A Revisit of the Integration of Metamorphic Testing and Test Suite Based Automated Program Repair

Mingyue Jiang 1,2, Tsong Yueh Chen 2, Fei-Ching Kuo 2, Zuohua Ding 1, Eun-Hye Choi 3, Osamu Mizuno 4

1 Zhejiang Sci-Tech University, China
2 Swinburne University of Technology, Australia
3 National Institute of Advanced Industrial Science and Technology (AIST), Japan
4 Kyoto Institute of Technology, Kyoto, Japan
Automatic Program Repair (APR)

Program Under Repair (PUR)

Test Suite
Failing test cases
Passing test cases

Ex.
int Min(int x, int y){
  if(x<=y-2) //fault
    return x;
  else
    return y;
}

APR

Repair
passing all test cases

Test oracle is necessary.

<table>
<thead>
<tr>
<th>Inputs (x,y)</th>
<th>Expected Output</th>
<th>Pass/ Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁ (8,1)</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>t₂ (0,5)</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>t₃ (1,2)</td>
<td>1</td>
<td>Fail</td>
</tr>
</tbody>
</table>

int Min(int x, int y){
  if(x<=y)
    return x;
  else
    return y;
}
Metamorphic Testing (MT)

To identify the correctness of test outputs, use metamorphic relations (MRs) instead of test oracle.

```c
int Min(int x, int y){
    if(x<=y-2) //fault
        return x;
    else
        return y;
}
```

**Min**

**MR**

\[ \text{Min}(a,b) = \text{Min}(b,a) \]

<table>
<thead>
<tr>
<th>Metamorphic test group (MTG)</th>
<th>Conditions to be satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_{tg1} ) ((8,1),(1,8))</td>
<td>( \text{Min}(8,1) = \text{Min}(1,8) )</td>
</tr>
<tr>
<td>( m_{tg2} ) ((0,5),(5,0))</td>
<td>( \text{Min}(0,5) = \text{Min}(5,0) )</td>
</tr>
<tr>
<td>( m_{tg3} ) ((1,2),(2,1))</td>
<td>( \text{Min}(1,2) = \text{Min}(2,1) )</td>
</tr>
</tbody>
</table>
To identify the correctness of test outputs, use **metamorphic relations** (MRs) instead of test oracle.

Ex.

```cpp
int Min(int x, int y) {
    if(x<=y-2) return x;
    else return y;
}
```

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Expected Output</th>
<th>Pass/Fail</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8,1)</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td>(0,5)</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td>(1,2)</td>
<td>1</td>
<td>Fail</td>
</tr>
</tbody>
</table>

**Test oracle**

**MR**

Min(a,b) = Min(b,a)

<table>
<thead>
<tr>
<th>MTG</th>
<th>MR is</th>
</tr>
</thead>
<tbody>
<tr>
<td>((8,1),(1,8))</td>
<td>non-violated</td>
</tr>
<tr>
<td>((0,5),(5,0))</td>
<td>non-violated</td>
</tr>
<tr>
<td>((1,2),(2,1))</td>
<td>violated</td>
</tr>
</tbody>
</table>

Min(1,2)=2  
Min(2,1)=1
**Test Suite**

- **Failing test cases**
- **Passing test cases**

**APR + MT**

**Program Under Repair (PUR)**

- Metamorphic test groups
  - Violating ones
  - Non-violating ones

**APR-MT**

- Repair satisfying MR for all MTGs

**Use MR.**

**APR**

**Program Under Repair (PUR)**

- Test Suite
  - Failing test cases
  - Passing test cases

**Use test oracle.**

**Repair passing all test cases**
APR + MT

Program Under Repair (PUR)

Metamorphic test groups
violating ones
non-violating ones

Use MR.

APR-MT

Repair satisfying MR for all MTGs

Ex.

int Min(int x, int y){
  if(x<=y-2) //fault
    return x;
  else
    return y;
}

Min(a,b) = Min(b,a)

<table>
<thead>
<tr>
<th>MTG</th>
<th>MR is</th>
</tr>
</thead>
<tbody>
<tr>
<td>((8,1),(1,8))</td>
<td>non-violated</td>
</tr>
<tr>
<td>((0,5),(5,0))</td>
<td>non-violated</td>
</tr>
<tr>
<td>((1,2),(2,1))</td>
<td>violated</td>
</tr>
</tbody>
</table>

int Min(int x, int y){
  if(x<=y)
    return x;
  else
    return y;
}
APR + MT

- We apply MT to a semantics based APR technique.
  - CETI-MT = CETI [Nguyen+17] + MT
  - MT is applied to a generate-and-validate APR technique [Jiang+17].
  - GenProg-MT = GenProg [Forrest+09] + MT

PUR variants

Repair passing all test cases
↓
satisfying all MTGs
1. Fault localization*. 

```
int Min(int x, int y, int uk_0){
    if(x<=2) //fault
        return x;
    else
        return y;
}
```

*tarantula[Jones+05].
1. Fault localization.

