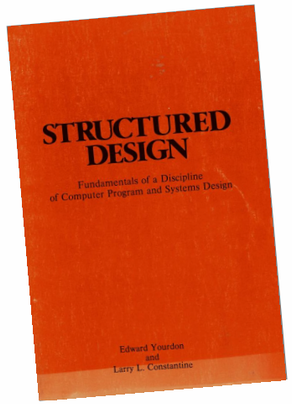
Clone Coverage

Yegor Bugayenko

Lecture #11 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video



Edward Yourdon

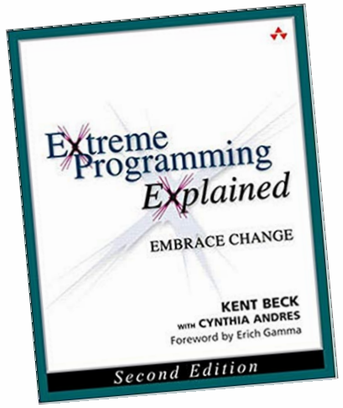
"Whenever possible, we wish to maximize fan-in during the design process. Fan-in is the *raison d'être* of modularity: Each instance of multiple fan-in means that some duplicate code has been avoided."

— Edward Yourdon and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice Hall, 1979. doi:10.5555/578522

Bertrand Meyer

"The challenge of reusability is to avoid unneeded duplication of software by taking advantage of the commonality between variants. If identical or near-identical fragments appear in different modules, it will be difficult to guarantee their integrity and to ensure that changes or corrections get propagated to all the needed places."

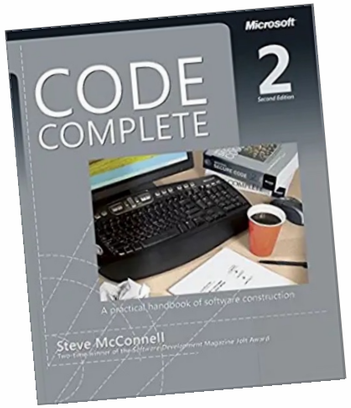
— Bertrand Meyer. *Object-Oriented Software Construction.* Prentice Hall, 1988. doi:10.5555/534929



Kent Beck

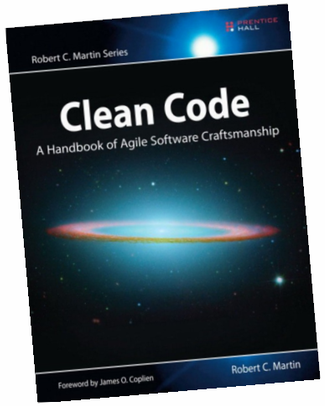
"You must find a way to eliminate all the duplicated logic in the system. This is the hardest part of design for me, because you first have to find the duplication, and then you have to find a way to eliminate it. Eliminating duplication naturally leads you to create lots of little objects and lots of little methods, because otherwise there will inevitably be duplication."

— Kent Beck. *Extreme Programming Explained: Embrace Change.* Addison-Wesley, 2000. doi:10.5555/318762



“With code in one place, you save the space that would have been used by duplicated code. Modifications will be easier because you’ll need to modify the code in only one location. The code will be more reliable because you’ll have to check only one place to ensure that the code is right.”

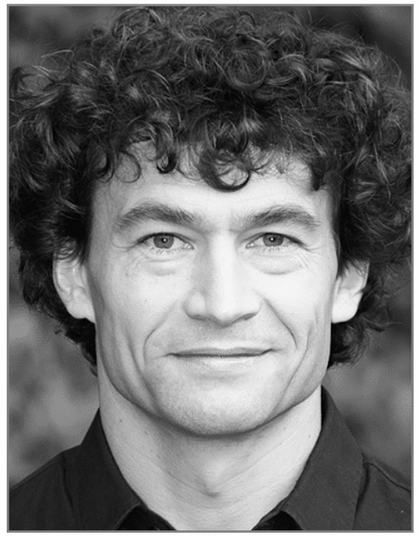
— Steve McConnell. *Code Complete.* Pearson Education, 2004. doi:10.5555/1096143



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



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*, 33(9), 2007. doi:10.1109/tse.2007.70725

