

TL;DR



Large-scale collaborative scientific software projects require more knowledge than any one person typically possesses.



As researchers scale up these projects, they are beset by problems of communication and coordination. Many software-related problems may, in large part, be organizational issues.

We present a case study on the coordination of expertise in a scientific software project, how it relates to development challenges, and explore possible solutions.

Who Am I?

- A postdoc working within the Software Engineering and Research Department at the Computer Science Research Institute at Sandia.
- My Areas of Interest:
  - Software Engineering
  - Compilers
  - Formal Verification
- My funding comes from the Interoperable Design of Extreme-scale Application Software (IDEAS) project, an arm of the Exascale Computing Project (ECP).

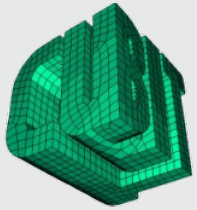
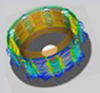
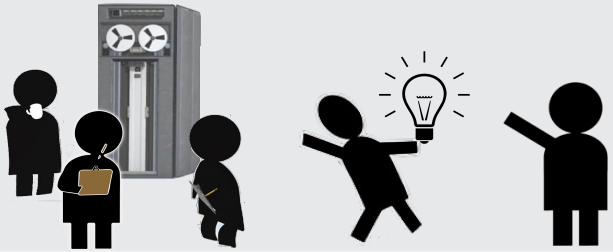
Large-Scale Scientific Software Development

Sandia National Laboratories

- Scientific software projects, such as those developed at Sandia, are among the most complex, knowledge-intensive undertakings in all of human history.
- When is software "large-scale"? The hallmark of large-scale software is that **nobody can know everything** (Moe et al. 2014).
- This threshold is reached far faster in scientific software than in most conventional software.

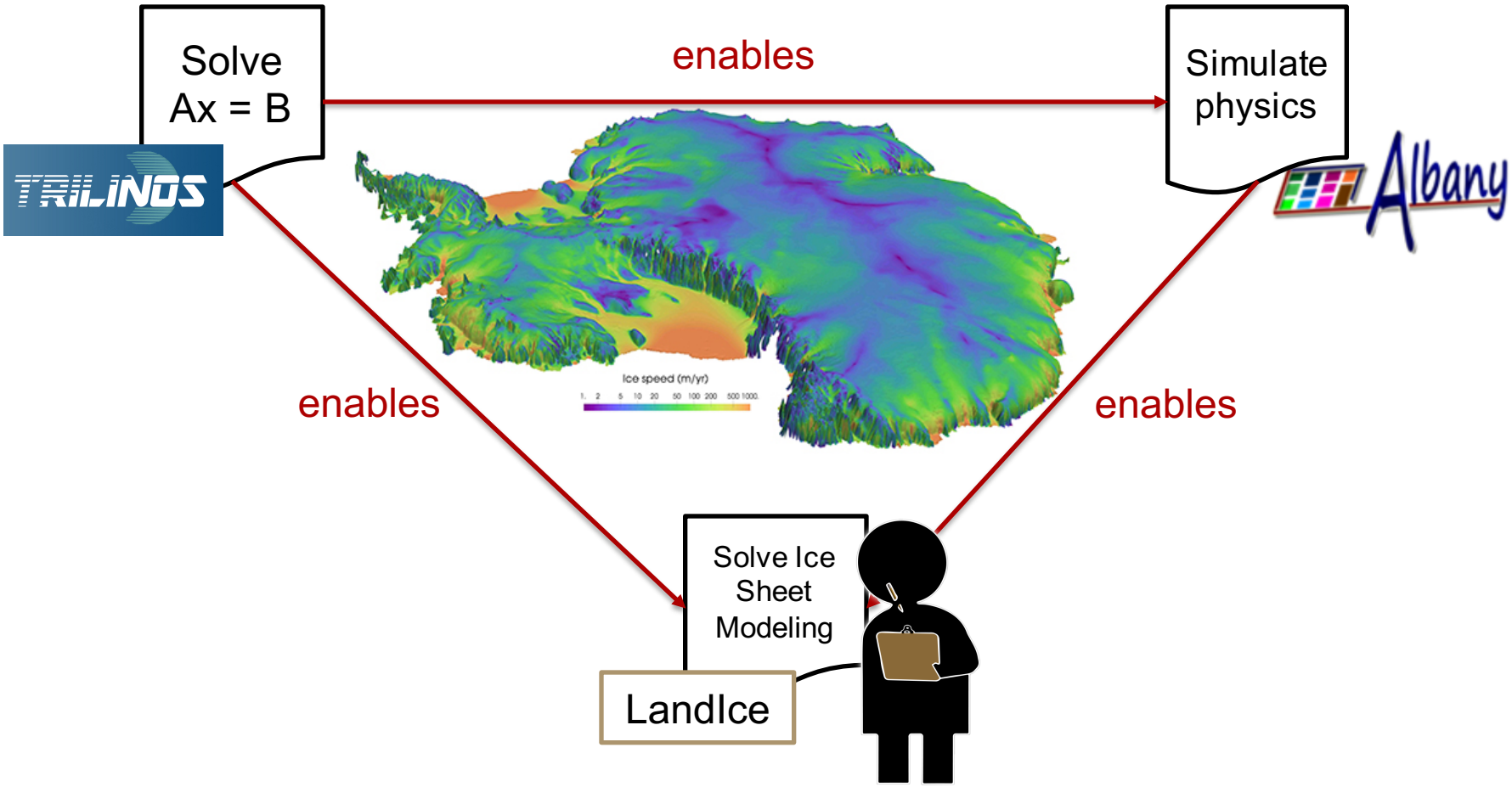
Large-Scale Scientific Software Development

