

Improving Software Maintainability through Automated Refactoring of Code Clones

Author: Simon Baars

Company Supervisor: Xander Schrijen

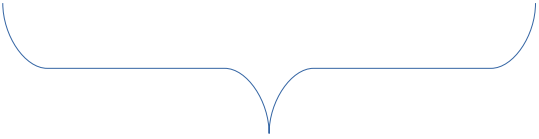
Host Company: Software Improvement Group

Academic Supervisor: Ana Opreescu

Second Reader: Clemens Grelck

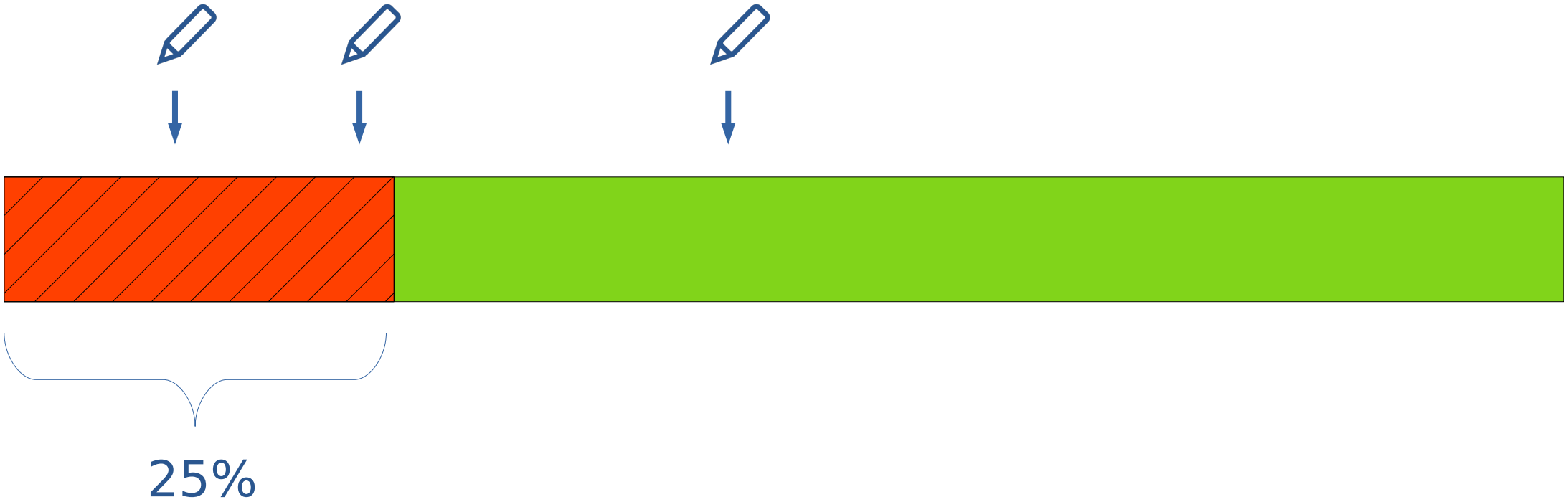
Location: Amsterdam

Date: 28 August 2019



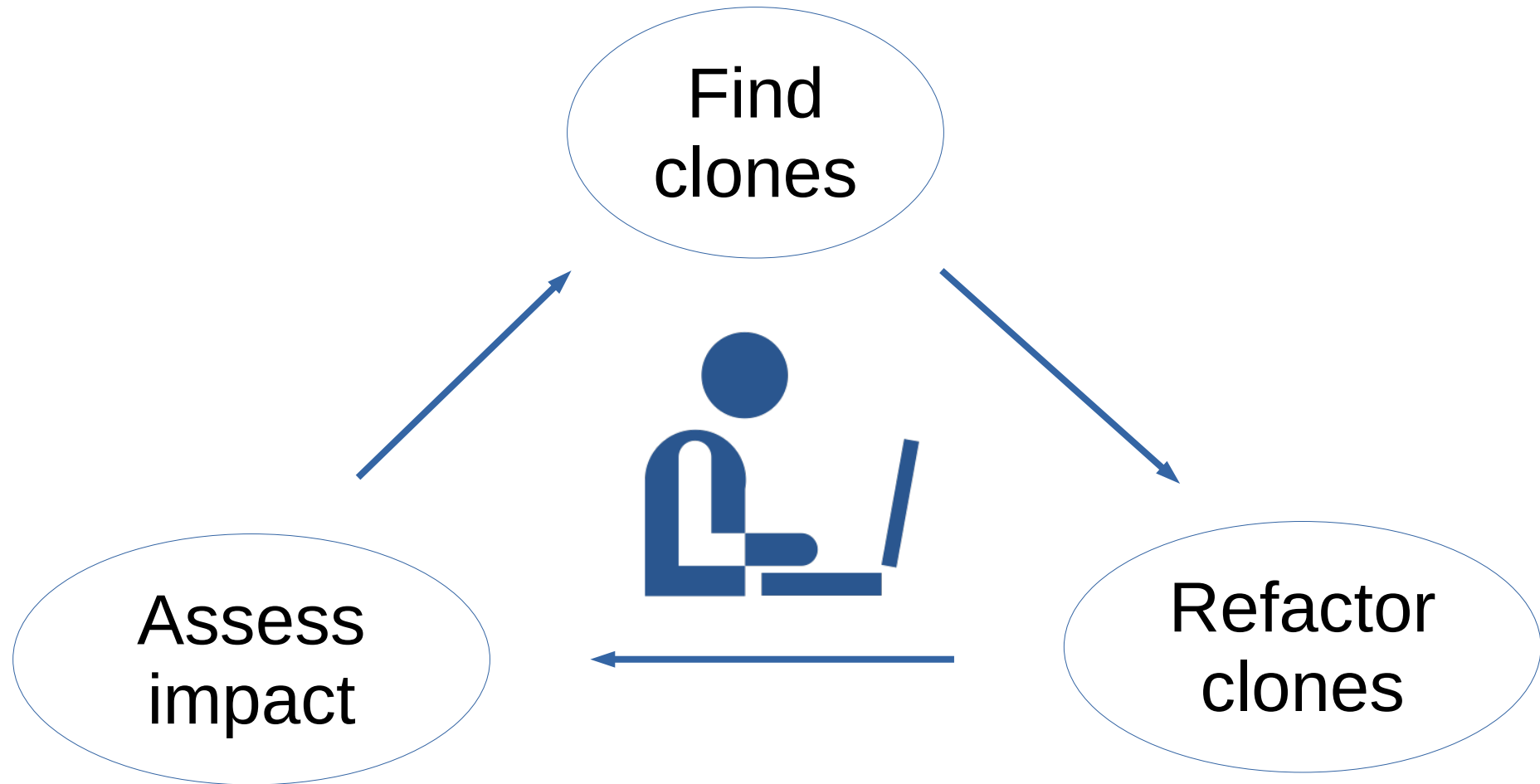
25%

One “simple” change...



