Metrics and Measurement

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

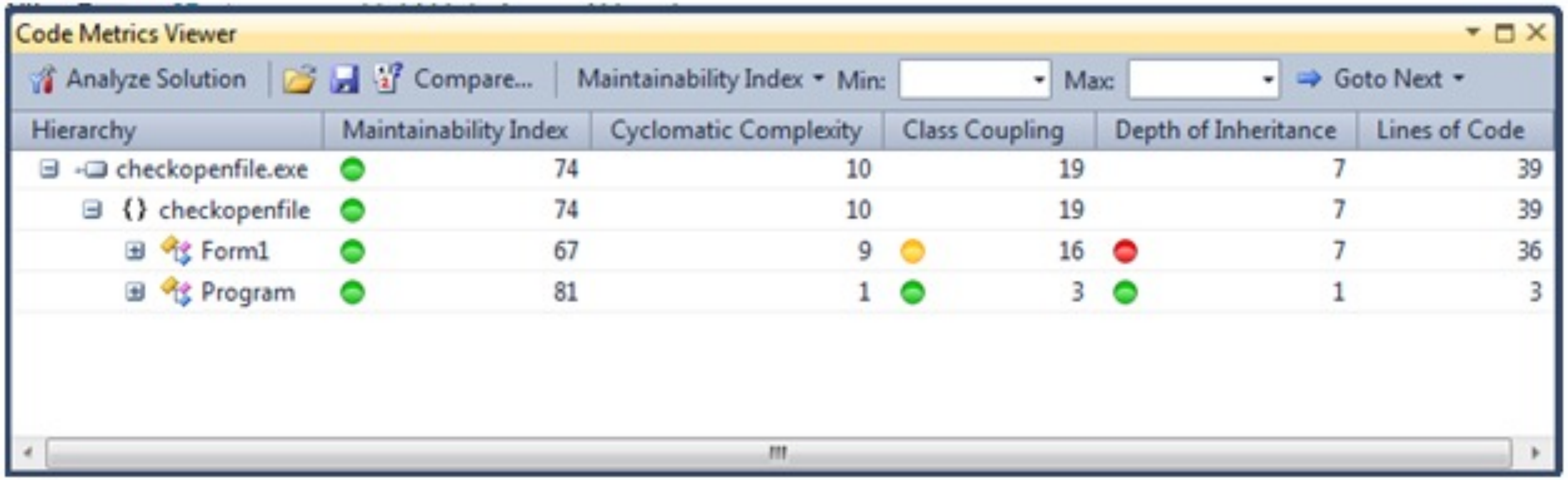
Software Engineering: Principles, practices (technical and non-technical) for **confidently** building **high-quality software**.

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**.

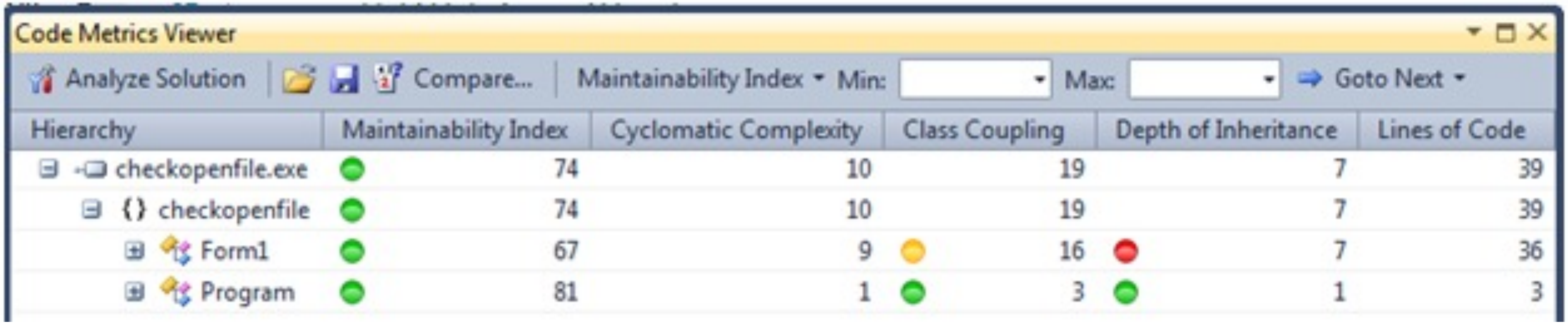
- 0-9 = Red
- 10-19 = Yellow
- 20-100 = Green



Hierarchy	Maintainability Index	Cyclomatic Complexity	Class Coupling	Depth of Inheritance	Lines of Code
checkopenfile.exe	🟢 74	10	19	7	39
{} checkopenfile	🟢 74	10	19	7	39
Form1	🟢 67	9	🟡 16	🔴 7	36
Program	🟢 81	1	🟢 3	🟢 1	3

