ECE444: Software Engineering

Metrics and Measurement 1

Shurui Zhou

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Learning Goals

- Use measurements as a decision tool to reduce uncertainty
- Understand difficulty of measurement; discuss validity of measurements
- Provide examples of metrics for software qualities and process
- Understand limitations and dangers of decisions and incentives based on measurements

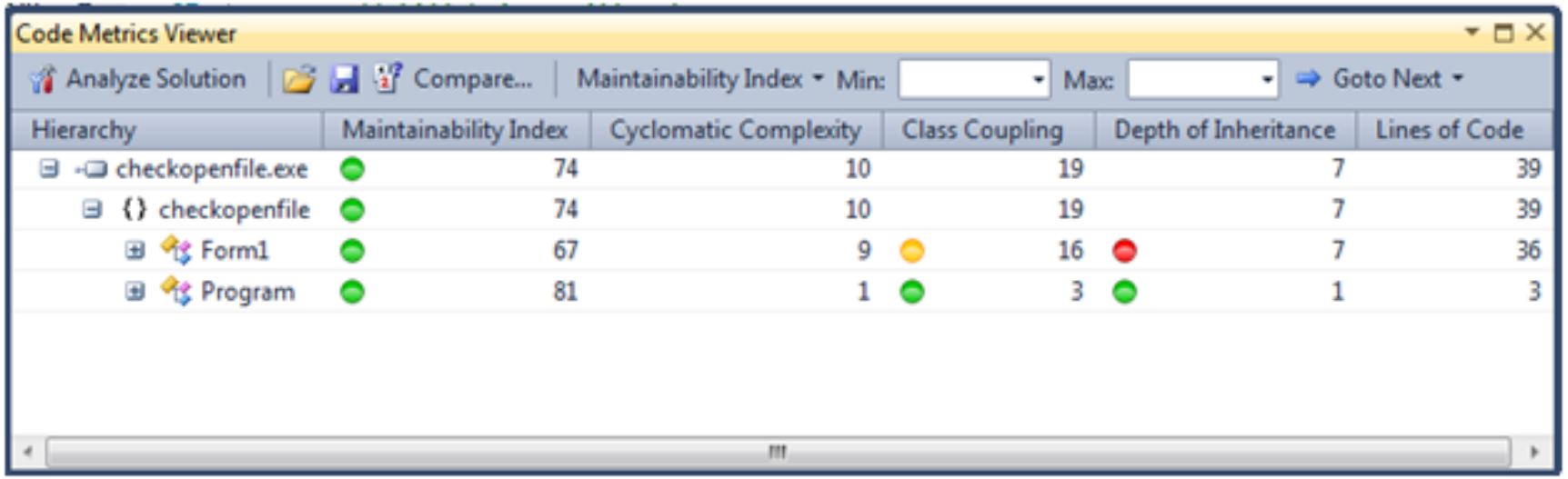
Software Engineering: Principles, practices (technical and non-technical) for **confidently** building **high-quality software**.

Case Study:
The Maintainability Index

Maintainability Index (Visual Studio since 2007)

Maintainability Index calculates an index value between 0 and 100 that represents the relative ease of maintaining the code. A **high value means better maintainability**.

- 0-9 = Red
- 10-19 = Yellow
- 20-100 = Green



https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2019
https://docs.microsoft.com/en-us/archive/blogs/codeanalysis/maintainability-index-range-and-meaning

Design Rational (from MSDN blog)

- "We noticed that as code tended toward 0 it was clearly hard to maintain code and the difference between code at 0 and some negative value was not useful."
- "The desire was that if the index showed red then we would be saying with a high degree of confidence that there was an issue with the code."

http://blogs.msdn.com/b/codeanalysis/archive/2007/11/20/maintainability-index-range-and-meaning.aspx

Maintainability Index (Visual Studio since 2007)

= 171

- 5.2 * log(Halstead Volume)
- 0.23 * (Cyclomatic Complexity)
- 16.2 * log(Lines of Code)

= MAX (0, (171
- 5.2 * log(Halstead Volume)
- 0.23 * (Cyclomatic Complexity)
- 16.2 * log(Lines of Code)
)*100 / 171)

Lines of Code

- Easy to measure

> wc –l file1 file2…

The wc (i.e., word count) command

-l : count only the number of lines

-w: count only the number of words

-m: count only the number of characters

-c: count only the number of bytes.

