RT-Selection

(A Regression Test Selection Technique Using Textual Differencing and Change Impact Analysis)

Eui-Sub Kim
Dependable Software Laboratory
KONKUK University
Contents

• Introduction

• RT-Selection
  – 7 phase

• Case study

• Conclusion
Introduction

- Some of questions and answer

- To seek to uncover new software bugs after changes have been made

What is regression testing?

Why regression testing cost is high?

How to reduce the cost of regression testing?
Introduction

• Some of questions and answer

**What** is regression testing?

**Why** regression testing cost is high?

**How** to reduce the cost of regression testing?

• To seek to uncover new software bugs after changes have been made

• The frequent changes
• The complexity and size of the modern software
Introduction

- Some of questions and answer

**What** is regression testing?

**Why** regression testing cost is high?

**How** to reduce the cost of regression testing?

- To seek to uncover new software bugs after changes have been made

- The frequent changes
- The complexity and size of the modern software

- To identify the changes
- To select a subset.

→ RT-Selection.
What is RT-Selection?

How to perform RT-Selection?
What is RT-Selection?

How to perform RT-Selection?

Overview of RT-Selection
1. Canonical Formatting

Overview of RT-Selection
1. Canonical Formatting

- **What** is canonical formatting?

  Each different style forms

  Style A  |  Style B

  ![Image of Style A and Style B]

  > Convert

  Canonical form

  ![Image of Canonical Form]

- **Why** convert into canonical formatting?
1. Canonical Formatting

• **What** is canonical formatting?

Each different style forms

Style A

... ... .....

Style B

... ...

Canonical form

convert

• **Why** convert into canonical formatting?

- To reduce unwanted result.

Blank
“(“, “{“ Comment
All of different coding styles so on
2. Textual Differencing and Change Analysis

Overview of RT-Selection
2. **Textual Differencing** and Change Analysis

- **What** is Textual differencing?
  - Comparison between two codes with line by line

- **Why** perform Textual differencing?

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

Old

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

New
2. **Textual Differencing** and Change Analysis

- **What** is Textual differencing?
  - Comparison between two codes with line by line

- **Why** perform Textual differencing?
  - To identify the changes

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

**Old**

**New**
2. Textual Differencing and Change Analysis

- **What** is change analysis?
  - To identify what elements is affected by changes

- **How** perform change analysis?

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

Change analysis List
[Calc(); 1_for_condition]
2. Textual Differencing and **Change Analysis**

- **What** is change analysis?
  - To identify what elements is affected by changes

- **How** perform change analysis?
  - Inspection
  - Inserting TAG

```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}

int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

Change analysis List
- `Calc(); 1_for_condition`

Old
```
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

New
```
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```
### TAG

<table>
<thead>
<tr>
<th>TAG</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Obsolete</td>
<td>Format of unit has changed or has no target units in the new version.</td>
</tr>
<tr>
<td>2. NewSpec</td>
<td>New elements have added</td>
</tr>
</tbody>
</table>

#### Change analysis List

- [ ] Obsolete

```c
int init(int a, int b) {
    return 0;
}

int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

Old

```c
int init(char a) {
    return 0;
}

int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

New

The unit is deleted
### Change analysis List

1. **Obsolete**
   - Format of unit has changed or has no target units in the new version.

2. **NewSpec**
   - New elements have added

```c
int init(int a, int b) {
    return 0;
}

int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

Old

```c
int init(char a) {
    return 0;
}

int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

New
```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```

**Change analysis List**

- 1. **Obsolete**: Format of unit has changed or has no target units in the new version.
- 2. **NewSpec**: New elements have added

```c
int change;
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    for(i = a; i <= b; i++){
        sum = sum + 1;
    }
    return sum;
}
```
3. Inserting Footprint

Overview of RT-Selection
3. Inserting Footprint

- **What** is Footprint?
  - Footprint is an interrelation of elements

- **Why** insert footprints?

- **How** insert the footprints?

1. identify the elements.
   - 1. assignment
   - 2. function return
   - 3. function call
   - 4. condition sentence
   - 5. iteration sentence

2. identify the interaction between elements.

3. select an appropriate footprint

4. insert selected footprint.

**Footprint process**

EX)

<table>
<thead>
<tr>
<th>Context:  $a = b + 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>interaction: $b$ affects to $a$</td>
</tr>
<tr>
<td>Footprint: $a \leftarrow b$</td>
</tr>
</tbody>
</table>
3. Inserting Footprint

- **What** is Footprint?
  - Footprint is an interrelation of elements

- **Why** insert footprints?
  - To know a riffle of software by test case

- **How** insert the footprints?