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

Maintainability Index (Visual Studio since 2007)



Hierarchy	Maintainability Index	Cyclomatic Complexity	Class Coupling	Depth of Inheritance	Lines of Code
⊟ checkopenfile.exe	🟢 74	10	19	7	39
⊟ { } checkopenfile	🟢 74	10	19	7	39
⊞ Form1	🟢 67	9	🟡 16	🔴 7	36
⊞ Program	🟢 81	1	🟢 3	🟢 1	3

- Index between 0 and 100 representing the relative ease of maintaining the code.
- Higher is better. Color coded by number:
  - 0-9 = Red
  - 10-19 = Yellow
  - 20-100 = Green

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

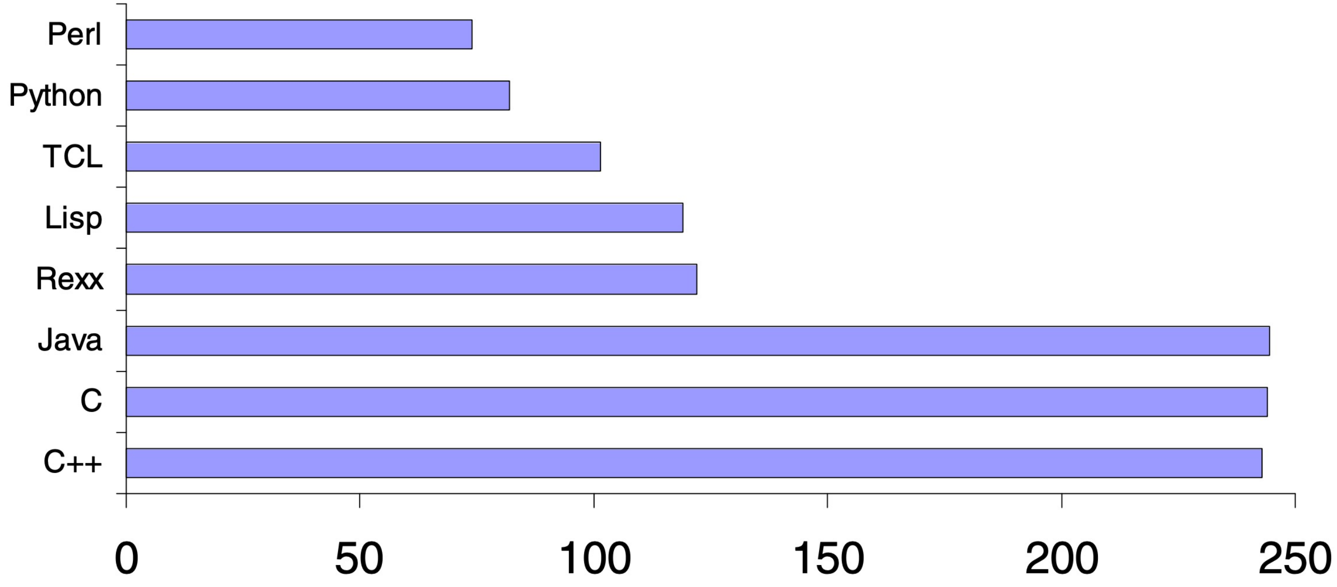
```
for (
        i = 0;
        i < 100;
        i += 1
    ) {
        printf("hello");
}
```

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/

Solving a particular ‘string processing problem’



Median Hours to Solve Problem

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

Maintainability Index (Visual Studio since 2007)