Sandia National Laboratories

Stakeholder	Such as…	In short…
Funding Agencies; Supporting Institutions	NSF; Office of Science U.S. Department of Energy; National Nuclear Security Administration	We need X! (X is some unit of science).
Specialized Codes and Libraries	TRILINOS;  DAKOTA Explore and predict with confidence.	We provide foundational capabilities to solve X.
Engineering Physics Integrated Codes	SIERRA;  ALEGRA; Xyce™ Parallel Electronic Simulator	We provide simulations for X.
End-users/ Analysts (and their codes)		We solve the problem and deliver X.

Large-Scale Scientific Software Development

Sandia National Laboratories

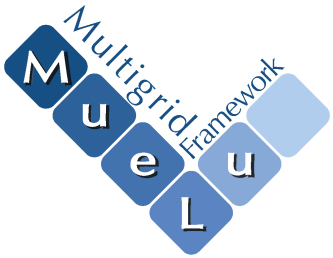


Case Study: Trilinos

Sandia National Laboratories

- Trilinos is a confederation of (~60) object-oriented software packages for building scalable scientific and engineering applications, written in C++.
- Provides foundational capabilities for many different applications both inside and outside of the labs.

Examples Include…



Multigrid solvers and preconditioners for sparse linear systems

Optimization algorithms for use in large-scale engineering applications

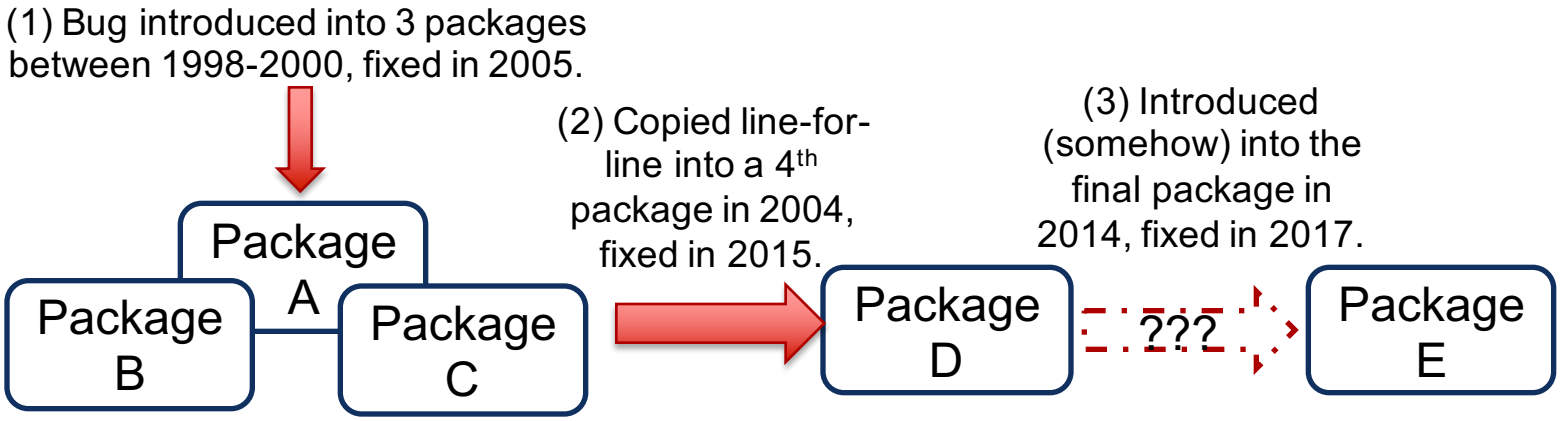
Motivation

Sandia National Laboratories

- **Problem**: A critical application using Trilinos failed a major acceptance test to an unexpected explosion in memory usage when running with more than 2^{17} MPI progresses.
- A team of researchers was given several months to locate the bug to no avail, but the issue was finally resolved when one heroic Trilinos scientist-developer volunteered three weeks of his time to uncover it.
- The cause of the bug?
  - A misuse of an MPI function due to a misunderstanding about its semantics.
  - An inefficient vendor implementation of that MPI function

