

Improving Software Maintainability through Automated Refactoring of Code Clones

Author: Simon Baars

Company Supervisor: Xander Schrijen

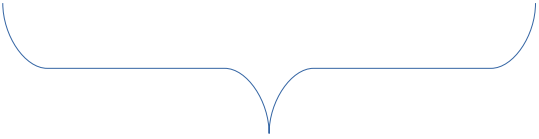
Host Company: Software Improvement Group

Academic Supervisor: Ana Opreescu

Second Reader: Clemens Grelck

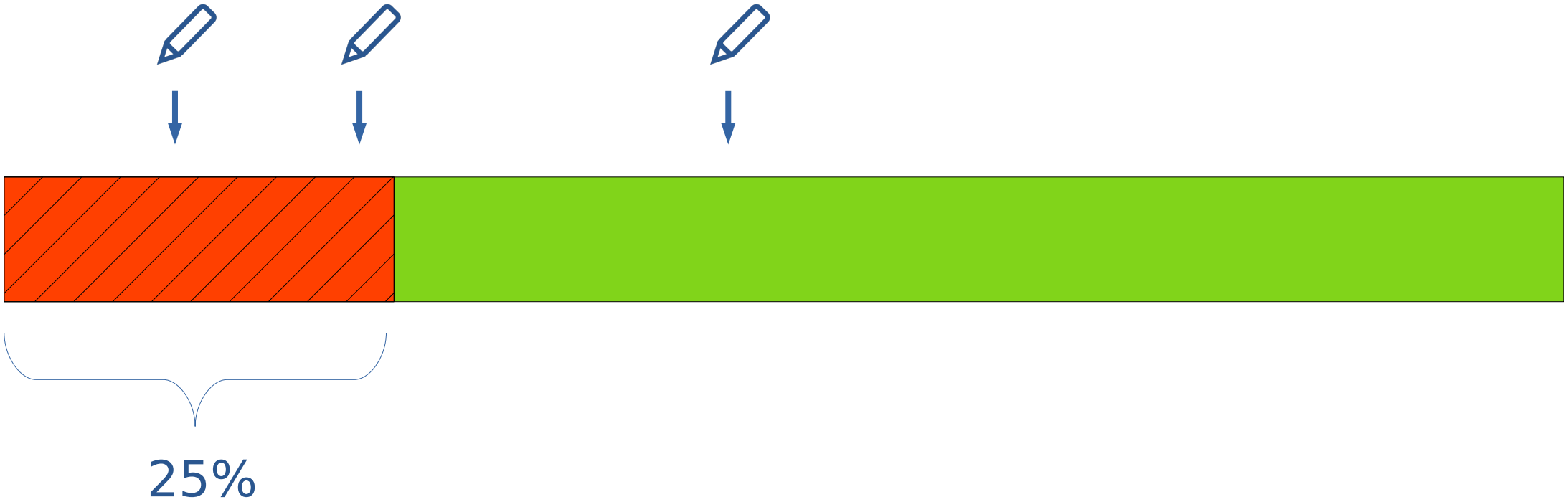
Location: Amsterdam

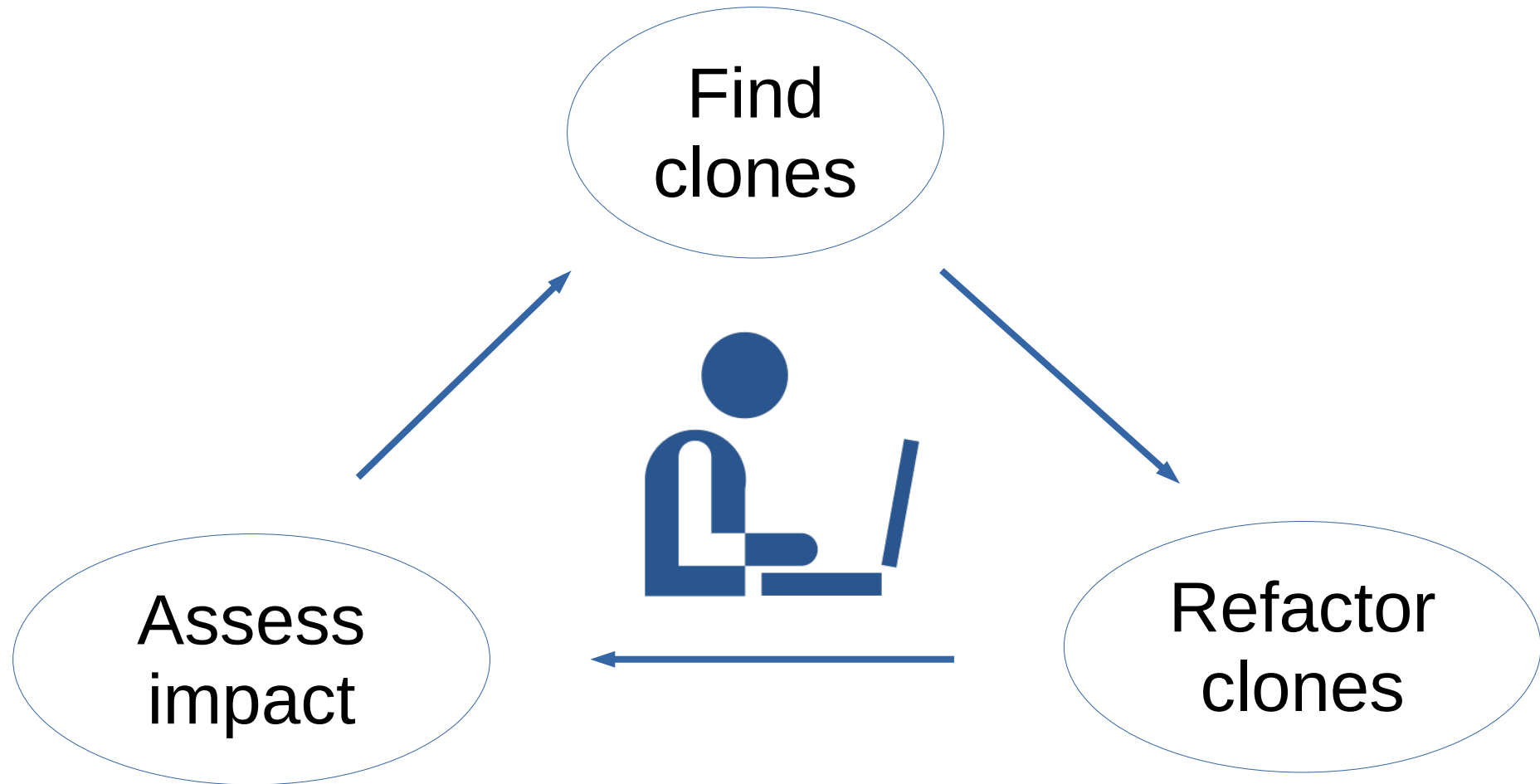
Date: 28 August 2019



25%

One “simple” change...





