RT-Selection

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

Eui-Sub Kim
Dependable Software Laboratory
KONKUK University

Contents

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

Introduction

- Some of questions and answer

What is regression testing?

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

Why regression testing cost is high?

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

How to reduce the cost of regression testing?

Introduction

- Some of questions and answer

What is regression testing?

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

Why regression testing cost is high?

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

How to reduce the cost of regression testing?

- To identify the changes
- To select a subset.

→ **RT-Selection**.

RT-Selection

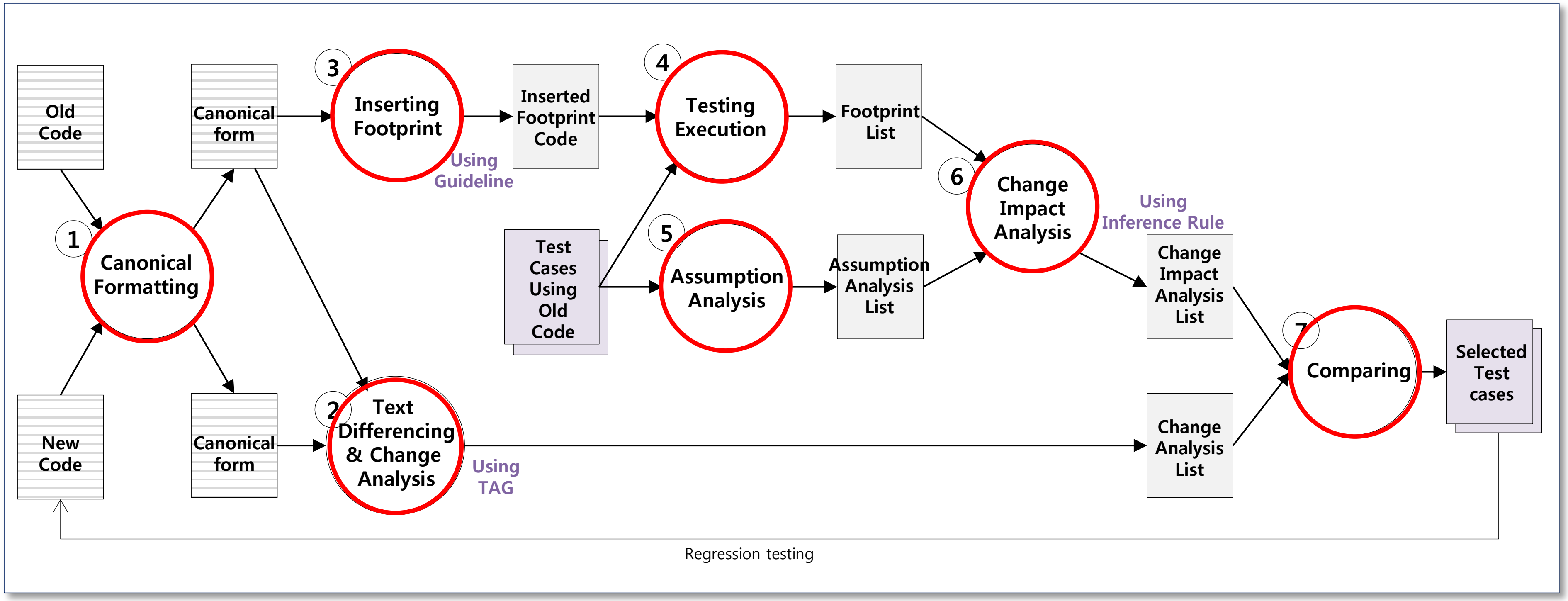
What is RT-Selection?

How to perform RT-Selection?

RT-Selection

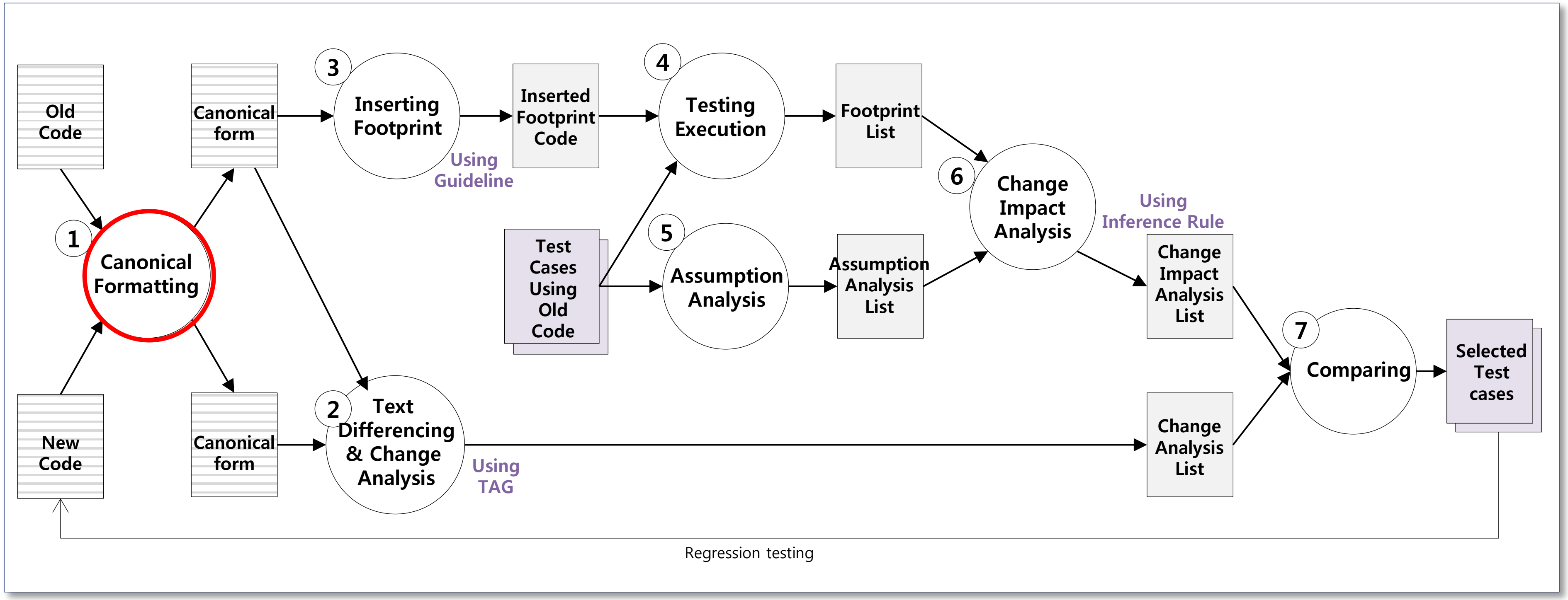
What is RT-Selection?

How to perform RT-Selection?



Overview of RT-Selection

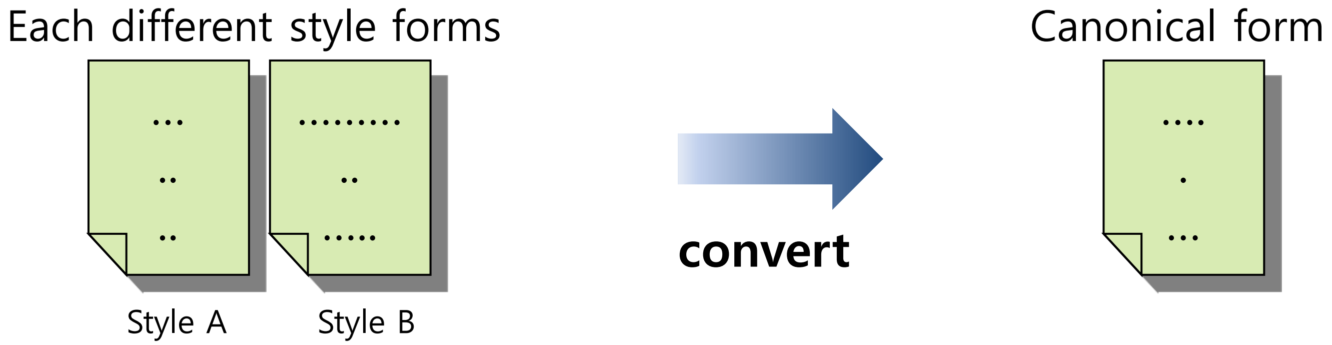
1. Canonical Formatting



Overview of RT-Selection

1. Canonical Formatting

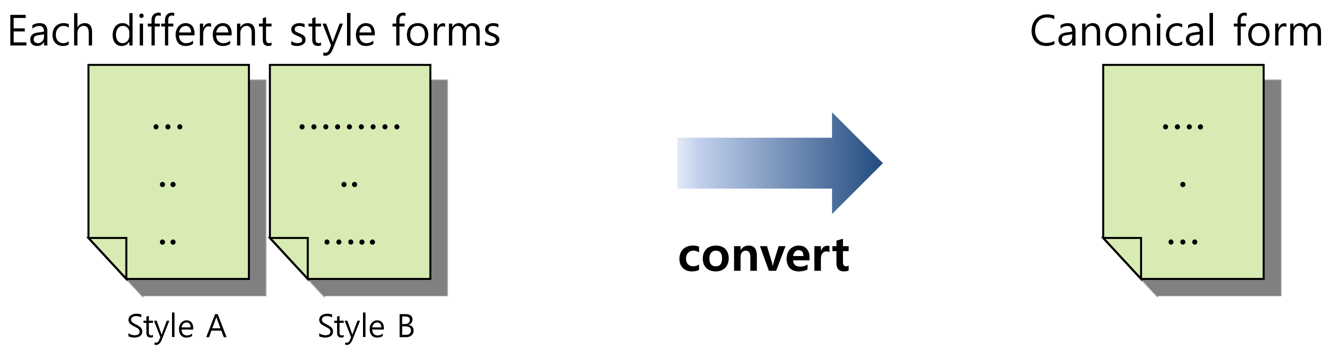
- **What** is canonical formatting?



- **Why** convert into canonical formatting?

