Reduce, Reuse, Recycle, Recover: Techniques for Improved Regression Testing

Mary Jean Harrold
ADVANCE Professor of Computing
College of Computing
Georgia Institute of Technology
ICSM—Many Firsts for Me

First paper—CSM 1988
An Incremental Approach to Unit Testing During Maintenance (with M. L. Soffa)
General Chair: Bob Arnold
Program Chair: Wilma Osborne
Session Chair: Tom Pigoski
Location: Phoenix, AZ

Other papers in my session
Post-maintenance Testing Based on Path Change Analysis
  (Benedusi, Cimitile, De Carlini)
Approaches to Regression Testing
  (Hartmann, Robson)

ICSM—Many Firsts for Me

First program committee—CSM 1991
General Chairs: John Munson, Roberto Ciampoli
Program Chairs: Malcolm Munro, Vaclav Rajlich
Location: Sorrento, Italy
ICSM—Many Firsts for Me

First Ph.D. student’s first paper—ICSM 1993
A safe, efficient algorithm for regression test selection (Gregg Rothermel)

General Chair: Marc Kellner
Program Chair: David Card
Session Chair: Mari Georges
Location: Montreal, Quebec, Canada

ICSM—Many Firsts for Me

First program committee chair—ICSM 1997

General Chair: Aniello Cimitili
Program Chairs: Mary Jean Harrold, Giuseppe Visaggio
Location: Bari, Italy
Regression Testing at ICSM

Maximum: 9 papers in 2009
Minimum: 1 paper in 2000
Average: 9%

Aspects of Green Regression Testing

REDUCE
RECOVER
Regression Testing
REUSE
RECYCLE
REDUCE in Green Regression Testing

Regression Testing

RECOVER

RECYCLE

Reuse RT Suite

Program P

P’ Version of P

Test Suite T

?
Papers at ICSM 2009

- Prioritizing JUnit Test Cases in Absence of Coverage Information, Zhang, Zhou, Hao, Zhang, Mei
  (Research Session 1: Tuesday)
- Prioritizing Component Compatibility Tests via User Preferences, Yoon, Sussman, Memon, Porter
  (Research Session 1: Tuesday)
Many companies automatically run RT suite

In 2002 OOPSLA talk, Bill Gates
• in-house tool—prioritizes regression tests
• “the impact had been very dramatic.”

In 2004 talk at Georgia Tech, Jim Gray
• prioritizing regression tests used at MS
• significant impact

Some Reuse Issues

Changes in function, interface, input, etc.
→ Obsolete test cases

Concurrency
→ Different results

Complex process
→ Difficult to integrate technique
REDUCE in Green Regression Testing

Regression Testing

RECOVER

REUSE

RECYCLE

REDUCE in Green Regression Testing

Program P

P' Version of P

Test Suite T

Test Suite T
REDUCE Select Subset of RT Suite

Program P

P' Version of P

T-T'

T'

T

Agrawal, Benedusi, Binkley, Briand, Chen, Cimitile, De Carlini, Dias, Do, Frankl, Grama, Gupta, Harrold, Hartmann, He, Horgan, Jagannathan, Krauser, Labiche, Leung, London, Muccini, Notkin, O'Sullivan, Porter, Ramanathan, Richardson, Robson, Rosenblum, Rothermel, Soffa, Vo, Vokolos, White, Xie

REDUCE Test Selection in Practice

Motorola

- Using regression-test selection; in-house tool

In 2002 OOPSLA talk, Bill Gates

- Using regression-test selection; in-house tool

ABB

- Used Firewall Technology
- Applied to 18 projects (16 procedural, 2 OO)
- Required 40% fewer test cases but found all faults of previous test suite
- Reports that ABB using approach worldwide

Jabar, Rajlich, Robinson, Smiley, White, Williams, Zheng
Reduce Size of RT Suite

Program P

P’ Version of P

Test Suite T

Agrawal, Gupta, Horgan, Harrold, Jones, London, McMaster, Memon, Soffa, Wong
Reduce Issues

Techniques assume
- No obsolete test cases
- Deterministic behavior
- Non-distributive

Transition to practice
- Validate real systems
- Integrate in process
- Assess trade offs
- Tools, infrastructure