http://www.linfo.org/wc.html

Lines of Code

LOC	projects
450	Expression Evaluator
2.000	Sudoku, Functional Graph Library
40.000	OpenVPN
80-100.000	Berkeley DB, SQLlight
150-300.000	Apache, HyperSQL, Busybox, Emacs, Vim, ArgoUML
500-800.000	gimp, glibc, mplayer, php, SVN
1.600.000	gcc
6.000.000	Linux, FreeBSD
45.000.000	Windows XP

Normalizing Lines of Code

- Ignore comments and empty lines
- Ignore lines < 2 characters
- Pretty print source code first
- Count statements (logical lines of code)

```
/* How many lines of code is this? */
```

1

```
for (i = 0; i < 100; i += 1) printf("hello");
```

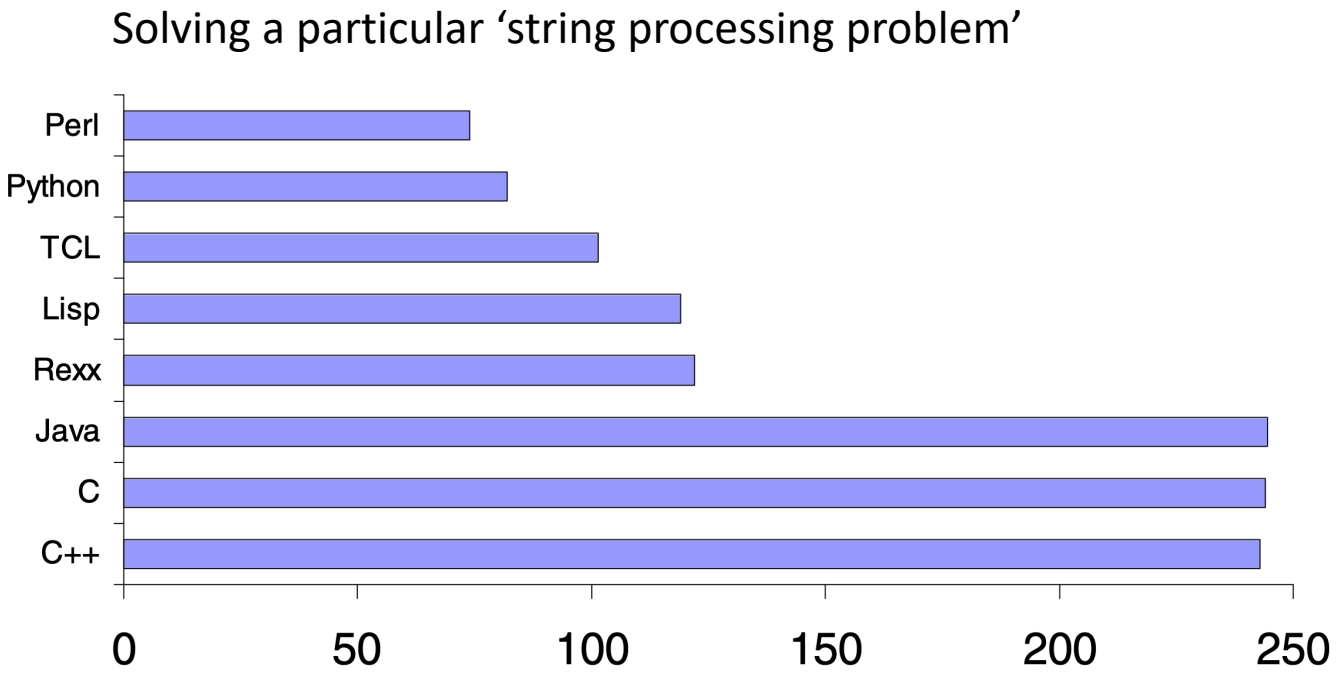
2

```
for (
            i = 0;
            i < 100;
            i += 1
    ) {
            printf("hello");
}
```

Normalizing per Language

Language	Statement factor (productivity)
C	1
C++	2.5
Fortran	2
Java	2.5
Perl	6
Python	6
Smalltalk	6

https://blog.codinghorror.com/are-all-programming-languages-the-same/



https://www.connellybarnes.com/documents/language_productivity.pdf

Maintainability Index (Visual Studio since 2007)

= 171

- 5.2 * log(Halstead Volume)
- 0.23 * (Cyclomatic Complexity)
- 16.2 * log(Lines of Code)