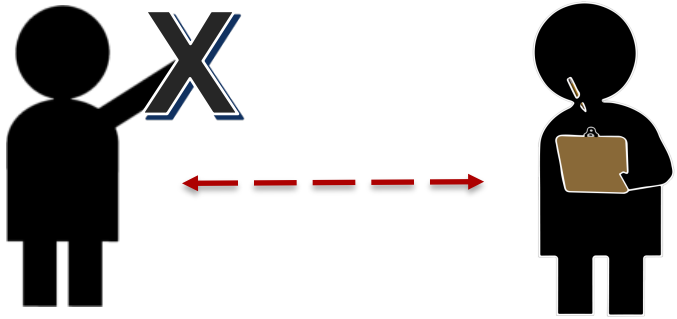
Motivation

- I conducted my own investigation by interviewing different parties and analyzing source code histories.
- **The deeper mystery**: The *exact same bug* had been fixed twice before!



- In each case, the discovery and solutions were socialized, notes were made, etc. However, the information did not flow to the right parties in each subsequent incident.

Motivation



Knows X

Needs to know X

Sandia National Laboratories

Knowledge: A fluid mix of framed experience, values, contextual information, and expert insights that provides a framework for evaluating and incorporating new experiences and information. It originates in and is applied in the minds of knowers. In organizations, it often becomes embedded not only in documents or repositories but also in organizational routines, processes, practices, and norms.

Davenport and Prusak 1998

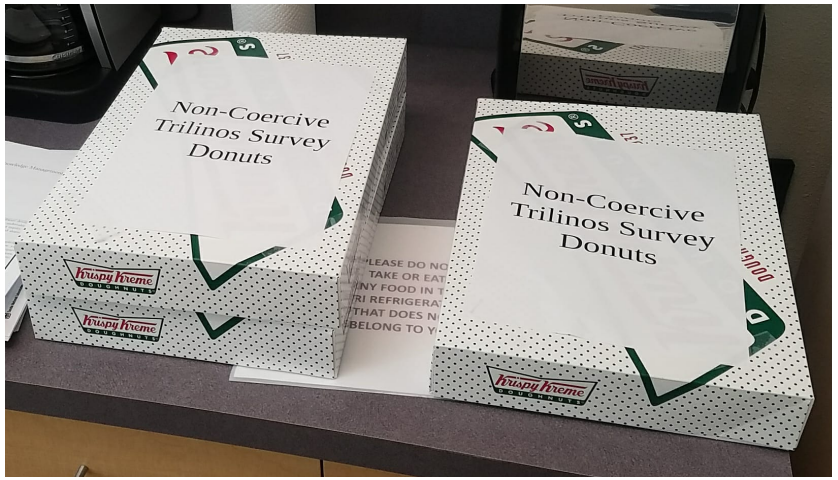
- It can be tacit or explicit.
- It can be socialized, combined, internalized, or externalized.
- It can include "knowing what", "knowing who", "knowing why", etc.
- It can be lost or unable to be communicated.

Research Questions

- **RQ1**: Do scientific software developers face challenges in sharing their knowledge? If so, what are the challenges?
- **RQ2**: How does individual and organizational knowledge affect those problems?
- **RQ3**: How is that knowledge communicated?