Find
clones



Refactor
clones

Assess
impact

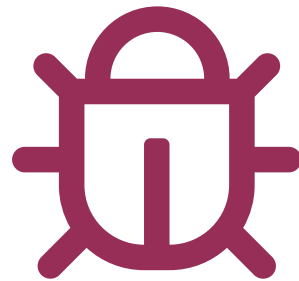


Find
clones



Refactor
clones

Assess
impact

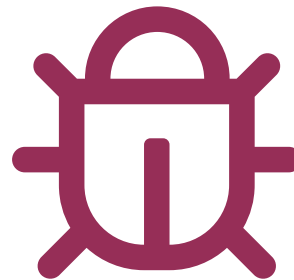


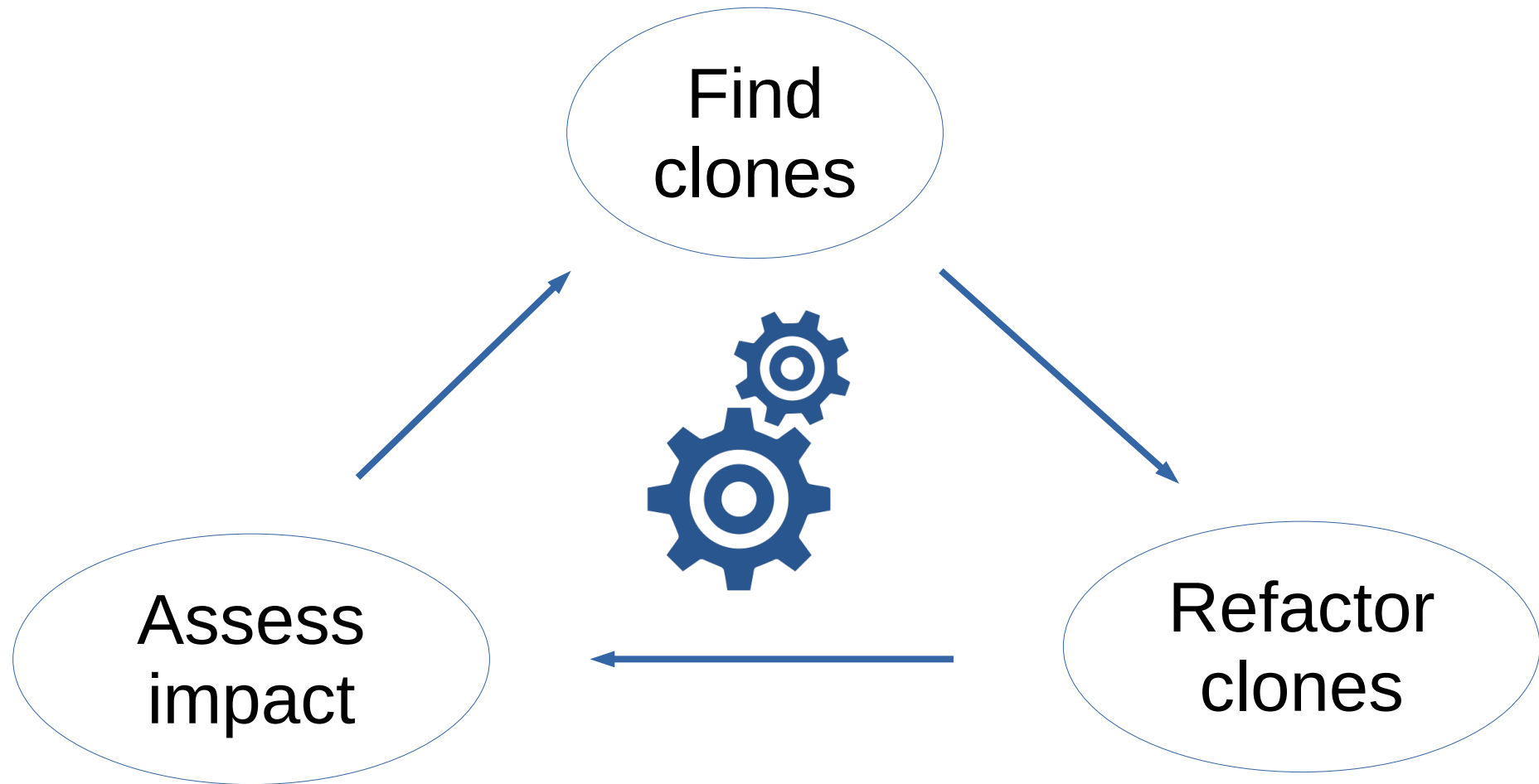
Find
clones



Refactor
clones

Assess
impact





RQ1.

*How can we define clone types such that they
can be automatically refactored?*

RQ1.

*How can we define clone types such that they **can** be automatically refactored?*

RQ2.

*How can we prioritize refactoring opportunities based on the **context** of clones?*

RQ1.

*How can we define clone types such that they **can** be automatically refactored?*

RQ2.

*How can we prioritize refactoring opportunities based on the **context** of clones?*

RQ3.

*What are the discriminating factors to decide when a clone **should** be refactored*

RQ1.

*How can we define clone types such that they **can** be automatically refactored?*

RQ2.

*How can we prioritize refactoring opportunities based on the **context** of clones?*

RQ3.

*What are the discriminating factors to decide when a clone **should** be refactored*

Clone Types

Each clone type allows more variance between cloned fragments.



Type 1

Type 2

Type 3

Type 1.

Textually identical code fragments except for variations in whitespace and comments.

Type 1.

Textually identical code fragments except for variations in whitespace and comments.

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.List;

public class AdditionUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.List;

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concatTen(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Type 1R.

Contextually & textually identical code fragments except for variations in layout and comments.

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.List;

public class AdditionUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.List;

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concatTen(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```


Type 1R.

Contextually & textually identical code fragments except for variations in layout and comments.

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.java.util.List

public class AdditionUtils
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.Ljava.awt.List

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concatTen(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Type 1R.

Contextually & textually identical code fragments except for variations in layout and comments.

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.List;

java.util.List.add(java.lang.Object)

public class AdditionUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.List;

java.awt.List.add(java.lang.String)

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concatTen(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Type 1R.

Contextually & textually identical code fragments except for variations in layout and comments.

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.List;

public class AdditionUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Diagram illustrating the code fragment for `AdditionUtils`. The code is annotated with type information (int) and a yellow dashed box highlights the method body.

```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.List;

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concat(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Diagram illustrating the code fragment for `StringUtils`. The code is annotated with type information (`java.lang.String`) and a yellow dashed box highlights the method body.

Type 1R.

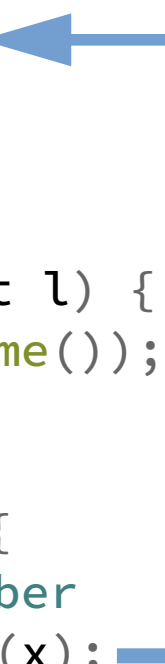
Retrieving contextual information can be challenging!

```
package com.sb.cryo.addition;

import com.notificationlib.*;
import java.util.List;

public class AdditionUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public int addTen(int x) {
        x = x + 10; // add number
        Notifier.notifyChanged(x);
        return x;
    }
}
```



```
package com.sb.cryo.util;

import com.sb.cryo.notifier.*;
import java.awt.List;

public class StringUtils {
    public void addToList(List l) {
        l.add(getClass().getName());
    }

    public String concatTen(String x) {
        x = x + 10; // concat number
        Notifier.notifyChanged(x);
        return x;
    }
}
```

Type 1R Summarized

In addition to type 1 rules, we compare contextual information of:

- 1.** Type references (Fully Qualified Identifier)
- 2.** Variable references (type)
- 3.** Method references (Fully Qualified Signature)

Retrieving fully qualified identifiers/signatures can be challenging!

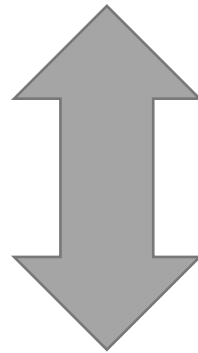
Type 2.

Structurally/syntactically identical fragments except for variations in identifiers, literals, types, layout, and comments.

Type 2.

Structurally/syntactically identical fragments except for variations in identifiers, literals, types, layout, and comments.

