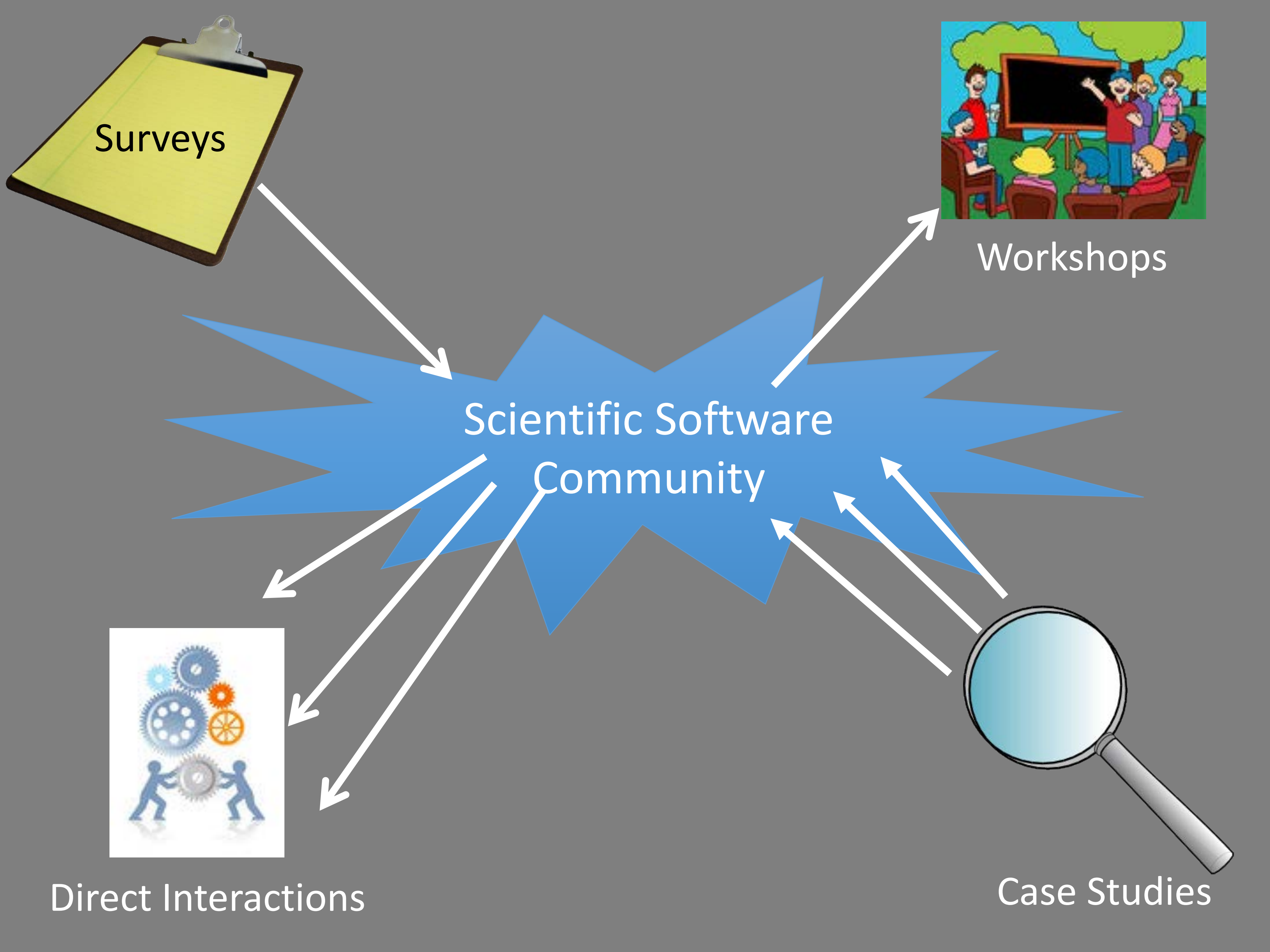
What We Have Learned About Using Software Engineering Practices in Scientific Software