2. Make a parameterized statement with a suspicious statement using a repair template*.

*constant, operator, etc.
CETI: Semantics based APR

1. Fault localization.
2. Generate a parameterized statement.
3. Generate a reachability instance program.

<table>
<thead>
<tr>
<th>Input</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8,1)</td>
<td>1</td>
</tr>
<tr>
<td>(0,5)</td>
<td>0</td>
</tr>
<tr>
<td>(1,2)</td>
<td>1</td>
</tr>
</tbody>
</table>

Precondition = passing all test cases.

```cpp
int Min(int x, int y, int uk_0){
    if(x<=y-uk_0) //fault
        return x;
    else
        return y;
}

void main(){
    int uk_0;
    if( Min(8, 1, uk_0) == 1 &&
        Min(0, 5, uk_0) == 0 &&
        Min(1, 2, uk_0) == 1) {
        Location L;
    }
}
```
CETI: Semantics based APR

1. Fault localization.
2. Generate a parameterized statement.
3. Generate a reachability instance program.

4. Explore values that make L reachable, i.e. construct a repair.

```cpp
cdef int Min(int x, int y, int uk_0):
    if(x<=y-uk_0) //fault
        return x;
    else
        return y;

cdef void main():
    int uk_0;
    if( Min(8, 1, uk_0) == 1 &&
        Min(0, 5, uk_0) == 0 &&
        Min(1, 2, uk_0) == 1) {
            Location L;
    }
```
```c
int Min(int x, int y, int uk_0){
    if(x<=y-uk_0) //fault
        return x;
    else
        return y;
}

int MR1Checker(int a, int b) {
    if(a == b) return 1;
    else return 0;
}

void main(){
    int uk_0;
    /* Apply the MTG set. */
    if( MR1Checker(Min(8, 1, uk_0), Min(1, 8, uk_0)) == 1 &&
       MR1Checker(Min(0, 5, uk_0), Min(5, 0, uk_0)) == 1 &&
       MR1Checker(Min(1, 2, uk_0), Min(2, 1, uk_0)) == 1 ) {
        Location L;
    }
}
```

**Precondition**
- satisfying MR for all MTGs

---

**Conditions to be satisfied**

<table>
<thead>
<tr>
<th>MTG</th>
<th>Conditions to be satisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>((8,1),(1,8))</td>
<td>Min(8,1) = Min(1,8)</td>
</tr>
<tr>
<td>((0,5),(5,0))</td>
<td>Min(0,5) = Min(5,0)</td>
</tr>
<tr>
<td>((1,2),(2,1))</td>
<td>Min(1,2) = Min(2,1)</td>
</tr>
</tbody>
</table>

---

**CETI-MT**
CETI-MT

<table>
<thead>
<tr>
<th>Inputs</th>
<th>CETI</th>
<th>CETI-MT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A PUR</td>
<td>A PUR</td>
<td>A set of MTGs (violating &amp; non-violating test inputs for MRs)</td>
</tr>
<tr>
<td>A test suite (passing &amp; failing test cases)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Repair Process</td>
<td>Use information of the test suite.</td>
<td>Use information of the MTGs.</td>
</tr>
<tr>
<td>Output</td>
<td>A repair passing all test cases or null.</td>
<td>A repair satisfying all MTGs or null.</td>
</tr>
</tbody>
</table>
### Experiments

Using 6 IntroClass benchmark programs in C [Goues+15].
Totally, 1143 versions.

<table>
<thead>
<tr>
<th>Subject programs.</th>
<th># versions</th>
<th>LOC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checksum</td>
<td>69</td>
<td>max: 45  min: 15 avg: 27</td>
<td>Computing the sum of a string</td>
</tr>
<tr>
<td>digits</td>
<td>236</td>
<td>max: 190 min: 15 avg: 40</td>
<td>Listing all digits of an integer</td>
</tr>
<tr>
<td>grade</td>
<td>268</td>
<td>max: 53  min: 18 avg: 29</td>
<td>Determining the grade of a score</td>
</tr>
<tr>
<td>median</td>
<td>232</td>
<td>max: 62  min: 13 avg: 24</td>
<td>Computing the median of three integers</td>
</tr>
<tr>
<td>smallest</td>
<td>177</td>
<td>max: 51  min: 17 avg: 25</td>
<td>Computing the smallest of four integers</td>
</tr>
<tr>
<td>syllables</td>
<td>161</td>
<td>max: 58  min: 12 avg: 29</td>
<td>Counting vowel characters of a string</td>
</tr>
</tbody>
</table>
We compare the success rate and the repair quality.
## Result: Success Rate