```
public boolean containsOnlyRedCircles(List<Circle> circles){  
    return circles.stream().allMatch(Shape::isRed);  
}
```



```
public Apple getEdibleApple(FruitBasket<Apple> basket){  
    return basket.getFruit().getApple(Fruit::notEaten);  
}
```

Type 2R.

Type 1R clones except for variations in a controlled set of expressions.

No design tradeoff

- Type declaration names
 - Method names
- Variable names/references (sometimes)

Design tradeoff

- Literals (of the same type)
- Variable references (sometimes)
- Called methods (same type & parameters)

Type 2R clones

> No design tradeoff

- **Type declaration names**
- **Method names**
- Variable names/references (sometimes)

> Design tradeoff

- Literals (of the same type)
- Variable references (sometimes)
- Called methods (same type & parameters)

```
public class A {  
    public void doA() {  
        print("hello");  
    }  
}  
  
public class B {  
    public void doB() {  
        print("hello");  
    }  
}
```

Type 2R clones

> No design tradeoff

- Type declaration names
- Method names
- **Variable names/references (sometimes)**

> Design tradeoff

- Literals (of the same type)
- Variable references (sometimes)
- Called methods (same type & parameters)

```
public void doA() {  
    String message = "hello";  
    print(message);  
}  
  
public void doB() {  
    String hello = "hello";  
    print(hello);  
}
```

Type 2R clones

> No design tradeoff

- Type declaration names
- Method names
- Variable names/references (sometimes)

```
// Original
void doABC(){
    doA();
    doB("abc");
    doC();
}

void doDEF(){
    doA();
    doB("def");
    doC();
}
```

> Design tradeoff

- **Literals (of the same type)**
- Variable references (sometimes)
- Called methods (same type & parameters)

```
// Refactored
void doABC(){
    doThis("abc");
}

void doDEF(){
    doThis("def");
}

void doThis(String letters){
    doA();
    doB(letters);
    doC();
}
```

Type 2R clones

> No design tradeoff

- Type declaration names
- Method names
- Variable names/references (sometimes)

```
// Original
String abc = "abc";
String def = "def";
void doABC(){
    doA();
    doB(abc);
    doC();
}
void doDEF(){
    doA();
    doB(def);
    doC();
}
```

> Design tradeoff

- Literals (of the same type)
- **Variable references (sometimes)**
- Called methods (same type & parameters)

```
// Refactored
String abc = "abc";
String def = "def";
void doABC(){
    doThis(abc);
}
void doDEF(){
    doThis(def);
}
void doThis(String letters){
    doA();
    doB(letters);
    doC();
}
```

Type 2R clones

> No design tradeoff

- Type declaration names
- Method names
- Variable names/references (sometimes)

```
// Original
void doABC(){
    doA();
    doB();
    doC();
}

void doADC(){
    doA();
    doD();
    doC();
}
```

> Design tradeoff

- Literals (of the same type)
- Variable references (sometimes)
- **Called methods (same type & parameters)**

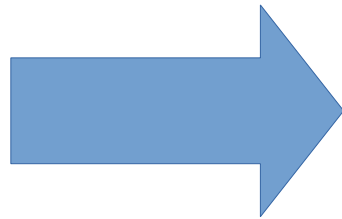
```
// Refactored
void doABC(){
    doThis(this::doB);
}

void doADC(){
    doThis(this::doD);
}

void doThis(Runnable r){
    doA();
    r.run();
    doC();
}
```

$$\text{T2R Variability} = \frac{\text{Number of different expressions}}{\text{Total number of expressions in clone instance}} * 100$$

```
void doABC(){  
    doA();  
    doB();  
    doC();  
}  
  
void doADC(){  
    doA();  
    doD();  
    doC();  
}
```



$$\frac{1}{3} = 33\%$$

Type 3.

Copied fragments with further modifications. Statements can be changed, added or removed in addition to variations in identifiers, literals, types, layout, and comments.

```
void doCwithA(){  
    int a = getA();  
    doC(a);  
}  
  
void multiplyA(){  
    int a = getA();  
    a *= 5;  
    doC(a);  
}
```

Type 3R.

Type 2R clones with optional gaps of non cloned statements.

```
// Original
void doCwithA(){
    int a = getA();
    doC(a);
}

void multiplyA(){
    int a = getA();
    a *= 5;
    doC(a);
}
```

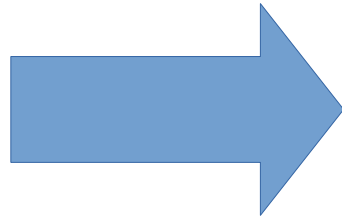
```
// Refactored
void doCwithA(){
    modifyA(false);
}

void multiplyA(){
    modifyA(true);
}

void modifyA(boolean
multiply){
    int a = getA();
    if(multiply) a *= 5;
    doC(a);
}
```


$$\text{T3R Gap Size} = \frac{\text{Number of statements in gap}}{\text{Number of statements in clones}} * 100$$

```
// Original  
void doCwithA(){  
    int a = getA();  
    doC(a);  
}  
  
void multiplyA(){  
    int a = getA();  
    a *= 5;  
    doC(a);  
}
```



$$\frac{1}{2} = 50\%$$

Summarized

Type 1R.

Contextually & textually identical code fragments except for variations in layout and comments.

Type 2R.

Type 1R clones except for variations in a controlled set of expressions.

Type 3R.

Type 2R clones with optional gaps of non cloned statements.

$$\text{Type 1R} \subseteq \text{Type 2R} \subseteq \text{Type 3R}$$

Clone Context

Determining how a clone should be refactored.

Relation

- Common Class
- Common Hierarchy
- Common Interface
 - Unrelated

Location

- Method Level
 - Class Level
- Interface Level
 - Enum Level

Contents

- Full Declaration
 - Partial Body
 - Only Fields
- Several Methods
 - Other

Refactorability

Determining whether a clone can be refactored using “Extract Method”.

Partial Block

```
if(result == 1){  
    println("Error!");  
    handleError(result);  
}  
doSomething();  
if(result == 1){  
    println("Error!");  
}
```

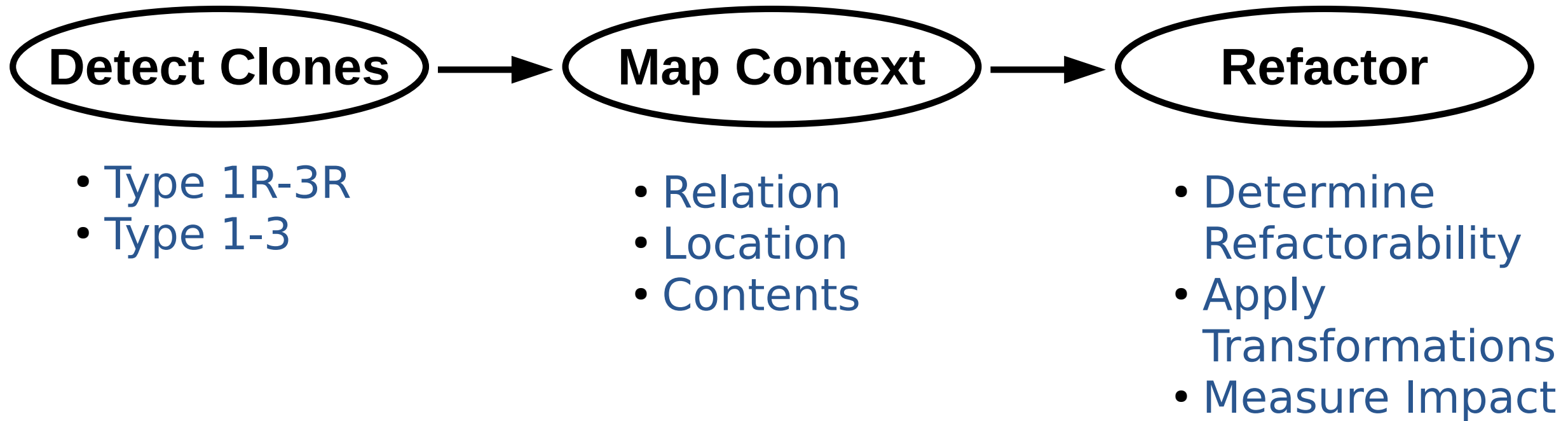
Top-Level Node is not a Statement

