Research Software Engineering: An Empirical Software Engineering Researcher's Perspective

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

@SE4Science

January 15, 2024

- I am a software engineering researcher who studies RSE, not an RSE
- My work is all done empirically, I think it is important to collect various types of data through various approaches to understand the needs and effectiveness of practices – I will talk more about this as we go

Research Software

Use	90%	95%
Can't live without	70%	63%

DOI:10.5281/zenodo.843607

Software Engineering

- People vs. tools/languages
- Important to understand human factors
- Most of my work could be considered ”applied sociology” – even though I don’t have a formal background

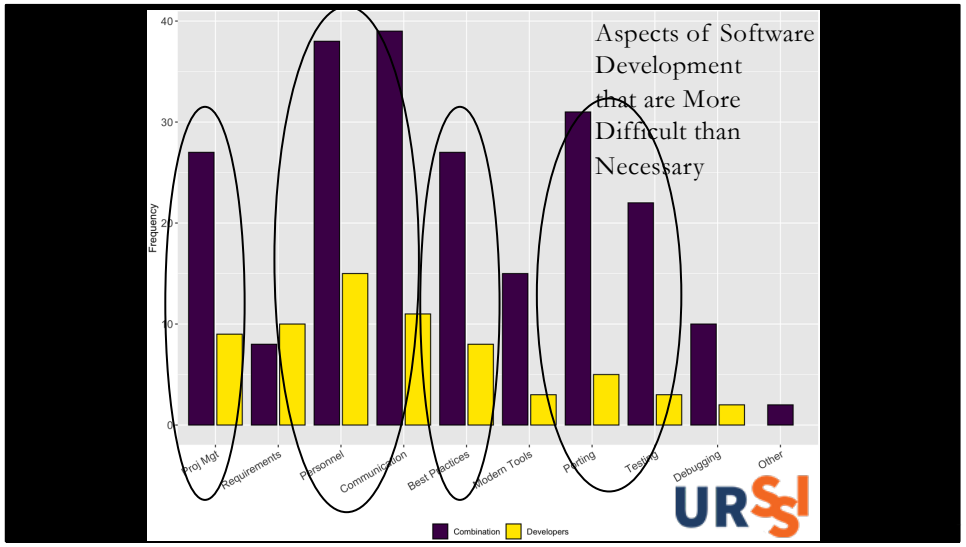
- There is an increasing focus on humans in the general SE world (see previous ICSE foci and papers)
- Many of the issues that come up are not technical
- We must understand the needs of the developer and/or user community before choosing the right solution (this is especially true when choosing a new tool/language/…)
- It is also important to choose the appropriate SE methods based on the types of software mentioned earlier and the developer dimension (individual up to group)
- We've done research in Software Process, Human Errors, Social Debt, Peer Code Review
- One area we could learn a lot from is Open-Source – we've done a lot of research in community building in the Open Source context

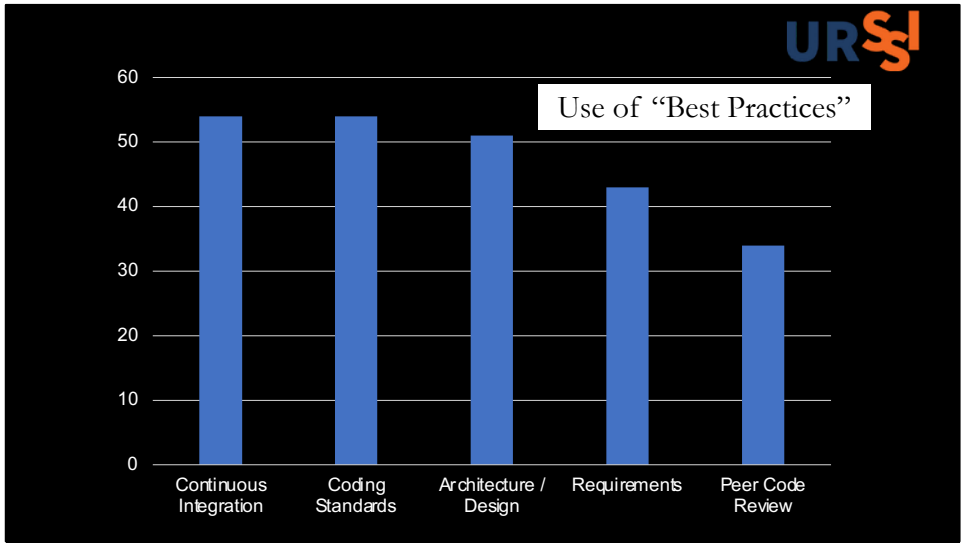
- Peer Code Review Survey – currently completing a follow-up
- Testing Survey – Finalizing write-up on follow-up
- URSSI Survey

