μMT: A Data Mutation Directed Metamorphic Relation Acquisition Methodology

Chang-ai Sun¹, Yiqiang Liu¹, Zuoyi Wang¹, W.K. Chan²

¹ University of Science and Technology Beijing, China
² City University of Hong Kong, China
Background and Motivation
- Oracle problem, metamorphic testing, data mutation
- Motivation

Our Approach: $\mu$ MT
- Principle, mapping rules, supporting tool

Evaluation
- Experimental design, results & analysis

Conclusion
Software testing presents a **practical approach**, including:

- Test cases generation
- Execution of test cases
- Comparison of actual outputs against the expected ones.

**The Oracle Problem** [E.J. Weyuker 1982; Antonia Bertolino 2007]

- In some situation, it is **impossible** or **practically too difficult** to decide whether the program outputs on test cases are correct.
- Most of existing testing techniques assume the existence of oracle. However, in some situations, tester are unable to determine whether the output is correct (**namely the oracle does not exist**).

**Example: sine function**

\[
\begin{align*}
sin(30^\circ) &= 0.5 \\
sin(30.5^\circ) &= ?
\end{align*}
\]
Metamorphic Testing (MT) [Chen et al., Technical Report’98]

- Can we effectively and efficiently test a program even without oracles?

- **Metamorphic Relation** (MR) denoted as \( \{R, R_f\} \): a kind of inherent properties of program under test. E.g. \( \sin(x) = \sin(x + 360^\circ) \)

**MR: \( \sin(x) = \sin(x + 360^\circ) \)**

- Source test case: 29.8°
- Follow-up test case: 389.8°
- Result verification: \( \sin(29.8^\circ) = \sin(389.8^\circ) \) ?
MR Acquisition

- MT has been observed as an simple yet effective way to alleviate the oracle problem. [Chen, SOSE’10]

- MR plays a crucial role in MT, which is used to generate follow-up test cases and verify test results.
  - Identifying MRs is a difficult yet crucial task, which usually involves extensive manual work.

- Related approaches on MR acquisition:
  - Machine learning approach [Kanewala and Bieman, ISSRE’13]
  - Search-based approach [Zhang et al. ASE’14]
  - Category-choice framework-based approach [Xie et al. JSS 2015]
  - MR composition method [Liu et al. QSIC’12]
### Data Mutation [Shan and Zhu, Computer Journal 2009]

- **Mutation Testing**
  - Mutation operator injects an fault into the program
  - “original program” → “mutant”

- **Data Mutation**
  - Data mutation operator executes a transformation to test data
  - “seed test case” → “mutant”

```plaintext
if (a && b) 
c = 1;
else 
c = 0;
```

```plaintext
if (a || b) 
c = 1;
else 
c = 0;
```

\[ tc_0 = (x, y) \]

- **mutation operator**

\[ tc_1 = (x, y) \]

- **seed test case**

- **data mutation operator**

- **mutant**
By comparing MT and DM, we have the two observations:

- Both MT and DM involve generating more test cases based on an original set of test cases
- The input relation $R$ in MRs is used for generating the follow-up test cases from the source test cases;
- DMOs are used to generate the mutant test cases from the seed test cases.

- Both the input relation in an MR and DMOs are certain kinds of transformational operation (denoted as $g$) on the inputs: $x' = g(x)$.

Can we leverage DM to alleviate the MR acquisition problem?

- Report an exploratory study, and present a DM-directed MR acquisition methodology $\mu$ MT
- Develop a tool to facilitate the application of $\mu$ MT
Our approach

**Overview of \( \mu \) MT**

1. **Seed Input:** \( x \) → **DMO** \( f \) → **Mutant Input:** \( x' = f(x) \)

   **Mapping rule**

   - Both can be used to generate test cases
   - Both can be express as a function

2. **Source Test Case:** \( x \)
   - Output of \( x \): \( y \)
   - Input Relation \( R \)
   - Follow-up test case: \( x' = R(x) \)

   \( \langle y, y' \rangle \in R_f \) (Output relation)?

   **Output of \( x' \):** \( y' \)

**DMO**

- Data mutation

- Test case generation
- Test execution
- Test result verification

"Overview of \( \mu \) MT"
Our approach (cont.)