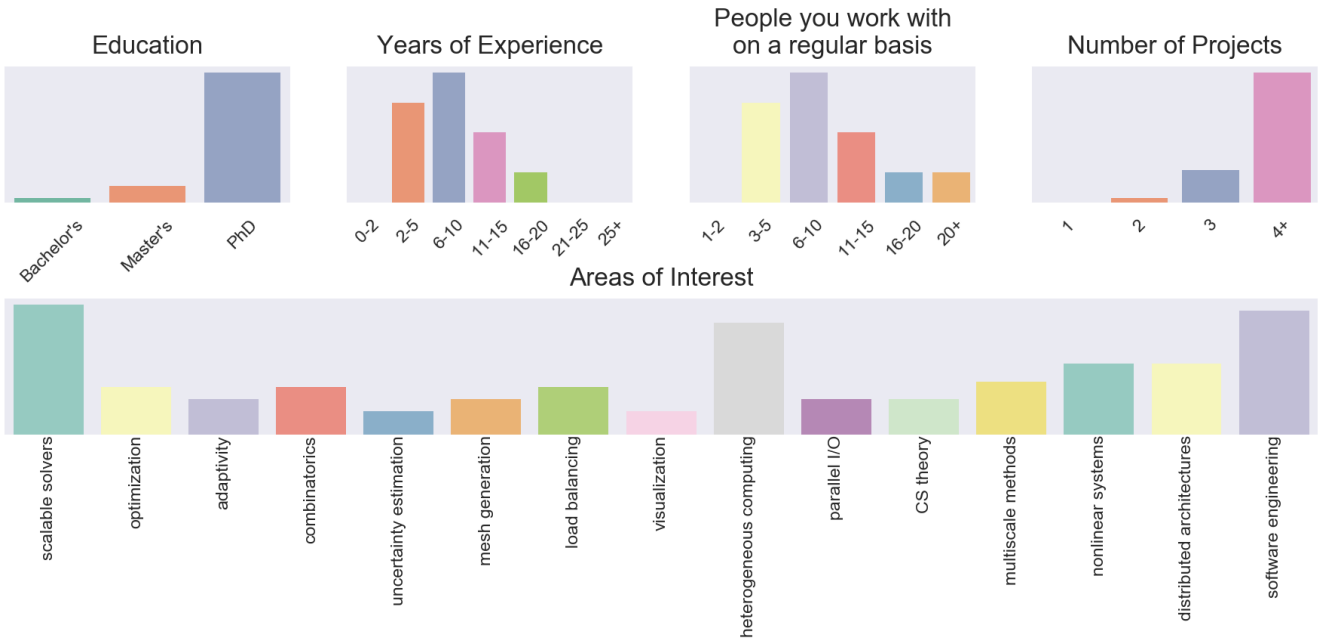
Conducting the Survey

- Survey data was collected over a period of 3 months.
- 36 developers responded, covering 95% of the "main" development group.
- Survey topics:
  - Background and demographic information
  - Career priorities
  - What and who people know
  - How they communicate
  - What problems they face



Demographics

- 86% of respondents have completed a PhD. The median respondent had between 11 and 15 years of experience.
- 73% work on 4 or more projects. Most people work regularly with 6 to 10 other people.



The 19 Problems (categories and examples)

Sandia National Laboratories

- **Code Understanding** (4 problems)
  - Understanding code that someone else wrote (83.3% agree).
- **Task Switching** (3 problems)
  - Having to divide my attention between many different projects (94.4% agree).
- **Modularity** (2 problems)
  - Understanding the impact of changes that I make on code elsewhere (61.1% agree).
- **Links Between Artifacts** (5 problems)
  - Finding code related to a bug (83.3% agree).
- **Team** (2 problems)
  - Convincing developers to make changes to code that I depend upon (61.1% agree).
- **Expertise Problems** (3 problems)
  - Finding the right person to talk about a piece of code (50.0% agree).