1. Canonical Formatting

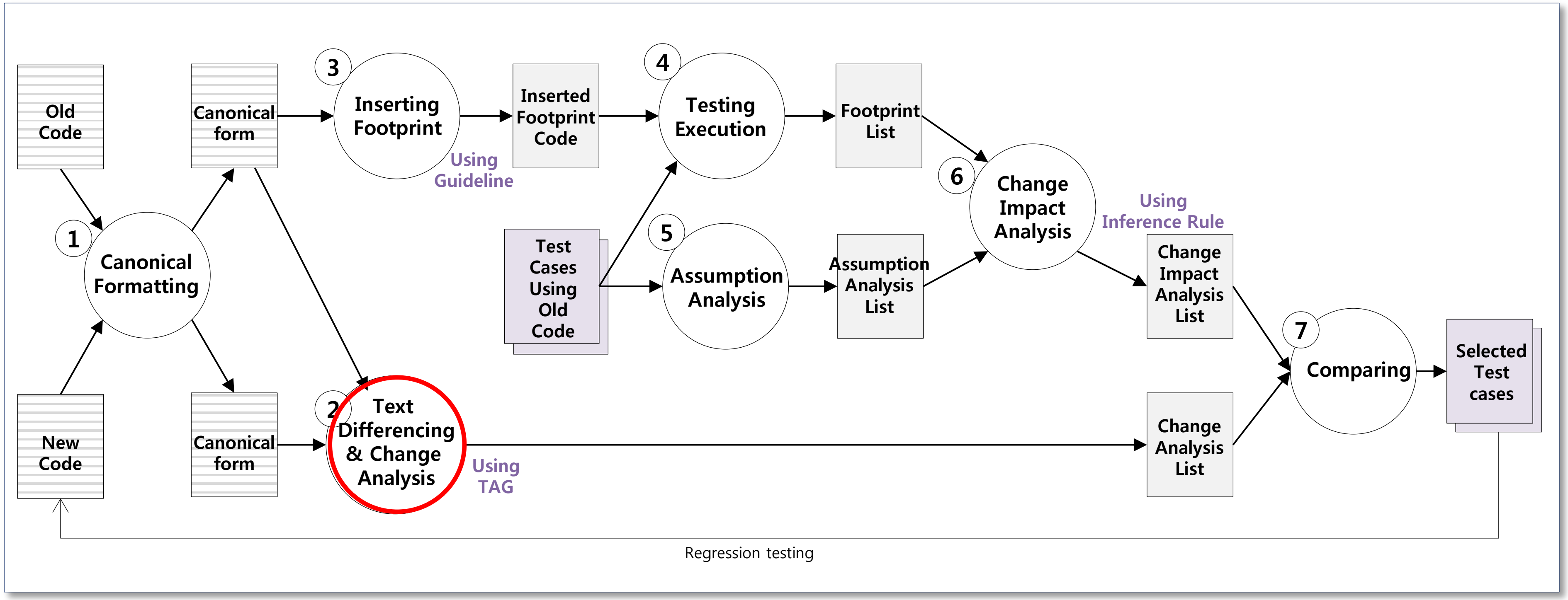
- **What** is canonical formatting?



- **Why** convert into canonical formatting?
  - ➢ To reduce unwanted result.

Blank
"(", "{"
Comment
All of different coding styles
so on

2. Textual Differencing and Change Analysis

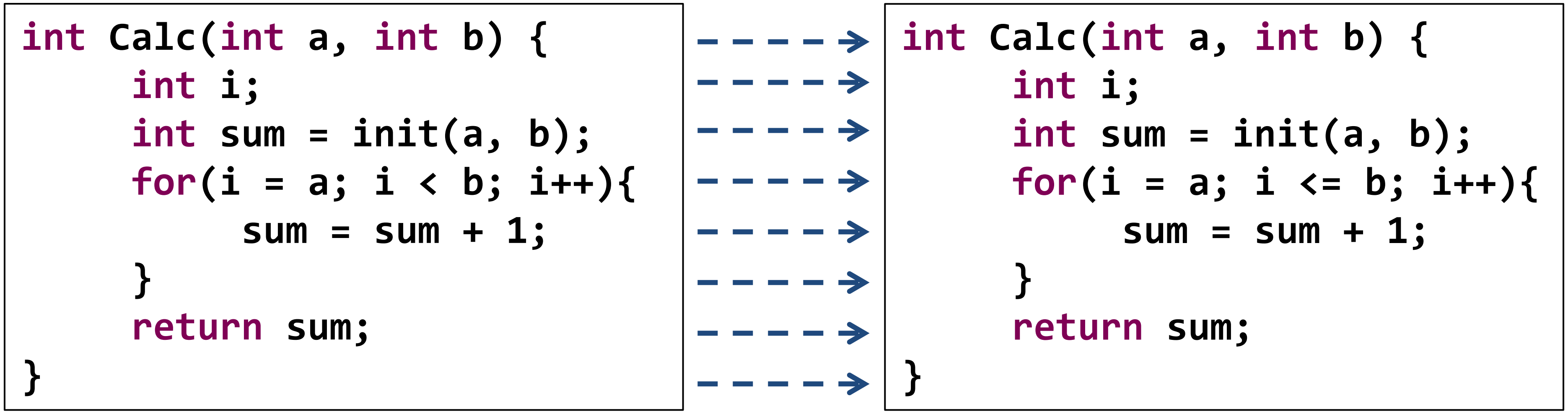


Overview of RT-Selection

2. Textual Differencing and Change Analysis

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

- **Why** perform Textual differencing?



```
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

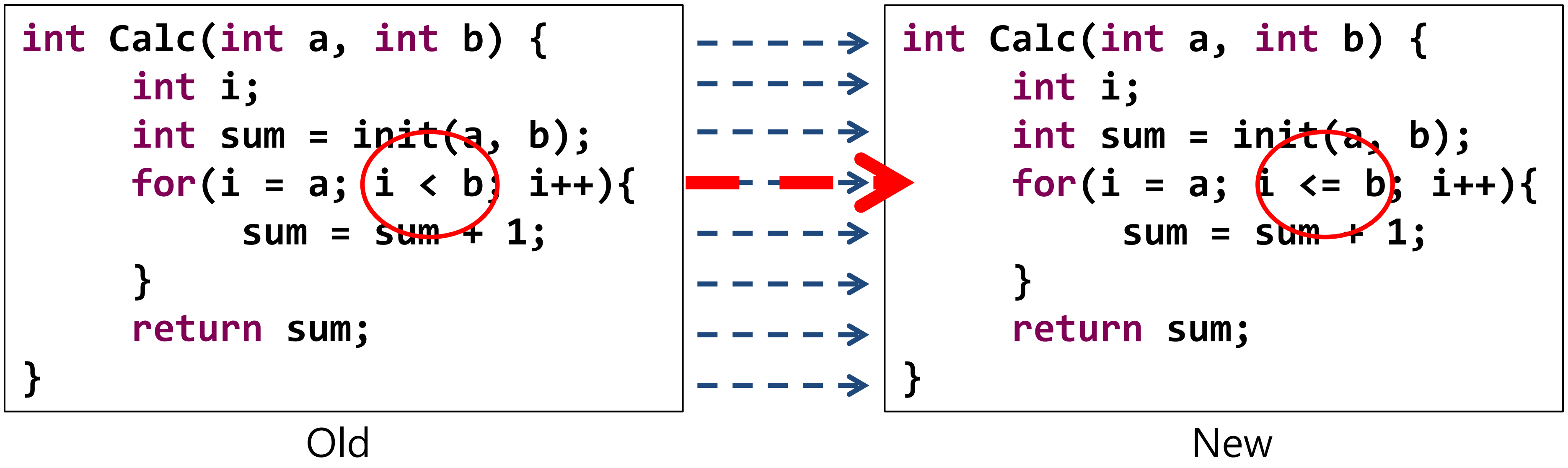
```
int Calc(int a, int b) {
     int i;
     int sum = init(a, b);
     for(i = a; i <= b; i++){
          sum = sum + 1;
     }
     return sum;
}
```

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

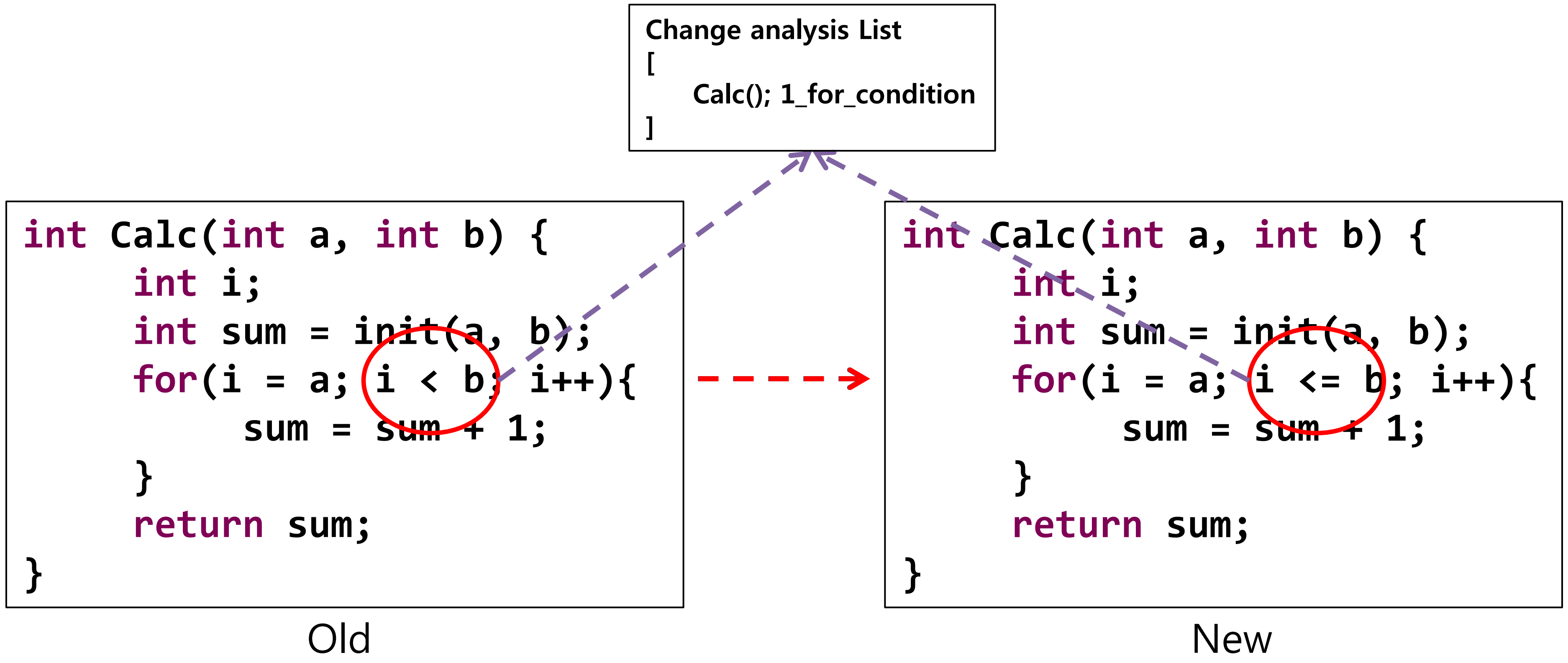