```
try {  
    doSomethingDangerous();  
} catch (DangerException e) {  
    println("Danger!");  
}  
doSomething();  
try {  
    doSomethingCool();  
} catch (DangerException e) {  
    println("Danger!");  
}
```

CloneRefactor



CloneRefactor



CloneRefactor



Refactoring

Determining the impact of refactoring the clone.

Characteristics

- Clone Size
 - Relation
- Return Category
- Parameters

Metrics

- Duplication
 - Volume
- Complexity
- Number of Parameters

Risk Profiles

- Low Risk
- Moderate Risk
 - High Risk
- Very High Risk

Current Repository
joda-time

Current Branch
CloneRefactor

Publish repository
Publish this repository to GitHub

Changes

History

Select Branch to Compare...

Formatted MonthDay
CloneRefactor authored and Simon Baar...

Formatted BasePartial
CloneRefactor authored and Simon Baar...

Created unified method in ISODateTimeFormat
CloneRefactor authored and Simon Baar...

Created unified method in LocalDate
CloneRefactor authored and Simon Baar...

Formatted LocalDate
CloneRefactor authored and Simon Baar...

Created unified method in ZoneInfoCo...
CloneRefactor authored and Simon Baar...

Formatted ZoneInfoCompiler
CloneRefactor authored and Simon Baar...

Created unified method in BaseDateTim...
CloneRefactor authored and Simon Baar...

Formatted BaseDateTimeField
CloneRefactor authored and Simon Baar...

Created unified method in ISODateTime...
CloneRefactor authored and Simon Baar...

Created unified method in ISODateTime...
CloneRefactor authored and Simon Baar...

Formatted ISODateTimeFormat
CloneRefactor authored and Simon Baar...

Initial commit
CloneRefactor authored and Simon Baar...

Created unified method in ISODateTimeFormat

CloneRefactor authored and Simon Baars committed 2e52b0e 1 changed file

CloneRefactor refactored a clone class with 2 clone instances. For the common code we created a new method and named this method "cloneRefactor0". These clone instances have an Same Method relation with each other. The newly created method has been placed in ISODateTimeFormat. Each duplicated fragment has been replaced with a call to this method.

Expand

.../ISODateTimeFormat.java

318	-	bld.appendLiteral('W');
319	-	bld.appendLiteral('-');
320	-	bld.appendDayOfWeek(1);
318	+	cloneRefactor0(bld);
321	319	} else {
322	320	// YYYY/YYYY
323	321	reducedPrec = true;
		@@ -338,9 +336,7 @@ public class ISODateTimeFormat {
338	336	} else if (fields.remove(DateTimeFieldType.dayOfWeek())) {
339	337	// -W-D/-W-D
340	338	bld.appendLiteral('-');
341	-	bld.appendLiteral('W');
342	-	bld.appendLiteral('-');
343	-	bld.appendDayOfWeek(1);
	339	cloneRefactor0(bld);
344	340	}
345	341	return reducedPrec;
346	342	}
		@@ -1700,4 +1696,10 @@ public class ISODateTimeFormat {