What We Know About These Problems

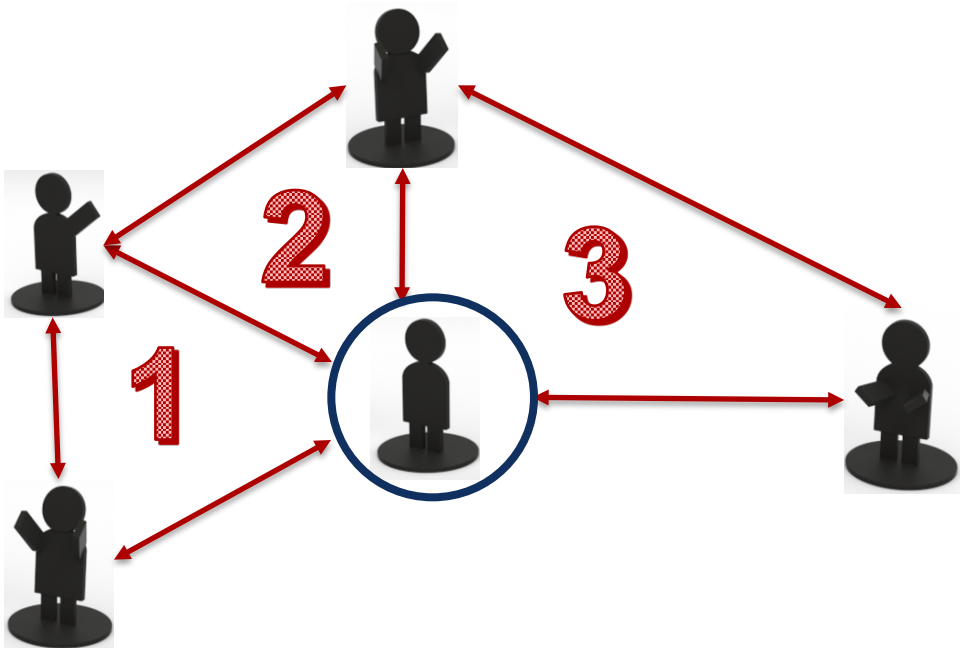
Sandia National Laboratories

- The majority of respondents reported having 13 out of 19 problems. The median respondent reported having 11 problems, 2 of which were considered serious.
- Problems within each category don't tend to correlate well with each other (as measured by Cronbach's alpha).

Category	Cronbach's alpha
Code Understanding	0.770
Task Switching	0.715
Modularity	0.474
Artifacts	0.595
Team	0.594
Expertise Finding	0.579

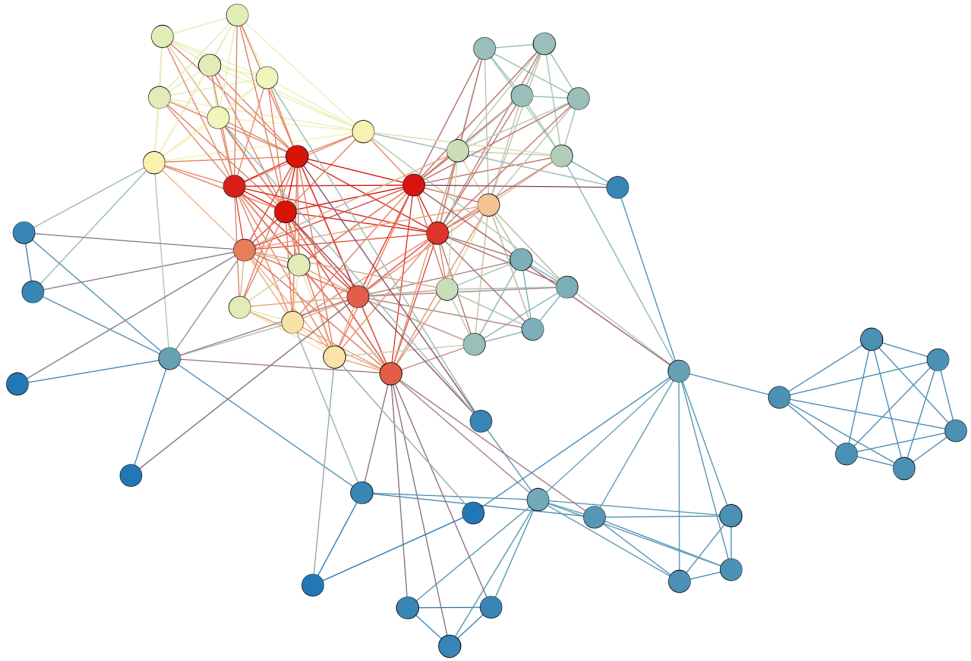
- The data suggests that these problems have multiple independent, latent causes.

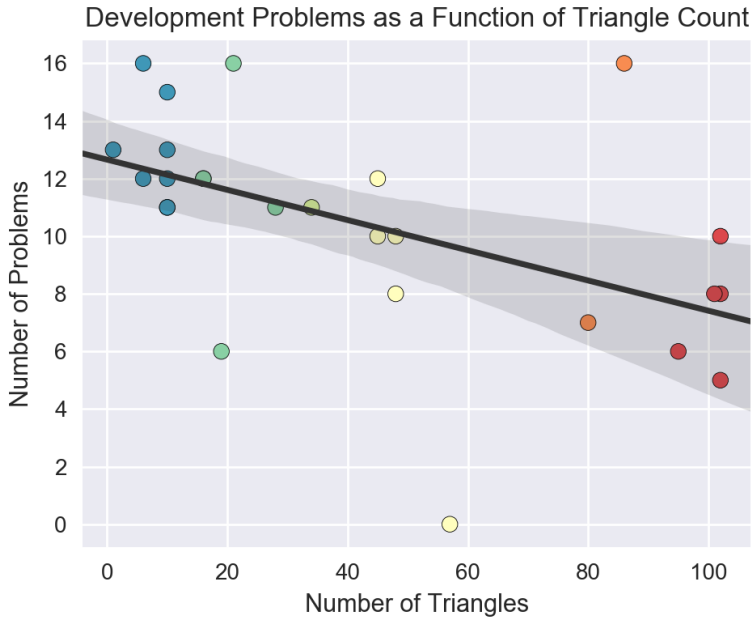
Let's try a naïve network analysis



- Triangle counting: For each vertex, how many cycles of length three can we find that include that vertex?
- For this, we examined Github project team membership.