```
int Calc(int a, int b) {
     int i;
     int sum = init(a, b);
     for(i = a; i < b; i++){
          sum = sum + 1;
     }
     return sum;
}
```

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

2. Textual Differencing and Change Analysis

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



Change analysis List
[
    Calc(); 1_for_condition
]

```
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

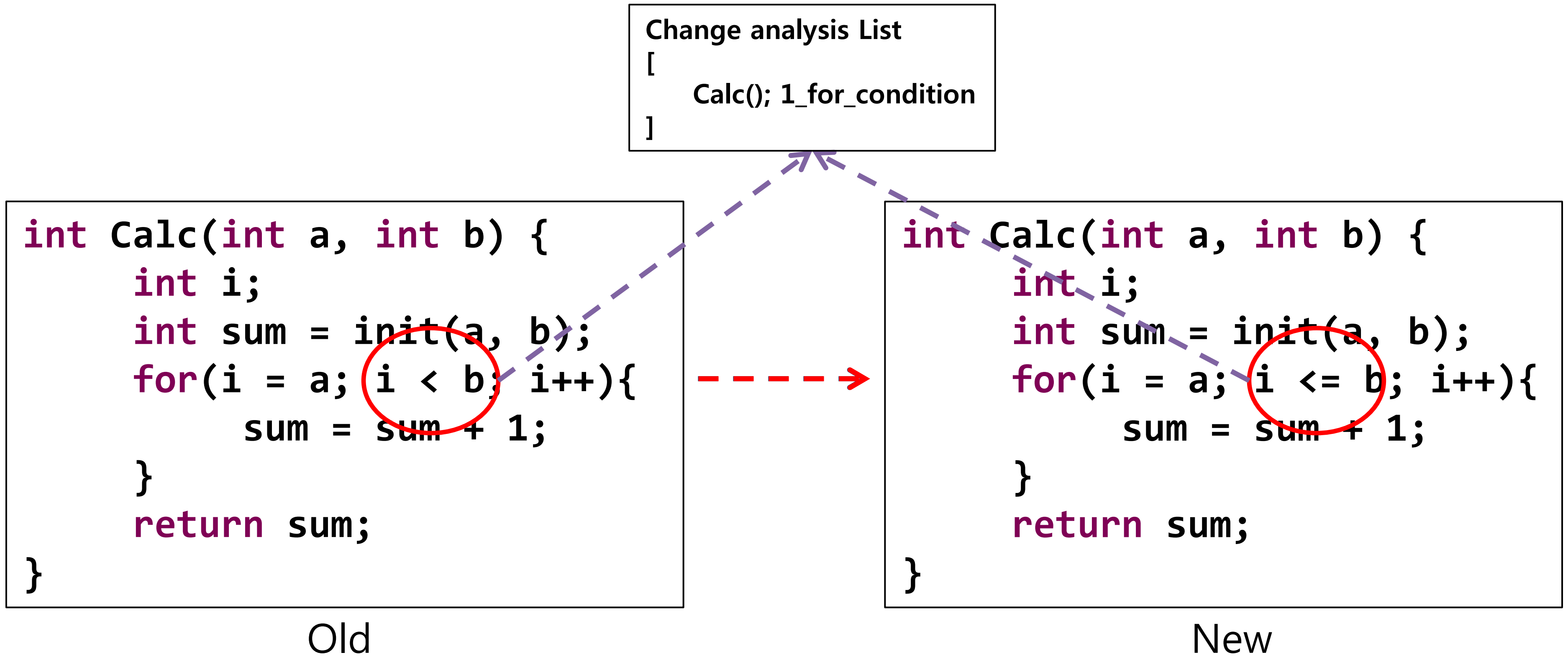
```
int Calc(int a, int b) {
     int i;
     int sum = init(a, b);
     for(i = a; i <= b; i++){
          sum = sum + 1;
     }
     return sum;
}
```

New

2. Textual Differencing and Change Analysis

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

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



```
Change analysis List
[
	Calc(); 1_for_condition
]
```

```
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

```
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

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

Change analysis List
[
Obsolete
]

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

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

The unit is deleted

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

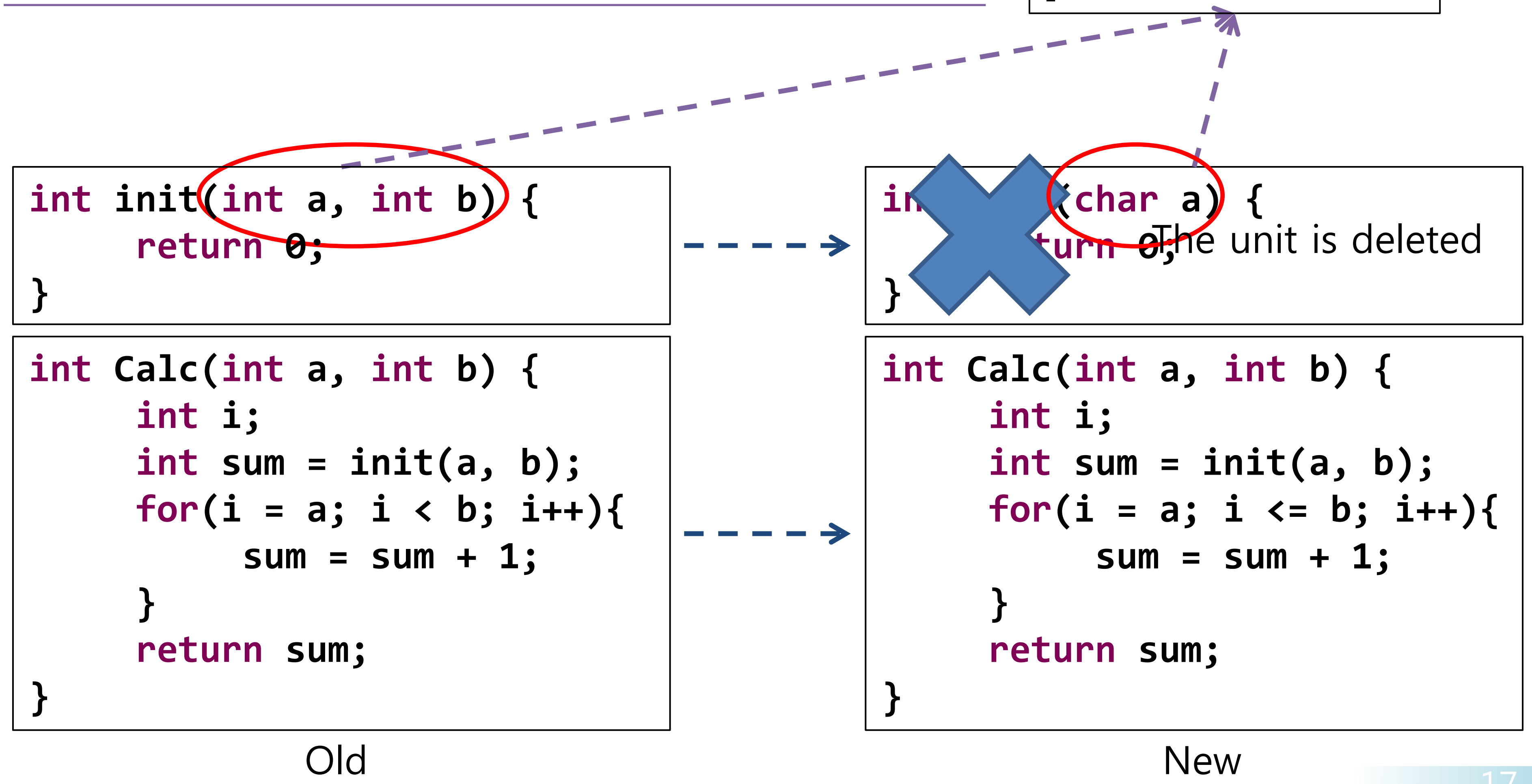
Old

```
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

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

Change analysis List
[
Obsolete
]



```
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

```
in      (char a) {
         turn 0;
}
```