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

Development Methodologies

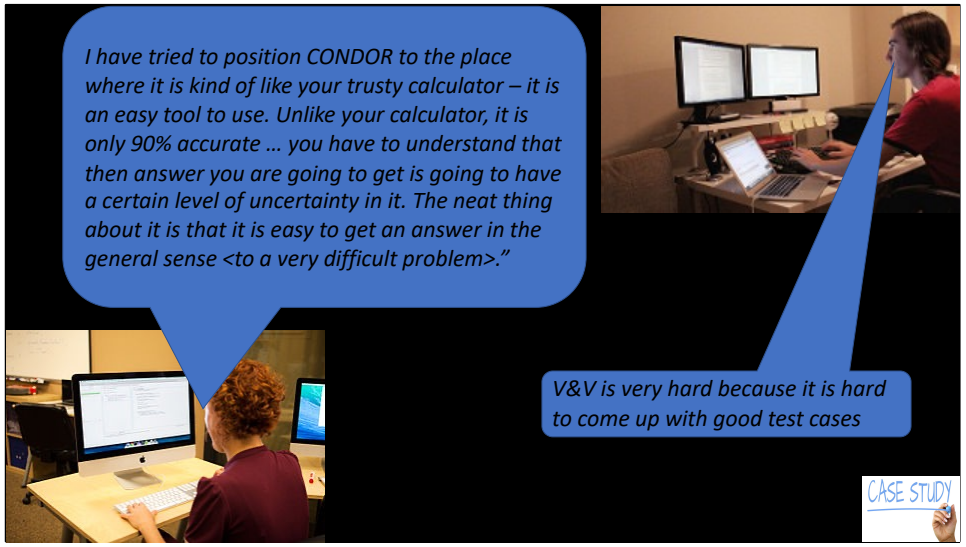
Where People Spend Software Time

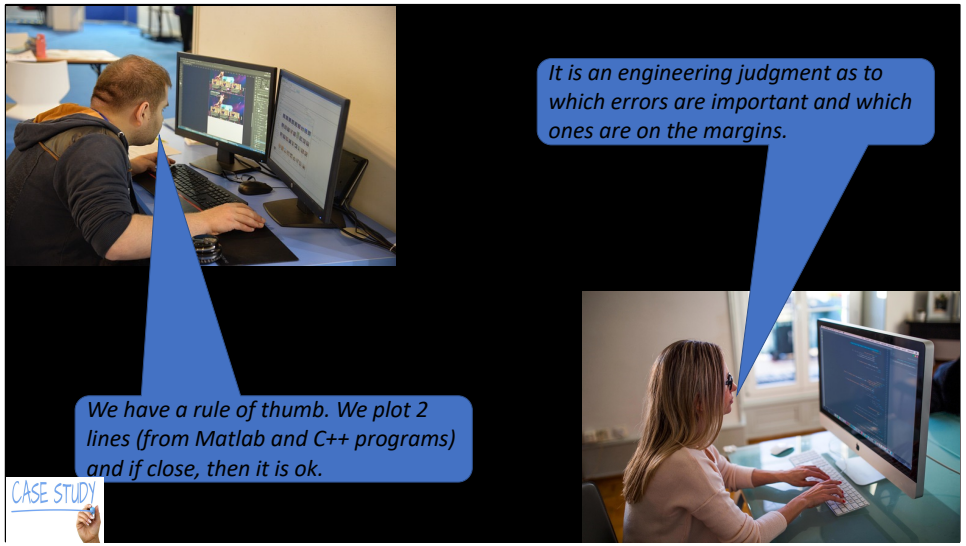

Aspects of Software Development that are More Difficult than Necessary
Frequency
40
30
20
10
0
Proj Mgt
Requirements
Personnel
Communication
Best Practices
Modern Tools
Porting
Testing
Debugging
Other
Combination
Developers
URSSI