1700	1696	return ze;
1701	1697	}
1702	1698	}
	1699	+
	1700	+ private void cloneRefactor0(DateTimeFormatterBuilder bld) {
	1701	+ bld.appendLiteral('W');
	1702	+ bld.appendLiteral('-');
	1703	+ bld.appendDayOfWeek(1);
	1704	+ }

Created unified method in ISODateTimeFormat

 CloneRefactor authored and Simon Baars committed  2e52b0e  1 changed file

CloneRefactor refactored a clone class with 2 clone instances. For the common code we created a new method and named this method "cloneRefactor0". These clone instances have an Same Method relation with each other. The newly created method has been placed in ISODateTimeFormat. Each duplicated fragment has been replaced with a call to this method.

== System Quality Metrics ==

Total Cyclomatic Complexity increased by 1 from 6994 to 6995.

Total Unit Interface Size increased by 1 from 3715 to 3716.

Total Unit Line Size increased by 1 from 23024 to 23025.

Total Unit Token Size decreased by 2 from 153190 to 153188.

Total Nodes decreased by 1 from 18866 to 18865.

Duplicated Nodes decreased by 6 from 502 to 496.

Duplicated Tokens decreased by 42 from 5064 to 5022.

Duplicated Lines decreased by 6 from 504 to 498.

== Risk Profiles ==

Unit Complexity

Created a new method with a low risk Unit Complexity of 1.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 8 to 8 Unit Complexity. This did not influence the risk category of this method, it is still low risk.

Line Volume

Created a new method with a low risk Line Volume of 5.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 47 to 39 Line Volume. This did not influence the risk category of this method, it is still high risk.

Token Volume

Created a new method with a low risk Token Volume of 30.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 303 to 271 Token Volume. This decreased the risk category of this method from high to moderate

The new method has a low risk Unit Interface Size of 1.

Duplication went from 2.66% to 2.63%. This did not influence the risk category of the duplication in this codebase, it is still low risk.

Downloaded from <http://ajph.org/> at University of California - San Francisco on June 11, 2015

Handelsregister eingetragen & kann nach § 15 Abs. 1 Nr. 1 HGB vertreten werden. Die hier vorgesehene Vertretung ist zulässig & der Vorstand hat keine Vertretungsbefugnis. Nach § 15 Abs. 1 Nr. 1 HGB kann der Vorstand auch durch einen oder mehrere Vorständeure vertreten werden. Die hier vorgesehene Vertretung ist zulässig & der Vorstand hat keine Vertretungsbefugnis. Nach § 15 Abs. 1 Nr. 1 HGB kann der Vorstand auch durch einen oder mehrere Vorständeure vertreten werden. Die hier vorgesehene Vertretung ist zulässig & der Vorstand hat keine Vertretungsbefugnis.