The unit is deleted

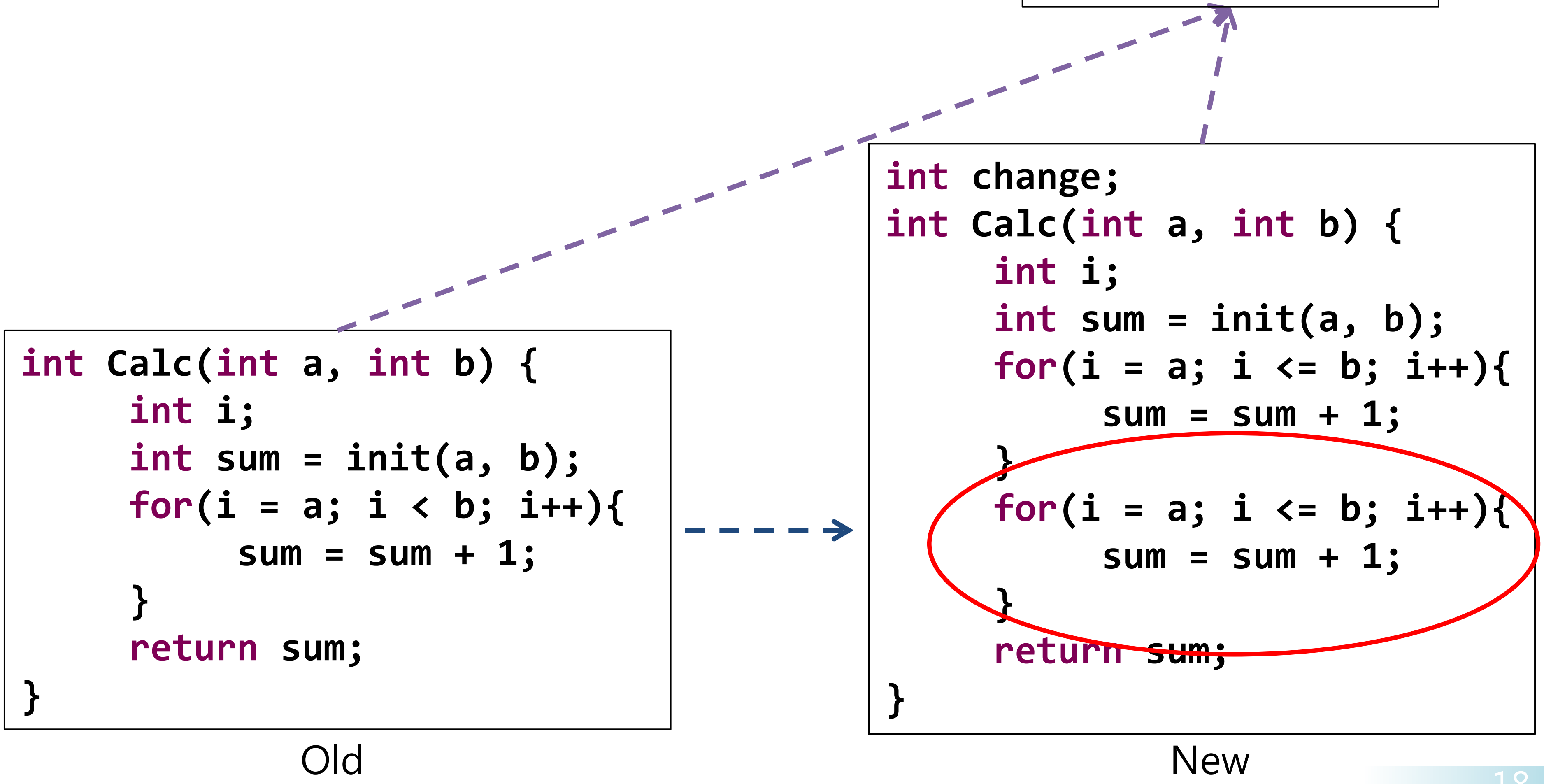
```
int Calc(int a, int b) {
      int i;
      int sum = init(a, b);
      for(i = a; i <= b; i++){
            sum = sum + 1;
      }
      return sum;
}
```

New

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

Change analysis List

```
[
    NewSpec
]
```



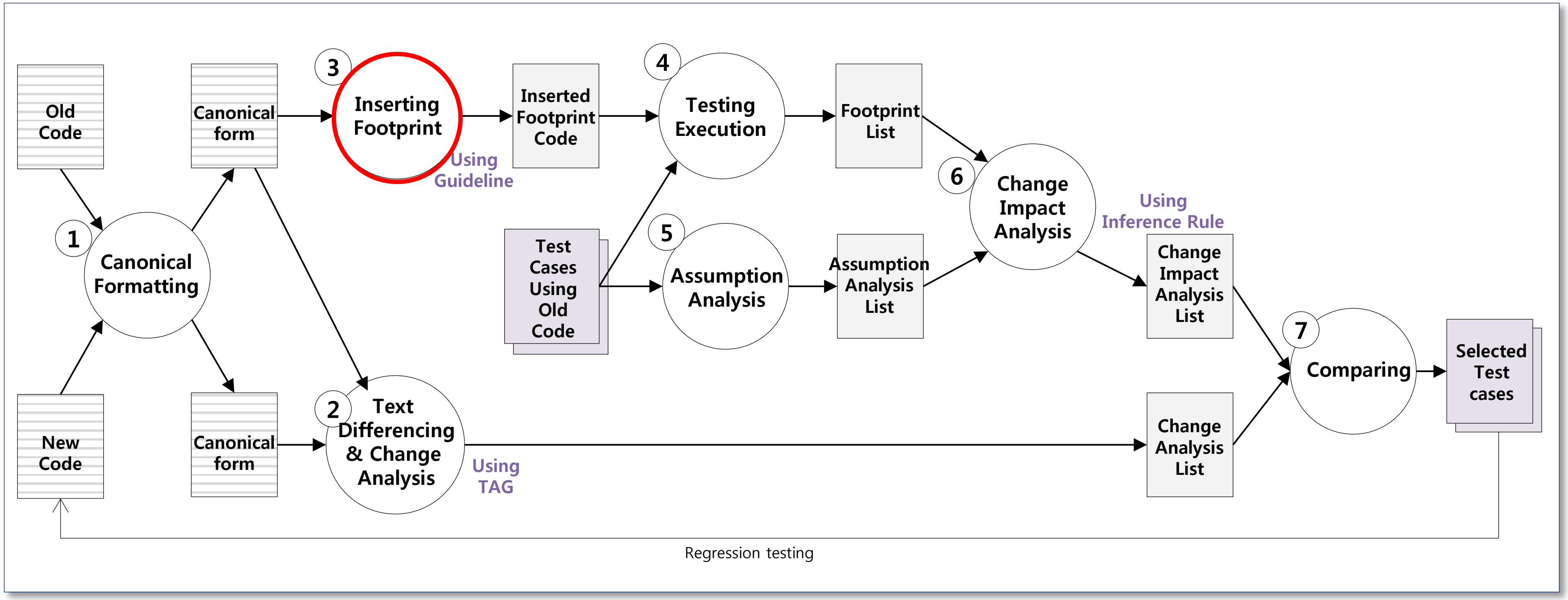
```
int Calc(int a, int b) {
      int i;
      int sum = init(a, b);
      for(i = a; i < b; i++){
            sum = sum + 1;
      }
      return sum;
}
```

Old

```
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;
}
```

New

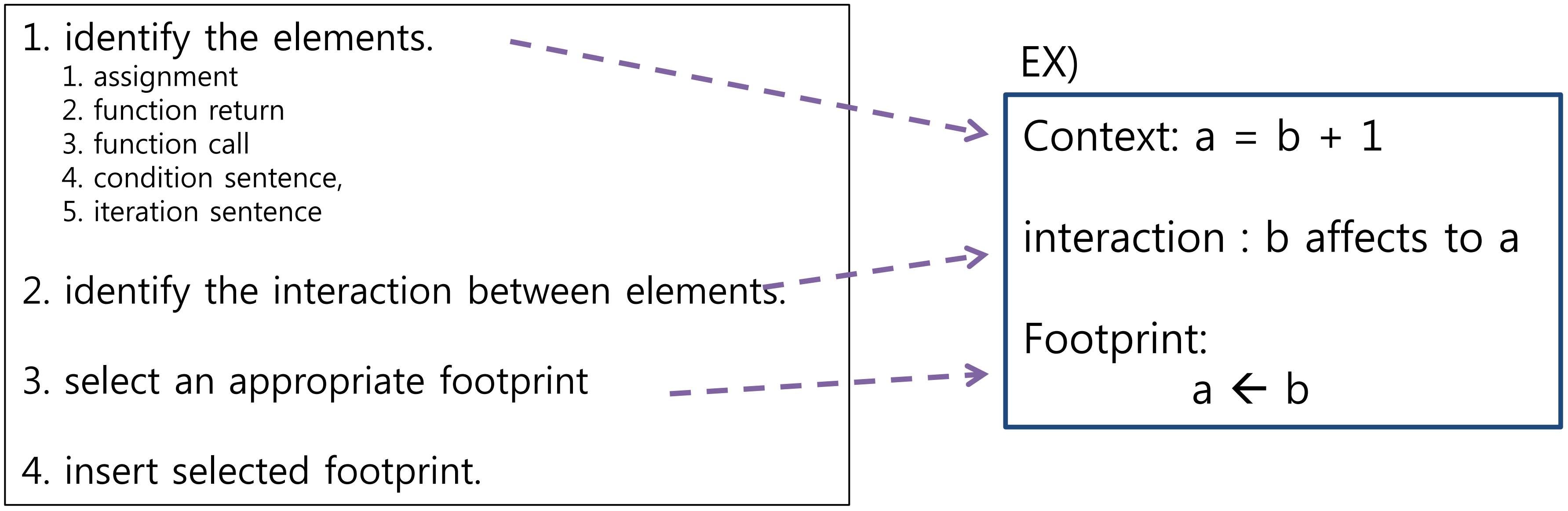
3. Inserting Footprint



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.

EX)

Context: a = b + 1

interaction : b affects to a

Footprint:
a ← b

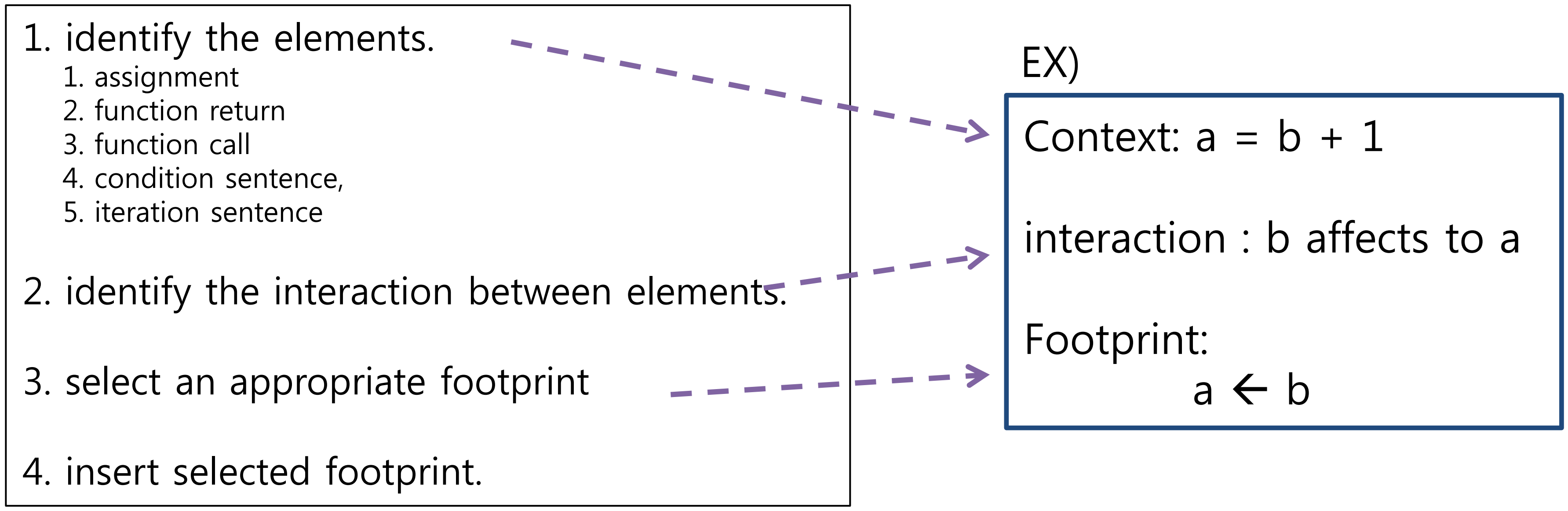
Footprint process

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.