URSSI
Use of "Best Practices"
60
50
40
30
20
10
0
Continuous Integration
Coding Standards
Architecture / Design
Requirements
Peer Code Review

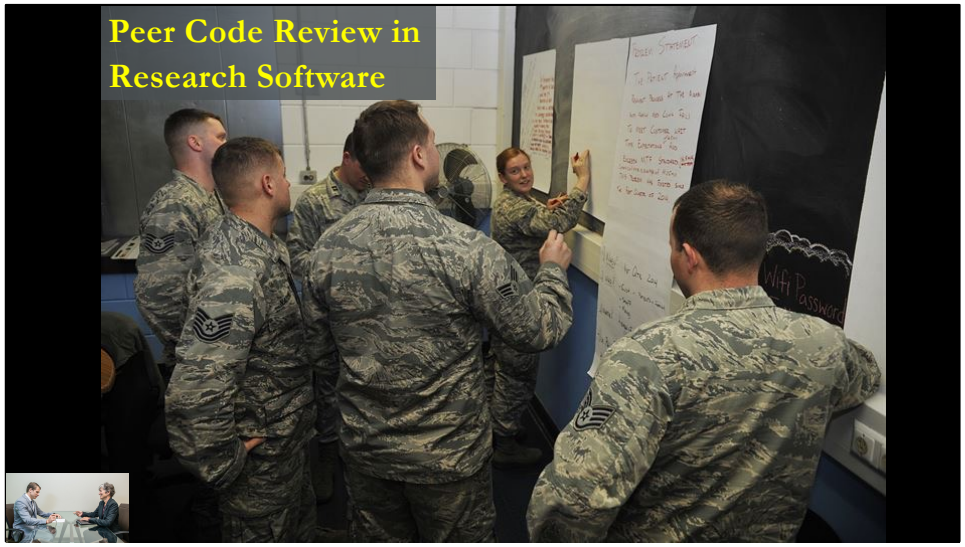
Validation & Verification

- 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

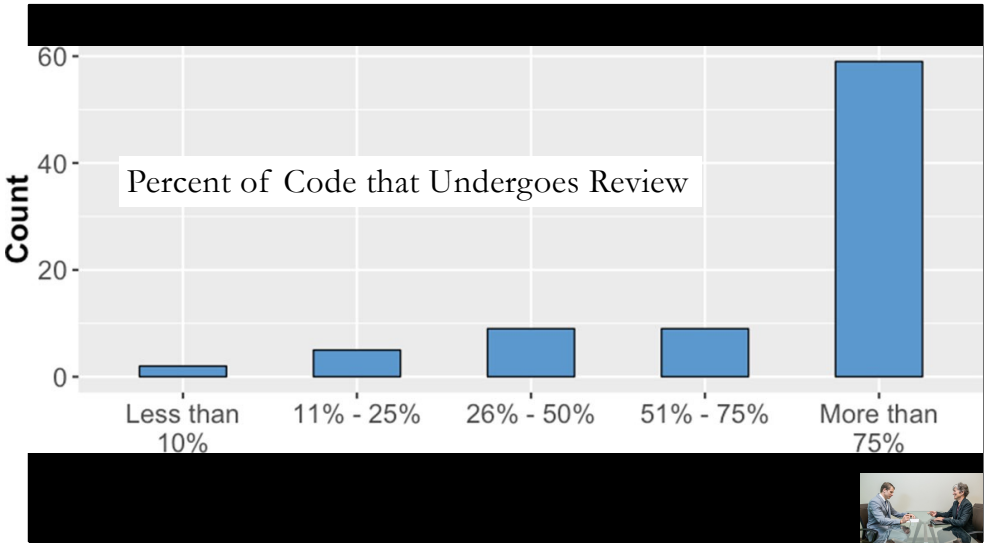

I have tried to position CONDOR to the place where it is kind of like your trusty calculator – it is an easy tool to use. Unlike your calculator, it is only 90% accurate … you have to understand that then answer you are going to get is going to have a certain level of uncertainty in it. The neat thing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