© 2004 Blackwell Publishing Ltd, *Journal of Internal Medicine* 255: 105–112

[illegible]

== Risk Profiles ==

[illegible]

The authors have nothing to disclose regarding potential conflicts of interest. The authors confirm that they did not perform or analyze any part of the study.

[illegible]

Token Volume

Created a new method with a low risk Token Volume of 30.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 303 to 271 Token Volume. This decreased the risk category of this method from high to moderate

[illegible]

Created unified method in ISODateTimeFormat

 CloneRefactor authored and Simon Baars committed  2e52b0e  1 changed file

CloneRefactor refactored a clone class with 2 clone instances. For the common code we created a new method and named this method "cloneRefactor0". These clone instances have an Same Method relation with each other. The newly created method has been placed in ISODateTimeFormat. Each duplicated fragment has been replaced with a call to this method.

== System Quality Metrics ==

Total Cyclomatic Complexity increased by 1 from 6994 to 6995.

Total Unit Interface Size increased by 1 from 3715 to 3716.

Total Unit Line Size increased by 1 from 23024 to 23025.

Total Unit Token Size decreased by 2 from 153190 to 153188.

Total Nodes decreased by 1 from 18866 to 18865.

Duplicated Nodes decreased by 6 from 502 to 496.

Duplicated Tokens decreased by 42 from 5064 to 5022.

Duplicated Lines decreased by 6 from 504 to 498.

== Risk Profiles ==

Unit Complexity

Created a new method with a low risk Unit Complexity of 1.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 8 to 8 Unit Complexity. This did not influence the risk category of this method, it is still low risk.

Line Volume

Created a new method with a low risk Line Volume of 5.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 47 to 39 Line Volume. This did not influence the risk category of this method, it is still high risk.

Token Volume

Created a new method with a low risk Token Volume of 30.

Removing duplicate blocks changed 1 methods.

The method "dateByWeek(DateTimeFormatterBuilder, Collection, boolean, boolean)" went from 303 to 271 Token Volume. This decreased the risk category of this method from high to moderate

The new method has a low risk Unit Interface Size of 1.

Duplication went from 2.66% to 2.63%. This did not influence the risk category of the duplication in this codebase, it is still low risk.

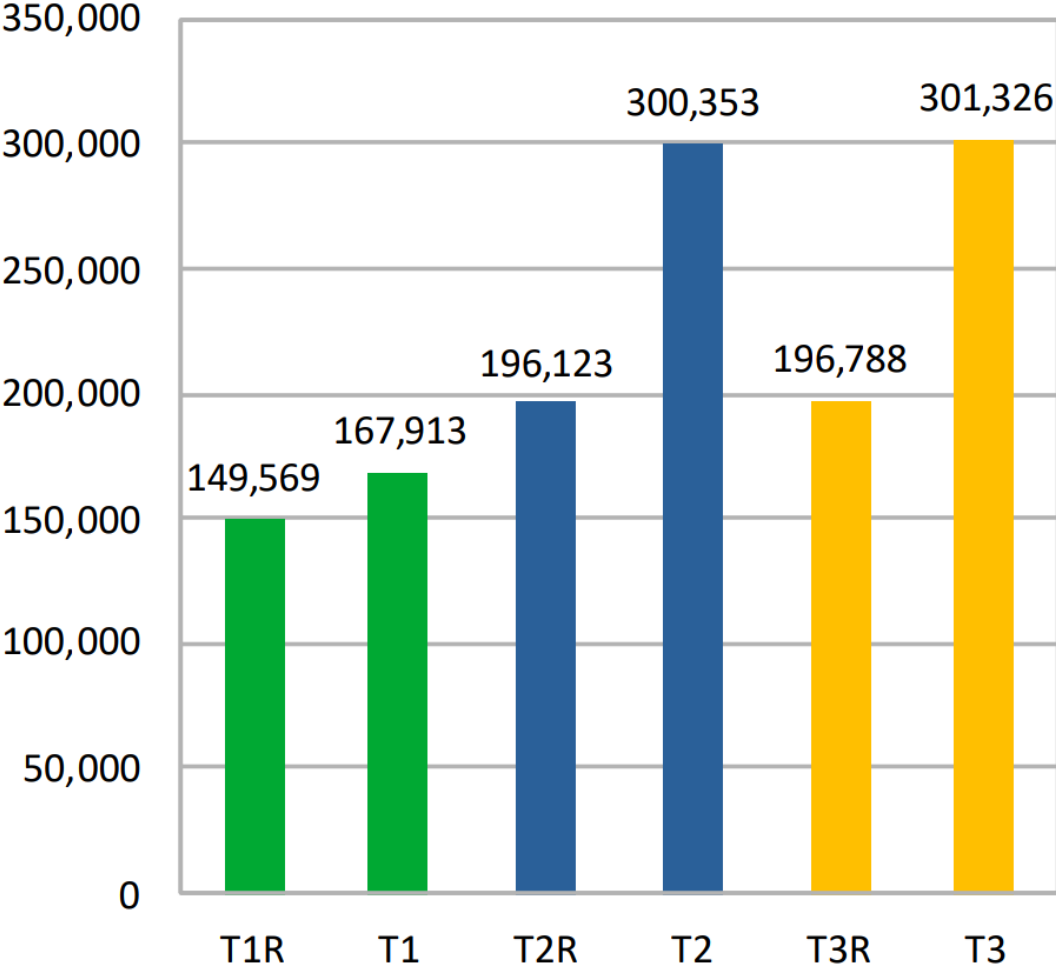
Experiments

- > GitHub Corpus with 2.267 Java projects
- > Experiments:
 - > Clone Types
 - > Context
 - > Refactorability
 - > Thresholds



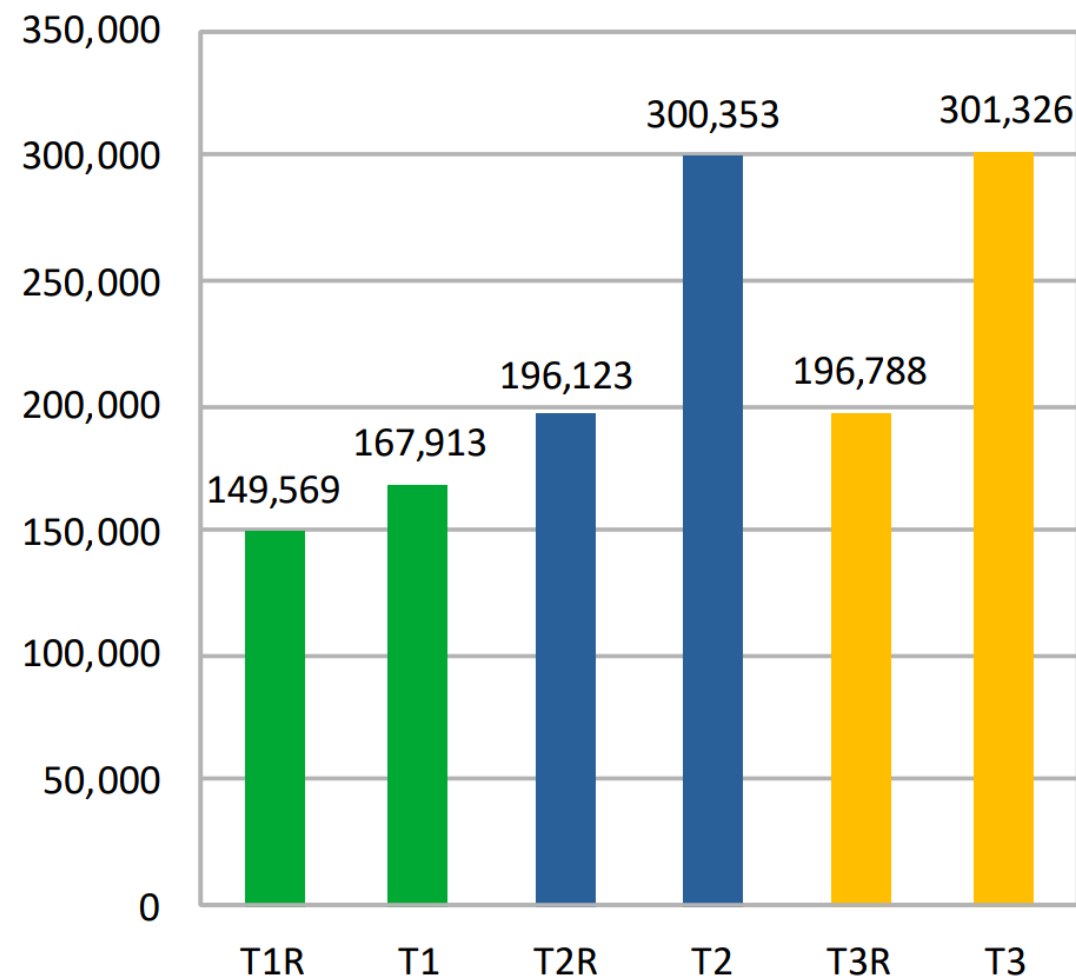
Clone Types

Amount of clones found

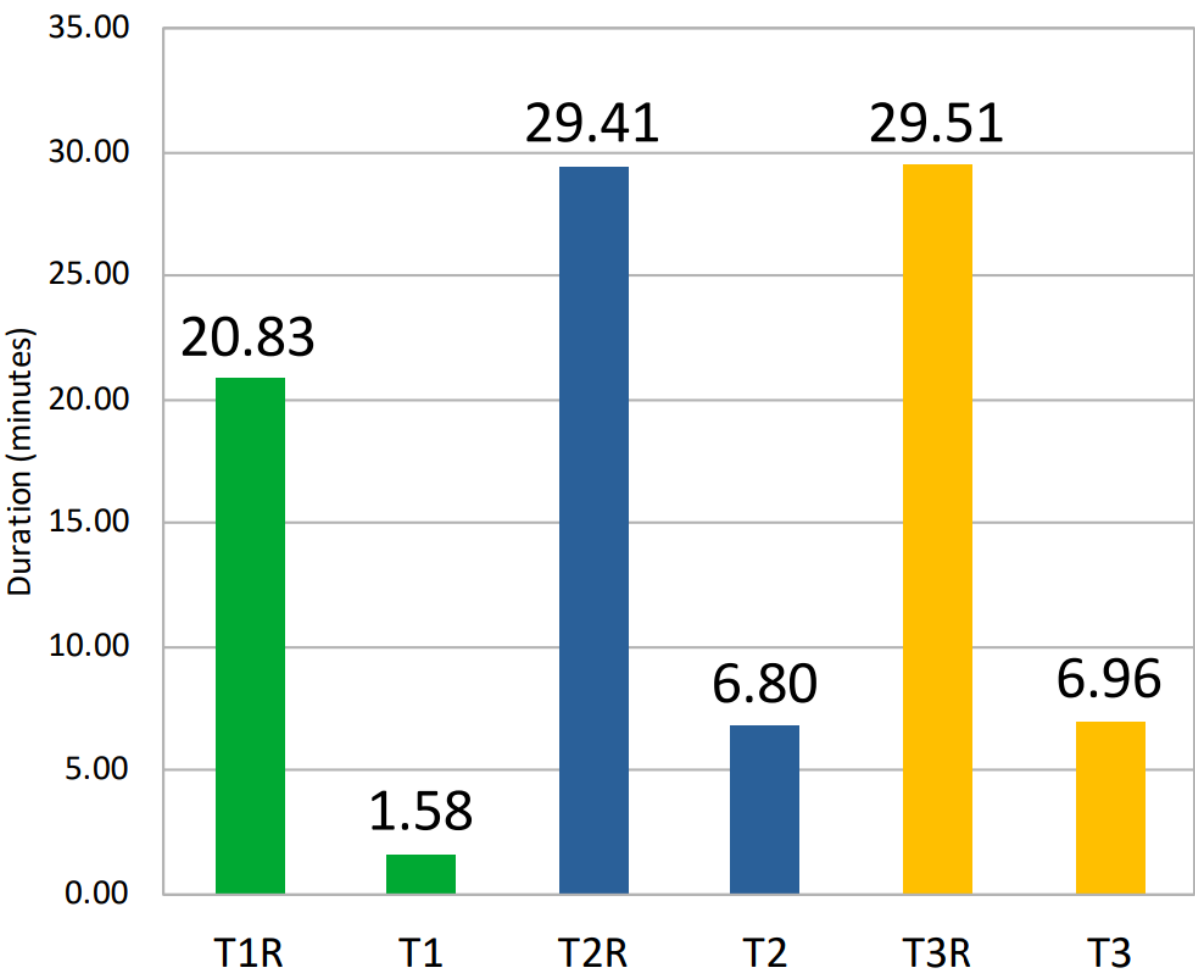