Team Network vs. Problems



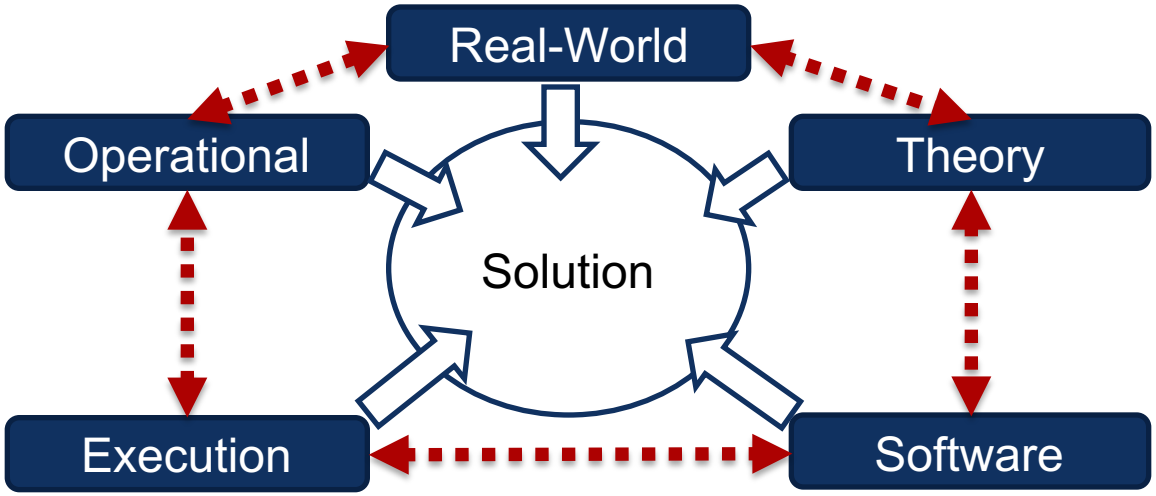


People on the organizational periphery tend to report more problems than people closer to the core. But why?

Is there a simple explanation?

- *Is this a matter of experience?*
  - **No**, only 3/19 problems can be correlated with experience.
- *Is this a matter of the number of people that respondents work with or the number of projects they work on?*
  - **No**, people don't tend to report more or fewer problems based on the number of contacts or projects they have. 2/19 problems can be correlated with the number of projects people have.
- This is not surprising: studies of R&D organizations have often drawn attention to the value of network centrality in amplifying an individual's impact and increasing their access to knowledge.

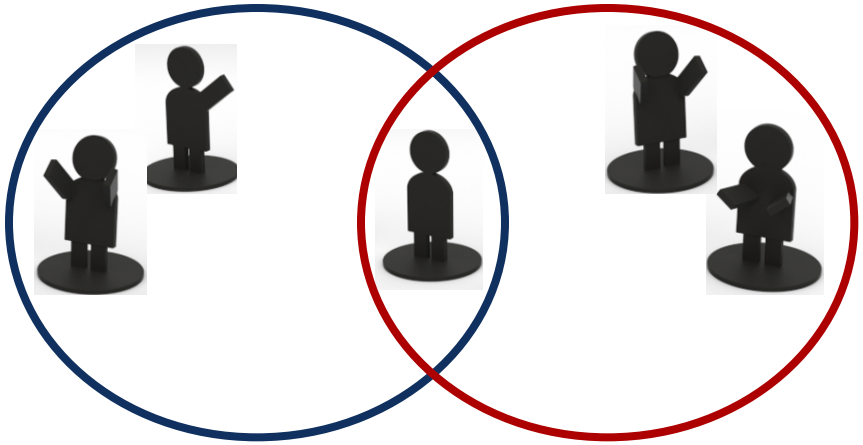
What/who do they know?



