Questionnaire for the Survey on Functionally Similar Code Clones

Verena Käfer, Stefan Wagner (University of Stuttgart)
Rainer Koschke (University of Bremen)

Progress: 0%

We are a researcher team from the University of Stuttgart and the University of Bremen in Germany.

We are interested in code created by copy&paste, so-called "code clones". This survey will first give you a short introduction into code clones and especially into Functionally Similar Clones. We then would like to know if you have seen certain clones in your work and what you think about them.

The survey will take about 15 minutes and contains 29 questions.

We will make the responses openly available, but will not include any individual information by which you could be identified.

If you have any questions about this survey, please contact verena.kaefer@informatik.uni-stuttgart.de

Thank you for participating!

Continue

Progress: 3%

A code clone is a piece of code that is similar to another piece of code. In common use there are three types of clones.

Type 1: Similar code fragments except for variation in whitespace, layout and comments.

Type 2: Similar code fragements except for variation in identifiers, literals, types, whitespaces, layouts and comments.

Type 3: Similar code fragments except that some statements may be added or deleted in addition to variation in identifiers, literals, types, whitespaces, layouts or comments.

These clone types can easily be found. We want to investigate a new type of clone, so called "Functionally Similar Clones" (FSCs).

FSCs are two code fragments that provide a similar functionality but can be implemented quite differently.

```
/**
 * @param input
 * @return the reversed input string
 */
public String reverse(String input) {

    StringBuffer buffer = new StringBuffer(input.length());
    for (int i = input.length() - 1; i >= 0; i--) {
        buffer.append(input.charAt(i));
    }
    return buffer.toString();
}

/**
 * @param str
 * @return the reversed input string or null, if the input is null
 */
public static String reverse1(final String str) {
    if (str == null) {
        return null;
    }
    return new StringBuilder(str).reverse().toString();
}
```

**The picture above shows an example of an FSC. Both developers wrote a function to reverse a string independently.
The functionality is similar (both reverse strings) but not identical, as only one method does a null check.**

**While this is only a small sample, FSCs can be much bigger. They can include methods or even classes.
Nevertheless, if the sample is too small there is no need to handle it as a clone.**

Back Continue

Progress: 6%

Please state whether you agree or disagree with the following statements:

	Disagree	Agree	No answer
I have known that FSCs exist before this study.	○	○	●
I have known the term "FSC" before this study.	○	○	●
I have experienced FSCs before.	○	○	●
I have created FSCs on purpose.	○	○	●
I have created FSCs by accident.	○	○	●
I have experienced problems that were caused by an FSC.	○	○	●
I have experienced problems after removing an FSC.	○	○	●

Please elaborate on the given answers, if you would like to add something.

Back Continue

Progress: 12%

In the following we will present some code fragments. Please state if you would refactor these fragments to merge them into one fragment. We would also like to know for which reasons you would like a tool to show you this clone (if at all).

Back Continue

Progress: 15%

Would you refactor these methods to merge them into one method?

```
public static class StringHandling {

    /**
     * @param input
     * @return the first letter of the input string
     */
    public static String getFirstLetter(String input) {
        return String.valueOf(input.charAt(0));
    }

    /**
     * @param s1
     * @param s2
     * @return the sum of the strings' integer values
     */
    public static int sumOfStrings(String s1, String s2) {
        return Integer.valueOf(s1) + Integer.valueOf(s2);
    }

}
```

```
public class StringHandler {

    /**
     * @param input
     * @return the first letter of the input as String
     */
    public String getFirstCharacter(String input) {
        return Character.toString(input.charAt(0));
    }

    /**
     * @param s1
     * @param s2
     * @return the sum of the input strings
     */
    public int sum(String s1, String s2) {
        return Integer.parseInt(s1) + Integer.parseInt(s2);
    }

    /**
     * @param s
     * @return the value of the input multiplied by -1
     */
    public int negativeValueOfString(String s) {
        return Integer.parseInt(s) * -1;
    }
}
```

	No	Probably not	It depends	Probably yes	Yes	No answer
I would refactor these fragments.	○	○	○	○	○	◉

Please explain why or why not and in which context.

I would like to see this clone in a tool because...