*“If you see the **same code structure** in more than one place, you can be sure that your program will be **better** if you find a way to **unify** them”*

~ Martin Fowler & Kent Beck in
Refactoring



Find
clones



Refactor
clones

Assess
impact

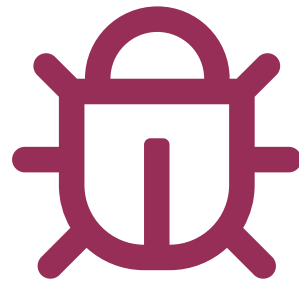


Find
clones



Refactor
clones

Assess
impact

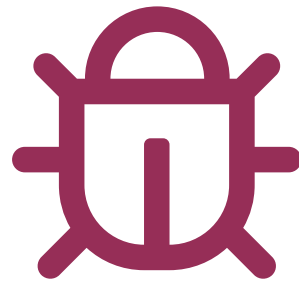


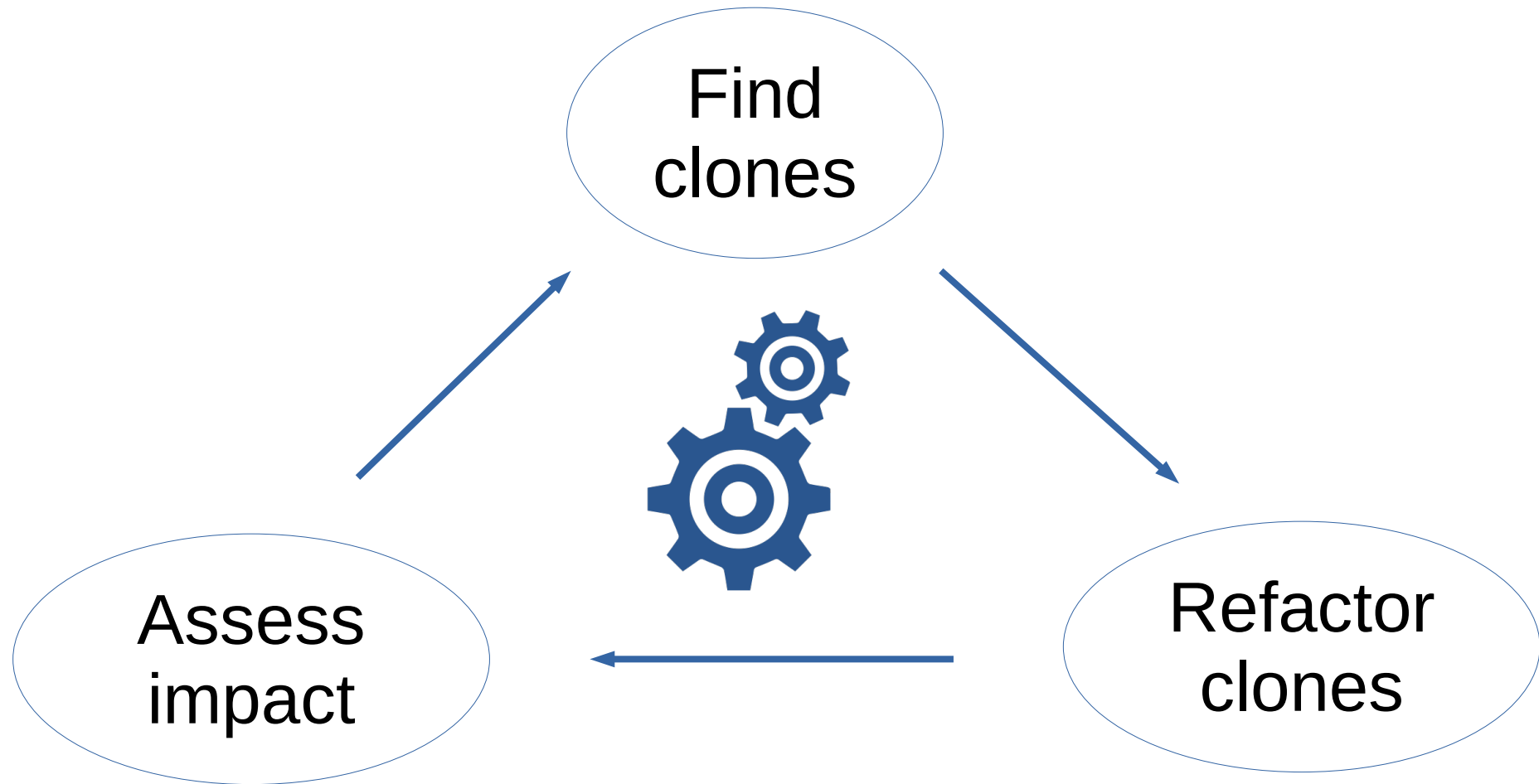
Find
clones



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.