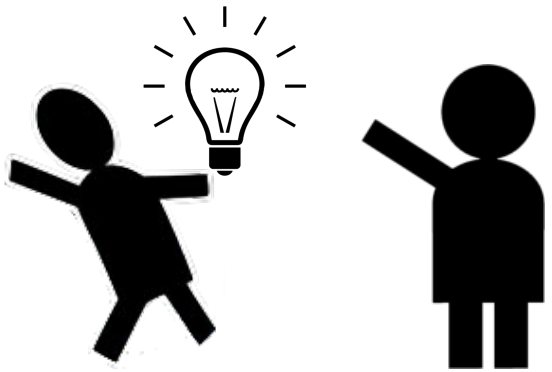
- We used a five factor knowledge model of scientific software development (Kelly 2015). From that, we created a list of topics and had respondents rank their expertise in those areas.
- For each topic, we also asked respondents to indicate whether they knew someone they could "turn to for help" for it.

D. Kelly, "Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software," *Journal of Systems and Software*, vol. 109, pp. 50–61, 2015.

What/who do they know?



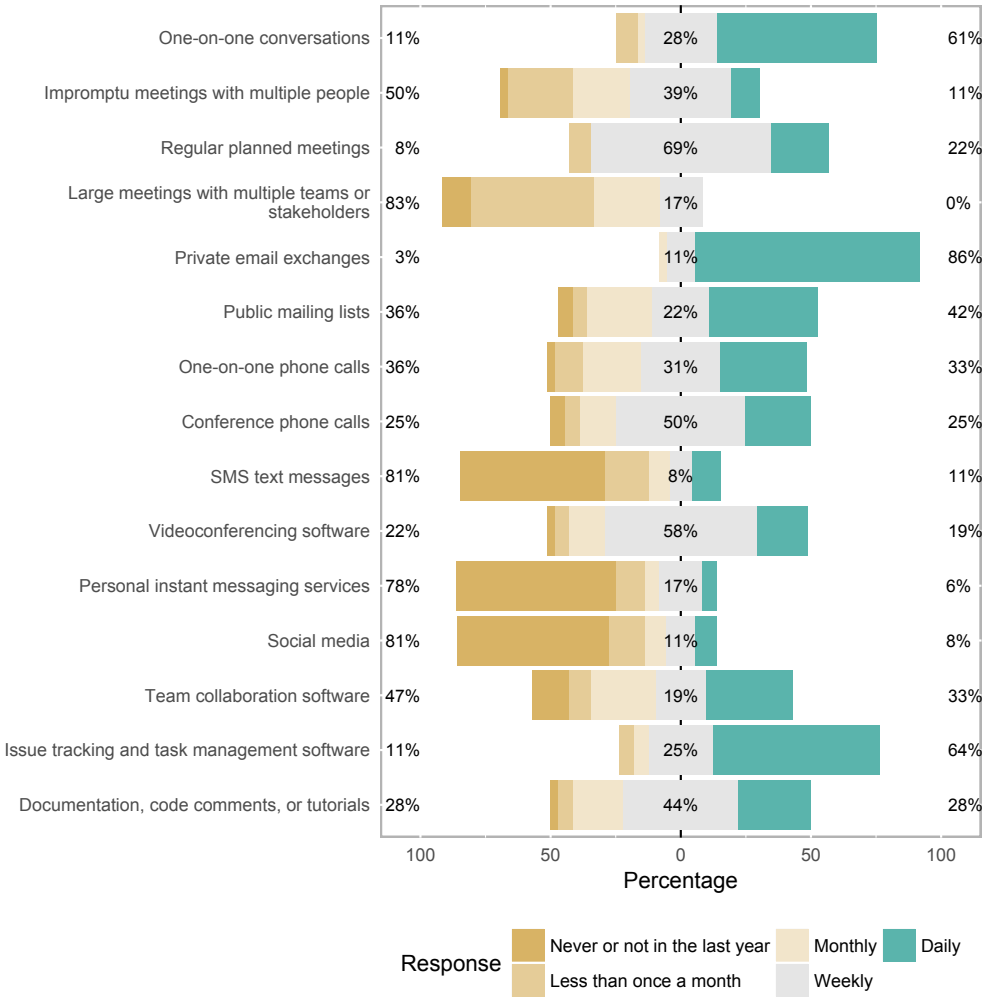
The most useful forms of expertise are those that allow respondents to position themselves between domains of activity.



"Knowing who" is instrumental for maintaining awareness as well as negotiating and coordinating with others.

