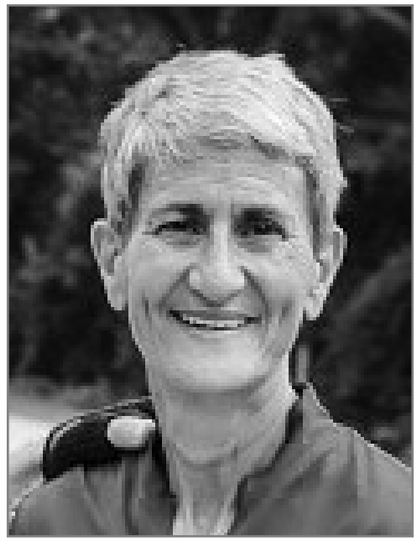
Lines of Code (LoC)

Yegor Bugayenko

Lecture #1 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video



Shari Lawrence Pfleeger

> “It’s not enough to make claims about your software; you must support your claims with measurable evidence.”

— Shari Lawrence Pfleeger. Software Metrics: Progress After 25 Years? *IEEE Software*, 25(6):32–34, 2008. doi:10.1109/MS.2008.160

1. Everybody wants higher quality of code, but nobody knows how to measure it.

> 2. Code maintainability, probably, is the ultimate objective of increasing the quality of source code.

> 3. Code size makes the biggest negative impact on code maintainability.

> 4. It may be wrong to measure productivity of a programmer by counting lines of code, but for the quality of code the LoC metric is a perfect indicator.

> 5. Instead of counting lines, it may be more reasonable to count NCSS (Non Commenting Source Statements), but not always.

6. There are 27.8M lines of C code in Linux kernel. What does it tell us?

7. In 2011, Uncle Bob suggested that 200 lines per Java class is a good guideline to stay below.

8. Java is two times more verbose than Ruby. Does it mean the quality of an average Ruby code is higher?

Read this:

The Formula for Software Quality (2017)

How Much Do You Pay Per Line of Code? (2014)

Hits-of-Code Instead of SLoC (2014)

Strict Control of Java Code Quality (2014)

References

Shari Lawrence Pfleeger. Software Metrics: Progress After 25 Years? *IEEE Software*, 25(6):32–34, 2008. doi:10.1109/MS.2008.160.