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

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 (purple boxes) and a yellow dashed box highlights the body of the `addTen` method.

```
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 (purple boxes) and a yellow dashed box highlights the body of the `concat` method.

Type 1R.

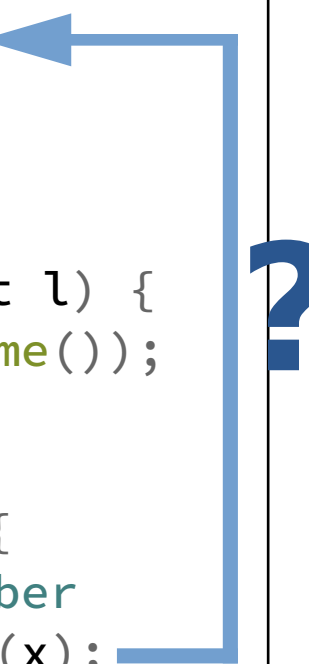
Retrieving contextual information can be challenging!

```
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!

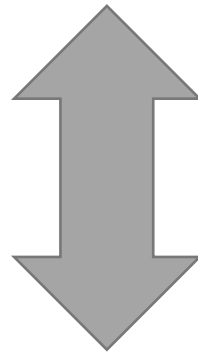
Type 2.

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);
}

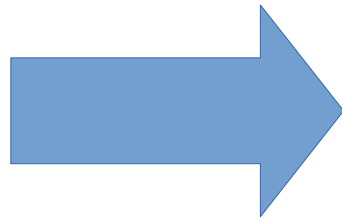
void doADC(){
    doThis(this::doD);
}

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);
}
```

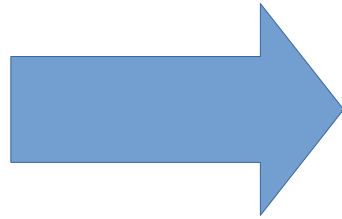
```
// Refactored
void doCwithA(){
    modifyA(false);
}

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 ISODateTime...
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.

Knowledge Graphs and Graphs in R

have an Same Method relation with each other.

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Knowledge Graphs and Graphs in R

Downloaded from <http://ajphaphysiol.org/> at www.ajphaphysiol.org/ on September 11, 2014

Handelsregister eingetragen & kann nach § 15 Abs. 1 Nr. 1 StB-Gesetz, für die Zwecke des § 15 Abs. 1 Nr. 1 StB-Gesetz, als „Handelsregister“ bezeichnet werden. Nach § 15 Abs. 1 Nr. 1 StB-Gesetz ist das Handelsregister ein öffentliches Register, das die Daten und Fakten über die Person, die das Unternehmen betreibt, enthält. Das Handelsregister ist ein öffentliches Register, das die Daten und Fakten über die Person, die das Unternehmen betreibt, enthält.

© 2000 Blackwell Science Ltd, *Journal of Internal Medicine* 247: 395–400

1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810

1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609 2610 2611 2612 2613 2614 2615 2616 2617 2618 2619 2620 2621 2622 2623 2624 2625 2626 2627 2628 2629 2630 2631 2632 2633 2634 2635 2636 2637 2638 2639 2640 2641 2642 2643 2644 2645 2646 2647 2648 2649 2650 2651 2652 2653 2654 2655 2656 2657 2658 2659 2660 2661 2662 2663 2664 2665 2666 2667 2668 2669 2670 2671 2672 2673 2674 2675 2676 2677 2678 2679 2680 2681 2682 2683 2684 2685 2686 2687 2688 2689 2690 2691 2692 2693 2694 2695 2696 2697 2698 2699 2700 2701 2702 2703 2704 2705 2706 2707 2708 2709 2710 2711 2712 2713 2714 2715 2716 2717 2718 2719 2720 2721 2722 2723 2724 2725 2726 2727 2728 2729 2730 2731 2732 2733 2734 2735 2736 2737 2738 2739 2740 2741 2742 2743 2744 2745 2746 2747 2748 2749 2750 2751 2752 2753 2754 2755 2756 2757 2758 2759 2760 2761 2762 2763 2764 2765 2766 2767 2768 2769 2770 2771 2772 2773 2774 2775 2776 2777 2778 2779 2780 2781 2782 2783 2784 2785 2786 2787 2788 2789 2790 2791 2792 2793 2794 2795 2796 2797 2798 2799 2800 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811

Copyright © 2006 John Wiley & Sons, Ltd.

[illegible]

Copyright © 2004 John Wiley & Sons, Inc.

Copyright © 2004 John Wiley & Sons, Ltd.

== Risk Profiles ==

Copyright © 2006 John Wiley & Sons, Inc. All rights reserved.

the authors' responsibility. The authors, however, do not accept any responsibility for any errors or omissions in this paper.

The authors have nothing to disclose.