Jeffrey Carver
University of Alabama
carver@cs.ua.edu

November 29, 2017

@SE4Science

1243887, 1445344


Surveys
Workshops
Scientific Software Community
Direct Interactions
Case Studies

Community Surveys

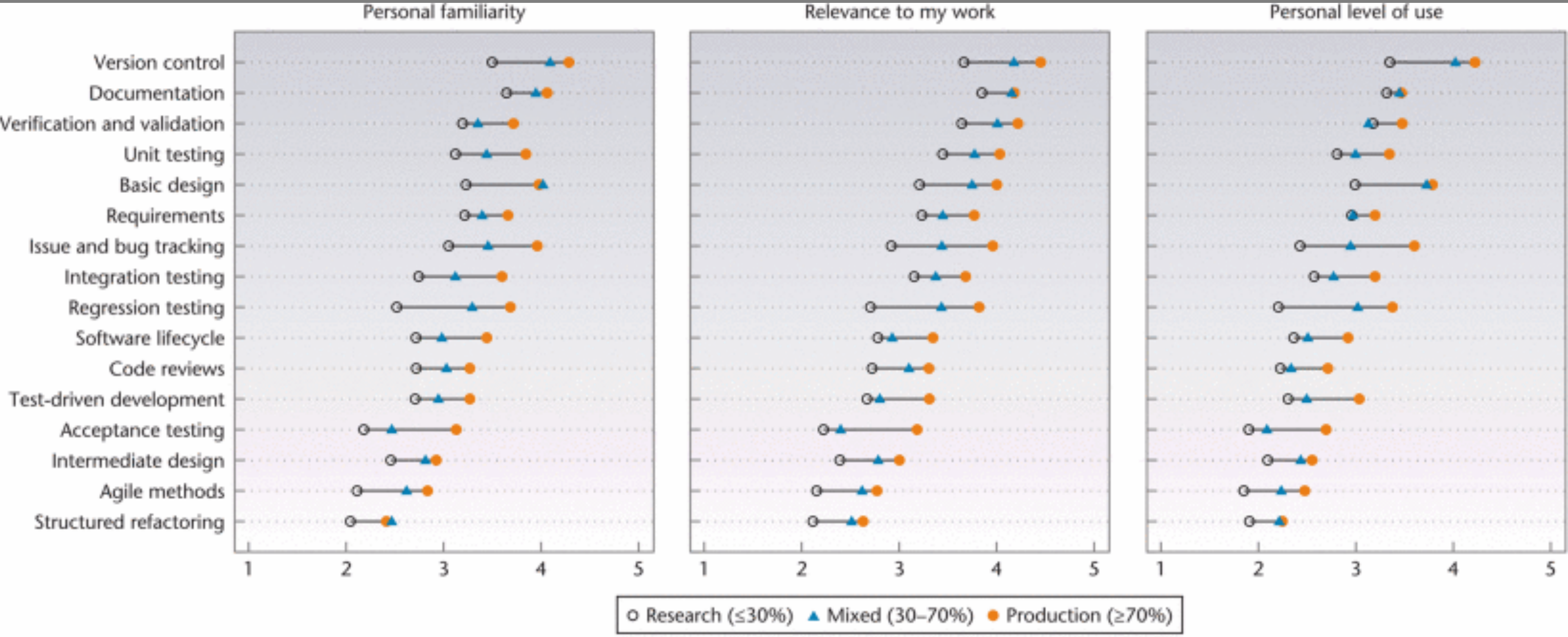
Community Surveys:
First Survey

- Sufficiency of SE Knowledge
  - Personally - 92% said yes
  - CSE community - 63% said yes
- Research vs. Production
- Reported 4 Key Problems
  - Rework
  - Performance issues
  - Regression
  - Forgetting to fix bugs not tracked

Community Surveys: Second Survey

- Broad subset of Computational Science audience – 151 responses
- Level of usage of various SE practices
- Generally agreed with our definitions of SE terminology

Community Surveys: Second Survey



Carver, J., et al. "Self-Perceptions about Software Engineering: A Survey of Scientist and Engineers." *Computing in Science & Engineering,* 15(1):7-11

Case Studies

Case Studies:
Goals

- Support scientific developers
- Gather information about effective and ineffective practices
- Understand and document software development practices
- Provide feedback to teams

Case Studies







	FALCON	HAWK	CONDOR	EAGLE	NENE	OSPREY	HARRIER
Application Domain	Prediction of Product Performance	Predication of Manufacturing Process	Analysis of Product Performance	Signal Processing	Calculate Molecule Properties	Weather Forecasting	Engineering Mesh Generation
Duration (Years)	~ 10	~ 6	~ 20	~ 3	~ 25	~10	~8
# of Releases	9 (production)	1	7	1	?	?	~16
Staffing (FTEs)	15	3	3-5	3	~10 (100's of contributors)	~10	5 primary + students
Customers	< 50	10s	100s	None	~ 100,000	100s	10s
Code Size (LOC)	~ 405,000	~ 134,000	~200,000	< 100,000	750,000	150,000	50,000
Primary Languages	F77 (24%), C (12%)	C++ (67%), C (18%)	F77 (85%)	C++, Matlab	F77 (95%)	Fortran	C++ (50%), Python (50%)
Other Languages	F90, Python, Perl, ksh/csh/sh	Python, F90	F90, C, Slang	Java Libraries	C	C	None
Target Hardware	Parallel Supercomputer	Parallel Supercomputer	PCs to Parallel Supercomputer	Embedded Hardware	PCs to Parallel Supercomputer	Parallel Supercomputer	Linux, Windows