RECYCLE in Green Regression Testing
Adapt RT Techniques

- Kinds of programs—e.g., object-oriented
- Phases of software—e.g., software architecture, UML
- Applications—e.g., Web-based, GUI, databases, spreadsheets

P \arrow{LT} T-T' \arrow{LT} T'

P' \arrow{LT} T'

Abdulla, Almezen, Andreas, Burgermeister, Gao, Grasso, Harrold, Hsia, Jones, Kaiser, Kung, Li, Memon, Orso, Perry, Ren, Robinson, Ryder, Sastry, Shah, Shi, Sinha, Smiley, Soffa, Tip, White, Williams, Xie, Ziegler, Zhao, Zheng

Select Subset of RT Suite

Papers in ICSM 2009

- **Model-Driven Testing in Software Product Lines**, Lamancha, Polo, García (Research Session 3: Wednesday)
- **Regression Model Checking**, Yang, Dwyer, Rothermel (Research Session 4: Wednesday)
Create Test Cases From ...

Program P

Executions
- Record executions
  - User interactions
  - Inputs, outputs at subsystem, unit, etc.
- Create test cases from recorded information

Artzi, Dwyer, Elbaum, Ernst, Orso, Saff
Create Test Cases From …

Papers in ICSM 2009

- Using Dynamic Execution Data to Generate Test Cases, Dara, Li, Liu, Ghorbani, Tahvildari
  (Industry Session 4: Thursday)

Recycle Issues

- New kinds of systems, classes of faults—e.g.,
  - Parallel, distributed, dynamic, component-based
  - Security patches
- Recycling of test cases, scripts,
- Better use of executions or field data for generating tests
RECOVER in Green Regression Testing

RECOVER in Green Regression Testing

Repair Test Cases

- Fix and repair
- Obsolete test cases
- Oracles,
- Scripts
- Etc.

Memon, Soffa
Papers in ICSM 2009

- Experimental Assessment of Manual Versus Tool-Based Maintenance of GUI-Directed Test Scripts, Grechanik, Xie, Fu

(Research Session 1: Tuesday)
How well do T, T', T'' or any test suites exercise P' with respect to changes?

Solution

Change-impact criterion

- assess adequacy
- create new test cases

Anand, Apiwattanapong, Chittimalli, Harrold, Orso, Park Santelices
Program Example
1. read X
2. read Y
3. if (X > 2)
   4. Y ++
   else
5. Y --
6. if (Y > 2)
7. print "1"
   else
8. print "-1"

Program and Control-flow Graph (CFG)

X, Y in [1,10] → 100 test cases

<table>
<thead>
<tr>
<th>Paths</th>
<th>Conditions</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: 1,2,3,4,6,7</td>
<td>X&gt;2, Y&gt;1</td>
<td>72</td>
</tr>
<tr>
<td>P2: 1,2,3,4,6,8</td>
<td>X&gt;2, Y≤1</td>
<td>8</td>
</tr>
<tr>
<td>P3: 1,2,3,5,6,7</td>
<td>X≤2, Y&gt;3</td>
<td>14</td>
</tr>
<tr>
<td>P4: 1,2,3,5,6,8</td>
<td>X≤2, Y≤3</td>
<td>6</td>
</tr>
</tbody>
</table>
Program and Control-flow Graph (CFG)

1. read X
2. read Y
3. X>2
4. Y++
5. Y--
6. Y>2
7. print “1”
8. print “-1”

Change-Impact Criterion

Want
- Test cases that reveal differences between P and P’
- Criterion that describes those test cases
  → Change-impact criterion
Want

- Test cases that reveal differences between P and P'
- Criterion that describes those test cases

→ Change-impact criterion

**Change-Impact Criterion**

Criterion for change-impact propagation (PIE model [Voas] for change propagation)

- Execution of the change
- Infection of the state after change
- Propagation of state to output where it can be observed

**Paths**

<table>
<thead>
<tr>
<th>Paths</th>
<th>Conditions</th>
<th>Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: 1,2,3,4,6,7</td>
<td>X&gt;2, Y&gt;1</td>
<td>72</td>
</tr>
<tr>
<td>P2: 1,2,3,4,6,8</td>
<td>X&gt;2, Y ≤1</td>
<td>8</td>
</tr>
<tr>
<td>P3: 1,2,3,5,6,7</td>
<td>X≤2, Y&gt;3</td>
<td>14</td>
</tr>
<tr>
<td>P4: 1,2,3,5,6,8</td>
<td>X≤2, Y ≤3</td>
<td>6</td>
</tr>
</tbody>
</table>