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

Copyright © 2000 by John Wiley & Sons, Inc. All rights reserved. This publication is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Organizations in the USA who are also registered with the Copyright Clearance Center may therefore copy material (beyond the limits permitted by sections 107 and 108 of US copyright law) subject to payment to CCC of the per copy fee of \$05.00. This consent does not extend to multiple copying for promotional or commercial purposes. ISI Tear Sheet Service, 3501 Market Street, Philadelphia, PA 19104, USA, is authorized to supply single copies of separate articles for private use only. Organizations authorized by the Copyright Licensing Agency may also copy material subject to the usual conditions. For all other use, permission should be sought from John Wiley & Sons, Inc. Permissions may be obtained from the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA, (978) 750-8400, fax (978) 750-4744, and on the Internet <http://www.copyright.com>.

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.

Changes 8

History

Select branch to compare...

CloneRefactor authored and Simon Baars committed

Created unified method in AbstractReadableInstant...

CloneRefactor authored and Simon Baars committed

Created unified method in AbstractReadableInstant...

CloneRefactor authored and Simon Baars committed

Formatted MutableDateTime

CloneRefactor authored and Simon Baars committed

Formatted DateMidnight

CloneRefactor authored and Simon Baars committed

Formatted DateTime

CloneRefactor authored and Simon Baars committed

Formatted AbstractReadableInstant...

CloneRefactor authored and Simon Baars committed

Created unified method in Property

CloneRefactor authored and Simon Baars committed

Created unified method in BasePartial

CloneRefactor authored and Simon Baars committed

Created unified method in BaseSingleFieldPeriod

CloneRefactor authored and Simon Baars committed

Formatted Seconds

CloneRefactor authored and Simon Baars committed

Created unified method in BaseSingleFieldPeriod

CloneRefactor authored and Simon Baars committed 72a71af 8 changed files

CloneRefactor refactored a clone class with 7 clone instances. For the common code we created a new method and named this method "cloneRefactor?". These clone instances have an sibling relation with each other. The newly created method has been placed in BaseSingleFieldPeriod. Each duplicated fragment has been replaced with a call to this method.

src/main/java/org/joda/time/Days.java			@@ -344,4 +344,9 @@ public abstract class BaseSingleFieldPeriod implements ReadablePeriod, Comparable
src/main/java/org/joda/time/Hours.java	344	344	}
src/main/java/org/joda/time/Minutes.java	345	345	return 0;
src/main/java/org/joda/time/Months.java	346	346	}
src/main/java/org/joda/time/Seconds.java		347	+ protected Chronology cloneRefactor?(ReadablePartial start) {
src/main/java/org/joda/time/Weeks.java		349	+ Chronology chrono = DateTimeUtils.getChronology(start.getChronology());
src/main/java/org/joda/time/Years.java		350	+ return chrono;
src/main/java/org/joda/time/Years.java		351	+ }
src/main/java/org/joda/time/Years.java	347	352	}