# of program versions successfully repaired / # of all program versions

<table>
<thead>
<tr>
<th></th>
<th>CETI</th>
<th>CETI-MT</th>
<th>CETI</th>
<th>CETI-MT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CE-BTS</td>
<td>MCE-BTG1</td>
<td>MCE-BTG2</td>
<td>CE-WTS</td>
</tr>
<tr>
<td>checksum</td>
<td>0.313</td>
<td>0.000</td>
<td>0.100</td>
<td>0.132</td>
</tr>
<tr>
<td>digits</td>
<td>0.122</td>
<td>0.457</td>
<td>0.133</td>
<td>0.217</td>
</tr>
<tr>
<td>grade</td>
<td>0.521</td>
<td>0.521</td>
<td>0.582</td>
<td>0.534</td>
</tr>
<tr>
<td>median</td>
<td>0.958</td>
<td>1.000</td>
<td>1.000</td>
<td>0.993</td>
</tr>
<tr>
<td>smallest</td>
<td>0.832</td>
<td>0.733</td>
<td>0.750</td>
<td>1.000</td>
</tr>
<tr>
<td>syllables</td>
<td>0.318</td>
<td>0.500</td>
<td>0.727</td>
<td>0.067</td>
</tr>
<tr>
<td>Total</td>
<td>0.636</td>
<td>0.657</td>
<td>0.659</td>
<td>0.626</td>
</tr>
</tbody>
</table>

→ CETI-MT is comparable to CETI.
Result: Repair Quality

• **Mann-Whitney U Test** [Wilcoxon1945]:
  to verify whether CETI and CETI-MT are different with p=0.05.

• $\hat{A}_{12}$ **statistic** [Arcuri+11, Vargha&Delaney2000]:
  to measure the probability that program repairs by CETI are of high quality than those by CETI-MT.
  
  • $\hat{A}_{12} > 0.56$: CETI is better.
  • $\hat{A}_{12} < 0.44$: CETI-MT is better.
  • Others: similar.
Result: Repair Quality

- For 5 cases, CETI-MT is better.
- For 5 cases, both are similar.
- For 2 cases, CETI is better.

CETI-MT is comparable to CETI.

<table>
<thead>
<tr>
<th>Eval. Data</th>
<th>CE-BTS vs MCE-BTGI</th>
<th>CE-BTS vs MCE-BTGI</th>
<th>CE-WTS vs MCE-WTG1</th>
<th>CE-WTS vs MCE-WTG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_w$</td>
<td>$p &lt; 0.05$</td>
<td>$p &lt; 0.05$</td>
<td>$p=0.4345$</td>
<td>$p &lt; 0.05$</td>
</tr>
<tr>
<td></td>
<td>$\hat{A}_{12} = 0.436$</td>
<td>$\hat{A}_{12} = 0.544$</td>
<td>$\hat{A}_{12} = 0.483$</td>
<td>$\hat{A}_{12} = 0.565$</td>
</tr>
<tr>
<td>CETI-MT is better</td>
<td>similar</td>
<td>CETI is better</td>
<td>CETI-MT is better</td>
<td>similar</td>
</tr>
<tr>
<td>$M_w^1$</td>
<td>$p &lt; 0.05$</td>
<td>$p &lt; 0.05$</td>
<td>$p &lt; 0.05$</td>
<td>$p=0.090$</td>
</tr>
<tr>
<td></td>
<td>$\hat{A}_{12} = 0.380$</td>
<td>$\hat{A}_{12} = 0.566$</td>
<td>$\hat{A}_{12} = 0.370$</td>
<td>$\hat{A}_{12} = 0.473$</td>
</tr>
<tr>
<td>CETI-MT is better</td>
<td>CETI is better</td>
<td>CETI-MT is better</td>
<td>similar</td>
<td>similar</td>
</tr>
<tr>
<td>$M_w^2$</td>
<td>$p &lt; 0.05$</td>
<td>$p &lt; 0.05$</td>
<td>$p=0.648$</td>
<td>$p &lt; 0.05$</td>
</tr>
<tr>
<td></td>
<td>$\hat{A}_{12} = 0.377$</td>
<td>$\hat{A}_{12} = 0.344$</td>
<td>$\hat{A}_{12} = 0.492$</td>
<td>$\hat{A}_{12} = 0.462$</td>
</tr>
<tr>
<td>CETI-MT is better</td>
<td>CETI-MT is better</td>
<td>similar</td>
<td>similar</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

• We applied MT to a semantics based APR, CETI, and investigated its effectiveness.

• A test oracle is stronger than an MR. Nevertheless, CETI-MT shows comparable repair effectiveness to CETI.

• Two major factors affecting CETI-MT: MRs and test cases.

• Challenges: Automatic identification of MRs. Investigating a way of CETI-MT.