
Alessandro Orso
School of Computer Science
College of Computing
Georgia Institute of Technology
http://www.cc.gatech.edu/~orso

Gregg Rothermel
Department of Computer Science and Engineering
University of Nebraska – Lincoln
http://www.cse.unl.edu/~grother

Partially supported by: NSF, IBM, Google, and MSR
SOFTWARE IS BUGGY!

Partially supported by: NSF, IBM, Google, and MSR
Software Testing

Most used approach

and investigated

Number of papers on testing at ICSE from 2000 to 2014
Software Testing and investigated Most used approach

<table>
<thead>
<tr>
<th>Topic</th>
<th>Number of submissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model-driven software...</td>
<td>10</td>
</tr>
<tr>
<td>Components, services, and...</td>
<td>15</td>
</tr>
<tr>
<td>Mobile, ubiquitous and...</td>
<td>20</td>
</tr>
<tr>
<td>Program comprehension and...</td>
<td>25</td>
</tr>
<tr>
<td>Patterns and frameworks</td>
<td>30</td>
</tr>
<tr>
<td>Requirements engineering</td>
<td>35</td>
</tr>
<tr>
<td>Architecture and design</td>
<td>40</td>
</tr>
<tr>
<td>Security, safety and reliability</td>
<td>45</td>
</tr>
<tr>
<td>Reverse engineering and...</td>
<td>50</td>
</tr>
<tr>
<td>Formal methods</td>
<td>55</td>
</tr>
<tr>
<td>Specification and verification</td>
<td>60</td>
</tr>
<tr>
<td>Software tools and...</td>
<td>65</td>
</tr>
<tr>
<td>Empirical studies of software</td>
<td>70</td>
</tr>
<tr>
<td>Testing and analysis</td>
<td>75</td>
</tr>
<tr>
<td>Most used approach</td>
<td>80</td>
</tr>
</tbody>
</table>
A Special Acknowledgement

Mary Jean Harrold

(1947–2013)
A Travelogue – Our goals

Discuss most successful research in software testing since 2000

Identify most significant challenges and opportunities
A Travelogue – Our Approach

Over 50 colleagues
About 30 responses

Two questions

1. What do you think are the most significant contributions to testing since 2000?

2. What do you think are the biggest open challenges and opportunities in this area?
In a Nutshell
Challenges & Opportunities

Testing Real-World Systems
Oracles
Probabilistic Approaches
Testing Non-Functional Prop.
Domain-Based Testing
Leveraging Cloud and Crowd

Research Contributions

Automated Test Input Generation
Dynamic Symbolic Execution
Search-based Testing
Random Testing
Combined Techniques

Testing Strategies

Combinatorial Testing
Model-Based Testing
Mining/Learning from Field Data

Empirical Studies & Infrastructure

Regression Testing

Practical Contributions

Frameworks for Test Execution
Continuous Integration
So Many Things, So Little Time…

Automated Test Input Generation

Regression Testing

Empirical Studies & Infrastructure

Practical Contributions

Leveraging Cloud and Crowd
Automated Test Input Generation

Not new, but resurgence
• Symbolic execution
• Search-based testing
• Random/fuzz testing
• Combined techniques

Achieve coverage goal
Reach a given point/state

Technical improvements
Powerful machines
Powerful decision procedures
Careful engineering
Not new, but resurgence

- Symbolic execution
- Search-based testing
- Random/fuzz testing
- Combined techniques

Technical improvements
- Powerful machines
- Powerful decision procedures
- Careful engineering

Achieve coverage goal
Reach a given point/state
Symbolic Execution

symbolic state:  
\(x=x_0, \ y=y_0\)  
path condition:  
true

symbolic state:  
\(x=x_0, \ y=y_0, \ z=x_0-y_0\)  
path condition:  
\(x_0\geq0, \ y_0>x_0/2\)

