

Clone Coverage

YEGOR BUGAYENKO

Lecture #11 out of 24

80 minutes

The slidedeck was presented by the author in this [YouTube Video](#)

All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as web sites. Copyright belongs to their respected authors.



ROBERT C. MARTIN

“Duplication is the primary enemy of a well-designed system.”

— Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, 2008. doi:[10.5555/1388398](https://doi.org/10.5555/1388398)



QURAT UL AIN

“If a bug is identified in one segment of code, all the similar segments need to be checked for the same bug. Consequently, code duplication may lead to bug propagation that significantly affects the maintenance cost.”

— Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam, and Bilal Maqbool. A Systematic Review on Code Clone Detection. *IEEE Access*, 2019. doi:[10.1109/access.2019.2918202](https://doi.org/10.1109/access.2019.2918202)



RAINER KOSCHKE

“The problem with code cloning is that errors in the original must be fixed in every copy. Other kinds of maintenance changes, for instance, extensions or adaptations, must be applied multiple times, too. Yet, it is usually not documented where code was copied.”

— Stefan Bellon, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo. Comparison and Evaluation of Clone Detection Tools. *IEEE Transactions on Software Engineering*, 2007. doi:[10.1109/tse.2007.70725](https://doi.org/10.1109/tse.2007.70725)

Motivating Example (part I)

Before (**wrong**):

```
1 printf("Hi, %s!", getName(42));  
2 printf("Hi, %s!", getName(7));  
3 printf("Hi, %s!", getName(55));
```

After (**better**):

```
1 sayHello(42);  
2 sayHello(7);  
3 sayHello(55);  
4  
5 void sayHello(int id) {  
6     var n = getName(id);  
7     printf("Hi, %s!", n);  
8 }
```

Motivating Example (part II)

Before (**still not ideal**):

```
1 sayHello(42);  
2 sayHello(7);  
3 sayHello(55);  
4  
5 void sayHello(int id) {  
6     var n = getName(id);  
7     printf("Hi, %s!", n);  
8 }
```

After (**perfect**):

```
1 var users = [42, 7, 55];  
2 for (id : users) {  
3     sayHello(id);  
4 }  
5  
6 void sayHello(int id) {  
7     var n = getName(id);  
8     printf("Hi, %s!", n);  
9 }
```



BRENDA S. BAKER

“Two lines of code are considered to be identical if they contain the same sequence of characters after removing comments and white space; the semantics of the program statements are not analyzed.”

— Brenda S. Baker. A Program for Identifying Duplicated Code. In *Proceedings of the 24th Symposium on the Interface*, pages 1–9, 1993

Up to 38% of lines are involved in duplicates



The plots are dense near the main diagonal, implying that most copies tend to occur fairly locally, e.g. within the same file or module.

However, certain line segments occur away from the main diagonal; it would be interesting to investigate why the corresponding sections of code are duplicated.



ANDY HUNT

“Don’t Repeat Yourself (DRY): Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.”

— Andrew Hunt and Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999. doi:[10.5555/320326](https://doi.org/10.5555/320326)



KENT BECK

“The Rule of Three: The first time you do something, you just do it. The second time you do something similar, you wince at the duplication, but you do the duplicate thing anyway. The third time you do something similar, you refactor.”

— Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts.
Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999. doi:[10.5555/311424](https://doi.org/10.5555/311424)



YOSHIKI HIGO

“Code-clone analysis is a good vehicle to quantitatively understand the differences and improvements between two versions of the same software system”

— Simone Livieri, Yoshiki Higo, Makoto Matsushita, and Katsuro Inoue. Analysis of the Linux Kernel Evolution Using Code Clone Coverage. In *Proceedings of the 4th International Workshop on Mining Software Repositories*, 2007. doi:[10.1109/msr.2007.1](https://doi.org/10.1109/msr.2007.1)



WASI HAIDER BUTT

“We identified and analyzed 26 Code Clone Detection (CCD) tools, i.e., 13 existing and 13 proposed/developed. Moreover, 62 open-source subject systems whose source code is utilized for the CCD are presented.”

— Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam, and Bilal Maqbool. A Systematic Review on Code Clone Detection. *IEEE Access*, 2019. doi:[10.1109/access.2019.2918202](https://doi.org/10.1109/access.2019.2918202)

Type-1: Exact Clone

Original:

```
1 | printf("Hi, %s\n", name(42));
```

Identical code segments except for changes in comments, layouts and whitespaces.

Clone:

```
1 | // Here we print a message
2 | // to the console for a user
3 | printf(
4 |     "Hi, %s\n",
5 |     name(42)
6 | );
```

Type-2: Parameterized Clone

Original:

```
1 | var n = name(42);  
2 | printf("Hi, %s\n", n);
```

Clone:

```
1 | String name = name(42);  
2 | printf("Hi, %s\n", name);
```

Code segments which are syntactically or structurally similar other than changes in comments, identifiers, types, literals, and layouts.

Type-3: Gapped Clone

Original:

```
1 | printf("Hi, %s\n", name(42));
```

Clone:

```
1 | var msg = "Hi, %s\n";  
2 | var n = name(42);  
3 | printf(msg, n);
```

Copied pieces with further modification such as addition or removal of statements and changes in whitespaces, identifiers, layouts, comments, and types but outcomes are similar.

Type-4: Semantic Clone

Original:

```
1 | printf("Hi, %s\n", name(42));
```

Clone:

```
1 | var s = sprintf(  
2 |     "Hi, %s\n",  
3 |     name(42));  
4 | print(s);
```

More than one code segments that are functionally similar but implemented by different syntactic variants.

Clones in Linux Kernel

	alpha	arm	i386	ia64	mips	mips64	ppc	s390	sh	sparc	sparc64
alpha	100%	0%	5.0%	0%	5.0%	10%	10%	5%	5%	5%	0%
arm	0%	100%	2.4%	0%	8.6%	4.9%	4.9%	0%	1.2%	9.8%	0%
i386	3.5%	7.1%	100%	0%	7.1%	7.1%	10.7%	14.2%	32.1%	0 %	0%
ia64	0%	0%	0%	100%	0%	0%	0%	0%	4.7%	9.5%	0%
mips	0.6%	8.7%	1.3%	0%	100%	19.4%	4.2%	0%	0.6%	4.1 %	0%
mips64	2.5%	3.8%	2.5%	0%	38.4%	100%	3.8%	0%	1.2%	2.5 %	0%
ppc	3.2%	4.9%	4.9%	0%	8.1%	4.9%	100%	1.6%	3.2 %	0 %	0%
s390	5.2%	0%	21.1%	0%	0%	0%	5.2%	100%	5.2%	0%	0%
sh	2.3%	2.3%	20.9%	2.3%	2.3%	2.3%	4.6%	2.3%	100%	0%	0%
sparc	0.3%	2.2%	0%	0.6%	2.8%	0.6%	0%	0%	0%	100 %	1.9%
sparc64	0%	0%	0%	0%	0%	0%	0%	0%	0%	16.6%	100 %

Table 3. Cloning Percentage among ^{mm} Subsystem Architecture-Dependent Code

Identifying Clones in the Linux Kernel, Gerardo Casazza, Giuliano Antoniol, Umberto Villano, Ettore Merlo, Massimiliano Di Penta, Proceedings of the International Workshop on Source Code Analysis and Manipulation, 2001

Methods of clone detection:

1. Using text
2. Using tokens
3. Using metrics
4. Using “tree matching”
5. Using Program Dependency Graphs (PDG)
6. Using Machine Learning (ML)



JENS KRINKE

“For the three Java systems studied, the following results were found: 1) cloned code is usually older than non-cloned code, 2) cloned code in a file is usually older than the non-cloned code in the same file. Both results suggest that cloned code is more stable than non-cloned code.”

— Jens Krinke. Is Cloned Code Older Than Non-Cloned Code? In *Proceedings of the 5th International Workshop on Software Clones*, 2011.