Case Studies

Lessons Learned

Lessons Learned:
Validation and Verification

http://dilbert.com/strip/2010-11-07

Lessons Learned: Validation and Verification

- Vary in formality and completeness
  - Core algorithms vs. User Interactions
  - Percentage of code tested
  - Dedicated testers vs. End users
- Required by sponsor?
- Existing verification techniques not useful

"V&V is very hard because it is hard to come up with good test cases"

Lessons Learned:
Validation and Verification

"I have tried to position CONDOR to the place where it is kind of like your trusty calculator – it is an easy tool to use. Unlike your calculator, it is only 90% accurate … you have to understand that then answer you are going to get is going to have a certain level of uncertainty in it. The neat thing about it is that it is easy to get an answer in the general sense <to a very difficult problem>."

"We have a rule of thumb. We plot 2 lines (from Matlab and C++ programs) and if close, then it is ok."

"It is an engineering judgment as to which errors are important and which ones are on the margins"

Lessons Learned: Validation and Verification

- Implications
  - Traditional software testing methods are not sufficient
  - Need methods that ensure the quality and limits of software
- Suggestions
  - Inspections
  - Formal planning
  - Use of regression test suites

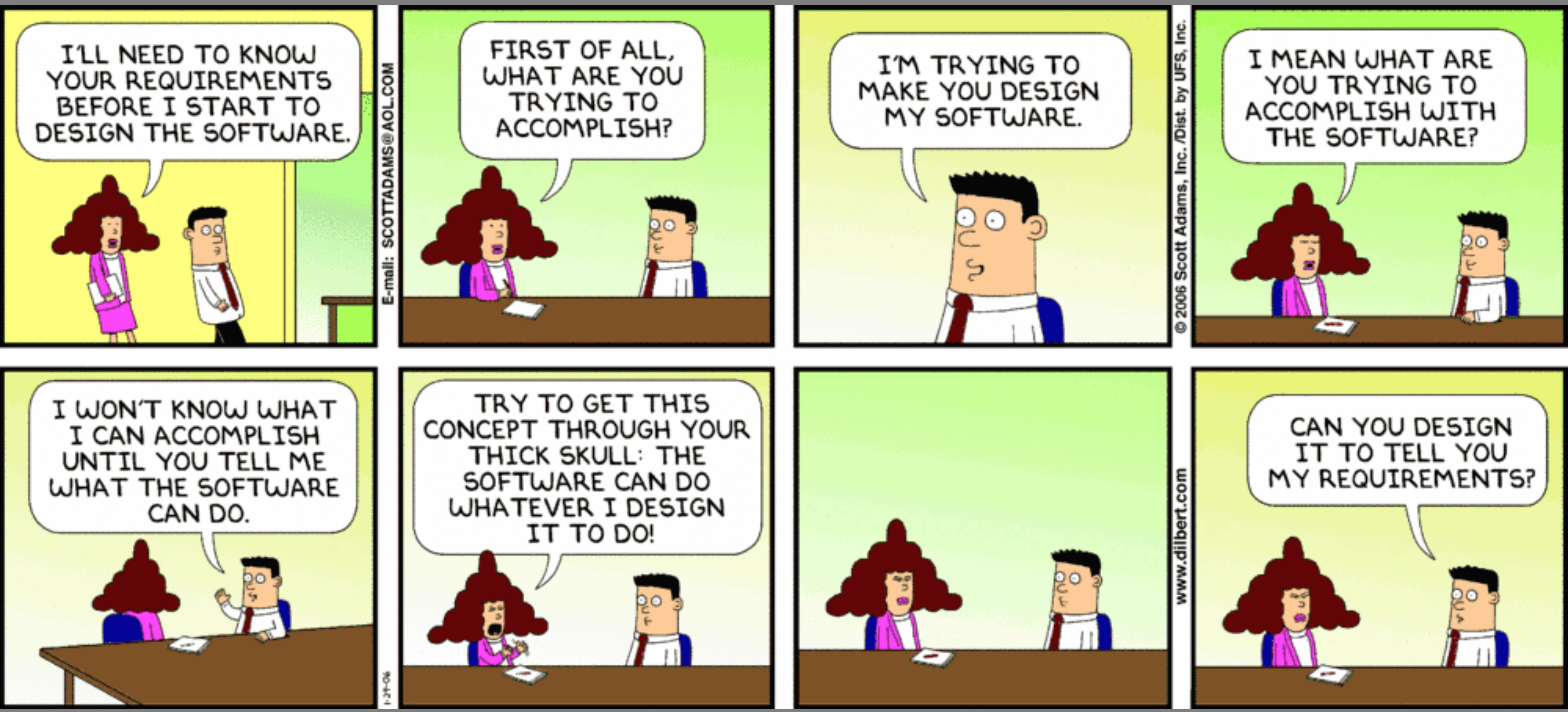
Lessons Learned: Development Goals

- Multiple goals are important
  - Performance – software is used on supercomputer
  - Portability and Maintainability – platforms change multiple times during a project