  1. identify the elements.
     1. assignment
     2. function return
     3. function call
     4. condition sentence
     5. iteration sentence
  2. identify the interaction between elements.
  3. select an appropriate footprint
  4. insert selected footprint.

**Footprint process**

```
EX)
Context: a = b + 1
interaction : b affects to a
Footprint:
    a ← b
```
3. Inserting Footprint

- **What** is Footprint?
  - Footprint is an interrelation of elements

- **Why** insert footprints?
  - To know a riffle of software by test case

- **How** insert the footprints?
  - Code analysis. (Using footprint process and Guidelines)

---

1. identify the elements.
   1. assignment
   2. function return
   3. function call
   4. condition sentence,
   5. iteration sentence

2. identify the interaction between elements.

3. select an appropriate footprint

4. insert selected footprint.

**Footprint process**

EX)

- Context: $a = b + 1$
- interaction : $b$ affects to $a$
- Footprint:
  - $a \leftarrow b$
# Guidelines for footprint

<table>
<thead>
<tr>
<th>Guideline 1.</th>
<th>Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Left element ← Right element</td>
</tr>
<tr>
<td>Function return</td>
<td>function name () ← something</td>
</tr>
<tr>
<td>Function call</td>
<td>function name () ← parameter</td>
</tr>
</tbody>
</table>

**EX)**

A = B + 1

A ← B

---

<table>
<thead>
<tr>
<th>Guideline 2.</th>
<th>Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition sentence</td>
<td>(ordinal number)_(context)_condition used element</td>
</tr>
<tr>
<td>(if, switch, etc.)</td>
<td></td>
</tr>
<tr>
<td>Enumeration</td>
<td>Left element ← (ordinal number)_(context)_condition</td>
</tr>
</tbody>
</table>

**EX)**

if ( a > b ){  
    sum=sum+1;  
}  

1_if_condition ← a  
1_if_condition ← b  
Sum ← 1_if_condition

Enumeration  
(Guideline 1.)
### Guideline 3.

**Footprint**

Iteration sentence (for, while, etc.)

(ordinal number)_(context)_condition used element

Enumeration (In case of Guideline 1.)

Left element $\leftarrow$ (ordinal number)_(context)_condition

**EX)**

```c
for(i=a;i<b;i++) {
    sum=sum+1;
}
```

Guideline 4.

**Footprint**

Enumeration (In case of Guideline 2, 3)

Left element $\leftarrow$ (ordinal number)_(context)_condition_ (ordinal number)_(context)_condition; ....

**EX)**

```c
for(i=a;i<b;i++) {
    for(i=a;i<b;i++) {
        sum=sum+1;
    }
}
```
```c
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
```
int Calc(int a, int b) {
    int i;
    int sum = init(a, b);
    for(i = a; i < b; i++){
        sum = sum + 1;
    }
    return sum;
}
Just execute the testing using previous test cases

```c
int Calc(int a, int b) {
    int i;
    printf("Calc(): init() \n");
    printf("Calc(): init() \n");
    printf("Calc(): sum <= init() \n");
    int sum = init(a, b);
    printf("Calc(): \_if; \_if_condition <= a\n");
    printf("Calc(): \_if; \_if_condition <= b\n");
    for (i = a; i < b; i++) {
        printf("Calc(): \_for; sum <= \_for_condition\n");
        sum = sum + 1;
    }
    printf("Calc(): Calc() \n");
    return sum;
}
```
5. Assumption Analysis

What elements are affecting to the assumption in the test case.

```c
Unit_test_1()
{
    int expected_value = 0;
    expected_value = Calc(1,3);
    Assumption(expected_value == 3);
}
```
6. Change Impact Analysis

1. Canonical Formatting

2. Text Differencing & Change Analysis

3. Inserting Footprint

4. Testing Execution

5. Assumption Analysis

6. Change Impact Analysis

7. Comparing

Regression testing

Old Code

Canonical form

Test Cases Using Old Code

Footprint List

Change Impact Analysis List

Change Analysis List

New Code

Canonical form

Using Guideline

Using TAG

Using Inference Rule
6. Change Impact Analysis

- **What** is Change Impact Analysis?
  - To identify what elements are affecting the assumption

- **Why** perform Change Impact Analysis?
  - To identify the actually effecting elements