[doi:10.1145/1985404.1985410](https://doi.org/10.1145/1985404.1985410)

These tools can help detecting duplicate code:

1. IntelliJ IDEA by JetBrains
2. Copy/Paste Detector (CPD) by PMD for Java
3. SonarQube
4. CloneDR by Semantic Designs
5. Simian by Quandary Peak Research

Simian 4.0.0

```
-bash
/code/cactos$ java -jar ~/Downloads/simian-4.0.0/simian-4.0.0.jar -threshold=17 **/*.java
Simian Similarity Analyzer 4.0.0 - https://simian.quandarypeak.com
Copyright (c) 2023 Quandary Peak Research. All rights reserved.
Subject to the Quandary Peak Academic Software License.
{failOnDuplication=true, ignoreCharacterCase=true, ignoreCurlyBraces=true, ignoreIdentifierCase=true, ignoreModifiers=true, ignoreStringCase=true, threshold=17}
Found 17 duplicate lines with fingerprint 8d01496ba38a19cb808ae9235ac8db2a in the following files:
  Between lines 87 and 107 in /Volumes/sec/code/cactos/src/test/java/org/cactos/io/InputOfTest.java
  Between lines 131 and 151 in /Volumes/sec/code/cactos/src/test/java/org/cactos/bytes/BytesOfTest.java
Found 17 duplicate lines with fingerprint b598bab8d6e4187f2109de9732ec2285 in the following files:
  Between lines 171 and 188 in /Volumes/sec/code/cactos/src/main/java/org/cactos/io/LoggingOutputStream.java
  Between lines 143 and 160 in /Volumes/sec/code/cactos/src/main/java/org/cactos/io/LoggingOutputStream.java
Found 18 duplicate lines with fingerprint 67a118fb204dfa3159a42f61ca6cb8f7 in the following files:
  Between lines 139 and 164 in /Volumes/sec/code/cactos/src/test/java/org/cactos/experimental/ThreadsTest.java
  Between lines 263 and 288 in /Volumes/sec/code/cactos/src/test/java/org/cactos/experimental/ThreadsTest.java
Found 19 duplicate lines with fingerprint 1800fef4f92055a4cacabe1d6c9cacb7 in the following files:
  Between lines 113 and 132 in /Volumes/sec/code/cactos/src/test/java/org/cactos/io/TempFolderTest.java
  Between lines 73 and 92 in /Volumes/sec/code/cactos/src/test/java/org/cactos/io/TempFolderTest.java
Found 21 duplicate lines with fingerprint 89b2815ad5cacc028593951c22a0440b in the following files:
  Between lines 48 and 82 in /Volumes/sec/code/cactos/src/main/java/org/cactos/list/ListIteratorEnvelope.java
  Between lines 48 and 82 in /Volumes/sec/code/cactos/src/main/java/org/cactos/list/ImmutableListIterator.java
Found 184 duplicate lines in 10 blocks in 7 files
Processed a total of 24317 significant (62285 raw) lines in 638 files
Processing time: 0.137sec
/code/cactos$
```

References

Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam, and Bilal Maqbool. A Systematic Review on Code Clone Detection. *IEEE Access*, 2019. doi:[10.1109/access.2019.2918202](https://doi.org/10.1109/access.2019.2918202).

Brenda S. Baker. A Program for Identifying Duplicated Code. In *Proceedings of the 24th Symposium on the Interface*, pages 1–9, 1993.

Stefan Bellon, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo. Comparison and Evaluation of Clone Detection Tools. *IEEE Transactions on Software Engineering*, 2007. doi:[10.1109/tse.2007.70725](https://doi.org/10.1109/tse.2007.70725).

Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving*

the Design of Existing Code. Addison-Wesley Professional, 1999. doi:[10.5555/311424](https://doi.org/10.5555/311424).

Andrew Hunt and Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999. doi:[10.5555/320326](https://doi.org/10.5555/320326).

Jens Krinke. Is Cloned Code Older Than Non-Cloned Code? In *Proceedings of the 5th International Workshop on Software Clones*, 2011. doi:[10.1145/1985404.1985410](https://doi.org/10.1145/1985404.1985410).

Simone Livieri, Yoshiki Higo, Makoto Matsushita, and Katsuro Inoue. Analysis of the Linux Kernel Evolution Using Code Clone Coverage. In *Proceedings of the 4th International Workshop on Mining Software Repositories*, 2007. doi:[10.1109/msr.2007.1](https://doi.org/10.1109/msr.2007.1).

Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Pearson Education, 2008. doi:[10.5555/1388398](https://doi.org/10.5555/1388398).