EX)

Context: a = b + 1

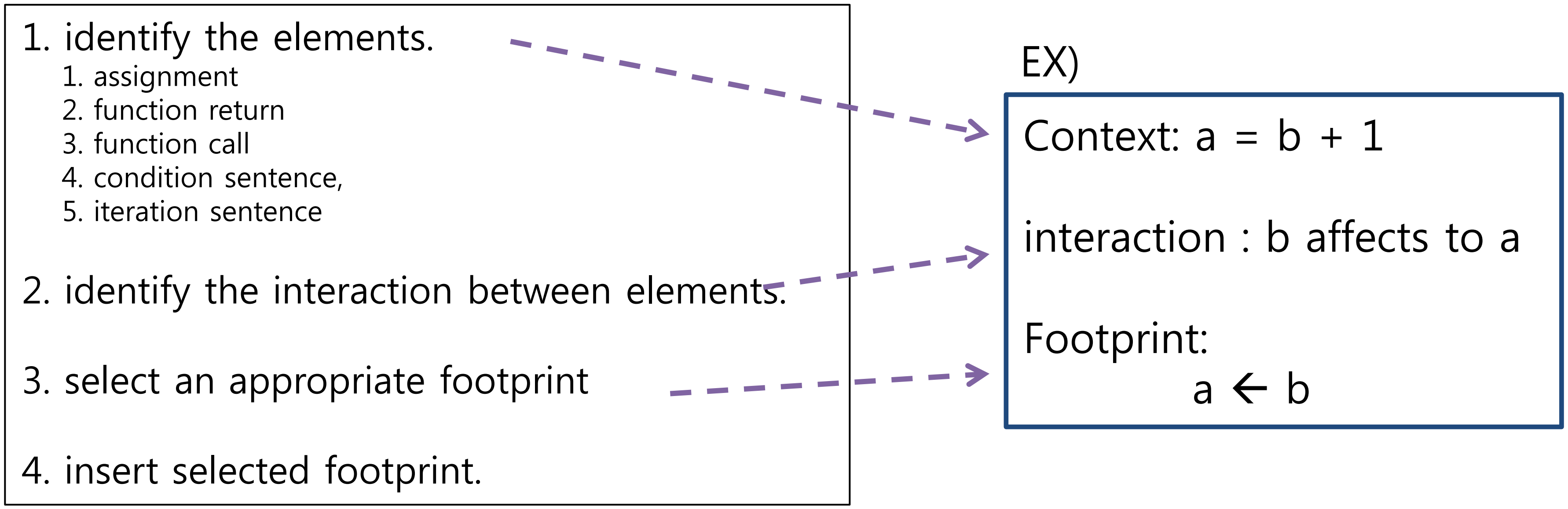
interaction : b affects to a

Footprint:
a ← b

Footprint process

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.

EX)

Context: a = b + 1

interaction : b affects to a

Footprint:
a ← b

Footprint process

Guidelines for footprint

Guideline 1.	Footprint
Assignment	Left element ← Right element
Function return	function name () ← something
Function call	function name () ← parameter

EX)

```
A = B + 1
```

```
A ← B
```

Guideline 2.	Footprint
Condition sentence (if, switch, etc.)	(ordinal number)_(context)_condition used element
Enumeration (In case of Guideline 1.)	Left element ← (ordinal number)_(context)_condition

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 ← (ordinal number)_(context)_condition

EX)

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

```
1_for_condition ← a
1_for_condition ← b
Sum ← 1_for_condition
```

Enumeration
(Guideline 1.)

Guideline 4.	Footprint
Enumeration (In case of Guideline 2, 3)	Left element ← (ordinal number)_(context)_condition_ (ordinal number)_(context)_condition; ….

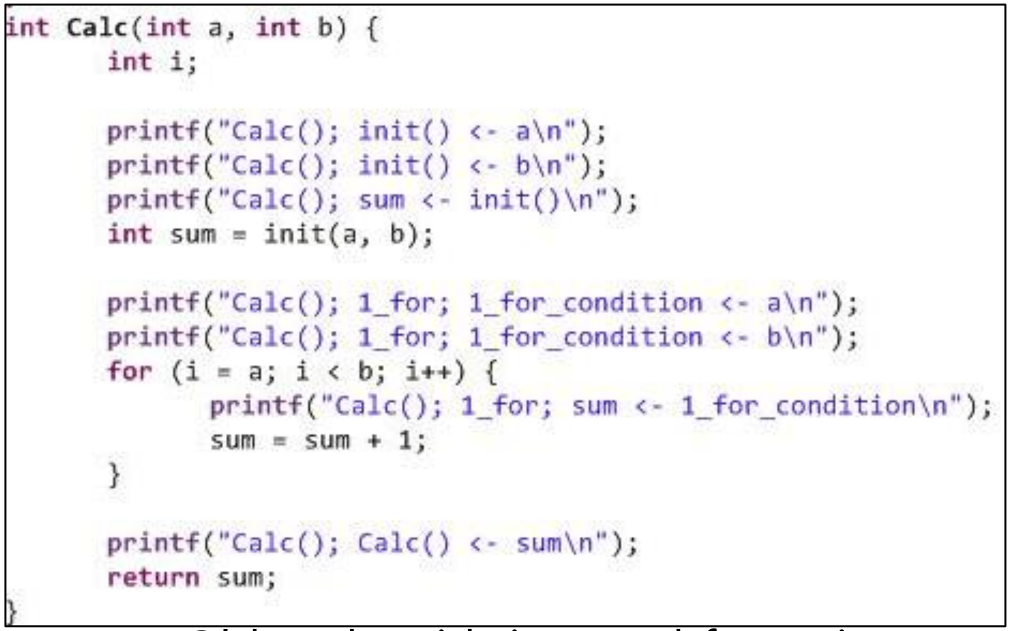
EX)

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

```
Sum ← 1_for_condition_
      1_for_condition
```

```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 code



```c
int Calc(int a, int b) {
	int i;

	printf("Calc(); init() <- a\n");
	printf("Calc(); init() <- b\n");
	printf("Calc(); sum <- init()\n");
	int sum = init(a, b);

	printf("Calc(); 1_for; 1_for_condition <- a\n");
	printf("Calc(); 1_for; 1_for_condition <- b\n");
	for (i = a; i < b; i++) {
		printf("Calc(); 1_for; sum <- 1_for_condition\n");
		sum = sum + 1;
	}

	printf("Calc(); Calc() <- sum\n");
	return sum;
}
```

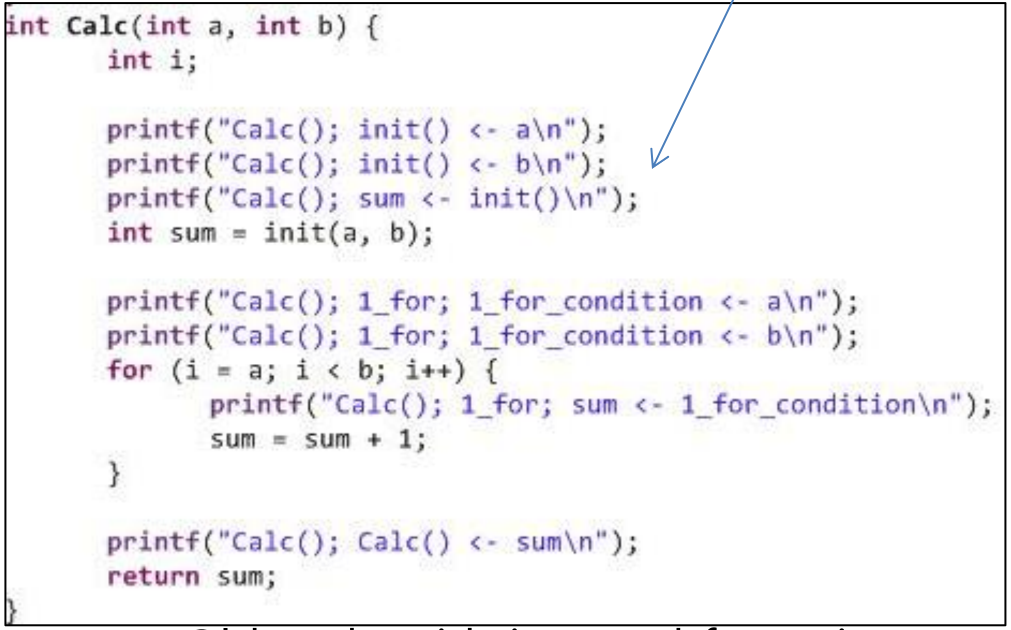
Old code with inserted footprint

Assignment

Sum ← init()