**Change-Impact Criterion**

1. read X
2. read Y
3. X>2
4. Y++
5. Y--
6. Y>2
7. print “1”
8. print “-1”
Change-Impact Criterion

To reveal difference, Y must be 2 or 3

Paths | Conditions | Tests
--- | --- | ---
P1: 1,2,3,4,6,7 | X>2, Y>1 | 72
P2: 1,2,3,4,6,8 | X>2, Y≤1 | 8
P3: 1,2,3,5,6,7 | X≤2, Y>3 | 14
P4: 1,2,3,5,6,8 | X≤2, Y≤3 | 6
Change-Impact Criterion

To reveal difference, Y must be 2 or 3

Paths | Conditions | Tests
--- | --- | ---
P1: 1,2,3,4,6,7 | X>2, Y>1 | 16/72
P2: 1,2,3,4,6,8 | X>2, Y≤1 | 0/8
P3: 1,2,3,5,6,7 | X≤2, Y>3 | 0/14
P4: 1,2,3,5,6,8 | X≤2, Y≤3 | 4/6

Existing Change-Impact Criteria

20/ 100 test cases
propagate difference

Criterion
• Random
  20% chance of finding fault
### Existing Change-Impact Criteria

**Coverage Criteria**

- Change or statement
  - $1 \rightarrow 20\%$

**Criterion for change-impact propagation** (PIE model for change propagation)

- Random
  - $20\%$ chance of finding fault

**Coverage Criteria**

- Change or statement
  - $1 \rightarrow 20\%$

<table>
<thead>
<tr>
<th>1</th>
<th>read X</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>read Y</td>
</tr>
<tr>
<td>3</td>
<td>$X &gt; 2$</td>
</tr>
<tr>
<td>4</td>
<td>$Y++$</td>
</tr>
<tr>
<td>5</td>
<td>$Y--$</td>
</tr>
<tr>
<td>6</td>
<td>$Y &gt; 2$</td>
</tr>
<tr>
<td>7</td>
<td>print “1”</td>
</tr>
<tr>
<td>8</td>
<td>print “-1”</td>
</tr>
</tbody>
</table>

**Propagate difference**

- 20/100 test cases
Existing Change-Impact Criteria

Criterion for change-impact propagation (PIE model for change propagation)

- Execution of the change
- 20/100 test cases propagate difference

Coverage Criteria
- Change or statement
  1 $\rightarrow$ 20%

Existing Change-Impact Criteria

20/100 test cases propagate difference

Coverage Criteria
- DU-Pairs

<table>
<thead>
<tr>
<th>DU-Pairs</th>
<th>Set P1, P2</th>
<th>Set P3, P4</th>
<th>Set P1, P3</th>
<th>Set P2, P4</th>
<th>Set P1</th>
<th>Set P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,(3,4),X), (2,4,Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,(3,5),X), (2,5,Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,(6,7),Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,(6,8),Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4,(6,7),Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4,(6,8),Y)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Existing Change-Impact Criteria

20/100 test cases
propagate difference

Coverage Criteria
• DU-Pairs

<table>
<thead>
<tr>
<th>DU-Pairs</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,(3,4),X), (2,4,Y)</td>
<td>P1, P2</td>
</tr>
<tr>
<td>(1,(3,5),X), (2,5,Y)</td>
<td>P3, P4</td>
</tr>
<tr>
<td>(2,(6,7),Y)</td>
<td>P1, P3</td>
</tr>
<tr>
<td>(2,(6,8),Y)</td>
<td>P2, P4</td>
</tr>
<tr>
<td>(4,(6,7),Y)</td>
<td>P1</td>
</tr>
<tr>
<td>(4,(6,8),Y)</td>
<td>P2</td>
</tr>
</tbody>
</table>

print “1”