Main steps of $\mu$MT

- Specification of $P$
- Source Test Case $T$
- Follow-up Test Case $T'$
- Mapping rules of DMO to MR
- Designing Data Mutation Operators (DMOs)
- Selecting DMOs
- Data Mutation (R)
- Verifying mutated test data
- Generating MRs
- Verifying derived MRs
### Mapping Rules

<table>
<thead>
<tr>
<th>Data Mutation Operator</th>
<th>Metamorphic Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap($a, b$)</td>
<td>$f(\text{Swap}(a, b)) = f(a, b)$</td>
</tr>
<tr>
<td>Inc($a$)</td>
<td>$f(\text{Inc}(a)) &gt; f(a)$</td>
</tr>
<tr>
<td>Dec($a$)</td>
<td>$f(\text{Dec}(a)) &lt; f(a)$</td>
</tr>
<tr>
<td>Double($a$)</td>
<td>$f(\text{Double}(a)) = k*f(a); \text{ or } f(\text{Double}(a)) &gt; f(a)$</td>
</tr>
<tr>
<td>Halve($a$)</td>
<td>$f(\text{Halve}(a)) = k*f(a); \text{ or } f(\text{Halve}(a)) &lt; f(a)$</td>
</tr>
<tr>
<td>IncT($a$)</td>
<td>$f(\text{IncT}(a)) = f(a)$</td>
</tr>
<tr>
<td>Neg($a$)</td>
<td>$f(\text{Neg}(a)) = f(a), \text{ or } f(\text{Neg}(a)) = -f(a)$</td>
</tr>
</tbody>
</table>
Our approach (cont.)

- **Mapping Rules (cont.)**

**Swap data mutation operator:**

- Swap the values of $x_i$ and $x_j$, where $0 < i \neq j \leq n$, which can be expressed as: $\text{swap}(x_i, x_j): x_i, x_j \rightarrow x_j, x_i$;

- Mapping MR: $f(x_1, \ldots, \text{swap}(x_i, x_j), \ldots, x_n) = f(x_1, \ldots, x_n);

*For a triangle area calculation program:*

- apply the swap data mutation operator to generate a MR:

  - $R: (x', y', z') \rightarrow (y, x, z)$,

  - $R_f: S' = S$. 
MT4WS was developed to support MT in the following ways [Sun et al. IJHPCN 2016]:

- A user-friendly interface enabling interactive and efficient MR input
- Verification mechanisms to identify contradictions or conflicts among input MRs
- Automatic follow-up test case generation and test result verification
- A highly flexible and configurable control over the testing process

Recently, we have enhanced MT4WS with the following features:

- Automatic composition of MRs
- Iterative Metamorphic Testing (IMT)
- uMT (this work)
2. Our approach (cont.)

- **Architecture of extended MT4WS**
2. Our approach (cont.)

- A snapshot of extended MT4WS supporting $\mu$ MT

**MT4WS**

**Step 2: Define Metamorphic Relations (MRs)**

- Define relations between Source test case (Source) and Follow-up test case (Follow-up) in Table R

- Define relations between Source test outputs and Follow-up test outputs in Table R

**Options:**
- Define Metamorphic Relations manually
- Import from XML file
- Data mutation

1. Select data mutation operator: **MULT2x**
   - Input a value of K (K is the multiple of the outputs): 1
2. Select data mutation inputs: □ accountFrom □ accountTo □ mode □ amount
3. Select data mutation outputs: □ balanceDeltaFrom □ balanceDeltaTo
4. Rf Relation □ □

Add to DM set
2. Our approach (cont.)

- A snapshot of extended MT4WS supporting $\mu$ MT

**MT4WS**

**Step 5: Configuration**
- Confirm information
  - Web Service Name
  - WSDL URI
  - Operation
  - Test Suite
  - Metamorphic Relations (MRs)
- Configuration

b. Configure the test settings
c. Start

**MT4WS**

**Testing Report**

- **Web Service**: ATMService
- **WSDL URI**: http://localhost:8080/axis2/services/ATMService?wsdl
- **Testing Period**: 2016-05-09 10:18:34 - 2016-05-09 10:18:34