- **How** perform Change Impact Analysis?
  - Inference rule
Footprint List
[
  Calc(); init() <- a
  Calc(); init() <- b
  Calc(); sum <- init()
  init(); 1_if; 1_if_condition <- a
  init(); 1_if; 1_if_condition <- b
  init(); 1_if; init() <- 1_if_condition
  Calc(); 1_for; 1_for_condition <- a
  Calc(); 1_for; 1_for_condition <- b
  Calc(); 1_for; sum <- 1_for_condition
  Calc(); Calc() <- sum
]

Change Impact analysis List
[
  init(); a
  init(); b
  init(); 1_if_condition
  Calc(); a
  Calc(); b
  Calc(); init()
  Calc(); Calc(); 1_for_condition
  Calc(); sum
  Calc(); Calc(); Calc()
]
7. Comparing

Same elements are existing? Just do comparison

1. Canonical Formatting

2. Text Differencing & Change Analysis

3. Inserting Footprint

4. Testing Execution

5. Assumption Analysis

6. Change Impact Analysis

7. Comparing

Selected Test cases

Regression testing
7. Comparing

Change Impact analysis List
[
  init(); a
  init(); b
  init(); 1_if_condition
  Calc(); a
  Calc(); b
  Calc(); init()
  Calc(); 1_for_condition
  Calc(); sum
  Calc(); Calc()
]

Change analysis List
[
  Calc(); 1_for_condition
]
7. Comparing

Comparing Old Code Canonical form
New Code Canonical form
Inserted Footprint Code
Change Analysis List

Same elements are existing? Just do comparison

Change Impact analysis List
[
    init(); a
    init(); b
    init(); 1_if_condition
    Calc(); a
    Calc(); b
    Calc(); init()
    Calc(); 1_for_condition
    Calc(); sum
    Calc(); Calc()
]

Change analysis List
[
    Calc(); 1_for_condition
]

Coincidence → Selection

TAG - Obsolete
- NewSpec

Using Inference Rule
Change Impact Analysis List
Change Analysis List
Comparing → Selected Test cases
Using Inference Rule

1. Canonical Formatting
   - Old Code
   - New Code

2. Text Differencing & Change Analysis
   - Using TAG

3. Inserting Footprint
   - Using Guideline
   - Inserted Footprint Code

4. Testing Execution
   - Test Cases Using Old Code

5. Assumption Analysis
   - Assumption Analysis List

6. Change Impact Analysis
   - Change Impact Analysis List

7. Comparing
   - Change Analysis List
   - Selected Test cases

Regression testing
Case study
• We performed RT-Selection with 9 teams code.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test cases for old version</td>
<td>55</td>
<td>49</td>
<td>50</td>
<td>72</td>
<td>54</td>
<td>57</td>
<td>35</td>
<td>73</td>
<td>50</td>
</tr>
<tr>
<td>Obsolete test cases</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>NewSpec test cases</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Re-testable test cases</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Candidate for regression testing</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>11</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>
Case study

- We performed RT-Selection with 9 teams' code.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test cases for old version</td>
<td>55</td>
<td>49</td>
<td>50</td>
<td>72</td>
<td>54</td>
<td>57</td>
<td>35</td>
<td>73</td>
<td>50</td>
</tr>
<tr>
<td>Obsolete test cases</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>NewSpec test cases</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Re-testable test cases</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Candidate for regression testing</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>11</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

2 test cases that have coincident elements
Case study

- We performed RT-Selection with 9 teams code.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test cases for old version</td>
<td>55</td>
<td>49</td>
<td>50</td>
<td>72</td>
<td>54</td>
<td>57</td>
<td>35</td>
<td>73</td>
<td>50</td>
</tr>
<tr>
<td>Obsolete test cases</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>3</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>NewSpec test cases</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Re-testable test cases</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>-</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>Candidate for regression testing</td>
<td>18</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
<td>11</td>
<td>7</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

Any other tags and coincident test cases is not existence.

2 test cases that have coincident elements
Conclusion

• We suggest the **RT-Selection** which is technique for regression testing.
  
  – It has two approach textual differencing and change impact analysis.
  – It has 7 phases.

• We are now planning to implement a set of automation tools.
  – It would increase usability.

• After all of tool are developed, we are perform how cost-effective than other regression tools.
Conclusion

• We suggest the **RT-Selection** which is technique for regression testing.
  
  – It has two approach textual differencing and change impact analysis.
  – It has 7 phases.

• We are now planning to implement a set of automation tools.
  – It would increase usability.

• After all of tool are developed, we are perform how cost-effective than other regression tools.
Thank you