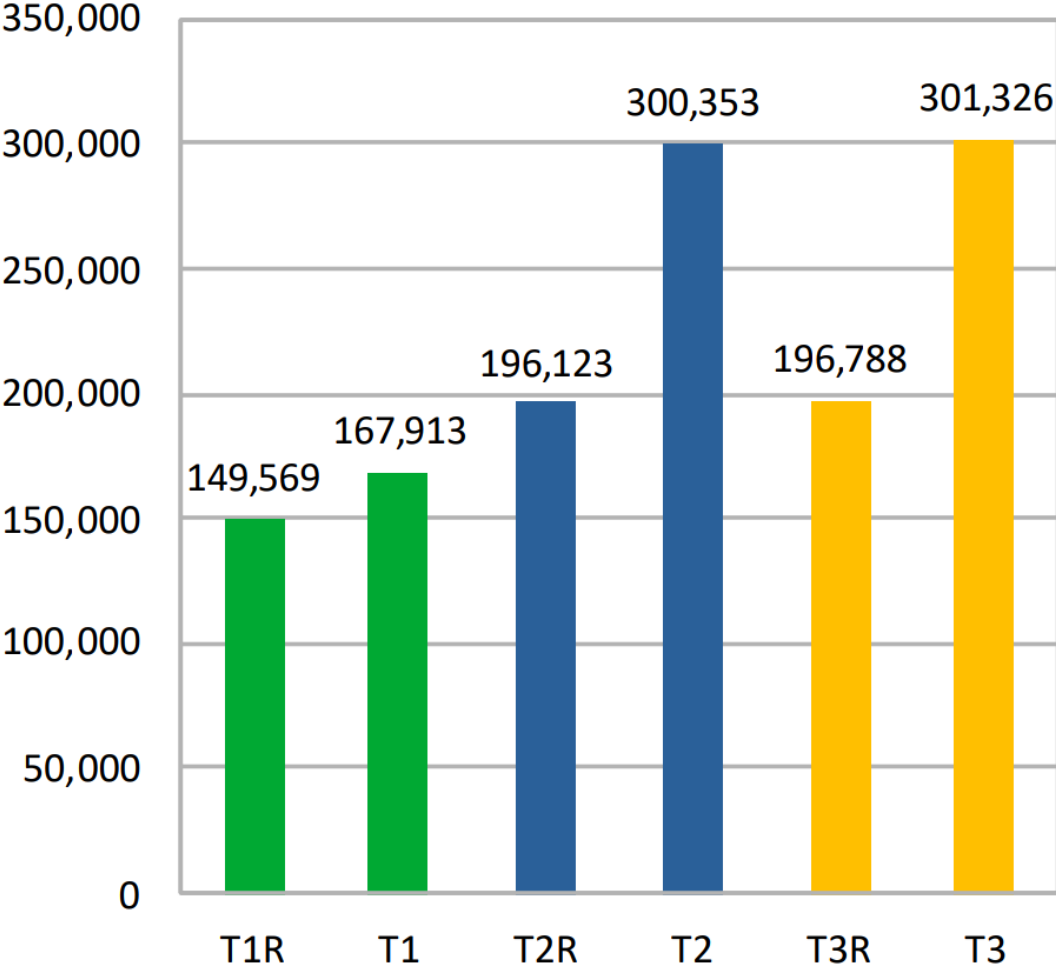
Experiments

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



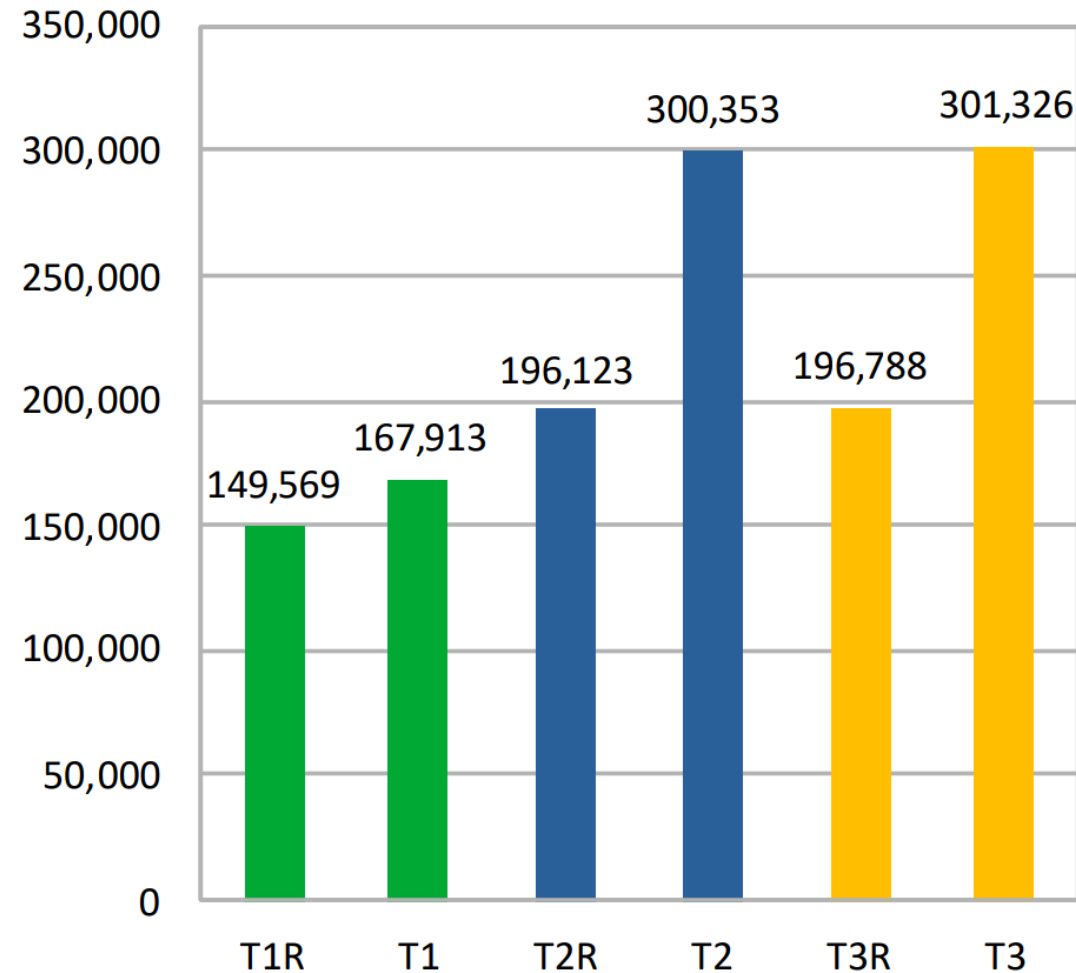
Clone Types

Amount of clones found

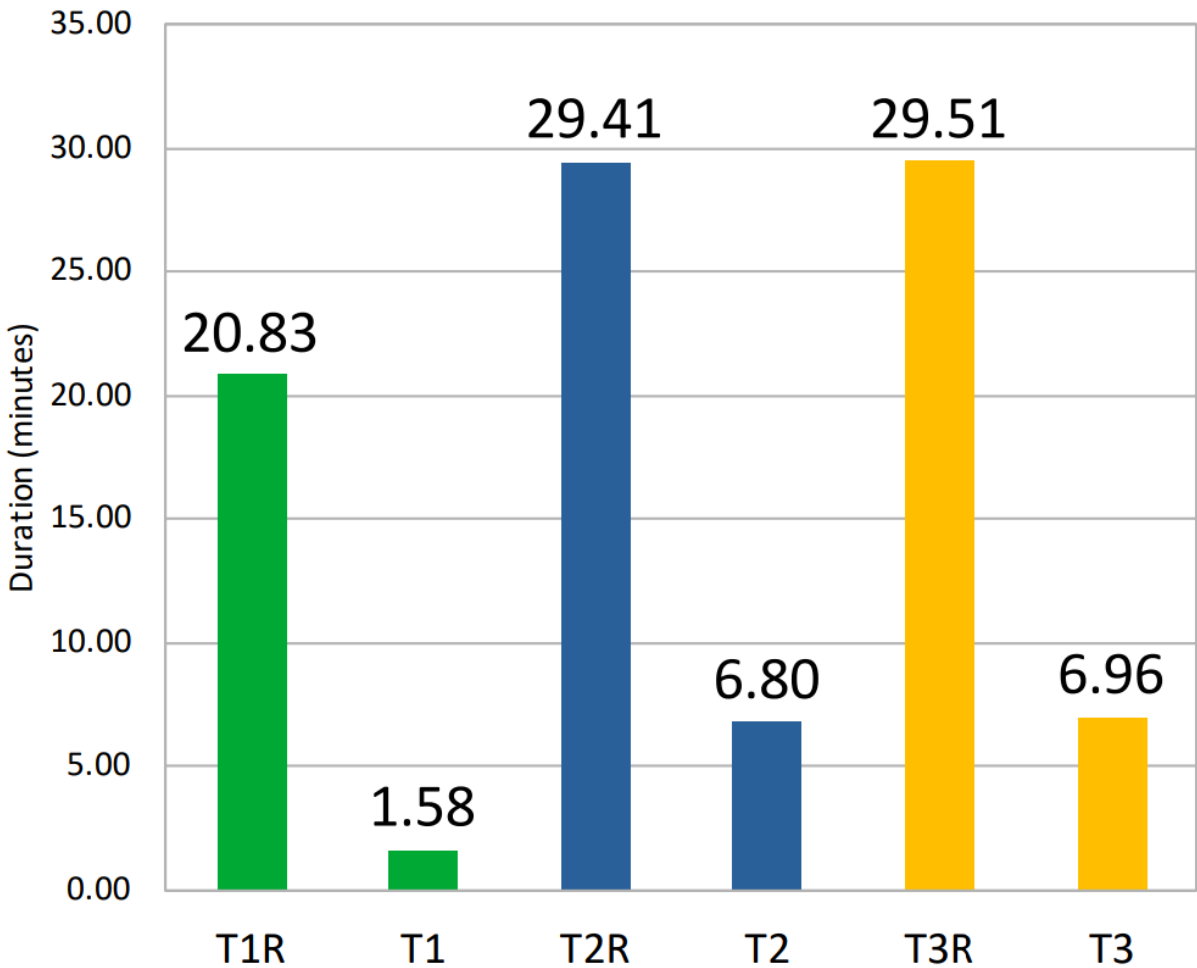


Clone Types

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