- Success of a project depends on the ability to port software to new machines

- Implications
  - The motivation for these projects may be different than for traditional IT projects
  - Methods must be chosen and tailored to align with the overall project goals

Lessons Learned:
Agile vs. Traditional Methodologies



http://dilbert.com/strip/2006-01-29

Lessons Learned: Agile vs. Traditional Methodologies

- Requirements constantly change as scientific knowledge evolves

- "Agile" software development methods
  - Tend to be more adaptable to change
  - Favor individuals and practices over process and tools

- Teams operate with agile philosophy by default

- Implications
  - Appropriate, flexible SE methodologies need to be employed for CSE software development
  - Agile-inspired approaches may be most appropriate

Lessons Learned:
Development Environments

http://dilbert.com/strip/1992-09-08

Lessons Learned:
Development Environments

They all [the IDEs] try to impose a particular style of development on me and I am forced into a particular mode

- Developers prefer flexibility of the command line over an Integrated Development Environment (IDE)

- Developers believe that:
  - IDEs impose too much rigidity
  - They are more efficient typing than navigating menus

- Implications – developers do not adopt IDEs because:
  - They do not trust the IDE to automatically perform a task in the same way they would do it manually
  - They expect greater flexibility than is currently provided
  - Prefer to use what they know rather than change

SE4Science Workshops

SE4Science Workshop Series
http://SE4Science.org

- Facilitate interaction between SE and Computational Scientists

- Held at ICSE, ICCS, and SC

- Discussion Topics
  - Testing scientific software
  - Trade-offs between quality goals
  - Research Software vs. IT Software
  - Crossing the communication chasm
  - Measuring impact on scientific productivity
  - Reproducibility of results

SE4Science Workshop Series
Domain Characteristics

- Complex domains
- Main focus on science
- Long lifecycles
- Investigation of unknown introduces risk
- Unique characteristics of developers
  - Deep knowledge of domain – lack formal SE
  - Often the main users of the software

SE4Science Workshop Series
Testing Scientific Software

- Stakes not high enough to make testing important
- Needs differ across domains
- Focus on process transparency
- Guaranteed not to give an incorrect output

SE4Science Workshop Series
Crossing the Communication Chasm

- Need to eliminate the stigma associated with SE
- Software Engineers need to
  - Understand domain constraints
  - Understand specific problems
  - Learn from Computational developers
  - Describe SE concepts in terms familiar to Computational developers
- Need people with expertise in both SE & Computational Science
- Computational teams need:
  - To realize a problem before needing help
  - Real examples of SE success within their domain

SE4Science Workshop Series
Scientific Impact

- Need to evaluate impact
- Scientific productivity ≠ Software productivity
- Need results in a relatively short time
  - Self-assessments
  - Word of mouth