Existing Change-Impact Criteria

20/100 test cases
propagate difference

Coverage Criteria
• DU-Pairs

<table>
<thead>
<tr>
<th>DU-Pairs</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,(3,4),X), (2,4,Y)</td>
<td>P1, P2</td>
</tr>
<tr>
<td>(1,(3,5),X), (2,5,Y)</td>
<td>P3, P4</td>
</tr>
<tr>
<td>(2,(6,7),Y)</td>
<td>P1, P3</td>
</tr>
<tr>
<td>(2,(6,8),Y)</td>
<td>P2, P4</td>
</tr>
<tr>
<td>(4,(6,7),Y)</td>
<td>P1</td>
</tr>
<tr>
<td>(4,(6,8),Y)</td>
<td>P2</td>
</tr>
</tbody>
</table>
Existing Change-Impact Criteria

20/100 test cases propagate difference

Coverage Criteria

• DU-Pairs

<table>
<thead>
<tr>
<th>DU-Pairs</th>
<th>P1, P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,(3,4),X), (2,4,Y)</td>
<td></td>
</tr>
<tr>
<td>(1,(3,5),X), (2,5,Y)</td>
<td></td>
</tr>
<tr>
<td>(2,(6,7),Y)</td>
<td></td>
</tr>
<tr>
<td>(2,(6,8),Y)</td>
<td></td>
</tr>
<tr>
<td>(4,(6,7),Y)</td>
<td></td>
</tr>
<tr>
<td>(4,(6,8),Y)</td>
<td></td>
</tr>
</tbody>
</table>

Existing Change-Impact Criteria

Criterion for change-impact propagation (PIE model for change propagation)

• Execution of the change
  
  • •

• Infection of the state
  
  • Propagation of state to output where it can be observed

<table>
<thead>
<tr>
<th>DU-Pairs</th>
<th>P1, P2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,(3,4),X), (2,4,Y)</td>
<td></td>
</tr>
<tr>
<td>(1,(3,5),X), (2,5,Y)</td>
<td></td>
</tr>
<tr>
<td>(2,(6,7),Y)</td>
<td></td>
</tr>
<tr>
<td>(2,(6,8),Y)</td>
<td></td>
</tr>
<tr>
<td>(4,(6,7),Y)</td>
<td></td>
</tr>
<tr>
<td>(4,(6,8),Y)</td>
<td></td>
</tr>
</tbody>
</table>
1: Change-Impact Criterion

20/100 test cases
propagate difference

Coverage Criteria
• Dependence chains
(all paths)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: 1,2,3,4,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2: 1,2,3,4,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3: 1,2,3,5,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4: 1,2,3,5,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1: Change-Impact Criterion

20/100 test cases
propagate difference

Coverage Criteria
• Dependence chains
(all paths)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: 1,2,3,4,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2: 1,2,3,4,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3: 1,2,3,5,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4: 1,2,3,5,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1: Change-Impact Criterion

20/100 test cases
propagate difference

Coverage Criteria
- Dependence chains
  (all paths)

<table>
<thead>
<tr>
<th></th>
<th>Coverage Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1,2,3,4,6,7</td>
</tr>
<tr>
<td>P2</td>
<td>1,2,3,4,6,8</td>
</tr>
<tr>
<td>P3</td>
<td>1,2,3,5,6,7</td>
</tr>
<tr>
<td>P4</td>
<td>1,2,3,5,6,8</td>
</tr>
</tbody>
</table>

P1: 1,2,3,4,6,7 16/72
P2: 1,2,3,4,6,8 0/8
P3: 1,2,3,5,6,7 0/14
P4: 1,2,3,5,6,8 4/6
1: Change-Impact Criterion

Criterion for change-impact propagation (PIE model for change propagation)

- Execution of the change
- Propagation of state to a point where it can be observed

20/100 test cases propagate difference

Coverage Criteria

- Dependence chains (all paths)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>1,2,3,4,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>1,2,3,4,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>1,2,3,5,6,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>1,2,3,5,6,8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2: Change-Impact Criterion

S3

Infection:
Path condition in P after S3 and path condition in P' after S3' differ

S4

Infection:
Value of state after execution of S4 in P and S4' in P' must differ
2: Change-Impact Criterion

Propagation

<table>
<thead>
<tr>
<th>PC</th>
<th>SS(n)</th>
<th>PC'</th>
<th>SS'(n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>X₀  Y₀</td>
<td>true</td>
<td>X₀  Y₀</td>
</tr>
</tbody>
</table>