**Statistics Summary**
You have tested 1/9 (total) operations of the Web Service, 1 operations have fault(s)

<table>
<thead>
<tr>
<th>Web Service Operations</th>
<th>login, withdraw, deposit, transfer, queryBalance, changePassword, insertAccount, deleteAccount, queryAccount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation(s) tested</td>
<td>transfer</td>
</tr>
<tr>
<td>Operation(s) passed</td>
<td></td>
</tr>
<tr>
<td>Operation(s) failed</td>
<td>transfer</td>
</tr>
</tbody>
</table>

**Failed Operations Details**

<table>
<thead>
<tr>
<th>Faulty</th>
<th>MRs which Detected the Fault</th>
<th>Test Cases which Detected the Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>transfer</td>
<td>ATMService_transfer_1</td>
<td>TC1, TC2</td>
</tr>
</tbody>
</table>
3. Evaluation

Research Questions

1. To what extent $\mu$MT can derive MRs?

2. How effectively can the MRs derived by $\mu$MT detect faults?

3. Is there any correlation between the MRs derived via different data mutation operators and their fault detection effectiveness?
Experimental Design

- **Mutation Analysis** is widely used to assess the effectiveness of testing techniques.
  - Mutation Operator: Mimic typical programming errors
  - Mutant: A variant that are seeded with a mutation operator
  - “Kill” a mutant: Detect the fault injected in a mutant.

- **Mutation Score** is used to measure how thoroughly a test suite can kill the mutants, which is defined as follows:

\[
MS(p, ts) = \frac{N_k}{N_m - N_e}
\]

- \( p \): program being mutated
- \( ts \): test suite under evaluation
- \( N_k \): number of killed mutants
- \( N_m \): total number of mutants
- \( N_e \): number of equivalent mutants
### Subject programs and seeded faults

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
<th>Line of Code</th>
<th>Number of Seeded Faults</th>
<th>Applicable Mutation Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>Transfer operation of China Agriculture Bank</td>
<td>263</td>
<td>163</td>
<td>AORB, AOIU, AOIS, ROR, COR, COD, COI, LOI</td>
</tr>
<tr>
<td>BillCal</td>
<td>Monthly bill calculation of China Unicom</td>
<td>112</td>
<td>112</td>
<td>AORB, AOIU, AOIS, ROR, COR, COI, LOI</td>
</tr>
<tr>
<td>BaggBill</td>
<td>Checked baggage billing of Air China</td>
<td>215</td>
<td>67</td>
<td>AORB, AOIU, AOIS, ROR, COI, LOI</td>
</tr>
</tbody>
</table>

In the study, *MuJava* is employed to seed faults into the implementation.
### Automatic Teller Machine

#### Commission fee calculation

<table>
<thead>
<tr>
<th></th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charge Percentage</td>
<td>0%</td>
<td>0.5%</td>
<td>0.5%</td>
<td>1%</td>
</tr>
<tr>
<td>Min(¥)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max(¥)</td>
<td>0</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Limit per Transfer(¥)</td>
<td>50000</td>
<td>50000</td>
<td>50000</td>
<td>50000</td>
</tr>
</tbody>
</table>

Test this interface

Withdrawal
Deposit
Print

Automatic Teller Machine
**Input (A, B, P, M):**

- **A, B**: the sender and recipient account numbers
- **P**: the transfer type, *Value ranges from 0 to 3, corresponding to type I to IV in the commission fee calculation table*
- **M**: the amount of transfer transaction, *Ranging from 0 to 50000*

**Output (ΔA, ΔB)**

- **ΔA**: Difference between the balances of account A *before* transaction and *after* transaction
- **ΔB**: Difference between the balances of account B *after* transaction and *before* transaction
### Derived MRs for ATM using $\mu$ MT

<table>
<thead>
<tr>
<th>MR</th>
<th>DMO</th>
<th>Mapping Rule</th>
<th>$\mathbf{MR}$</th>
<th>$\mathbf{R}$</th>
<th>$\mathbf{R_f}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR1</td>
<td>Inc</td>
<td>$f(\text{Inc} (M), A, B, P) &gt; f(M), A, B, P)$</td>
<td>$M' = M+1$</td>
<td>$\triangle B' &gt; \triangle B$</td>
<td></td>
</tr>
<tr>
<td>MR2</td>
<td>Dec</td>
<td>$f(\text{Dec} (M), A, B, P) &lt; f(M), A, B, P)$</td>
<td>$M' = M-1$</td>
<td>$\triangle B' &lt; \triangle B$</td>
<td></td>
</tr>
<tr>
<td>MR3</td>
<td>Double</td>
<td>$f(\text{Double} (M), A, B, P) \geq k \times f(M), A, B, P)$</td>
<td>$M' = 2^*M$</td>
<td>$\triangle B' \geq 2^*B$</td>
<td></td>
</tr>
<tr>
<td>MR4</td>
<td>Halve</td>
<td>$f(\text{Halve} (M), A, B, P) \leq k \times f(M), A, B, P)$</td>
<td>$M' = 0.5^*M$</td>
<td>$\triangle B' \leq 0.5^*\triangle B$</td>
<td></td>
</tr>
</tbody>
</table>
3. Evaluation (cont.)