Motivating Example (part I)

Before (wrong):

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

After (better):

```
sayHello(42);
sayHello(7);
sayHello(55);

void sayHello(int id) {
  var n = getName(id);
  printf("Hi, %s!", n);
}
```

Motivating Example (part II)

Before (still not ideal):

```
sayHello(42);
sayHello(7);
sayHello(55);

void sayHello(int id) {
  var n = getName(id);
  printf("Hi, %s!", n);
}
```

After (perfect):

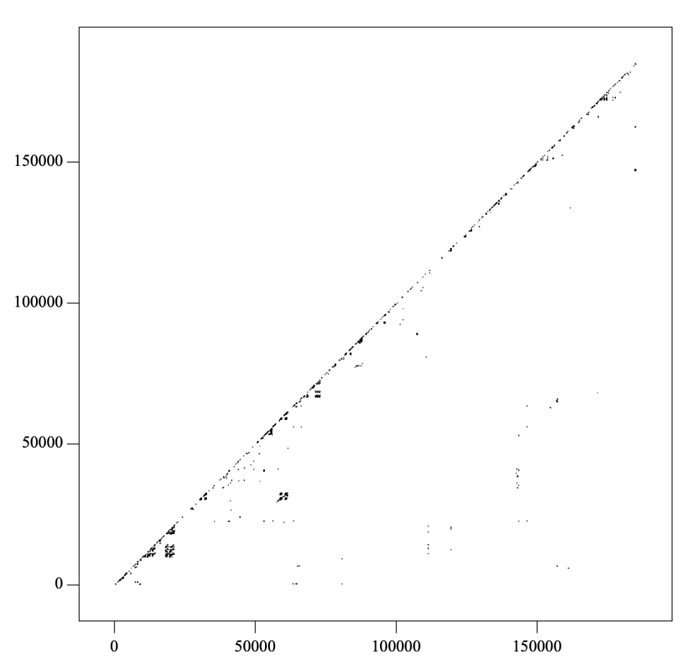
```
var users = [42, 7, 55];
for (id : users) {
  sayHello(id);
}

void sayHello(int id) {
  var n = getName(id);
  printf("Hi, %s!", n);
}
```

Brenda S. Baker

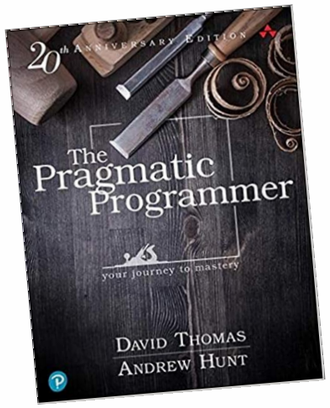
> “Two lines of code are considered to be identical if they contain the same sequence of characters after removing comments and white space; the semantics of the program statements are not analyzed.”
>
> — Brenda S. Baker. A Program for Identifying Duplicated Code, 1993

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."

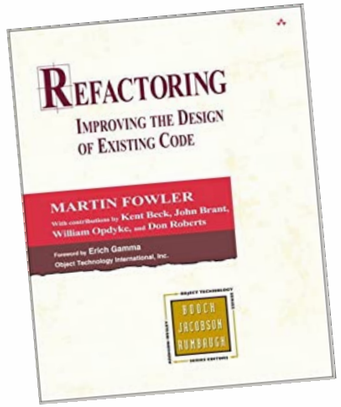
Source: Brenda S. Baker. A Program for Identifying Duplicated Code, 1993



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



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, 1999. doi: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

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*, 7(1), 2019. doi:10.1109/access.2019.2918202

Type-1: Exact Clone

Original:

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

Clone:

```
// Here we print a message
// to the console for a user
printf(
  "Hi, %s\n",
  name(42)
);
```

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

Type-2: Parameterized Clone

Original:

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

Clone:

```
String name = name(42);
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:

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

Clone:

```
var msg = "Hi, %s\n";
var n = name(42);
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:

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

Clone:

```
var s = sprintf(
  "Hi, %s\n",
  name(42));
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

Source: Gerardo Casazza, Giuliano Antoniol, Umberto Villano, Ettore Merlo, and Massimiliano Di Penta. Identifying Clones in the Linux Kernel. In *Proceedings of the 1st International Workshop on Source Code Analysis and Manipulation*, pages 90–97. IEEE, 2001. doi:10.1109/SCAM.2001.972670

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)
7. Using Large Language Models (LLM)

Jens Krinke

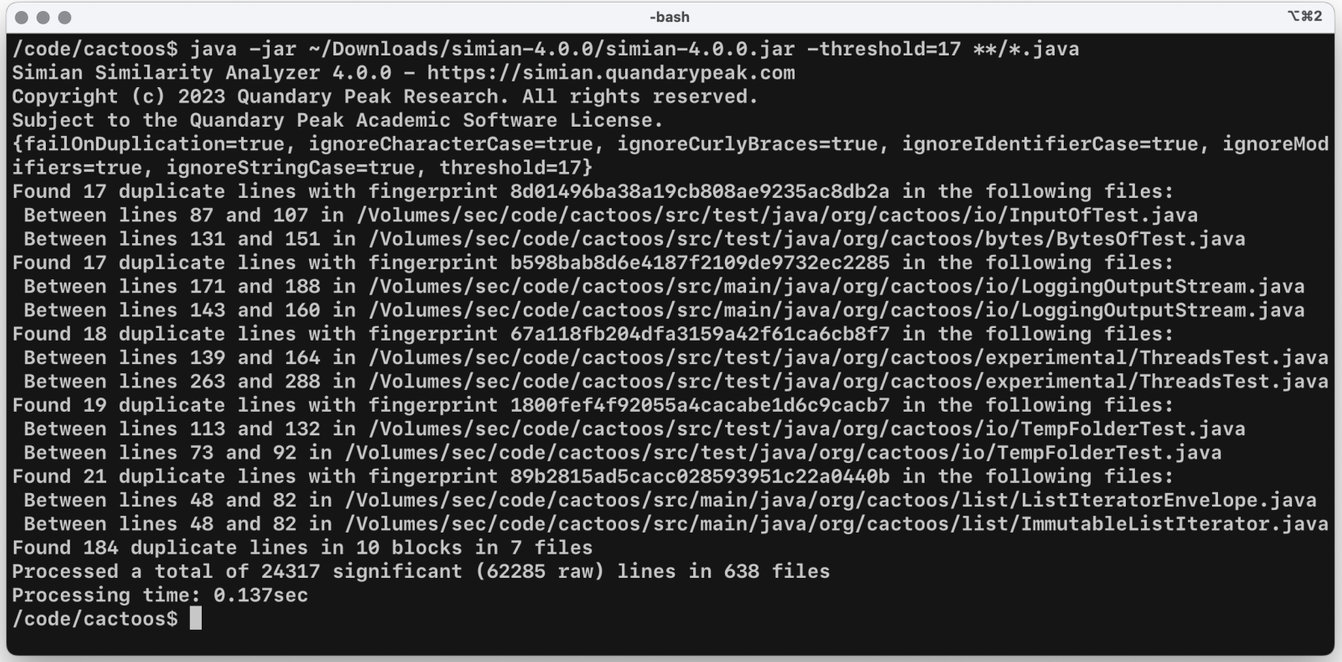
“For the three Java systems studied, the following results were found: 1) cloned code is usually older than non-cloned code, 2) cloned code in a file is usually older than the non-cloned code in the same file. Both results suggest that cloned code is more stable than non-cloned code.”

— 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

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
⌥⌘2
/code/cactoos$ 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, ignoreMod
ifiers=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/cactoos/src/test/java/org/cactoos/io/InputOfTest.java
 Between lines 131 and 151 in /Volumes/sec/code/cactoos/src/test/java/org/cactoos/bytes/BytesOfTest.java
Found 17 duplicate lines with fingerprint b598bab8d6e4187f2109de9732ec2285 in the following files:
 Between lines 171 and 188 in /Volumes/sec/code/cactoos/src/main/java/org/cactoos/io/LoggingOutputStream.java
 Between lines 143 and 160 in /Volumes/sec/code/cactoos/src/main/java/org/cactoos/io/LoggingOutputStream.java
Found 18 duplicate lines with fingerprint 67a118fb204dfa3159a42f61ca6cb8f7 in the following files:
 Between lines 139 and 164 in /Volumes/sec/code/cactoos/src/test/java/org/cactoos/experimental/ThreadsTest.java
 Between lines 263 and 288 in /Volumes/sec/code/cactoos/src/test/java/org/cactoos/experimental/ThreadsTest.java
Found 19 duplicate lines with fingerprint 1800fef4f92055a4cacabe1d6c9cacb7 in the following files:
 Between lines 113 and 132 in /Volumes/sec/code/cactoos/src/test/java/org/cactoos/io/TempFolderTest.java
 Between lines 73 and 92 in /Volumes/sec/code/cactoos/src/test/java/org/cactoos/io/TempFolderTest.java
Found 21 duplicate lines with fingerprint 89b2815ad5cacc028593951c22a0440b in the following files:
 Between lines 48 and 82 in /Volumes/sec/code/cactoos/src/main/java/org/cactoos/list/ListIteratorEnvelope.java
 Between lines 48 and 82 in /Volumes/sec/code/cactoos/src/main/java/org/cactoos/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/cactoos$