1. int testme(int x, int y) {  
2.   int z;  
3.   if (x < 0)  
4.     return -x;  
5.   z = x - y;  
6.   if (y > z)  
7.     fail();  
8.   return z;  
9. }

1. int testme(int x, int y) {  
2.   int z;  
3.   if (x < 0)  
4.     return -x;  
5.   z = x - y;  
6.   if (y > z)  
7.     fail();  
8.   return z;  
9. }

Solver

solution:  
\(x_0=1, \ y_0=1\)
Symbolic Execution

symbolic state: 
\( x = x_0, y = y_0 \)

path condition: 
true

1. int testme(int x, int y) {
2.   int z;
3.   if (x < 0)
4.     return -x;
5.   z = x - y;
6.   if (y > z)
7.     fail();
8.   return z;
9. }

1. int testme(int x, int y) {
2.   int z;
3.   if (x < 0)
4.     return -x;
5.   z = x - y;
6.   if (y > z)
7.     fail();
8.   return z;
9. }
Symbolic Execution

symbolic state:
\[ x = x_0, \ y = y_0 \]
path condition:
\[ \text{true} \]

1. int testme(int x, int y) {
2.   int z;
3.   if (x < 0)
4.     return -x;
5.   z = cmplxFnct(x, y);
6.   if (y > z)
7.     fail();
8.   return z;
9. }

path condition:
\[ x_0 \geq 0, \ y_0 > \text{cmplxFnct}(10, 10) \]
Symbolic Execution

Open Challenges

- Highly structured inputs
- External libraries
- Large complex programs
- Oracle problem

path condition: \( x_0 \geq 0, y_0 > \text{cmplxFnct}(10, 10) \)
solution: \( x_0 = 1, y_0 = 100 \)
So Many Things, So Little Time…

- Automated Test Input Generation
- Regression Testing
- Empirical Studies & Infrastructure
- Practical Contributions
- Leveraging Cloud and Crowd

Keywords:
- fault
- scale
- tools
- random
- system profiles
- mining practical
- search
- machine localization
- combinatorial
- process
- fixing
- ideas
- coverage
- oracle
- industry
- empirical
- practice
- search-based
- requirements
- verification
- cases
- real regression
- generation
- tool
- model-based
- different
- distributed
- studies
- quality
- security
- automated genetic
- algorithm
- symbolic
- optimization
- reliability
- continuous
- people
From a 2009 survey:

- 168 papers on regression testing
A 2009 survey: 87 papers
Test Suite Minimization

A 2009 survey: 37 papers
Test Case Prioritization

A 2009 survey: 47 papers
A 2015 Survey: over 170 papers
Regression Testing
(What is Happening?)

- Many new techniques are using more sophisticated approaches such as search-based and AI-based

- More extensive empirical studies being conducted

- New topics:
  - Test suite augmentation
  - New software domains (e.g. web-based)
Regression Testing
(What’s Needed Next?)

• Greater industrial uptake
• Requires better efforts to understand practitioners’ problems and needs
• Industrial case studies may help
• Consideration of specific practical processes
Regression Testing (Process Models)

- Traditional:
  - 
  - 
  - 

- Incremental:
  - 
  - 
  - 

- Hyper-Incremental:
  - 
  - 
  - 

- Continuous:
  - 
  - 
  - 

Note: The diagram illustrates the process models for regression testing, showing the timeline for maintenance, V & V, and t1, t2, t3.
So Many Things, So Little Time…
Empirical Studies & Infrastructure

• Testing is heuristic ➞ must be empirically evaluated
• Things have changed dramatically since then:
  • Empirical evaluations are almost required
  • Artifact evaluations at various conferences
Empirical Studies & Infrastructure
(What Changed?)

• Increased availability of experiment objects
• Repositories: SIR (over 2600 users registered, over 800 papers used it), Bugbench, iBugs, Marmoset, SAMATE Reference Dataset, …
• Open-source systems, often large and available with versions, tests, bug reports, …

• Increased availability of supporting infrastructure (analysis tools, coverage tools, mutation tools, …)
• Increased understanding of empirical methodologies
Empirical Studies & Infrastructure

(What’s Needed Next?)