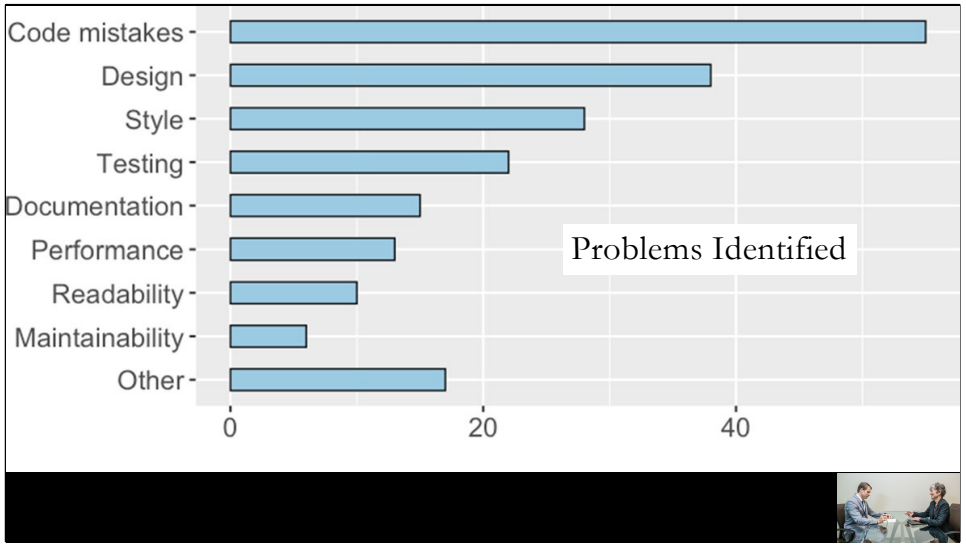


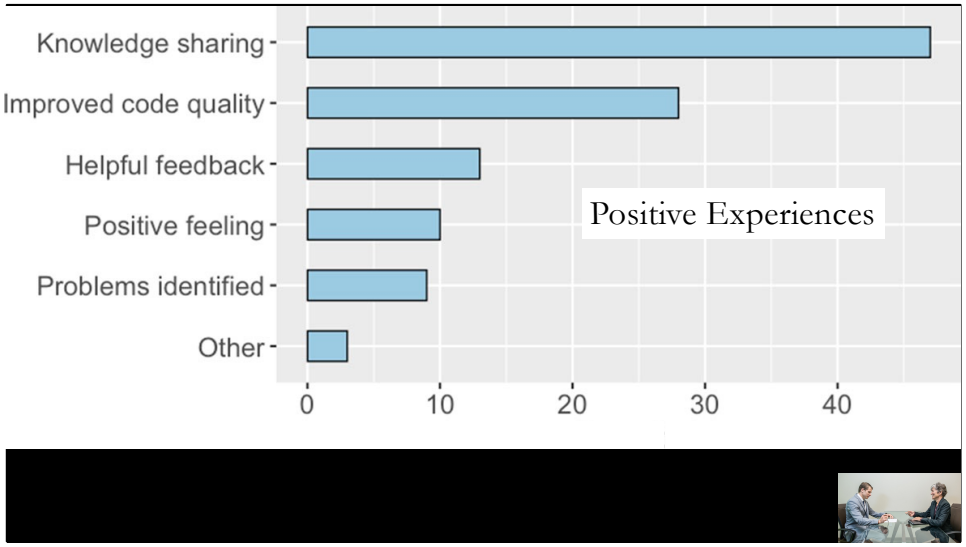
- Implications
  - Traditional software testing methods are not sufficient
  - Need methods that ensure the quality and limits of software
- Suggestions
  - Inspections
  - Formal planning
  - Use of regression test suites