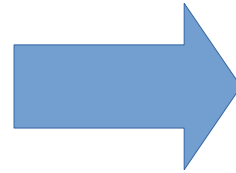
Metrics

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

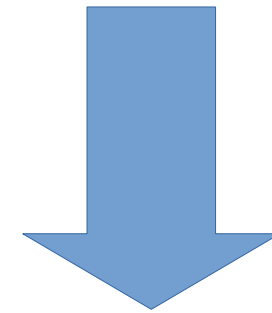
Calculate Maintainability Score

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

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

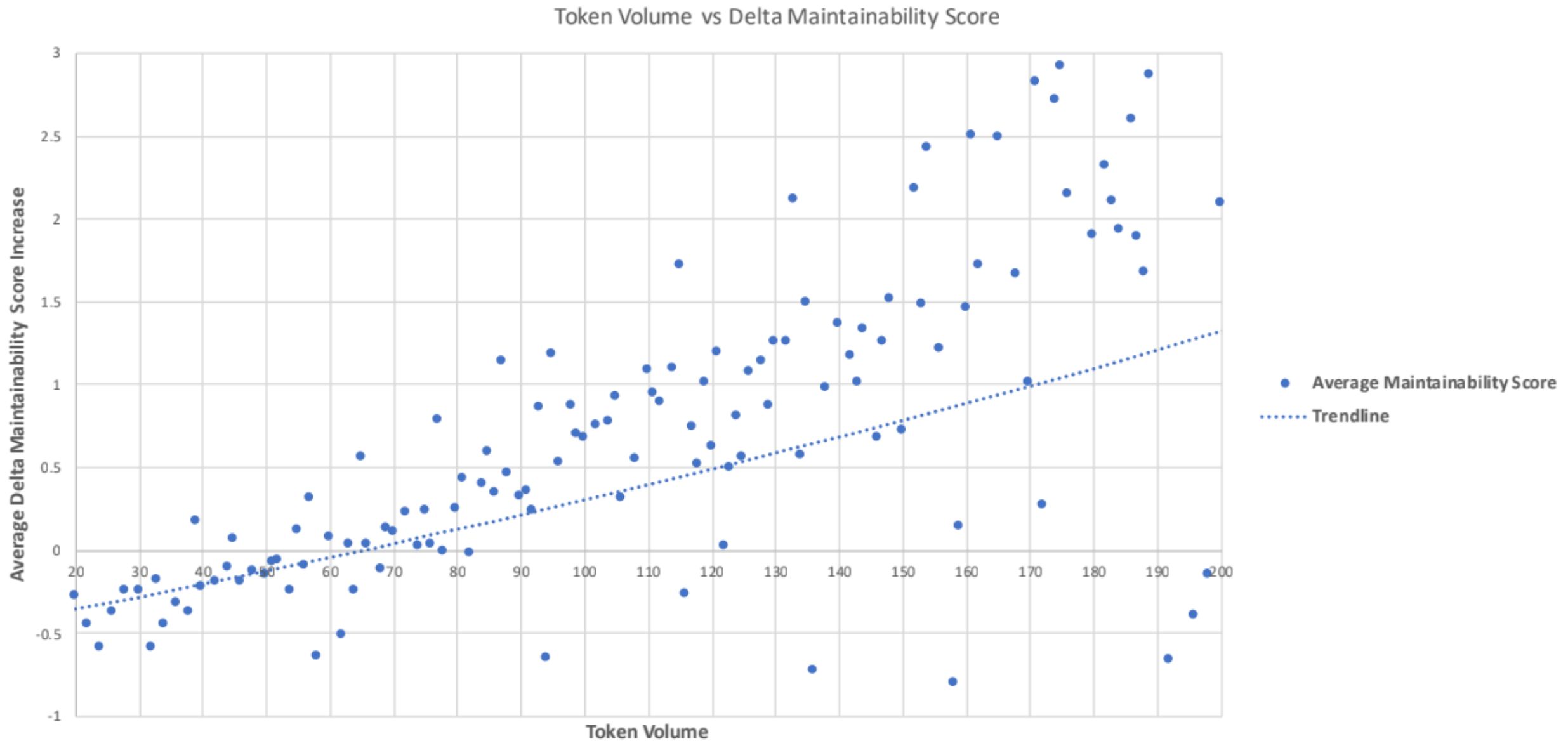