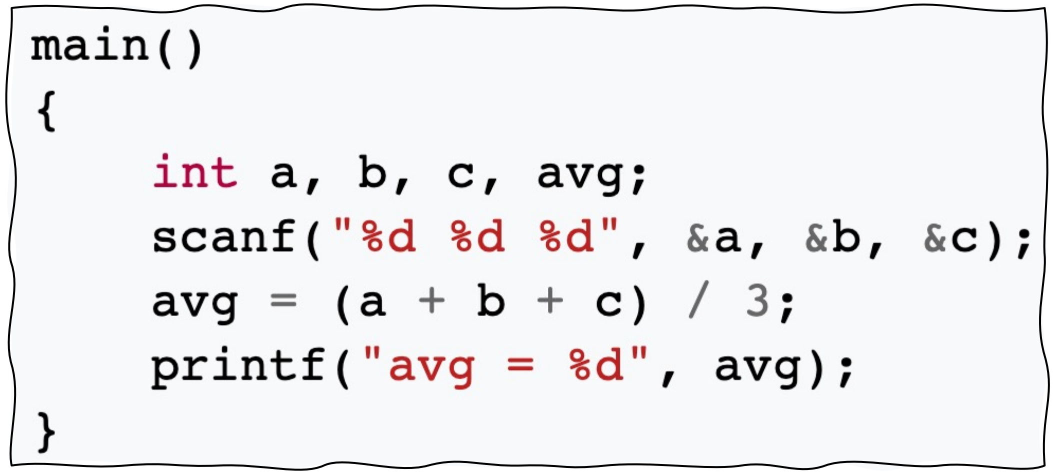
= 171

- 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

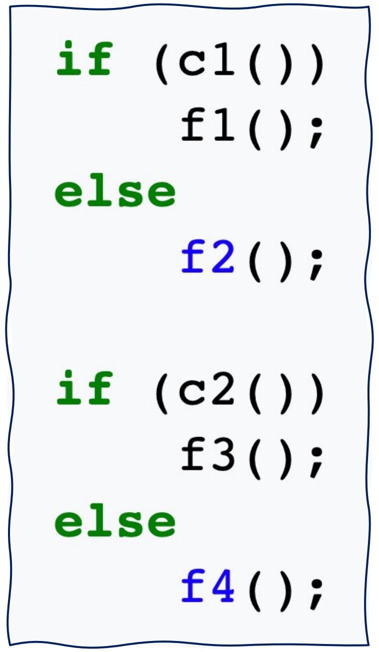
- Proposed by McCabe 1976
- Based on control flow graph, measures number of linearly independent paths through a program
- linearly independent: each path has at least one edge that is not in one of the other paths.
    - no control flow statement: Complexity = 1
    - 1 single-condition IF statement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

Cyclomatic Complexity

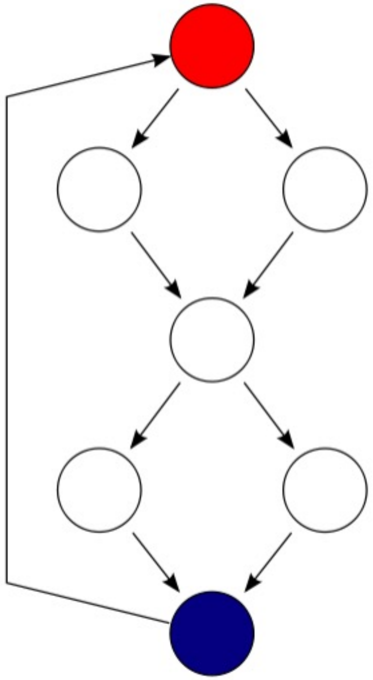
- Proposed by McCabe 1976
- Based on control flow graph, measures number of <u>linearly independent paths</u> through a program
- linearly independent: each path has at least one edge that is not in one of the other paths
- ~= number of decisions
- = Number of test cases needed to achieve branch coverage