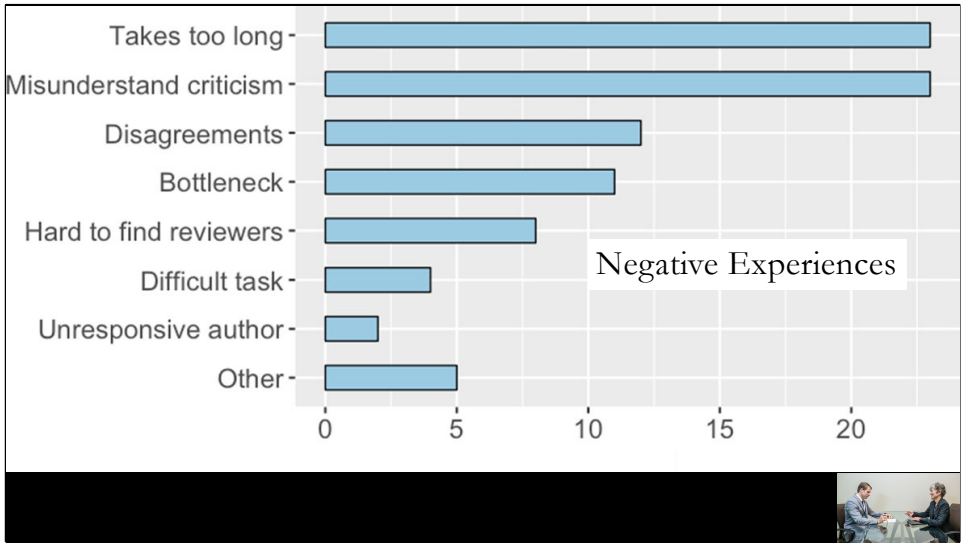


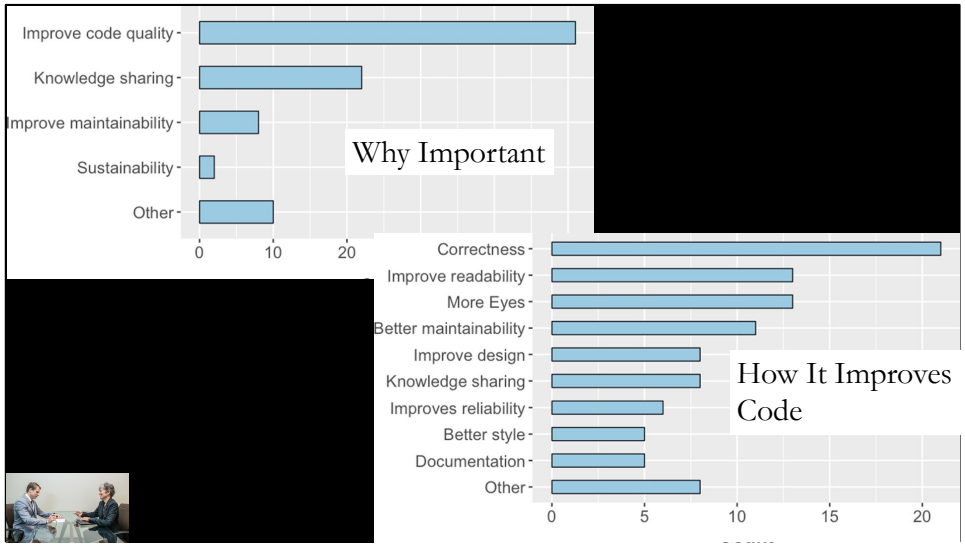
- Interviews and surveys of scientific software developers – will highlight some key findings

