Who Am I?
Research Software
Software Quality
Part #1

Testing Research Software: A survey
Online Survey
Part #1 (Testing) Outline

• Demographics
• Level of knowledge research software developers have on testing
• Current testing practices in research software community
• Difficulties to test research software
• Compatibility of Commercial/IT testing techniques
• Improvement of the testing process in research software
Part #1 (Testing) Outline

• Demographics
• Level of knowledge research software developers have on testing
• Current testing practices in research software community
• Difficulties to test research software
• Compatibility of Commercial/IT testing techniques
• Improvement of the testing process in research software
# Developers

<table>
<thead>
<tr>
<th>Count</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 5</td>
<td>67</td>
</tr>
<tr>
<td>6 to 10</td>
<td>25</td>
</tr>
<tr>
<td>11 to 15</td>
<td>17</td>
</tr>
<tr>
<td>16 to 20</td>
<td>5</td>
</tr>
<tr>
<td>More than 20</td>
<td>5</td>
</tr>
</tbody>
</table>
Part #1 (Testing) Outline

• Demographics
• Level of knowledge research software developers have on testing
• Current testing practices in research software community
• Difficulties to test research software
• Compatibility of Commercial/IT testing techniques
• Improvement of the testing process in research software
Knowledge of Testing

Count

Low  Average  High  Very High

10  50  30  10
Understanding Testing Concepts USED

[Bar chart showing counts for Very Low, Low, Average, High, and Very High]
Understanding Testing Concepts NEEDED
Part #1 (Testing) Outline

- Demographics
- Level of knowledge research software developers have on testing
- Current testing practices in research software community
- Difficulties to test research software
- Compatibility of Commercial/IT testing techniques
- Improvement of the testing process in research software
Level 0 – Level 0
Level 1 – The Purpose of testing is to show correctness
Level 2 – Level 2
Level 3 – The Purpose of testing is not to prove anything specific, but to reduce the risk of using the software
Level 4 – Level 4
Testing Methods Used

- Unit
- Integration
- System
- Acceptance
- Module
Part #1 (Testing) Outline

• Demographics
• Level of knowledge research software developers have on testing
• Current testing practices in research software community
• **Difficulties to test research software**
• Compatibility of Commercial/IT testing techniques
• Improvement of the testing process in research software
Part #1 (Testing) Outline

• Demographics
• Level of knowledge research software developers have on testing
• Current testing practices in research software community
• Difficulties to test research software
• Compatibility of Commercial/IT testing techniques
• Improvement of the testing process in research software
Commercial/IT Testing Methods - Team

Always: 32
Most of the time: 34
Sometimes: 21
Rarely: 16
Never: 20
Commercial/IT Testing Methods - Individual

- **Always**: 30
- **Most of the time**: 35
- **Sometimes**: 20
- **Rarely**: 15
- **Never**: 20
Value Seen in Comm/IT Testing Methods

- Very High: Count
- High: 40
- Average: 30
- Low: 10
- Very Low: 5
Challenges to Adapt Comm/IT Methods

- Not useful
- Lack of resources
- Mindset
- Lack of knowledge
- Infrastructure
- Cost
- Difficult to use
- Runtime restrictions
- Other

Count
Challenges Not Met by Comm/IT Methods

- Lack of expertise: 5
- Slow: 3
- CI issues: 2
- Benchmarks: 2
- Other: 8
Part #1 (Testing) Outline

• Demographics

• Level of knowledge research software developers have on testing

• Current testing practices in research software community

• Difficulties to test research software

• Compatibility of Commercial/IT testing techniques

• Improvement of the testing process in research software
• Researchers pose a clear goal of testing their project

• Complexity associated with the process needs further attention

• Make a culture of testing in the research software community.

• Providing proper training and resources can improve the testing process in research software.
Peer Code Review in Research Software
Interview & Survey
Part #2 (Code Review) Outline

- Demographics
- Current code review practices in research software
- Impacts of the code review process in research software
- Difficulties developers face during code review
- Potential areas of improvement in the review process
• Demographics

• Current code review practices in research software

• Impacts of the code review process in research software

• Difficulties developers face during code review

• Potential areas of improvement in the review process
Years Worked

<table>
<thead>
<tr>
<th>Years Worked</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 1 year</td>
<td>4</td>
</tr>
<tr>
<td>1 year to less than 5 years</td>
<td>26</td>
</tr>
<tr>
<td>5 years to 10 years</td>
<td>27</td>
</tr>
<tr>
<td>More than 10 years</td>
<td>24</td>
</tr>
</tbody>
</table>
Role

Count

Add new code 70
Fix bugs 60
Report bugs 60
Maintain project 50
Make strategic decisions 50
Other 10
Balance as a Reviewee and Reviewer

- Reviewee only: 1, 2, 3, 4 - Equally, 5, 6, 7
- Reviewer only: 1 - Reviewee only, 2, 3, 4 - Equally, 5, 6, 7 - Reviewer only
Part #2 (Code Review) Outline

• Demographics

• **Current code review practices in research software**

• Impacts of the code review process in research software

• Difficulties developers face during code review

• Potential areas of improvement in the review process
Percentage of Code Undergo Review