Clone Types

Amount of clones found



Performance



Relation

<i>Category</i>	<i>Relation</i>	<i>Clone Classes</i>	<i>%</i>	<i>Total</i>	<i>%</i>
Common Class	Same Class	22,893	26.8%	31,848	37.2%
	Same Method	8,955	10.5%		
Common Hierarchy	Sibling	15,588	18.2%	20,342	23.8%
	Superclass	2,616	3.1%		
	First Cousin	1,219	1.4%		
	Common Hierarchy	720	0.8%		
	Ancestor	199	0.2%		
Unrelated	No Direct Superclass	10,677	12.5%	20,314	23.7%
	External Superclass	4,525	5.3%		
	External Ancestor	3,347	3.9%		
	No Indirect Superclass	1,765	2.1%		
Common Interface	Same Direct Interface	7,522	8.8%	13,074	15.3%
	Same Indirect Interface	5,552	6.5%		

Location

<i>Category</i>	<i>Clone instances</i>	<i>%</i>
Method Level	232,545	78.43%
Class Level	50,402	17.00%
Constructor Level	10,039	3.39%
Interface Level	2,693	0.91%
Enum Level	788	0.27%

Contents

<i>Category</i>	<i>Contents</i>	<i>Clone instances</i>	<i>Total</i>		
Partial	Method Body	219,540	74.05%	229,521	77.42%
	Constructor Body	9,981	3.37%		
Other	Several Methods	22,749	7.67%	53,773	18.14%
	Only Fields	17,700	5.97%		
	Other	13,324	4.49%		
Full	Full Method	12,990	4.38%	13,173	4.44%
	Full Interface	64	0.02%		
	Full Constructor	58	0.02%		
	Full Class	37	0.01%		
	Full Enum	24	0.01%		

Refactorability

<i>Category</i>	<i>All</i>	<i>% (All)</i>
Can be Extracted	24,157	28.2%
Is not in a Method Body	21,625	25.3%
Top-level AST-Node is not a Statement	19,887	23.2%
Spans Part of a Block	12,964	15.2%
Multiple Return Values	5,622	6.6%
Complex Control Flow	1,106	1.3%
Overlap in Clone Class	147	0.2%
Not in Class or Interface	70	0.1%

Refactoring Experiments

> **12.683** refactorings performed

Characteristics

- Clone Size
 - Relation
- Return Category
 - Parameters

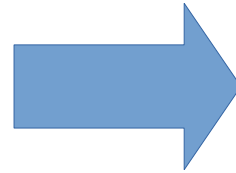
Metrics

- Δ Duplication
 - Δ Volume
 - Δ Complexity
- Δ Number of Parameters

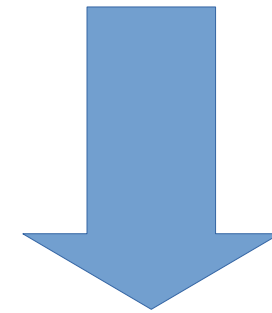
Calculate Maintainability Score

Aggregating the used maintainability metrics to give each refactoring a score.

- Δ Duplication
- Δ Volume
- Δ Complexity
- Δ Number of Parameters

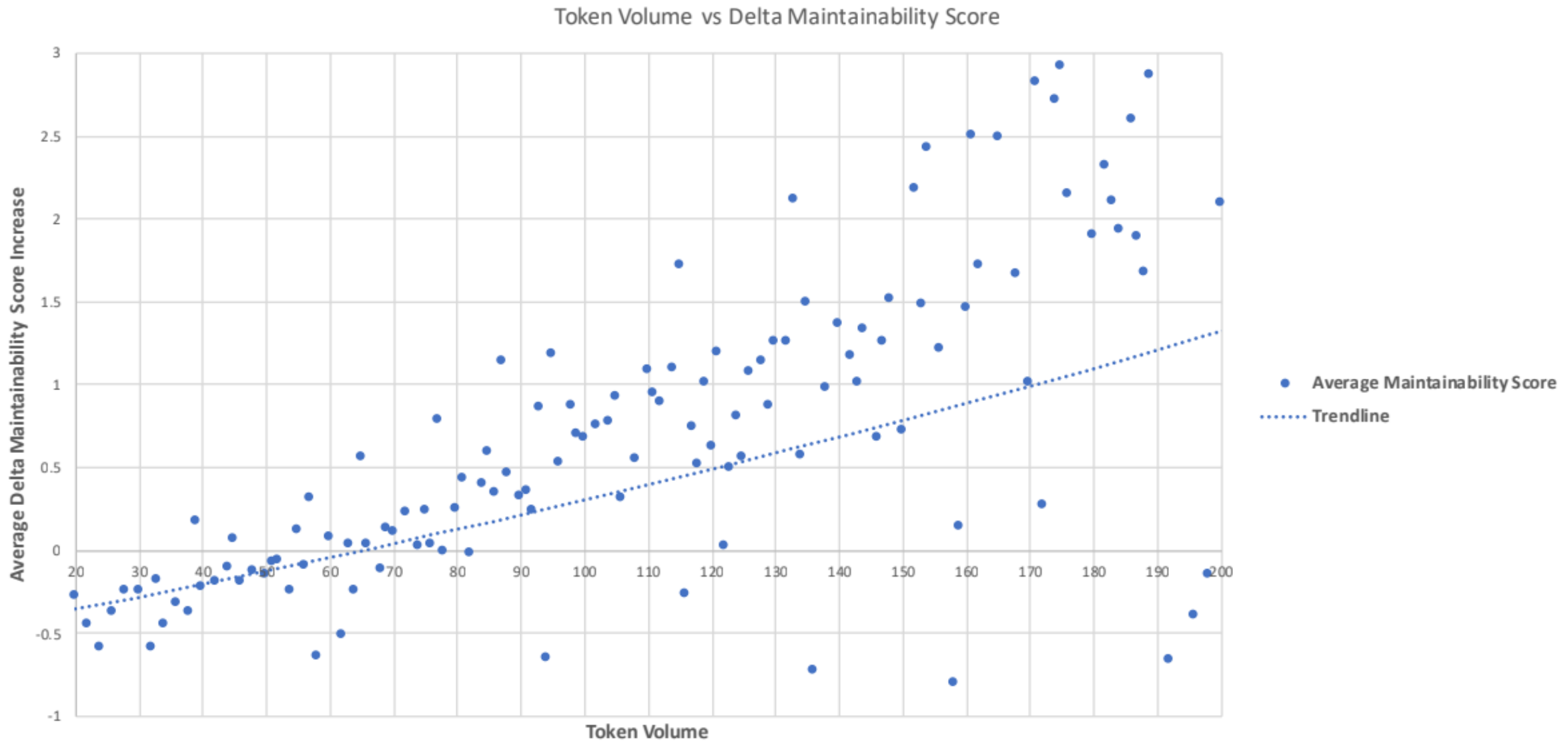


$$N_{metric} = \frac{\Delta X - \mu}{\sigma}$$



$$\text{Maintainability Score} = N_{duplication} + N_{complexity} + N_{volume} + N_{parameters}$$

Token Volume



Relation

<i>Relation</i>	<i>Duplication</i>	<i>Complexity</i>	<i>Parameters</i>	<i>Volume</i>	<i>#</i>	<i>Score</i>
Common Hierarchy	-66.33	0.73	1.20	-8.85	2,202	0.23
Superclass	-64.48	0.79	0.94	-7.22	229	0.42
Sibling	-70.07	0.69	1.28	-10.97	1,722	0.23
Same Hierarchy	-44.18	0.95	0.89	1.54	87	0.10
First Cousin	-42.69	0.89	0.93	4.86	144	0.02
Ancestor	-32.75	1.00	0.75	11.00	20	-0.03
Common Interface	-47.06	0.83	1.04	4.50	1,044	-0.02
Same Indirect Interface	-37.08	0.93	0.82	9.96	487	-0.01