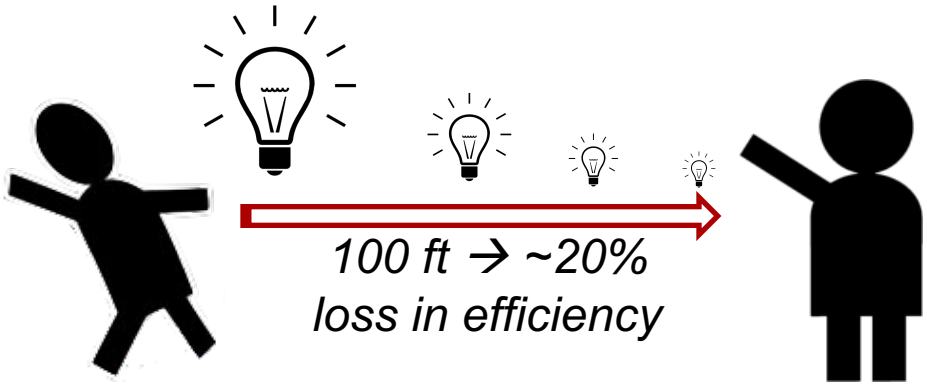
- Operational and execution domain knowledge are moderately correlated with four of the nineteen problems.
- Real-world, theory, and software domain knowledge provided **no** measurable benefit.
- Ten of the nineteen problems are influenced by seeking help from others.

How do they communicate?



- Knowledge scores appear to mirror communication scores; those who communicate more, know more.
- Face-to-face communications enable expertise-finding activities, affecting three of the nineteen problems.
- Digital communication strategies are useful for protecting modularity and understanding the links between artifacts, but the communication overhead also introduces new challenges (e.g. divided attention).

How do they communicate?



Face-to-face communication is important for collaborative problem solving. It's more than just knowing who to talk to, it's about cultivating close relationships with those people.

In fact, there was a recent study of an R&D organization that found a drop-off in collaboration frequency and success after about 100 feet of distance between offices (Kabo et al 2014) .

F. W. Kabo, N. Cotton-Nessler, Y. Hwang, M. C. Levenstein, and J. Owen-Smith, "Proximity effects on the dynamics and outcomes of scientific collaborations," *Research Policy*, vol. 43, no. 9, pp. 1469– 1485, 2014.

So, what can we say about the 19 problems?

Sandia National Laboratories

- They are not cured by time or experience.
- We found no proof to suggest that additional domain or software development training will make them go away.
- However, they are, in some sense, a function of a person's "embeddedness" in the team. More specifically, we found several factors that appear to influence the occurrence of problems:
  - Knowing what
  - Knowing who
  - Communication strategies

Problem Area	Backgrounc	What They Know	Knowing Who Knows	How They Commu-nicate
Code Understanding	(2/4)	(1/4)	(3/4)	
Switching Tasks		(2/3)	(3/3)	(1/3)
Modularity	(1/2)		(1/2)	(1/2)
Links Between Artifacts	(2/5)	(3/5)	(2/5)	(2/5)
Team			(1/2)	(1/2)
Expertise Finding	(1/3)	(1/3)		(3/3)

Recommendations

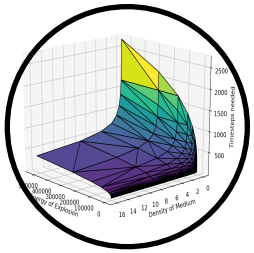
Sandia National Laboratories

- What might help?
  - Empowering knowledge brokers
    - 33% of our respondents knew no one they could turn to for help in any of the knowledge areas while having an average of 5 different problems that could potentially be mitigated by having useful contacts
    - Giving formal recognition and power to the people who know people
  - Cultivating organizational awareness
    - The value of serendipitous encounters
    - Interdepartmental seminars and luncheons
  - Encouraging integrative work (more expensive)
    - 36% of respondents reported having no daily face-to-face interactions with other coworkers
    - Occasional (temporary) team rotations
    - Pair programming for production code

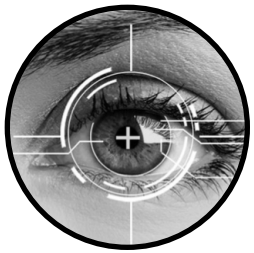
Ongoing/Future Work

Collected 27 hours of testimony from scientific software developers. Using this data to develop plans of action. Ongoing work between researchers at Sandia, Clemson, Para, and Pernamuco.

Now that 1400 has an official software engineering R&D department, we've talked about putting research into action more quickly to support teams (e.g. tools/frameworks).



Working with another researcher at LANL to develop better tools for performance modeling and analysis to aid decision support.



Working with researchers at Clemson University to study teams that use Trilinos and other libraries, targeting issues in code readability and usability.

Sandia National Laboratories

Questions?

Want to use our protocol in your own research? Here's a link:

https://github.com/rmmilewi/KnowledgeManagementSurvey