Results and Analysis

- $\mu$ MT demonstrates to be feasible

- Different MRs demonstrate varying mutation scores

- Inc and Dec have higher mutation scores than Double and Halve on BillCal and BaggBill but reversed on ATM

Table 5. Mutation scores on the ATM program

<table>
<thead>
<tr>
<th>DMO</th>
<th>MR</th>
<th>TC=10</th>
<th>TC=20</th>
<th>TC=50</th>
<th>TC=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inc</td>
<td>MR1</td>
<td>6.75</td>
<td>7.98</td>
<td>7.98</td>
<td>8.59</td>
</tr>
<tr>
<td>Dec</td>
<td>MR2</td>
<td>6.75</td>
<td>7.98</td>
<td>7.98</td>
<td>9.20</td>
</tr>
<tr>
<td>Double</td>
<td>MR3</td>
<td>14.11</td>
<td>15.34</td>
<td>16.56</td>
<td>17.79</td>
</tr>
<tr>
<td>Halve</td>
<td>MR4</td>
<td>14.11</td>
<td>15.95</td>
<td>16.56</td>
<td>16.56</td>
</tr>
</tbody>
</table>

Table 8. Average Mutation scores on the BillCal and BaggBill programs by DMO operators

<table>
<thead>
<tr>
<th></th>
<th>BillCal</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MS (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TC=1</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMO</td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>Inc</td>
<td>12.17</td>
<td>2.82</td>
<td>13.17</td>
<td>2.42</td>
<td>13.50</td>
<td>2.36</td>
</tr>
<tr>
<td>Dec</td>
<td>12.05</td>
<td>2.90</td>
<td>12.83</td>
<td>2.57</td>
<td>13.39</td>
<td>2.29</td>
</tr>
<tr>
<td>Double</td>
<td>5.13</td>
<td>0.41</td>
<td>6.36</td>
<td>0.57</td>
<td>6.36</td>
<td>0.57</td>
</tr>
<tr>
<td>Halve</td>
<td>5.25</td>
<td>1.11</td>
<td>6.47</td>
<td>0.92</td>
<td>6.70</td>
<td>0.67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>BaggBill</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MS (%)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TC=1</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMO</td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
<td>Mean</td>
<td>SD</td>
</tr>
<tr>
<td>Inc</td>
<td>12.17</td>
<td>2.82</td>
<td>13.17</td>
<td>2.42</td>
<td>13.50</td>
<td>2.36</td>
</tr>
<tr>
<td>Dec</td>
<td>12.05</td>
<td>2.90</td>
<td>12.83</td>
<td>2.57</td>
<td>13.39</td>
<td>2.29</td>
</tr>
<tr>
<td>Double</td>
<td>5.13</td>
<td>0.41</td>
<td>6.36</td>
<td>0.57</td>
<td>6.36</td>
<td>0.57</td>
</tr>
<tr>
<td>Halve</td>
<td>5.25</td>
<td>1.11</td>
<td>6.47</td>
<td>0.92</td>
<td>6.70</td>
<td>0.67</td>
</tr>
</tbody>
</table>
4. Conclusion

- We have proposed a methodology called $\mu$MT to deal with the MR acquisition issue:
  - Using data mutation to guide the MR acquisition
  - Developing a supporting tool for $\mu$MT through extending an existing tool called MT4WS
  - Conducting a preliminary evaluation on the feasibility and effectiveness of $\mu$MT

- For future work, we will explore the improvement of $\mu$MT through
  - Improving its efficiency through high code coverage and diversity
  - Conducting more extensive empirical evaluation with more programs and data mutation operators
Thanks for your attention!