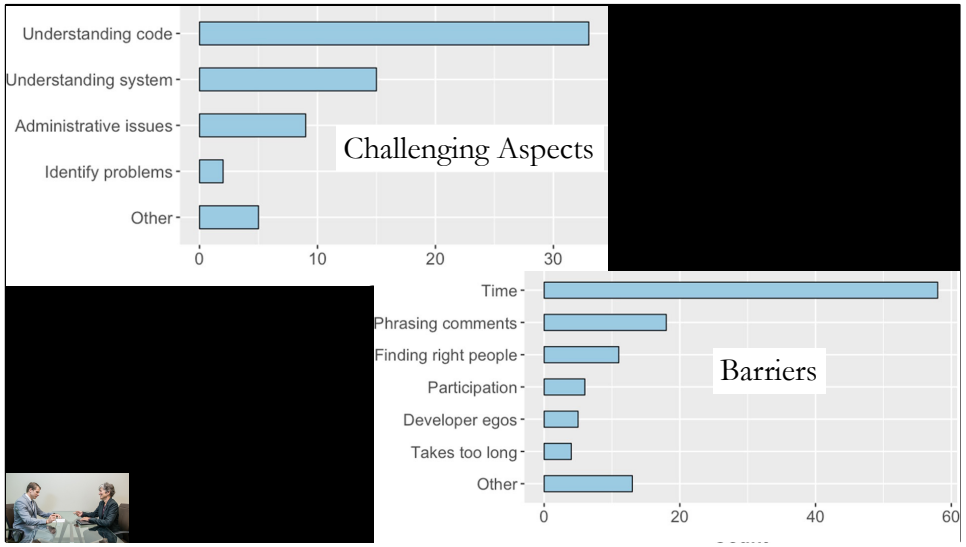

Percent of Code that Undergoes Review
Count
60
40
20
0
Less than 10%
11% - 25%
26% - 50%
51% - 75%
More than 75%

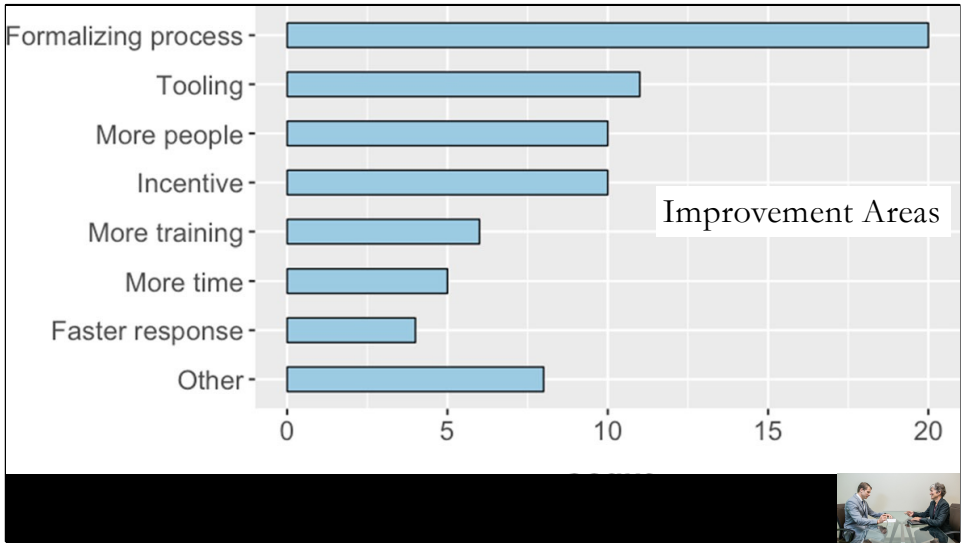

Code mistakes
Design
Style
Testing
Documentation
Performance
Readability
Maintainability
Other
0
20
40
Problems Identified


Knowledge sharing
Improved code quality
Helpful feedback
Positive feeling
Problems identified
Other
0
10
20
30
40
Positive Experiences