```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;
}
```

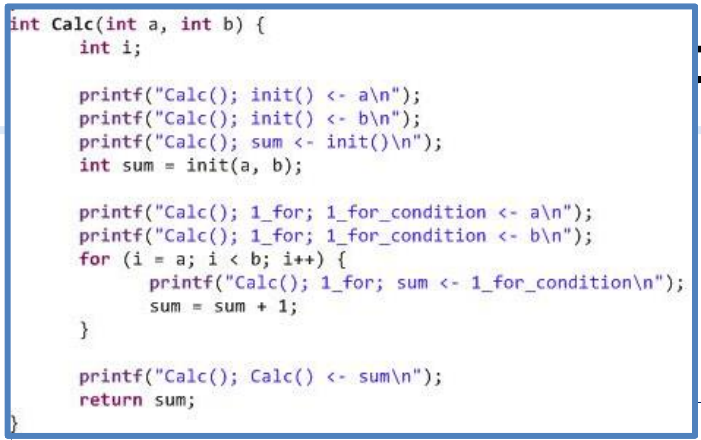
Old code



```c
int Calc(int a, int b) {
	int i;

	printf("Calc(); init() <- a\n");
	printf("Calc(); init() <- b\n");
	printf("Calc(); sum <- init()\n");
	int sum = init(a, b);

	printf("Calc(); 1_for; 1_for_condition <- a\n");
	printf("Calc(); 1_for; 1_for_condition <- b\n");
	for (i = a; i < b; i++) {
		printf("Calc(); 1_for; sum <- 1_for_condition\n");
		sum = sum + 1;
	}

	printf("Calc(); Calc() <- sum\n");
	return sum;
}
```

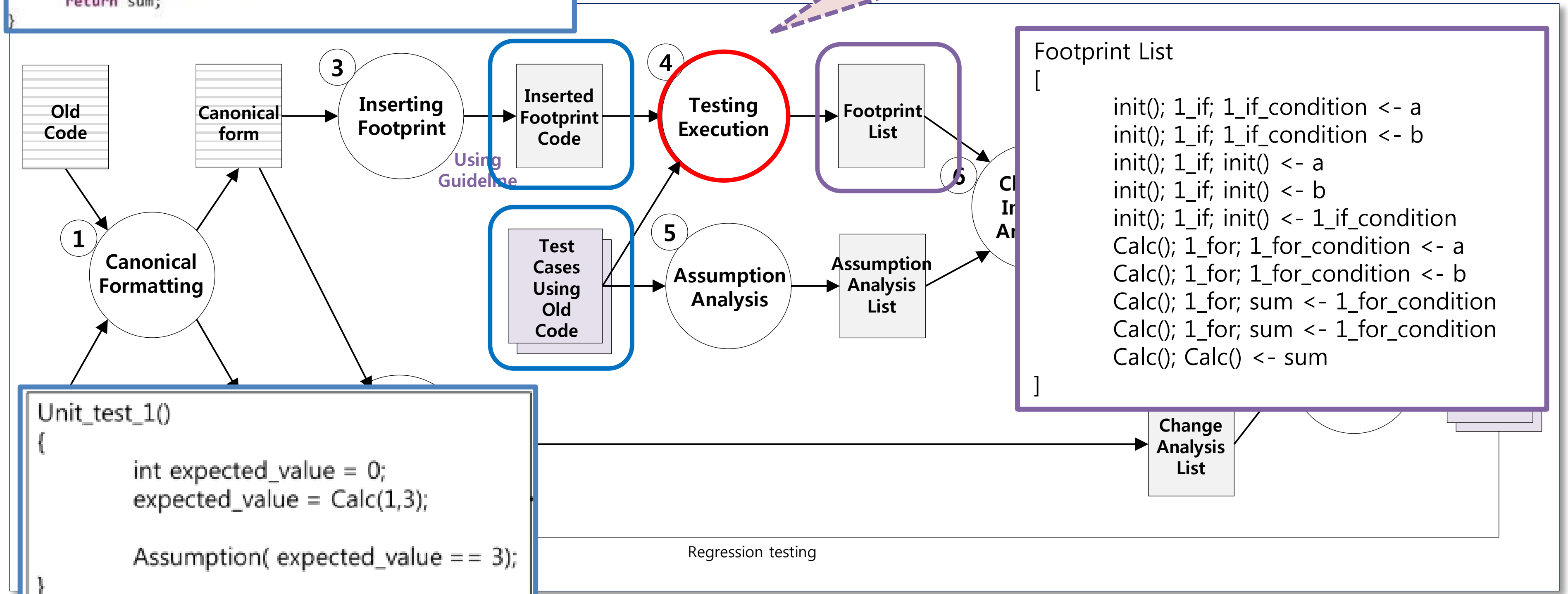
Old code with inserted footprint

tion



```
int Calc(int a, int b) {
        int i;

        printf("Calc(); init() <- a\n");
        printf("Calc(); init() <- b\n");
        printf("Calc(); sum <- init()\n");
        int sum = init(a, b);

        printf("Calc(); 1_for; 1_for_condition <- a\n");
        printf("Calc(); 1_for; 1_for_condition <- b\n");
        for (i = a; i < b; i++) {
                printf("Calc(); 1_for; sum <- 1_for_condition\n");
                sum = sum + 1;
        }

        printf("Calc(); Calc() <- sum\n");
        return sum;
}
```

Just execute the testing using **previous test cases**

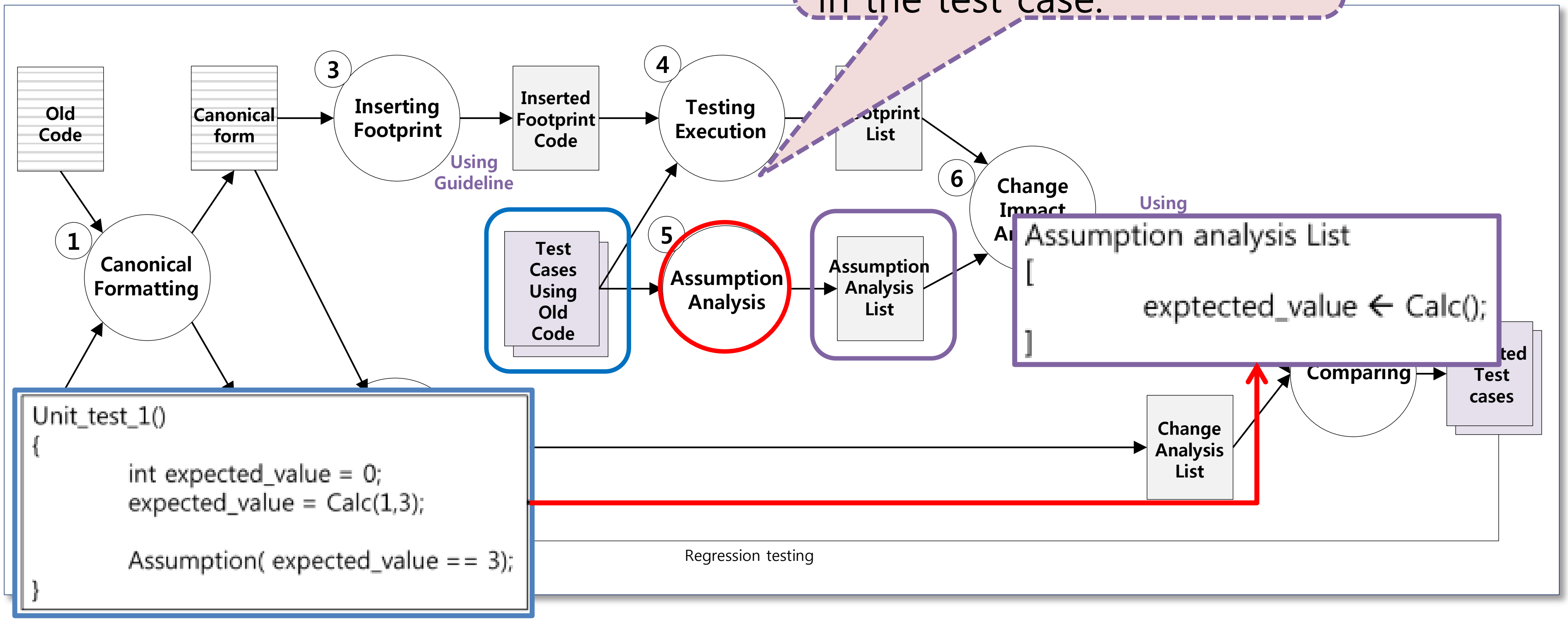


Footprint List
[
init(); 1_if; 1_if_condition <- a
init(); 1_if; 1_if_condition <- b
init(); 1_if; init() <- a
init(); 1_if; init() <- b
init(); 1_if; init() <- 1_if_condition
Calc(); 1_for; 1_for_condition <- a
Calc(); 1_for; 1_for_condition <- b
Calc(); 1_for; sum <- 1_for_condition
Calc(); 1_for; sum <- 1_for_condition
Calc(); Calc() <- sum
]

```
Unit_test_1()
{
        int expected_value = 0;
        expected_value = Calc(1,3);