Same Direct Interface	-55.79	0.75	1.24	-0.28	557	-0.02
Common Class	-52.42	0.87	1.13	1.47	7,239	-0.02
Same Class	-51.85	0.86	1.03	3.36	4,874	0.04
Same Method	-53.60	0.90	1.32	-2.44	2,365	-0.15
Unrelated	-45.86	0.88	1.08	9.56	2,198	-0.15
No Direct Superclass	-52.24	0.84	1.12	6.04	811	-0.06
External Superclass	-47.09	0.87	1.13	8.77	697	-0.17
External Ancestor	-35.73	0.93	0.95	14.58	586	-0.21
No Indirect Superclass	-44.89	0.84	1.18	14.08	104	-0.30
Grand Total	-53.26	0.84	1.12	1.33	12,683	0.00

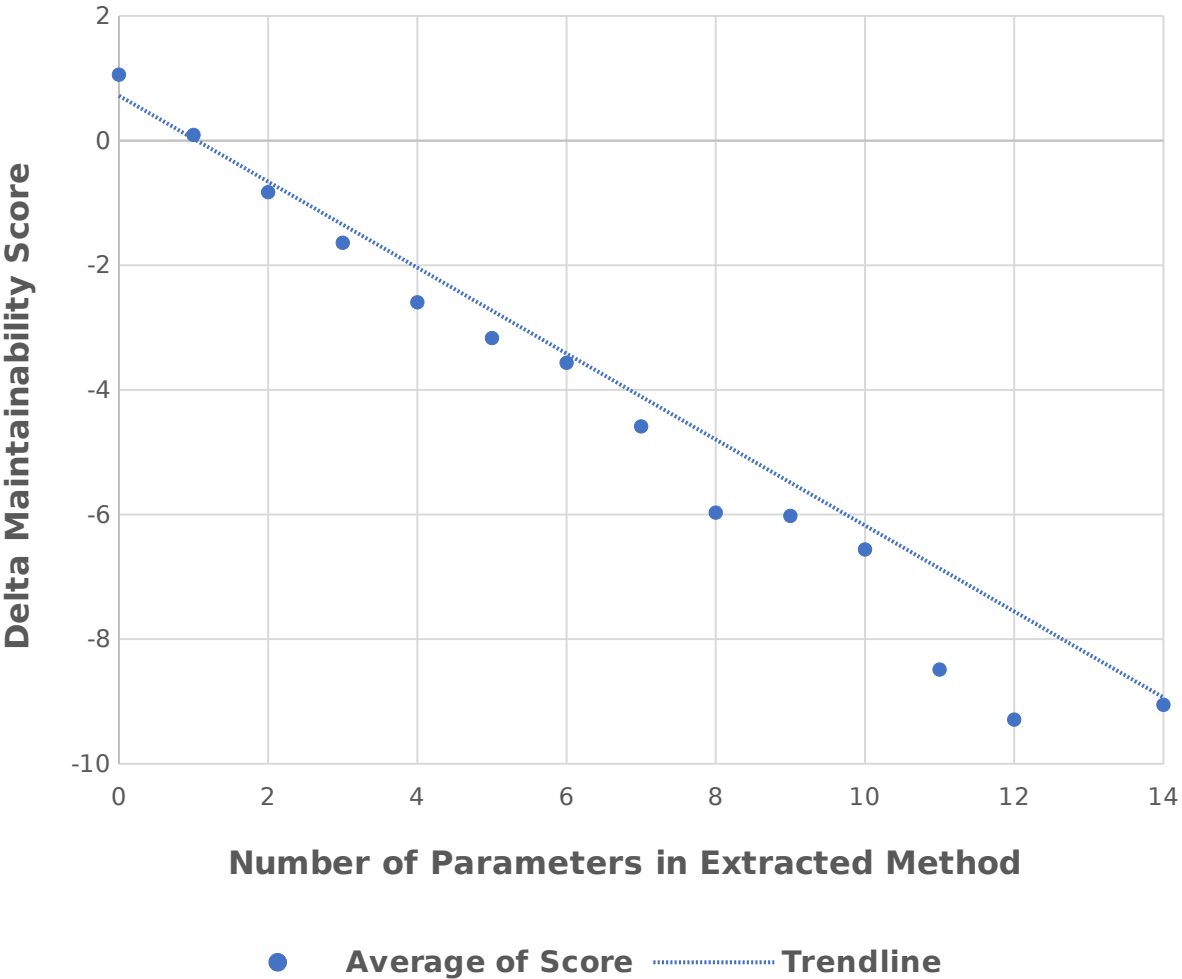
Return Category

<i>Return Category</i>	<i>Complexity</i>	<i>Parameters</i>	<i>Size</i>	<i>Duplication</i>	<i>#</i>	<i>Score</i>
Return	0.85	1.02	-3.84	-55.00	1,571	0.19
Declare	0.94	0.74	11.11	-49.19	5,177	0.15
Assign	0.79	1.07	0.43	-56.29	14	0.12
Void	0.76	1.49	-5.85	-56.35	5,921	-0.18
Grand Total	0.84	1.12	1.33	-53.26	12,683	0.00

Parameters

Any Token Volume

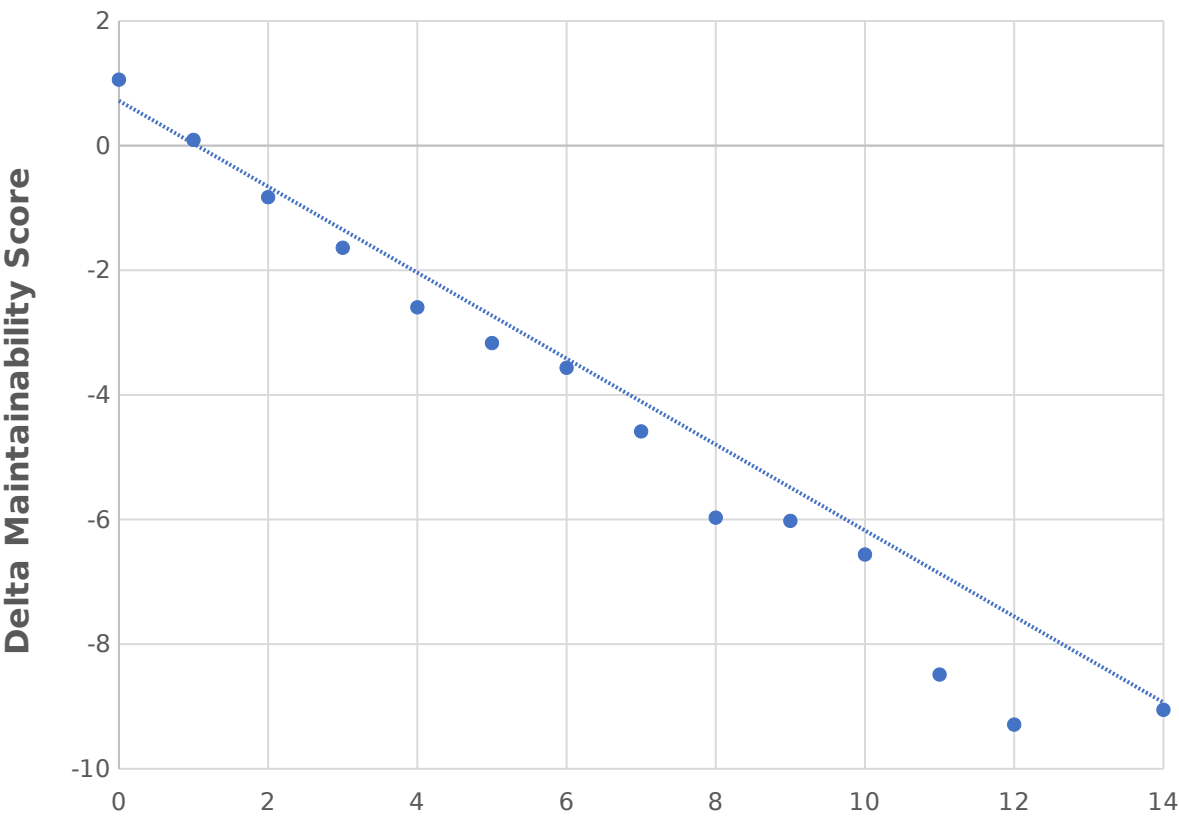
Number of Parameters in the Extracted Method vs Delta Maintainability Score



Parameters

Any Token Volume

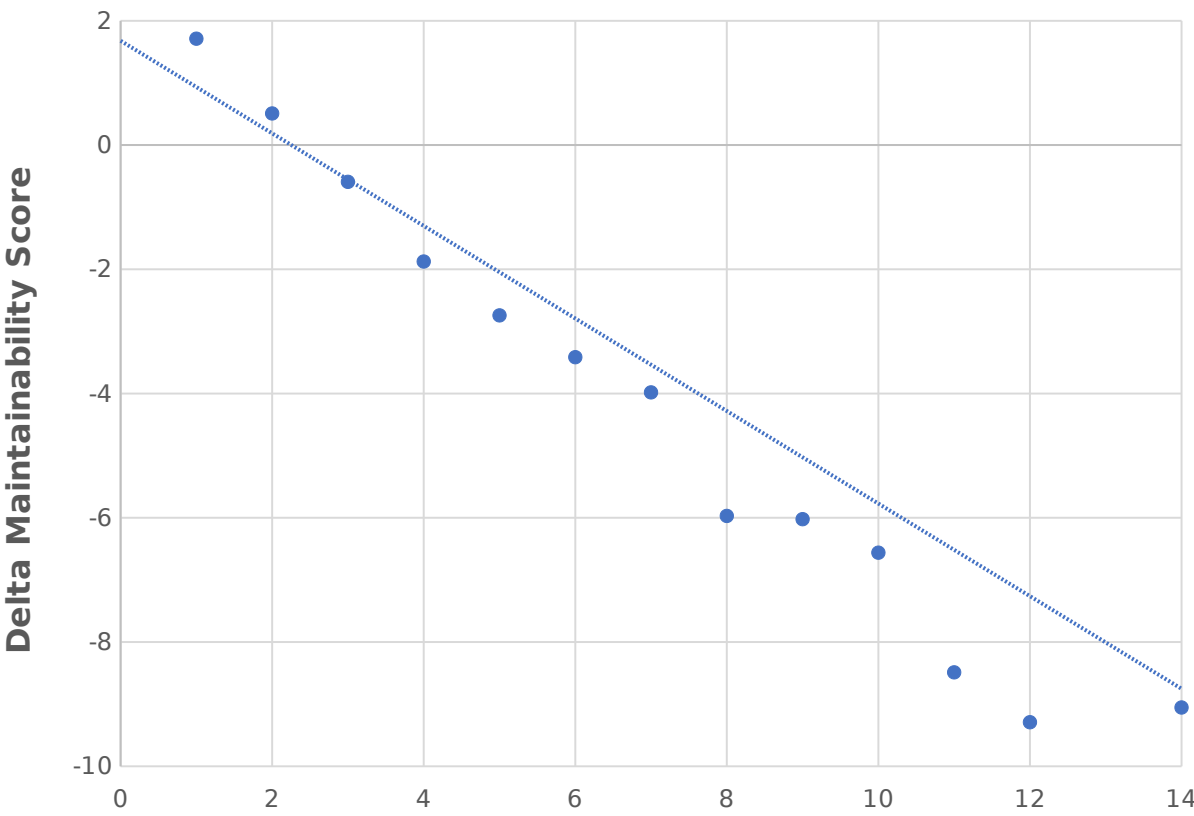
Number of Parameters in the Extracted Method vs Delta Maintainability Score



● Average of Score Trendline

63+ Token Volume

Number of Parameters in the Extracted Method vs Delta Maintainability Score



● Average of Score Trendline

Conclusion

RQ1.

*How can we define clone types such that they **can** be automatically refactored?*

RQ2.

*How can we prioritize refactoring opportunities based on the **context** of clones?*

RQ3.

*What are the discriminating factors to decide when a clone **should** be refactored*

Summary

- > **T1R.** Textually identical code fragments except for variations in whitespace and comments.
 - > **T2R.** Type 1R clones except for variations in a controlled set of expressions.
 - > **T3R.** Type 2R clones with optional gaps of non cloned statements.
-
- > **Relation.** 37% Same Class, 24% Same Hierarchy, 24% Unrelated and 7% Same Interface
 - > **Location.** 78% Method Level, 17% Class Level, 4% Constructor Level and 1% Other
 - > **Contents.** 74% Method Body (77% including constructor), 8% Several Methods, 6% Only Fields and 4% Full Method.
-
- > **Refactorability.** 28% can be extracted, 25% is not in a method body, 23% top-level AST-node is not a statement, 15% spans part of a block and 6% multiple return values.
-
- > **Token Volume.** Clone classes with a Token Volume higher than 63 results in refactorings that, on average, improve maintainability.
 - > **Number of Parameters.** When an extracted method has more than two parameters , it is more likely to decrease maintainability.
 - > **Relation and Return Category.** These two factors have a minor influence on maintainability.