$$N_{metric} = \frac{\Delta X - \mu}{\sigma}$$



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

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

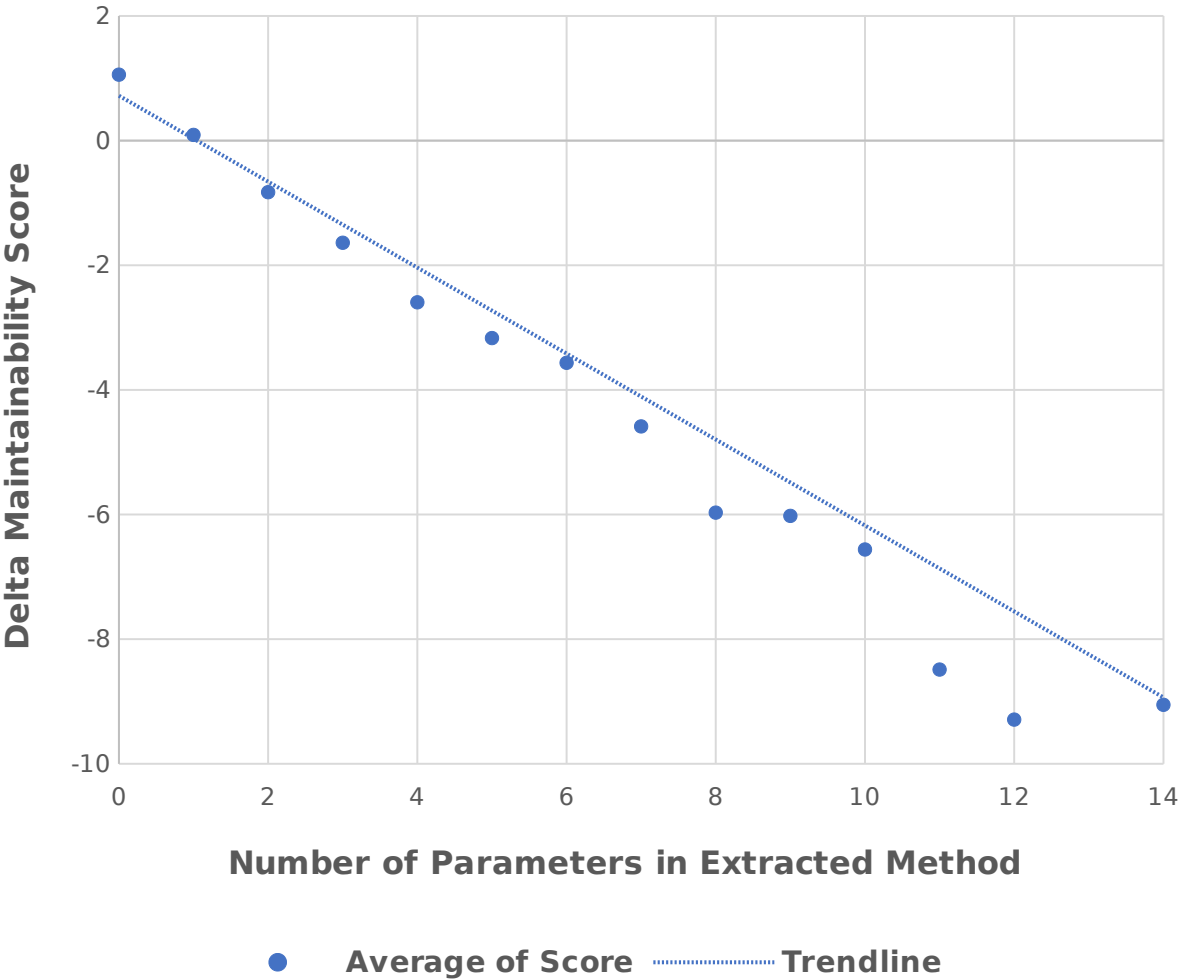
Return Category

<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