- Less than 10%
- 11% - 25%
- 26% - 50%
- 51% - 75%
- More than 75%
Time Spent on Code review

<table>
<thead>
<tr>
<th>Hour</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than 1</td>
<td>23</td>
</tr>
<tr>
<td>1 to 5</td>
<td>45</td>
</tr>
<tr>
<td>6 to 10</td>
<td>8</td>
</tr>
<tr>
<td>10 to 15</td>
<td>5</td>
</tr>
<tr>
<td>More than 20</td>
<td>1</td>
</tr>
</tbody>
</table>
Time For a First Response

- Less than 1 hour
- Less than 1 day
- 1 - 3 days
- 4 - 7 days
- More than 7 days
Positive Experience

- Knowledge sharing
- Improved code quality
- Good feedback
- Positive feeling
- Problems identified
- Other

Count

- Knowledge sharing: 47
- Improved code quality: 28
- Good feedback: 12
- Positive feeling: 10
- Problems identified: 8
- Other: 3
Part #2 (Code Review) Outline

• Demographics

• Current code review practices in research software

• Impacts of the code review process in research software

• Difficulties developers face during code review

• Potential areas of improvement in the review process
Why Code Review is Important

- Improve code quality
- Knowledge sharing
- Improve maintainability
- Sustainability
- Other

Count
How Code Review Improves Code

Correctness: 20
Improve readability: 13
More Eyes: 13
Better maintainability: 12
Improve design: 11
Knowledge sharing: 9
Improves reliability: 7
Better style: 5
Documentation: 5
Other: 7

Count
Part #2 (Code Review) Outline

• Demographics
• Current code review practices in research software
• Impacts of the code review process in research software
• Difficulties developers face during code review
• Potential areas of improvement in the review process
Challenges

- Understanding code: 32
- Understanding system: 14
- Administrative issues: 8
- Identify problems: 2
- Other: 2
Part #2 (Code Review) Outline

• Demographics

• Current code review practices in research software

• Impacts of the code review process in research software

• Difficulties developers face during code review

• Potential areas of improvement in the review process
Improvements

- Formalizing process: 20
- Tooling: 10
- More people: 10
- Incentive: 10
- More training: 6
- More time: 5
- Faster response: 4
- Other: 7

Count
Discussion

• Research software developers employ an informal code review process

• Code review has an overall positive impact

• Most common difficulty reported by participants is finding time to do it and understand other people’s code.

• Formalizing the review process by including more people, more training, and providing compensation could potentially improve the code review process.
Recommendations - Testing
Recommendations - Testing

• Provide enough training on software testing to all kinds of research software developers ranging from graduate students to experienced researchers

• Incorporate more tests that can solve specific needs of the research software

• Provide infrastructure support, for example, a public service for testing including many-tier pricing structure for machine time and a sophisticated testing dashboard
Recommendations - Testing

• Provide automation for setting tests and analysis of the results

• Improve continuous integration system to facilitate a better way of testing, especially, the incoming tests during down time

• Make a culture of testing in the team and encourage others by sharing the benefits from the experience of testing

• Improve the quality of the code so that developers can write tests easily
• Provide proper acknowledgement of developers for contributions in testing

• Make the testing process simpler so that it is easy to adopt in the project

• Provide enough resources to developers so that they can utilize the resources to develop test suits
Recommendations – Code Review
Recommendations – Code Review

• Make code review process more formal with a structured guideline for each step of the process

• Try to ensure at least one science review and one technical review

• Include automatic tools in the code review process and train your peer reviewers the best practices to use the tool
Recommendations – Code Review

• Encourage more people to participate in the review process and allocate some time to do the review.

• Provide incentives or rewards to reviewers to participate in code review.

• Allocate sufficient time in the development process to perform code review.

• Provide faster feedback to any incoming review request.
• Train reviewers on how to phrase good feedback

• Train developers to forget their egos and accept comments from the reviewers to improve their code

• Make the overall code review process faster

• Provide necessary support from the administrative level that encourages people to participate in the code review process
## Acknowledgement

<table>
<thead>
<tr>
<th>Dr. Jeffrey Carver</th>
<th>Dr. George Thiruvathukal</th>
<th>Dr. David Bernholdt</th>
</tr>
</thead>
<tbody>
<tr>
<td>University of Alabama</td>
<td>Loyola University Chicago</td>
<td>Oak Ridge National Laboratory</td>
</tr>
<tr>
<td>Dr. Hai Ah Nam</td>
<td>Dr. Danny Perez</td>
<td>Dr. J. Dave Moulton</td>
</tr>
<tr>
<td>Los Alamos National Laboratory</td>
<td>Los Alamos National Laboratory</td>
<td>Los Alamos National Laboratory</td>
</tr>
<tr>
<td>Dr. Roland Haas</td>
<td>Dr. Gabrielle Allen</td>
<td>Dr. Daniel Katz</td>
</tr>
<tr>
<td>National Center for Supercomputing Applications</td>
<td>National Center for Supercomputing Applications</td>
<td>National Center for Supercomputing Application</td>
</tr>
</tbody>
</table>

**NSF grant 1445344**