Clean Code
4

A Handbook of Agile Software Craftsmanship

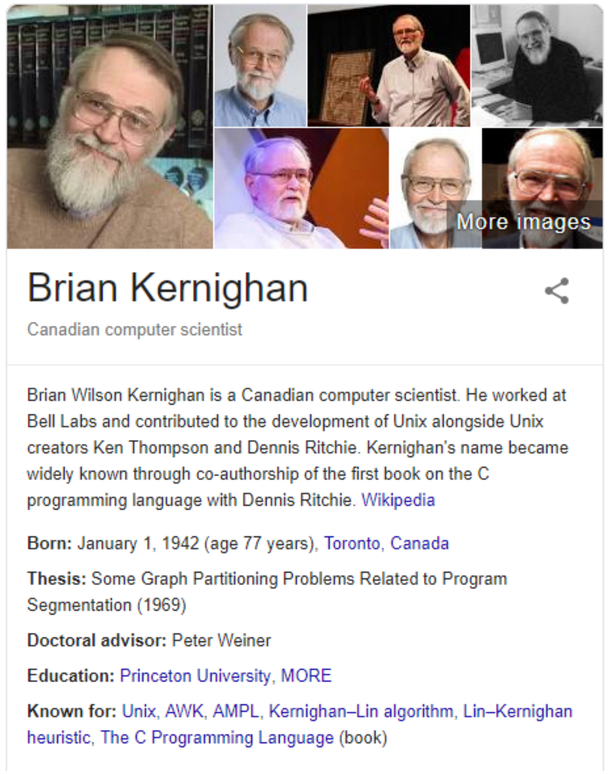
Robert C. Martin

Note by waegaein@github.com

CH4 Comments

"Don't comment bad code – rewrite it"

- Brian W. Kernighan and P. J. Plaugher



Brian Kernighan

Canadian computer scientist

Brian Wilson Kernighan is a Canadian computer scientist. He worked at Bell Labs and contributed to the development of Unix alongside Unix creators Ken Thompson and Dennis Ritchie. Kernighan's name became widely known through co-authorship of the first book on the C programming language with Dennis Ritchie. Wikipedia

Born: January 1, 1942 (age 77 years), Toronto, Canada

Thesis: Some Graph Partitioning Problems Related to Program Segmentation (1969)

Doctoral advisor: Peter Weiner

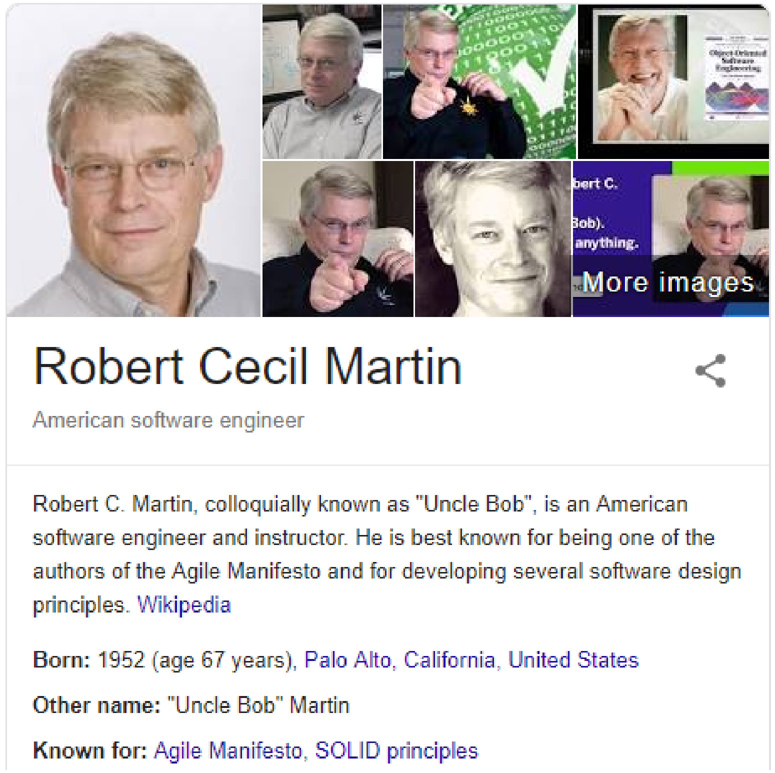
Education: Princeton University, MORE

Known for: Unix, AWK, AMPL, Kernighan–Lin algorithm, Lin–Kernighan heuristic, The C Programming Language (book)