Halstead Volume

- Introduced by Maurice Howard Halstead in 1977
- Halstead Volume =
  number of operators&operands *
  log2(number of distinct operators&operands)
- Approximates size of elements and vocabulary

Halstead Volume - example

```
main()
{
    int a, b, c, avg;
    scanf("%d %d %d", &a, &b, &c);
    avg = (a + b + c) / 3;
    printf("avg = %d", avg);
}
```

Halstead Volume =
number of operators&operands *
log2(number of distinct operators&operands)

The unique operators are: `main` , `()` , `{}` , `int` , `scanf` , `&` , `=` , `+` , `/` , `printf` , `,` , `;` — 12

The unique operands are: `a` , `b` , `c` , `avg` , `"%d %d %d"` , `3` , `"avg = %d"` — 7

Volume: $V = 42 \times log_2 19 = 178.4$

Maintainability Index (Visual Studio since 2007)

= 171

- 5.2 * log(Halstead Volume)
- 0.23 * (Cyclomatic Complexity)
- 16.2 * log(Lines of Code)

Cyclomatic Complexity

- Proposed by McCabe 1976
- Based on control flow graph, measures number of linearly independent paths through a program
- linearly independent: each path has at least one edge that is not in one of the other paths.
  - no control flow statement: Complexity = 1
  - 1 single-condition IF statement --> 2 path: Complexity = 2
  - ...

Cyclomatic Complexity

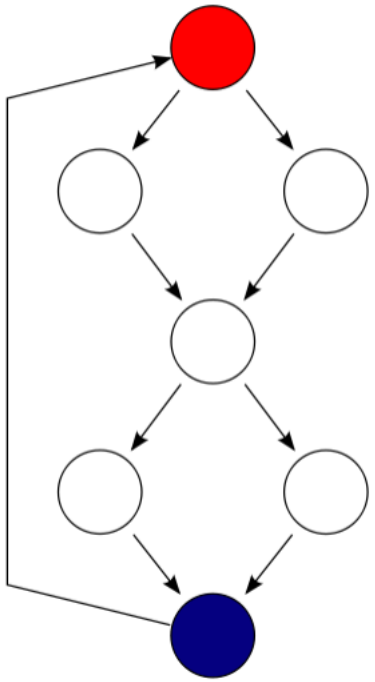
- Proposed by McCabe 1976
- Based on control flow graph, measures number of linearly independent paths through a program
- linearly independent: each path has at least one edge that is not in one of the other paths
- ~= number of decisions
- = Number of test cases needed to achieve branch coverage

Cyclomatic Complexity

M = #edges – #nodes + #end points

```
if (c1())
     f1();
else
     f2();

if (c2())
     f3();
else
     f4();
```



9 edges, 7 nodes and 1 end points:
M = 9 − 7 + 1 = 3

Application of Cyclomatic Complexity

- Limiting complexity during development
- Implications for software testing

```
if (c1())
    f1();
else
    f2();

if (c2())
    f3();
else
    f4();
```

c1() == True, c2() == True
c1() == False, c2() == False — Branch Coverage

c1() == True, c2() == False
c1() == False, c2() == True — Path Coverage

branch coverage $\leq$ cyclomatic complexity $\leq$ number of paths.

Maintainability Index (Origin)

Metrics for Assessing a Software System's Maintainability

Paul Oman and Jack Hagemeister

Software Engineering Test Lab
University of Idaho, Moscow, Idaho 83843
oman@cs.uidaho.edu

- Developers rated a number of HP systems in C and Pascal
- Statistical regression analysis to find key factors among 40 metrics

$$171 - 5.2 ln(\text{HV}) - 0.23\text{CC} - 16.2 ln(\text{LOC}) + 50.0 sin\sqrt{2.46 * \text{COM}}$$

= percentage of comments

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=242525

Maintainability Index (Origin)

Carnegie Mellon University
Software Engineering Institute

"*good and sufficient predictors of maintainability”*
"*potentially very useful for operational Department of Defense systems*".

Thoughts?

Thoughts

- Metric seems attractive
- Easy to compute
- Often seems to match intuition

- Parameters seem almost arbitrary, calibrated in single small study code (few developers, unclear statistical significance)
- All metrics related to size: just measure lines of code?
- Original 1992 C/Pascal programs potentially quite different from Java/JS/C# code

http://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/