How Effective LLMs Are?

Table 2. Performance of LLM Models on Different Clone Types

Clone Type	Model	TP	FP	Recall
T1	GPT-3.5	300	0	1.00
	GPT-4	300	0	1.00
T2	GPT-3.5	169	131	0.56
	GPT-4	259	41	0.86
VST3	GPT-3.5	156	144	0.52
	GPT-4	283	17	0.94
ST3	GPT-3.5	133	167	0.44
	GPT-4	290	10	0.97
MT3	GPT-3.5	70	230	0.23
	GPT-4	262	38	0.87
WT3/T4	GPT-3.5	20	280	0.07
	GPT-4	68	232	0.23

"A correlation was observed between the GPTs' accuracy at identifying code clones and code similarity, with both GPT models exhibiting low effectiveness in detecting the most complex Type-4 code clones."

Source: Zixian Zhang and Takfarinas Saber. Assessing the Code Clone Detection Capability of Large Language Models. In *Proceedings of the 4th International Conference on Code Quality (ICCQ)*, pages 75–83. IEEE, 2024. doi:10.1109/ICCQ60895.2024.10576803

References

Qurat Ul Ain, Wasi Haider Butt, Muhammad Waseem Anwar, Farooque Azam, and Bilal Maqbool. A Systematic Review on Code Clone Detection. *IEEE Access*, 7(1), 2019. doi:10.1109/access.2019.2918202.

Brenda S. Baker. A Program for Identifying Duplicated Code, 1993.

Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000. doi:10.5555/318762.

Stefan Bellon, Rainer Koschke, Giulio Antoniol, Jens Krinke, and Ettore Merlo. Comparison and Evaluation of Clone Detection Tools. *IEEE Transactions on Software Engineering*, 33(9), 2007. doi:10.1109/tse.2007.70725.

Gerardo Casazza, Giuliano Antoniol, Umberto Villano, Ettore Merlo, and Massimiliano Di Penta. Identifying Clones in the Linux Kernel. In *Proceedings of the 1st International Workshop on Source Code Analysis and Manipulation*, pages 90–97. IEEE, 2001. doi:10.1109/SCAM.2001.972670.

Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999. doi:10.5555/311424.

Andrew Hunt and Dave Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Pearson Education, 1999. doi: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.

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.

Robert C. Martin. *Clean Code: A Handbook of Agile*

Software Craftsmanship. Pearson Education, 2008. doi:10.5555/1388398.

Steve McConnell. *Code Complete*. Pearson Education, 2004. doi:10.5555/1096143.

Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1988. doi:10.5555/534929.

Edward Yourdon and Larry Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice Hall, 1979. doi:10.5555/578522.

Zixian Zhang and Takfarinas Saber. Assessing the Code Clone Detection Capability of Large Language Models. In *Proceedings of the 4th International Conference on Code Quality (ICCQ)*, pages 75–83. IEEE, 2024. doi:10.1109/ICCQ60895.2024.10576803.