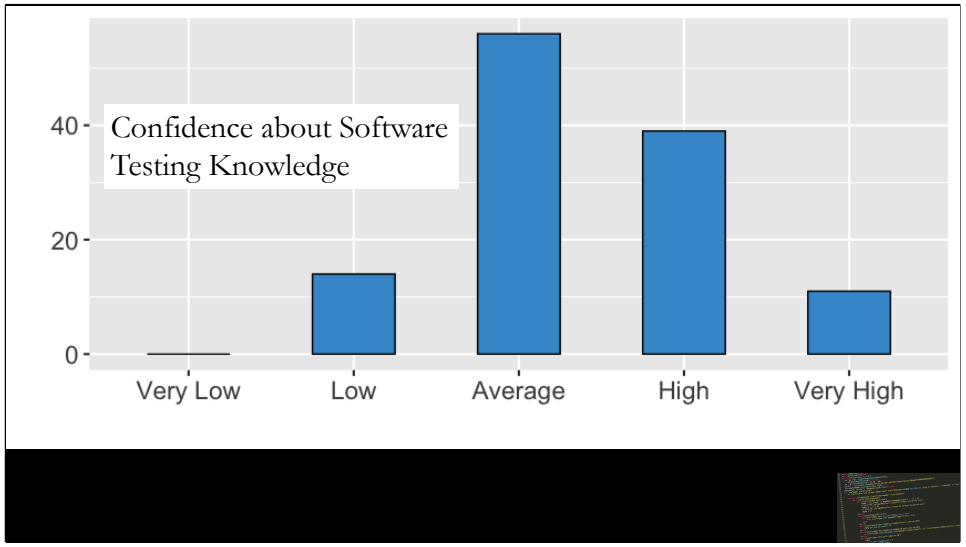

Takes too long
Misunderstand criticism
Disagreements
Bottleneck
Hard to find reviewers
Difficult task
Unresponsive author
Other
0
5
10
15
20
Negative Experiences

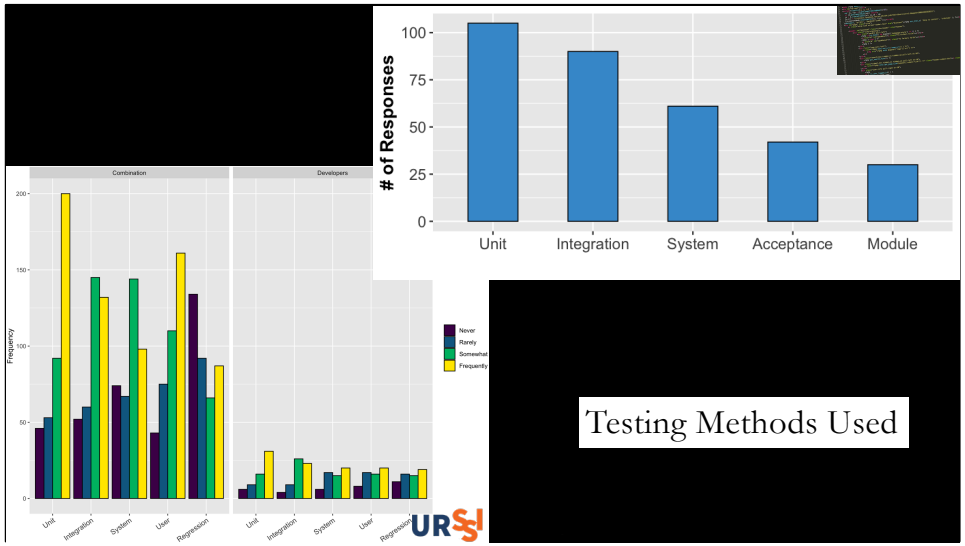

Improve code quality
Knowledge sharing
Improve maintainability
Sustainability
Other
0
10
20
Why Important
Correctness
Improve readability
More Eyes
Better maintainability
Improve design
Knowledge sharing
Improves reliability
Better style
Documentation
Other
0
5
10
15
20
How It Improves Code