• Address threats to external validity by doing:
  • Controlled studies of more realistic systems
  • Case studies of industrial systems

• Move beyond “tool” and “technique” studies
  • “If you build it they will come” is not necessarily the case with software engineering tools; we need to investigate whether engineers can and will actually use tools
So Many Things, So Little Time…
Practical Contributions

• Frameworks for test execution
• Shortening of the testing process life cycle
Practical Contributions

- Frameworks for test execution
- Dramatically improved the state of the art
- Indirectly affected research
- Examples:

- Shortening of the testing process life cycle
Practical Contributions

• Frameworks for test execution
• Shortening of the testing process life cycle
  • From integrating and testing “at the end”, to early integration and testing, to continuous integration (CI)
• Widely used in industry
• Examples:

Hudson
Jenkins
Travis
So Many Things, So Little Time…
Leveraging Cloud and Crowd

- From local to remote (data centers, servers)
- Software increasingly built and run on the net (e.g., cloud IDEs)
- Natural for testing to follow (e.g., symbolic execution, test farms, heavy-weight analysis)
Leveraging Cloud and Crowd

- Testing is still very much human intensive
- Leveraging the crowd makes much sense
- This has been happening for some time, both in academia and in industry
- Interesting new directions (game-based testing and verification, crowd oracles, ...)

![Image showing people in a brainstorming session with speech bubbles and a diagram illustrating the cycle of software engineering tasks involving in-house, in the field, and field data with SE Tasks: regression testing, performance testing, debugging, behavior classification, ...]
Leveraging Cloud and Crowd

- Testing is still very much human intensive
- Leveraging the crowd makes much sense
- This has been happening for some time, both in academia and in industry
- Interesting new directions (game based testing and verification, crowd oracles, …)

- Testing as a game?
- Must be a game people are willing to play
- Must be easier than the original problem
In Summary

- Incredible amount of work on testing
- Yet, things are not that different… …or are they?

👍 Automated testing
👍 Empirical evaluation
👍 Testing strategies

👍 Testing tools
👍 Testing process
…
Future Directions

• Stop chasing full automation
• True for other related areas too (e.g., debugging, program repair)
• Use the different players for what they are best at doing
  - Human: creativity
  - Computer: computation-intensive, repetitive, error-prone, etc.

Testing Real-World Systems
  - Oracles
Probabilistic Approaches
Testing Non-Functional Properties
Domain-Based Testing
Leveraging Cloud and Crowd
  - Leveraging Cloud and Crowd repetitive, error-prone, etc. tasks
With much appreciated input/contributions from

• Alex Groce
• Andrea Arcuri
• Andreas Zeller
• Andy Podgurski
• Antonia Bertolino
• Atif Memon
• Corina Pasareanu
• Darko Marinov
• David Rosenblum
• Elaine Weyuker
• John Regehr
• Lionel Briand
• Lori Pollock
• Mark Grechanik
• Natalia Juristo
• Paolo Tonella
• Patrice Godefroid
• Per Runeson
• Peter Santhanam
• Phil McMinn
• Phyllis Frankl
• Robert Hierons
• Satish Chandra
• Sebastian Elbaum
• Sriram Rajamani
• T. Y. Chen
• Tom Ostrand
• Wes Masri
• Willem Visser
• Yves Le Traon
Thanks for attending this ACM SIGSOFT Webinar!

Software Testing: A Research Travelogue

• Information on future and past SIGSOFT webinars is found at:
  • http://www.sigsoft.org/resources/webinars.html

• Dec 17 12PM EST: TBD
  • Speaker: Bram Adams, Polytechnique Montreal

• Jan 20, 12PM EST: TBD
  • Speaker: Gail Murphy, University of British Columbia