	Very unlikely	Unlikely	Likely	Very likely	No answer
...I want to get rid of one of the code fragements.	○	○	○	○	◉
...I want to trace changes.	○	○	○	○	◉
... I want to improve the maintainability.	○	○	○	○	◉
...I want to prevent bugs.	○	○	○	○	◉
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○

Back Continue

Progress: 24%

Would you refactor these methods to merge them into one method?

```
/**
 * @param input
 * @return the reversed input string
 */
public String reverse(String input) {

    StringBuffer buffer = new StringBuffer(input.length());
    for (int i = input.length() - 1; i >= 0; i--) {
        buffer.append(input.charAt(i));
    }
    return buffer.toString();
}

/**
 * @param str
 * @return the reversed input string or null, if the input is null
 */
public static String reverse1(final String str) {
    if (str == null) {
        return null;
    }
    return new StringBuilder(str).reverse().toString();
}
```

	No	Probably not	It depends	Probably yes	Yes	No answer
I would refactor these fragments.	○	○	○	○	○	◉

Please explain why or why not and in which context.

I would like to see this clone in a tool because...

	Very unlikely	Unlikely	Likely	Very likely	No answer
...I want to get rid of one of the code fragements.	○	○	○	○	◉
...I want to trace changes.	○	○	○	○	◉
... I want to improve the maintainability.	○	○	○	○	◉
...I want to prevent bugs.	○	○	○	○	◉
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○

Back Continue

Progress: 33%

Would you refactor these methods to merge them into one method?

```
public static double calculateDistance2D(Point p1, Point p2) {
    int xDiff = p2.x - p1.x;
    int yDiff = p2.y - p1.y;
    int sum = xDiff ^ 2 + yDiff ^ 2;
    return Math.sqrt(sum);
}
```

```
public static double calculateDistance3D(Point p1, Point p2) {
    int xDiff = p2.x - p1.x;
    int yDiff = p2.y - p1.y;
    int zDiff = p2.z - p1.z;
    int sum = xDiff ^ 2 + yDiff ^ 2 + zDiff ^ 2;
    return Math.sqrt(sum);
}
```

	No	Probably not	It depends	Probably yes	Yes	No answer
I would refactor these fragments.	○	○	○	○	○	◉

Please explain why or why not and in which context.

I would like to see this clone in a tool because...

	Very unlikely	Unlikely	Likely	Very likely	No answer
...I want to get rid of one of the code fragements.	○	○	○	○	◉
...I want to trace changes.	○	○	○	○	◉
... I want to improve the maintainability.	○	○	○	○	◉
...I want to prevent bugs.	○	○	○	○	◉
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○

Back Continue

Would you refactor these methods to merge them into one method?

```
/**
 * @param intArr
 * @return sorts the input array using bubble sort
 */
public int[] bubbleSort(int[] intArr) {
    int k;
    for (int i = 0; i < intArr.length - 1; i++) {
        if (intArr[i] < intArr[i + 1]) {
            continue;
        }
        k = intArr[i];
        intArr[i] = intArr[i + 1];
        intArr[i + 1] = k;
        bubbleSort(intArr);
    }
    return intArr;
}

/**
 * @param intArr
 * @return sorts the given array using insertion sort
 */
public int[] insertSort(int[] intArr) {
    int k;
    for (int i = 0; i < intArr.length; i++) {
        for (int j = intArr.length - 1; j > 0; j--) {
            if (intArr[j - 1] > intArr[j]) {
                k = intArr[j];
                intArr[j] = intArr[j - 1];
                intArr[j - 1] = k;
            }
        }
    }
    return intArr;
}
```

	No	Probably not	It depends	Probably yes	Yes	No answer
I would refactor these fragments.	○	○	○	○	○	◉

Please explain why or why not and in which context.

I would like to see this clone in a tool because...

	Very unlikely	Unlikely	Likely	Very likely	No answer
...I want to get rid of one of the code fragements.	○	○	○	○	◉
...I want to trace changes.	○	○	○	○	◉
... I want to improve the maintainability.	○	○	○	○	◉
...I want to prevent bugs.	○	○	○	○	◉
	○	○	○	○	○
	○	○	○	○	○
	○	○	○	○	○

Back Continue

ROGATOR

Progress: 52%

Now, we would like to know more details about your experiences with FSCs.

Back Continue

ROGATOR

Progress: 79%

How old are you?

What is your gender?

- Male
- Female
- Other

Where are you from?

- Germany
- Austria
- Switzerland
- USA
- Other

What is your level of programming experience?

- < 1 year
- 2-5 years
- 6-10 year
- > 10 years

What are your main programming languages?

- JavaScript
- Java
- Python
- Ruby
- C++
- C#
- C
- Go
- Other
- Other

FSCs are a problem in my project.					

What kind of software are you developing?

Back Continue

Progress: 76%

Thank you for your answers so far. We will now ask some demographic questions.

Back Continue