PC—path condition
SS—symbolic state
Perform symbolic execution from before change to get conditions

Resulting conditions that show differences

C1: \( Y₀ + 1 > 2 \) and \( Y₀ - 1 \leq 2 \)
\( Y₀ \) in \{2,3\}

C2: \( Y₀ + 1 \leq 2 \) and \( Y₀ - 1 > 2 \)
\( Y₀ \) in \{\}
2: Change-Impact Criterion

But

- symbolic execution on the entire program is expensive
- may not scale to large programs
- etc.

Technique has two ways to improve efficiency

Partial Symbolic Execution

1. Begins immediately before change
   - Computes conditions in terms of variables immediately before change
   - Avoids symbolic execution from beginning
   → No need to solve conditions

2. For some distance—but not to output
   - Computes conditions on states at intermediate points (i.e., distances)
   - Bounds depth using slicing-like dependences, avoids symbolic execution to outputs
   → Greater distances improve confidence
How well do T, T', T'' or any test suites exercise P' with respect to changes?

Solution
Change-impact criterion
- assess adequacy
- create new test cases
Assess Adequacy

Program Example’
1. read X
2. read Y
3. if (X ≤ 2)
   4. Y ++
      else
   5. Y --
6. if (Y > 2)
7.   print “1”
      else
8.   print “-1”

To record adequacy (coverage of conditions)
Instrument modified program so that probes check for satisfaction of condition before change (e.g., before S3’)

Empirical Study

Goal: Evaluate effectiveness of criteria

Subjects: Tcas (6 versions) and NanoXML (9 versions)

Criteria: Statement: changed statements
          Data: chains of data-only dependencies
          T1: chains of data/control dependencies
          T2: Technique 1 with infection conditions

Procedure:
• Generated 100 test suites per criterion
• Considered different distances of chains
• Computed % of test suites revealing output difference
Recent Improvements

- Partial symbolic execution applied to multiple paths together
  - improve effectiveness (much greater distances, improved confidence)
  - improve efficiency same distance (order of magnitude reduction in time)
- Handling multiple changes
  - existing techniques too imprecise
  - can determine when multiple changes interact

Augment Test Suite

How well do T, T’, T” or any test suites exercise P’ with respect to changes?

Solution
Change-impact criterion
- assess adequacy
- create new test cases

For more details, see Santelices, et al. ASE 2008
Augment Test Suite

How well do T, T’, T” or any test suites exercise P’ with respect to changes?

Solution
Change-impact criterion
• assess adequacy
• create new test cases

Create New Test Cases

For each unsatisfied condition C
Consider t in T”’
• Symbolically execute t’s path
• Change path to satisfy conditions
Until new t created for C
Add new t to T”’
Preliminary Results

Improvements to dynamic symbolic execution using static analysis
- able to generate test cases for realistic programs that use libraries
- able to improve efficiency because of better analysis (e.g., pointer)
- able to generate new test cases in many cases from T’”

Related Techniques

Guided Path Exploration for Regression Test Generation, Taneja, Xie, Tillmann, de Halleux, Schulte (ICSE 2009)
- Generates test cases to changes
- Uses PIE model to guide paths to try
- Eliminates paths that don’t lead to change

BERT (BEhavioral RT), Orso, Xie (WODA 2009)
- Generates many test cases targeted at changes
- Differences outcomes of original, modified programs
- Filters out multiple test cases that exhibit differences due to the same cause
Contributions of This Work

New change test criteria
- Based on entities and state
- Can be used to
  - Assess adequacy of test suite with respect to changes
  - Automatically generate new test cases to satisfy unsatisfied conditions
- Can be performed on large programs—only area around change evaluated
- Can improve confidence and reduce risk in changed software

Recover Issues
- Maturation of current techniques
- New techniques for automatic fixing of test cases, generation of new test cases, etc.
Aspects of **Green** Regression Testing

**REDUCE**

Regression Testing

**REUSE**

**RECOVER**

**RECYCLE**

How **Green** is Regression Testing?

**RED**

Regression Testing

**REUS**

**RE**
How *Green* is Regression Testing?

Still much to be done!