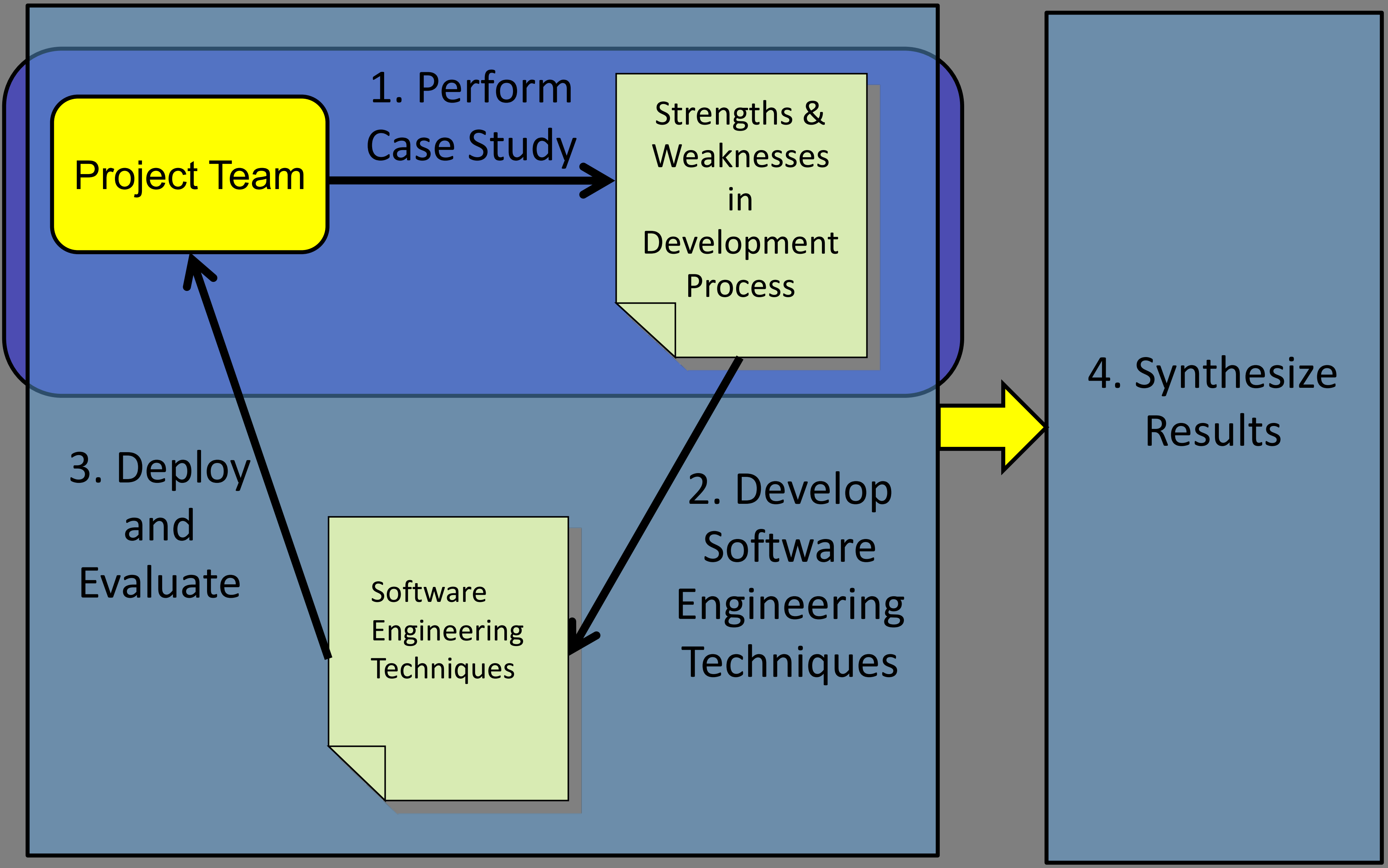
SE4Science Workshop Series
http://SE4Science.org