CH4 Comments

“Comments do not make up for bad code”

- Robert C. Martin



Robert Cecil Martin

American software engineer

Robert C. Martin, colloquially known as "Uncle Bob", is an American software engineer and instructor. He is best known for being one of the authors of the Agile Manifesto and for developing several software design principles. Wikipedia

Born: 1952 (age 67 years), Palo Alto, California, United States

Other name: "Uncle Bob" Martin

Known for: Agile Manifesto, SOLID principles

CH4 Comments

Good comments are

- legal comments for copyright and authorship statements.

```
// Copyright (C) 2003, 2004, 2005 by Object Mentor, Inc. All rights reserved.
// Released under the terms of the GNU General Public License version 2 or later.
```

CH4 Comments

Good comments are

- explanations of the intents behind decisions expressed as codes.

```
// This is our best attempt to get a race condition
// by creating large number of threads.
for (int i = 0; i  < 25000; i++) {
        WidgetBuilderThread widgetBuilderThread;
        ...
        thread.start();
}
assertEquals(false, falFlag.get());
```

CH4 Comments

Good comments are

- TODOs that explain why the implementation has defects.

```
// TODO-MdM these are not needed
// We expect this to go away when we do the checkout model
protected VersionInfo makeVersion() throws Exception
{
        return null;
}
```

CH4 Comments

Bad comments are

- redundant explanation on the information stated by codes.

```
// Utility method that returns when this.closed is true.
// Throws an exception if the timeout is reached.
public synchronized void waitForClose(final long timeoutMillis)
throws Exception
{
        if (!closed)
        {
                wait(timeoutMillis);
                if (!closed)
                        throw new Exception(“MockResponseSender could not be closed”);
                }
        }
}
```

CH4 Comments

Bad comments are

- redundant explanation on the information stated by codes.

Remove it.

CH4 Comments

Bad comments are

- misleading statements that are not enough accurate.

CH4 Comments

Bad comments are

- misleading statements that are not enough accurate.

Rename your functions and variables.

Rewrite your logic.

Then remove it.

CH4 Comments

Bad comments are

- replacements for functions or variables.

```
// does the module from the global list <mod> depend on the
// subsystem we are part of?
if (smodule.getDependSubsystems().contains(subSysMod.getSubSystem())
```

CH4 Comments

Bad comments are

- replacements for functions or variables.

Use functions or variables.

```
ArrayList moduleDependees = smodule.getDependSubsystems();
String ourSubSystem = subSysMod.getSubSystem();
if (moduleDependees.contains(ourSubSystem));
```

CH4 Comments

Bad comments are

- syntactic comments for braces for attributes.

```
try {
        while ((line = in.readLine()) != null) {
                lineCount++;
                charCount += line.length();
                String words[] = line.split(“₩₩W”);
                wordcount += words.length;
        } //while
} //try
catch (IOException e) {
        System.err.println(“Error:” + e.getMessage());
} //catch
```

CH4 Comments

Bad comments are

- syntactic comments for braces for attributes.

Try shorten your function instead.

CH4 Comments

Bad comments are

- commented-out codes.

CH4 Comments

Bad comments are

- commented-out codes.

We have version control system.

Put the history in Git history.

CH4 Comments

Comments are, at best, a necessary evil.
The proper use of comments is to compensate for our **failure** to express in code.
Comments are always failures.

Why am I so down on comments?
Because they lie.
Not always, and not intentionally, but **too often**.

The reason is simple.
Programmers can't realistically **maintain** them.

Does programming matter that much?

My code is fine mostly…

- A chain is as strong as its **weakest** link.
- Mistakes made in the worst part would blow up the project.

I program well if I take care…

- Your programming ability is determined by your worst code.

Does programming matter that much?

Isn’t architecture more important?

- Programming is small but also a sort of architecture.
- Big architecture is even harder if you cannot properly do small.

You would understand the code if you read it carefully…

- Problem is that we have to read it **carefully**.
- It is just a waste of time to investigate, deduce, in order to understand.