        Assumption( expected_value == 3);
}
```

5. Assumption Analysis

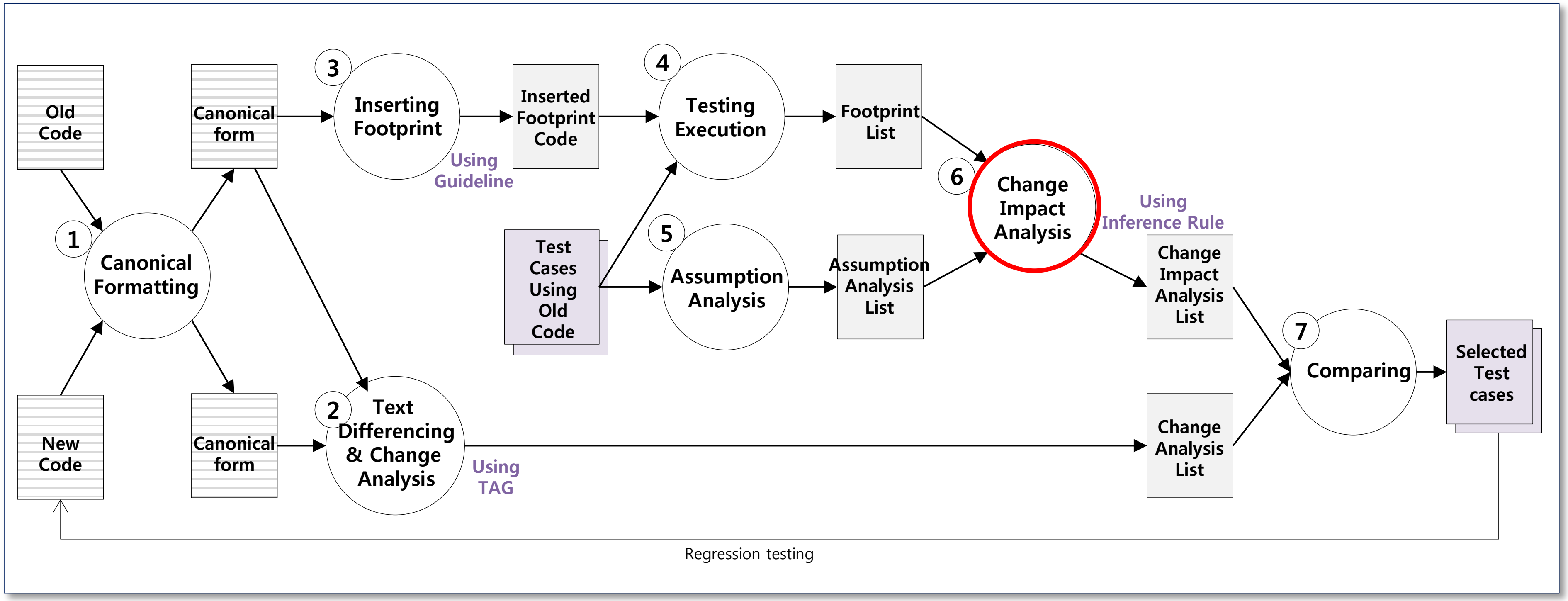
what elements are affecting to the assumption in the test case.



Assumption analysis List
[
    exptected_value ← Calc();
]

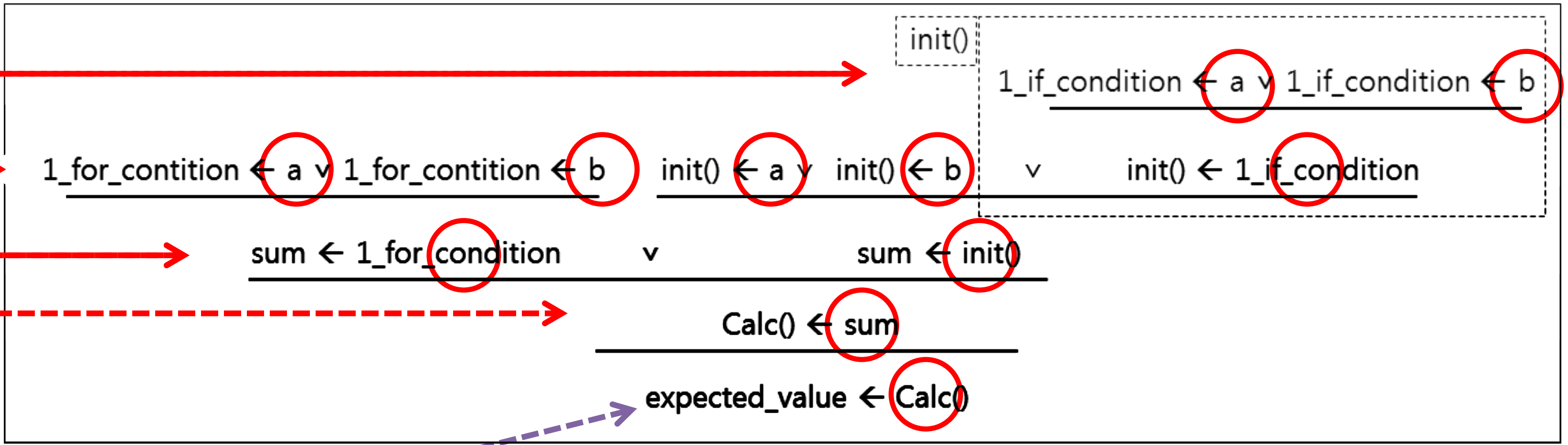
```
Unit_test_1()
{
        int expected_value = 0;
        expected_value = Calc(1,3);