- Next edition – during ICSE'18
- Gothenberg, Sweden
- Please consider attending

http://SE4Science.org/workshops/

Direct Interactions

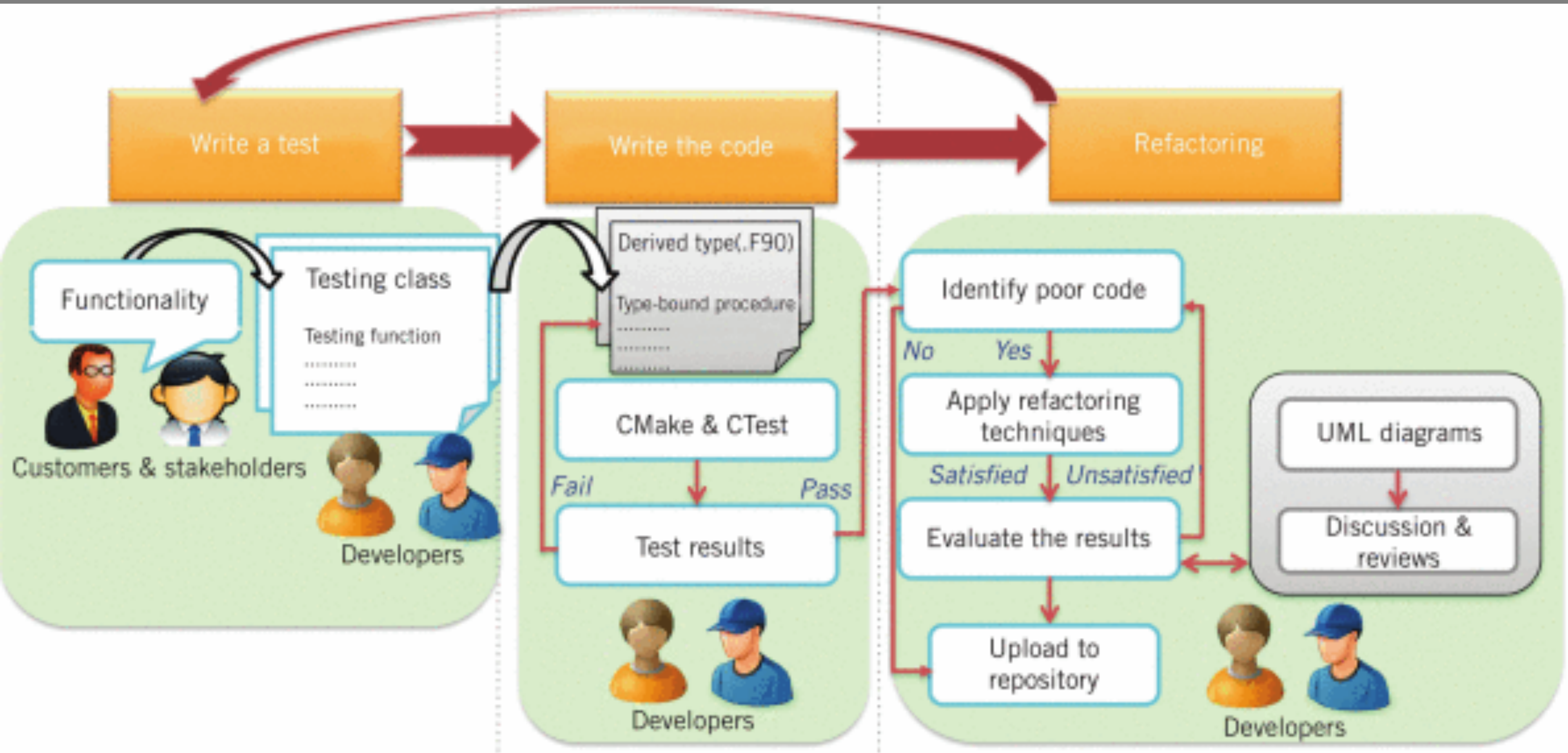
One Possible Methodology



Successful SE/CSE Interactions: TDD - Sandia

- Student spent semester at Sandia
- Taught and modeled TDD on a science code project
- Developed 2 tests for each PDE
  - Small number of steps
  - Whole time evolution
- Lessons Learned
  - Mitigated risks in changing requirements
  - Reduced developer effort
  - Continuous feedback from customer

Successful SE/CSE Interactions: TDD - Sandia



Nanthaamornphong, A. Carver, J., et al. "Building CLiiME via Test-Driven Development: A Case Study." *Computing in Science & Engineering,* 16(3): 36-46

Successful SE/CSE Interactions: Peer Review - ORNL

- Student spent summer with science team at ORNL
- Taught team peer code review process
- Team adopted and continued on own
- Anecdotal Benefits
  - Found faults that would not have been found with traditional testing
  - Adopted coding standard for readability

Ongoing Work

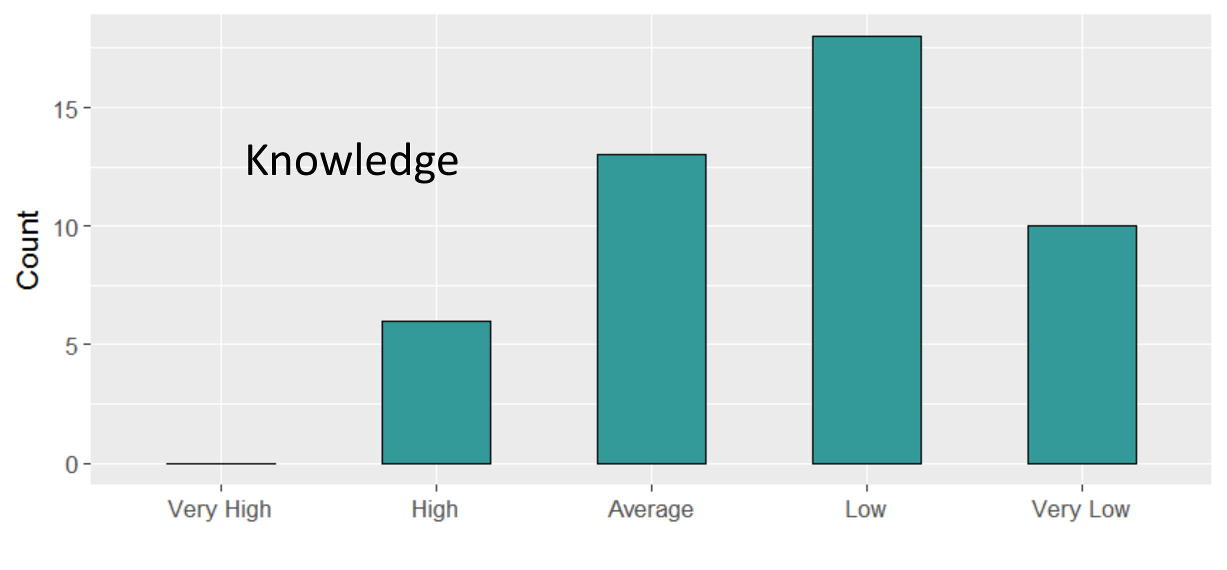
“Bad By Admission” Code:

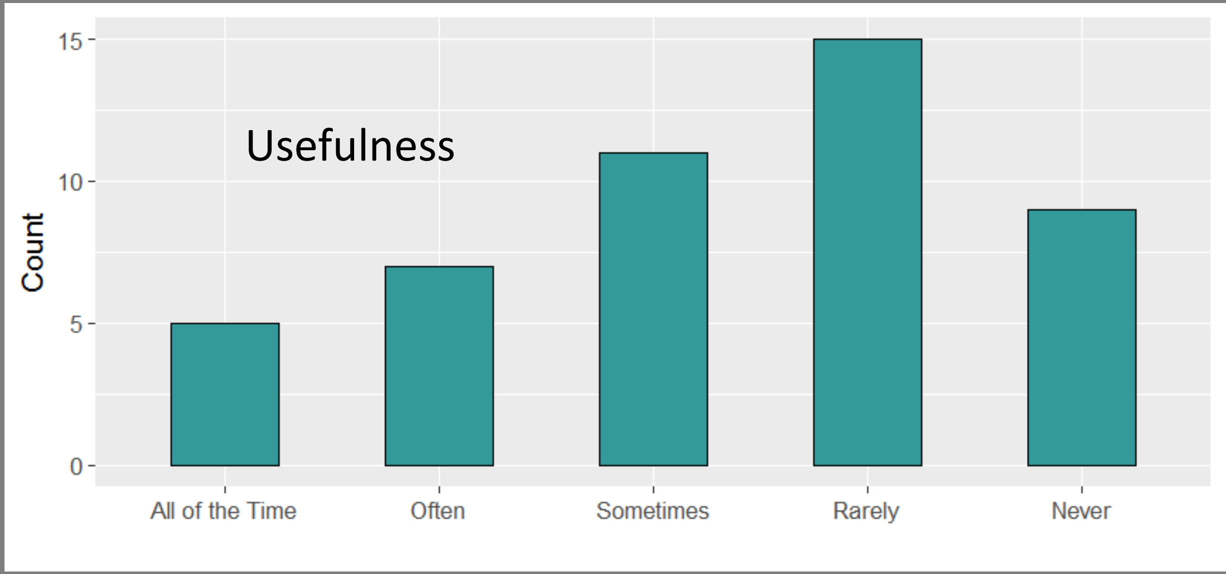
- Code that is actively recognized as deficient
  - Indicated by TODO or FIX
  - Often not fixed
- Compare Scientific and other software in GitHub
  - Compared 10 projects
  - Scientific code has 2x as many TODOs

Software Metrics in Scientific Software

- Survey of scientific software developers
- Goals
  - Understand knowledge and use of metrics
  - Understand perceived usefulness of metrics
  - Gain some insight into software process

Software Metrics in Scientific Software: Knowledge and Use of Metrics





Software Metrics in Scientific Software: Knowledge and Use of Metrics

Category	Number of Unique Metrics	Known (frequency)	Used (frequency)
Architecture	1	1	0
Code Complexity	13	49	10
General Quality	5	5	6
Methodology	2	3	3
Performance	9	13	17
Process	9	7	6
Recognition	5	4	4
Testing	12	20	13

Code Review in Scientific Software

- Interviews and surveys of scientific software developers
- Goals
  - Understand code review process
  - Understand impacts and expectations
  - Understand barriers
  - Identify areas of potential improvement

Code Review in Scientific Software: Importance

- Large portion of code is reviewed
- Shared expertise improves code quality
- Consistent style and reusability
- Good for new contributors and tricky features
- Saves debugging time

Code Review in Scientific Software: Challenges

- Underlying science viewed as more important than code
- Developers are attached to the way they have done things and resist change
- Lack of time and qualified contributors
- Lack of enough people to properly review
- Obtaining reviewer agreement

Summary

- Scientific Software Engineering needs:
  - Diverse
  - Deep
- Unique problems that lack simple solutions
- Successful interactions require
  - Time
  - Openness to new ideas

@SE4Science

carver@cs.ua.edu

Acknowledgements

- Roscoe Bartlett
- Victor Basili
- Neil Chue Hong
- Nasir Eisty – PhD student
- Thomas Epperly
- Christine Halverson
- Dustin Heaton – *Former PhD student*
- Lorin Hochstein
- Jeff Hollingsworth
- Dan Katz
- Richard Kendall
- Karla Morris
- Aziz Nanthaamornphong - *Former PhD student*
- Damian Rouson
- Forrest Shull
- Susan Squires
- Doug Post
- Marvin Zelkowitz

Further Readings:
Community Surveys

- Carver, J., Heaton, D., Hochstein, L., Bartlett, R. "Self-Perceptions about Software Engineering: A Survey of Scientists and Engineers." *Computing in Science and Engineering*. 15(1): 7-11. Jan/Feb 2013.