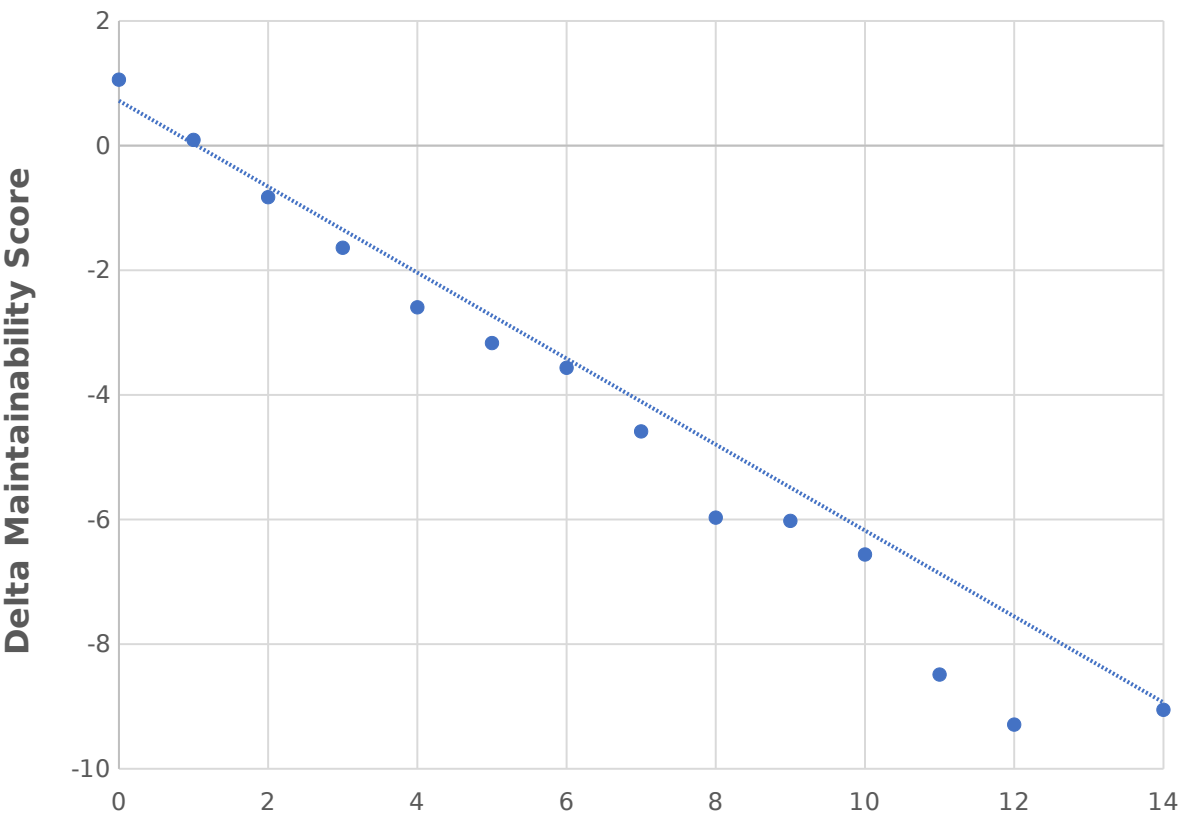
Number of Parameters in the Extracted Method vs Delta Maintainability Score



Parameters

Any Token Volume

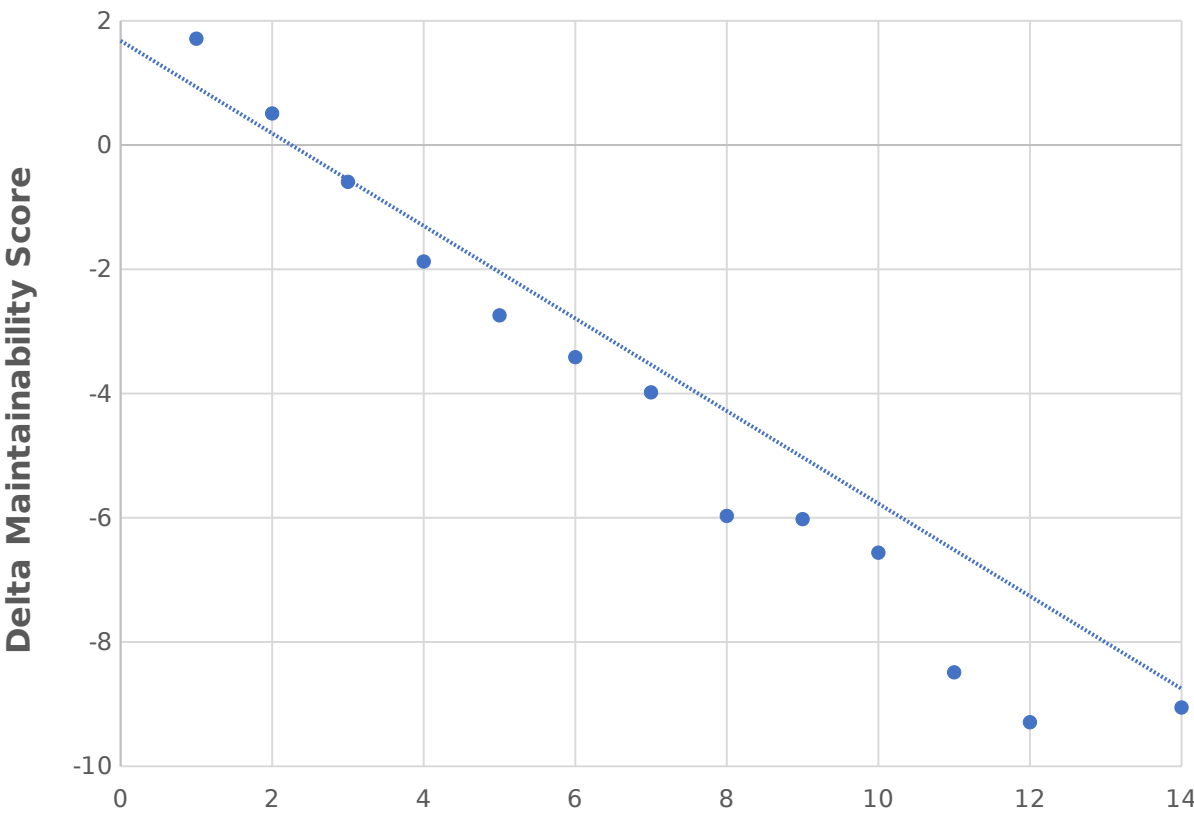
Number of Parameters in the Extracted Method vs Delta Maintainability Score



● 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.