        Assumption( expected_value == 3);
}
```

6. Change Impact Analysis



6. Change Impact Analysis

- **What** is Change Impact Analysis?
  - ➢ To identify what elements are affecting to the assumption
- **Why** perform Change Impact Analysis?
  - ➢ To identify the actually effecting elements
- **How** perform Change Impact Analysis?
  - ➢ Inference rule



Inference rule

Assumption analysis List
[
exptected_value ← Calc();
]

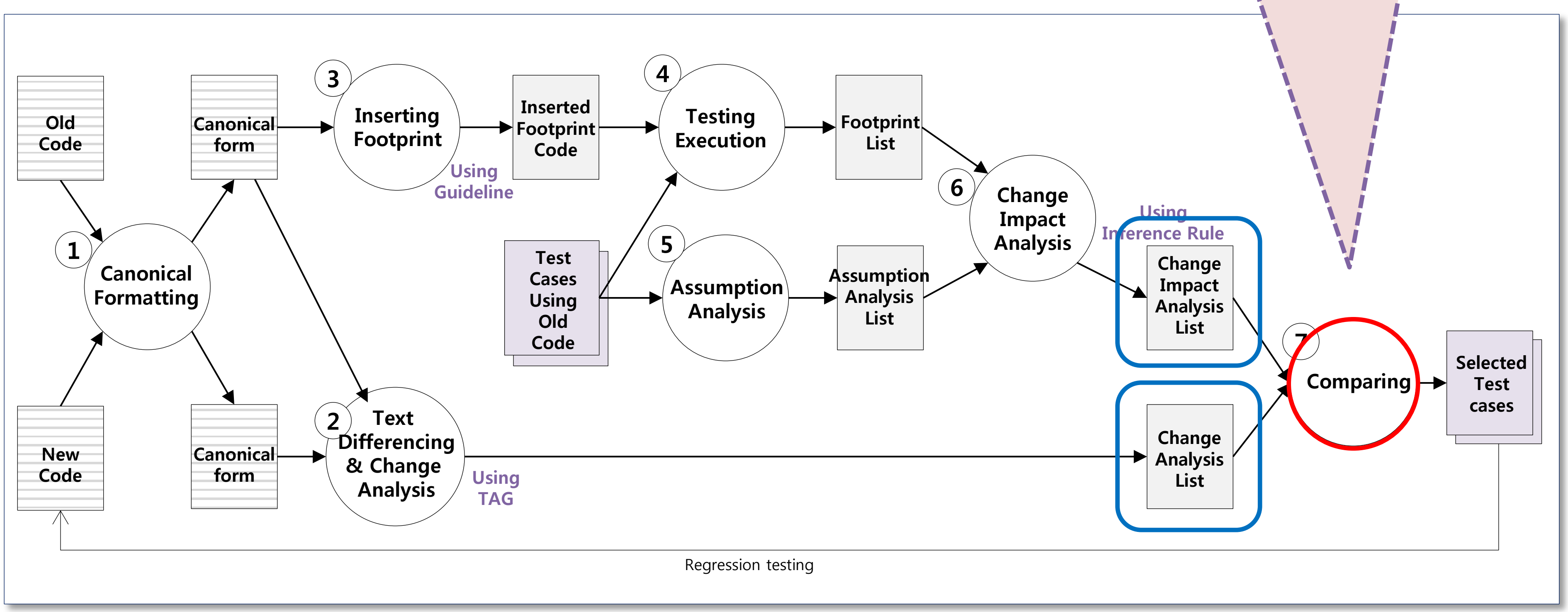
Footprint List
[
Calc(); init() <- a
Calc(); init() <- b
Calc(); sum <- init()
init(); 1_if; 1_if_condition <- a
init(); 1_if; 1_if_condition <- b
init(); 1_if; init() <- 1_if_condition
Calc(); 1_for; 1_for_condition <- a
Calc(); 1_for; 1_for_condition <- b
Calc(); 1_for; sum <- 1_for_condition
Calc(); Calc() <- sum
]

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

Change Impact Analysis List

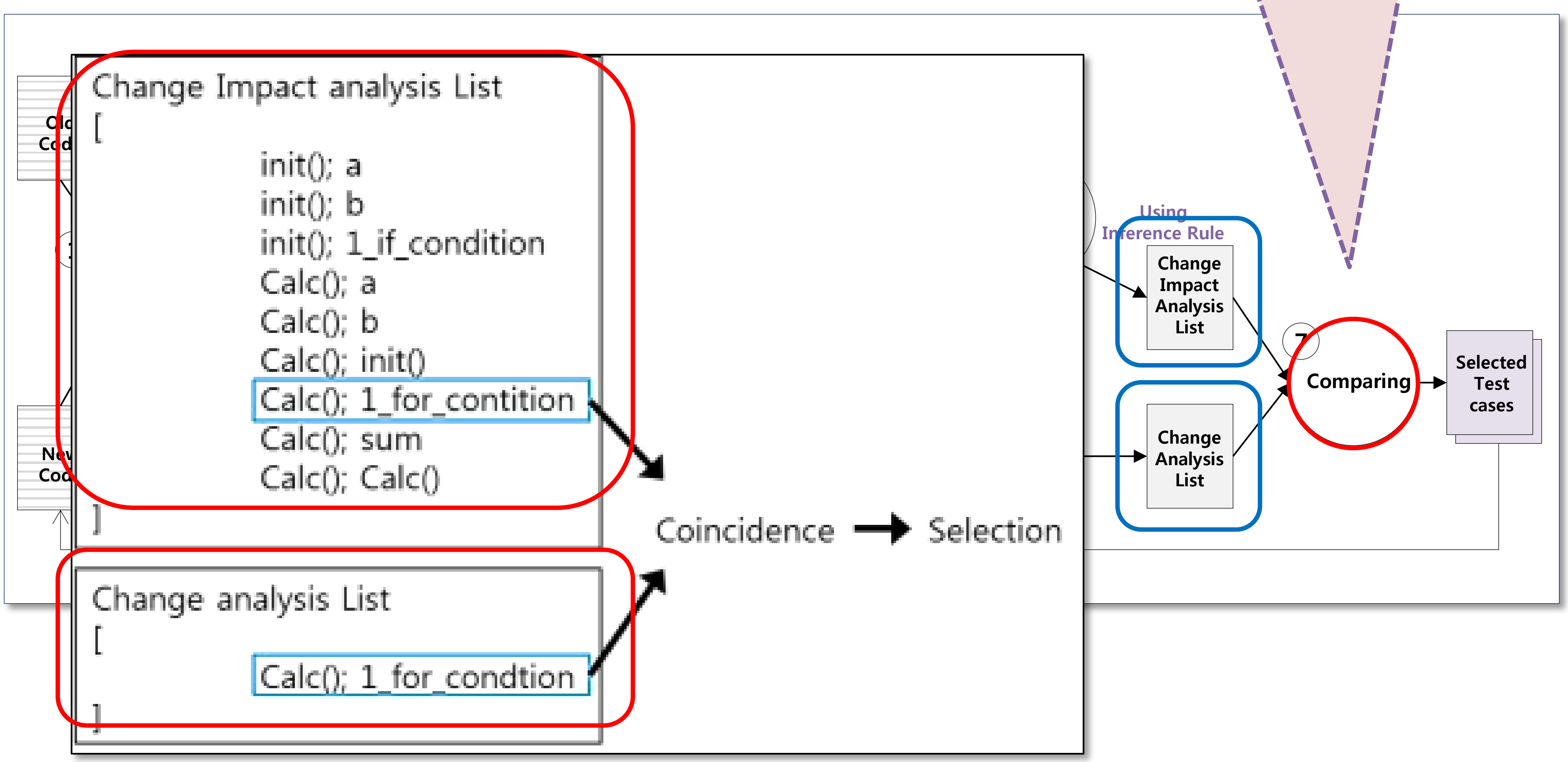
7. Comparing

Same elements are existing?
Just do comparison



7. Comparing

Same elements are existing?
Just do comparison



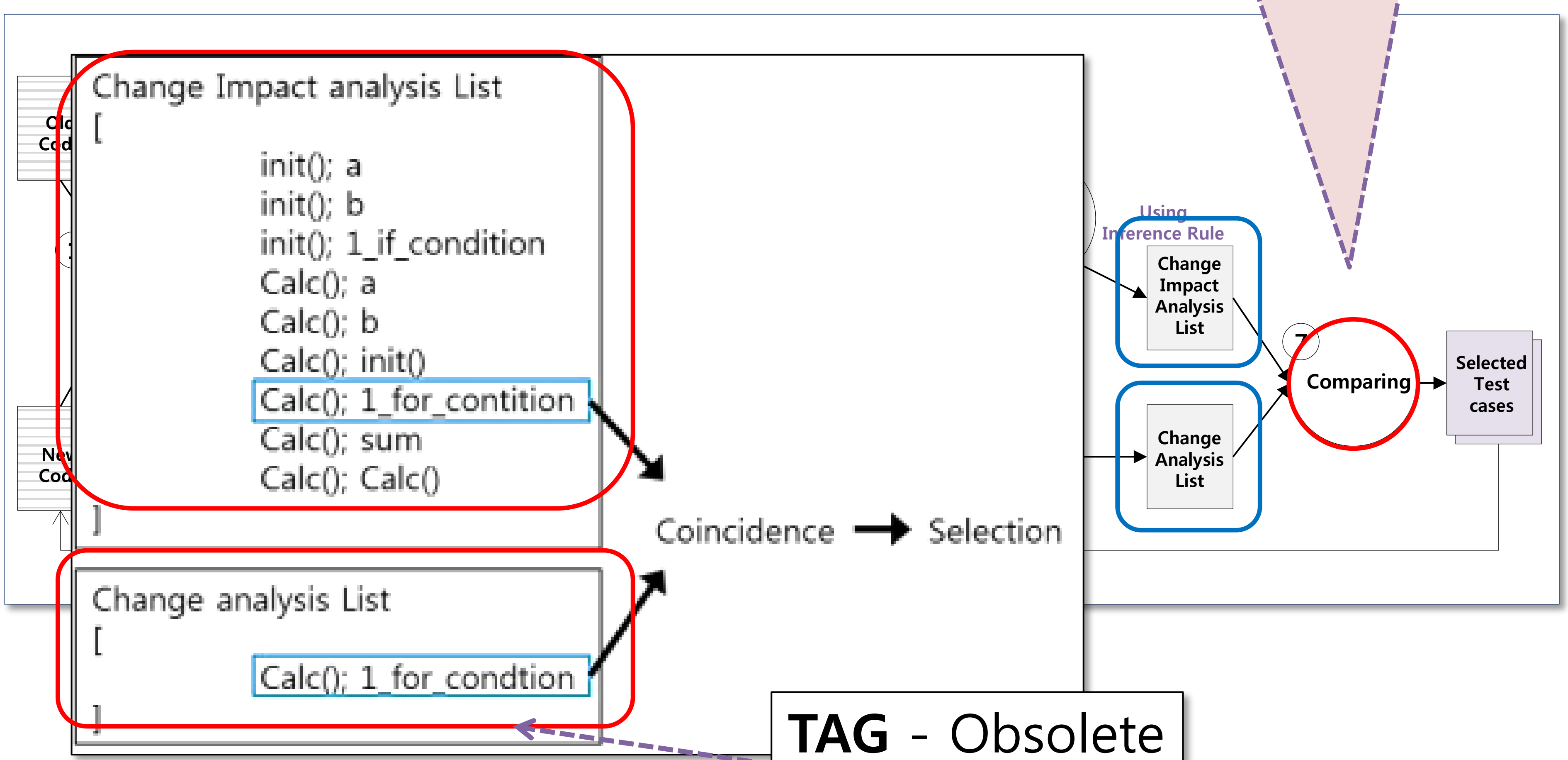
```
Change Impact analysis List
[
                init(); a
                init(); b
                init(); 1_if_condition
                Calc(); a
                Calc(); b
                Calc(); init()
                Calc(); 1_for_contition
                Calc(); sum
                Calc(); Calc()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

```
Change analysis List
[
                Calc(); 1_for_condtion
]
```

Coincidence → Selection

7. Comparing

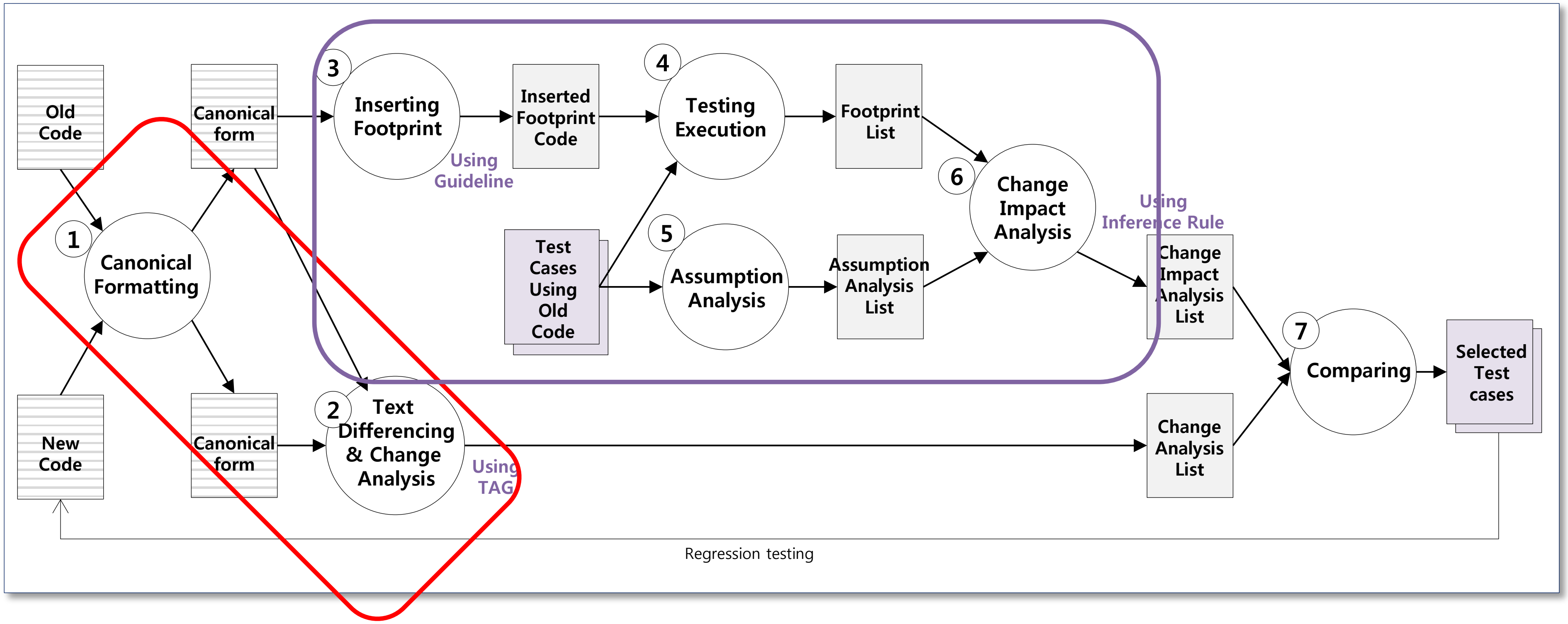
Same elements are existing?
Just do comparison



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

```
Change analysis List
[
              Calc(); 1_for_condtion
]
```

TAG - Obsolete
- NewSpec


Old Code
Canonical form
1
Canonical Formatting
New Code
Canonical form
3
Inserting Footprint
Using Guideline
Inserted Footprint Code
4
Testing Execution
Footprint List
Test Cases Using Old Code
5
Assumption Analysis
Assumption Analysis List
6
Change Impact Analysis
Using Inference Rule
Change Impact Analysis List
2
Text Differencing & Change Analysis
Using TAG
Change Analysis List
7
Comparing
Selected Test cases
Regression testing

Case study

- We performed RT-Selection whit 9 teams code.

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

Case study

- We performed RT-Selection whit 9 teams code.

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

2 test cases that have coincident elements

Case study

- We performed RT-Selection whit 9 teams code.

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

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