- Dustin Heaton, Jeffrey Carver, Roscoe Bartlett, Kimberly Oakes and Lorin Hochstein. “The Relationship Between Development Problems and Use of Software Engineering Practices in Computational Science.” *Proceedings of the First Workshop on Maintainable Software Practices in e-Science*.

Further Readings:
SE for CSE

- Carver, J., Kendall, R., Squires, S. and Post, D. "Software Development Environments for Scientific and Engineering Software: A Series of Case Studies." *Proceedings of the 2007 International Conference on Software Engineering*. Minneapolis, MN. May 23-25, 2007. p. 550-559.

- Basili, V., Carver, J., Cruzes, D., Hochstein, L., Hollingsworth, J., Shull, F. and Zelkowitz, M. "Understanding the High Performance Computing Community: A Software Engineer's Perspective." *IEEE Software*, 25(4): 29-36. July/August 2008.

- Carver, J., Hochstein, L., Kendall, R., Nakamura, T. Zelkowitz, M., Basili, V. and Post, D. "Observations about Software Development for High End Computing." *CTWatch Quarterly*. November, 2006. p. 33-37. (Invited Paper).

- Hochstein, L., Nakamura, T., Basili, V., Asgari, S., Zelkowitz, M. Hollingsworth, J., Shull, F., Carver, J., Voelp, M., Zazworka, N., and Johnson, P. "Experiments to Understand HPC Time to Development." *CTWatch Quarterly*. 2(4A): 24-32. November, 2006

Further Readings:
SE-CSE Workshops

- Carver, J., Chue Hong, N., and Ciraci, S. "Software Engineering for CSE." *Scientific Programming*. Volume 2015. Article ID 591562. DOI: 10.1155/2015/591562
- Carver, J. and Epperly, T. "Software Engineering for Computational Science and Engineering [Guest editors' introduction]." *Computing in Science and Engineering*. 16(3):6-9. May/June 2014.
- Carver, J. "Software Engineering for Computational Science and Engineering." (Guest Editor's Introduction). *Computing in Science and Engineering*, 14(2):8-11. March/April 2012.
- Carver, J. "Software engineering for computational science and engineering," *Computing in Science & Engineering*, vol. 14, no. 2, pp. 8–11, 2011.
- Carver, J. "Report from the Second International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE 09)." *Computing in Science & Engineering*. 11(6): 14-19. Nov/Dec. 2009.
- Carver, J. "First International Workshop on Software Engineering for Computational Science and Engineering." *Computing in Science & Engineering*. 11(2): 8-11. March/April 2009.

Further Readings:
Case Studies

- Kendall, R., Carver, J., Fisher, D., Henderson, D., Mark, A., Post, D., Rhoades, C. and Squires, S. "Development of a Weather Forecasting Code: A Case Study." *IEEE Software*, 25(4): 59-65. July/August 2008.
- Kendall, R.P., Carver, J., Mark, A., Post, D., Squires, S., and Shaffer, D. Case Study of the Hawk Code Project. Technical Report, LA-UR-05-9011. Los Alamos National Laboratories: 2005.
- Kendall, R.P., Mark, A., Post, D., Squires, S., and Halverson, C. Case Study of the Condor Code Project. Technical Report, LA-UR-05-9291. Los Alamos National Laboratories: 2005.
- Kendall, R.P., Post, D., Squires, S., and Carver, J. Case Study of the Eagle Code Project. Technical Report, LA-UR-06-1092. Los Alamos National Laboratories: 2006.
- Post, D.E., Kendall, R.P., and Whitney, E. "Case study of the Falcon Project". In Proceedings of Second International Workshop on Software Engineering for High Performance Computing Systems Applications (Held at ICSE 2005). St. Louis, USA. 2005. p. 22-26

Further Readings: Community Interactions

- Nanthaamornphong, A.; Morris, K.; Rouson, D.W.I.; Michelsen, H.A., "A case study: Agile development in the community laser-induced incandescence modeling environment (CLiiME)," *5th International Workshop on Software Engineering for Computational Science and Engineering (SE-CSE), 2013.* doi: 10.1109/SECSE.2013.6615094