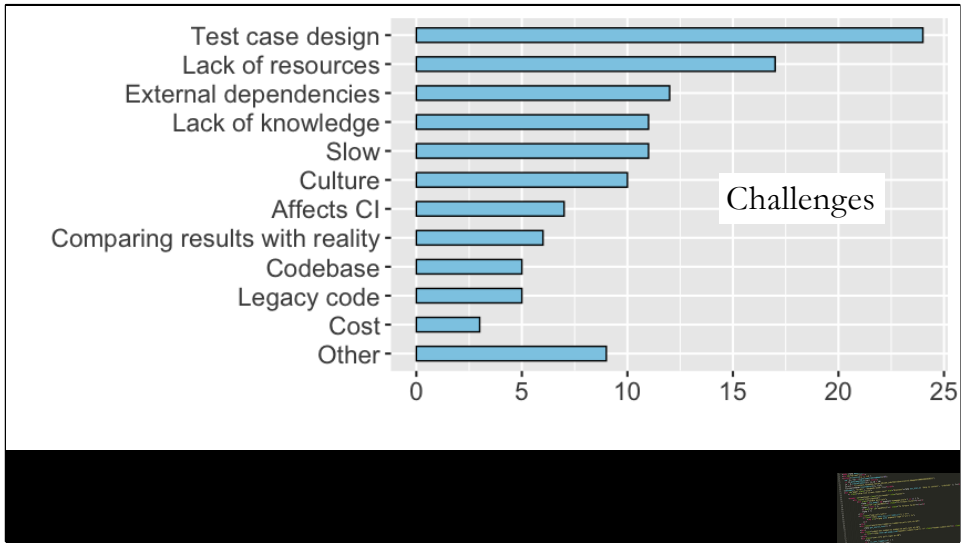

Understanding code
Understanding system
Administrative issues
Identify problems
Other
0
10
20
30
Challenging Aspects
Time
Phrasing comments
Finding right people
Participation
Developer egos
Takes too long
Other
0
20
40
60
Barriers

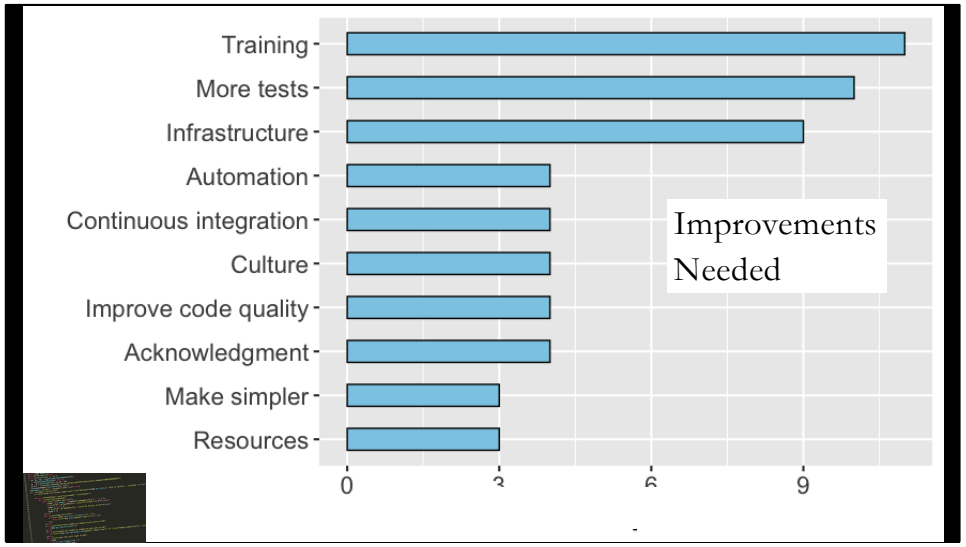

Formalizing process
Tooling
More people
Incentive
More training
More time
Faster response
Other
0
5
10
15
20
Improvement Areas

Software Testing


Confidence about Software Testing Knowledge
40
20
0
Very Low
Low
Average
High
Very High


# of Responses
100
75
50
25
0
Unit
Integration
System
Acceptance
Module
Combination
Developers
Frequency
Never
Rarely
Somewhat
Frequently
Unit
Integration
System
User
Regression
URSSI
Testing Methods Used


Test case design
Lack of resources
External dependencies
Lack of knowledge
Slow
Culture
Affects CI
Comparing results with reality
Codebase
Legacy code
Cost
Other
0
5
10
15
20
25
Challenges


Training
More tests
Infrastructure
Automation
Continuous integration
Culture
Improve code quality
Acknowledgment
Make simpler
Resources
0
3
6
9
Improvements Needed

Research Software Engineering: An Empirical Software Engineering Researcher's Perspective

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

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

Acknowledgements

- Roscoe Bartlett
- Victor Basili
- Neil Chue Hong
- Nasir Eisty
- Thomas Epperly
- Sandra Gesing
- Christine Halverson
- Dustin Heaton
- Lorin Hochstein
- Jeff Hollingsworth
- Dan Katz
- Richard Kendall
- Karla Morris
- Aziz Nanthaamornphong
- Karthik Ram
- Damian Rouson
- Forrest Shull
- Susan Squires
- Nic Weber
- Doug Post
- Marvin Zelkowitz

Photo Credits

- Alpha Stock Images - http://alphastockimages.com/
- By Joonspoon [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], from Wikimedia Commons
- https://www.hotelmanagement.net/nyu-lessons-learned
- By Roger McLassus - Picture taken and uploaded by Roger McLassus., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=575581