Cyclomatic Complexity

M = #edges – #nodes + #end points



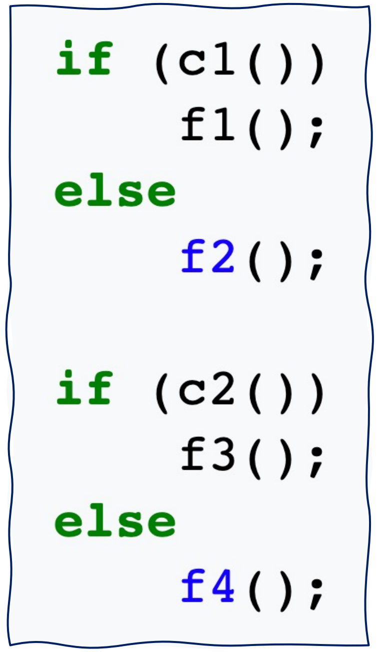
```
if (c1())
     f1();
else
     f2();

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

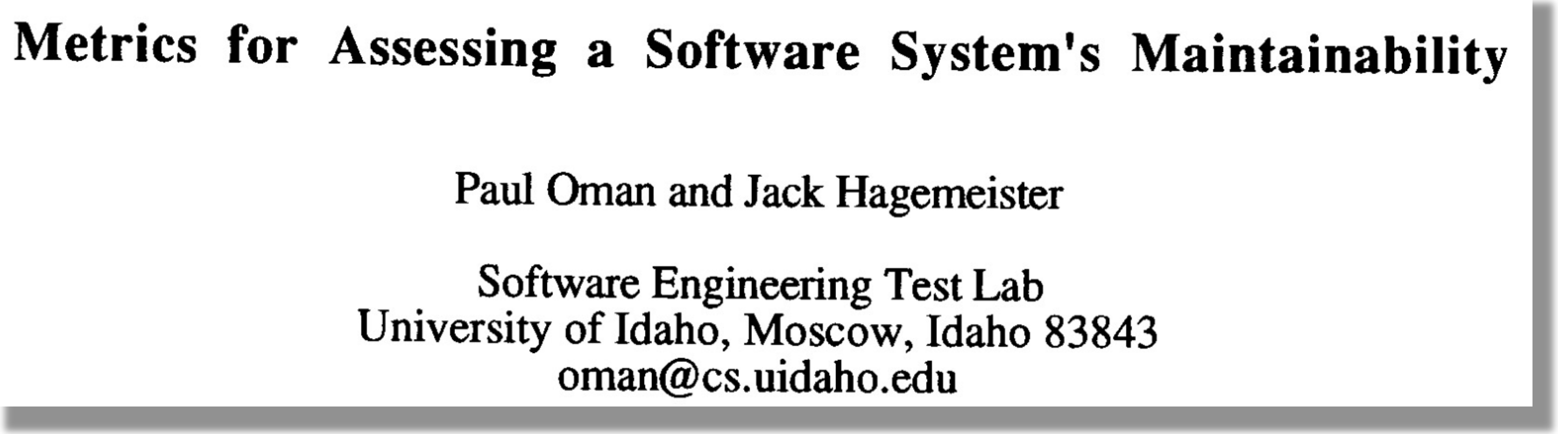
Branch Coverage

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/

Maintainability

- How easy is identifying and fixing a fault in software?
- Is it possible to identify the main cause of failure?
- How much effort will code modification require in case of a fault?
- How stable is the system performance while changes are being applied?

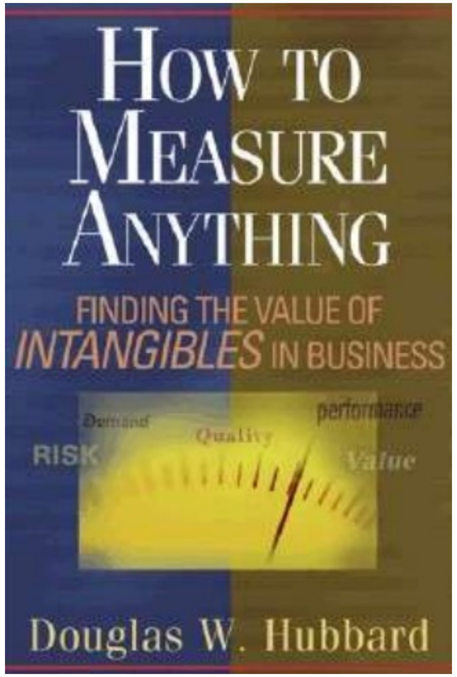
Key concerns of Maintainability Index

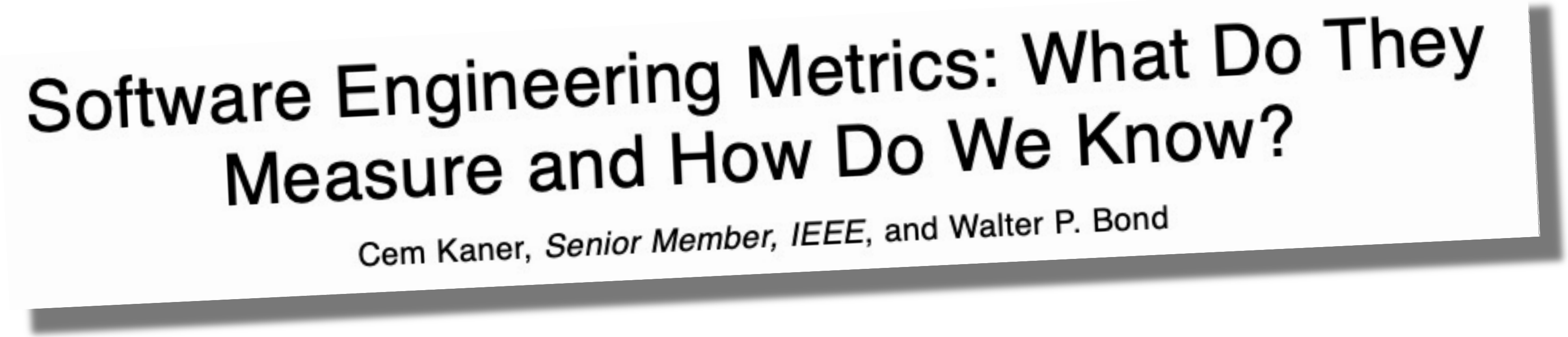
- There is no clear explanation for the specific derived formula.
- The only explanation that can be given is that all underlying metrics (Halstead, Cyclomatic Complexity, Lines of Code) are directly correlated with size (lines of code
- The set of programs used to derive the metric and evaluate it was small, and contained small programs only.
- Programs were written in C and Pascal, which may have rather different maintainability characteristics than current object-oriented languages such as C#, Java, or Javascript.
- For the experiments conducted, only few programs were analyzed, and no statistical significance was reported

Measurement for Decision Making in Software Development

What is Measurement?

- A quantitatively expressed reduction of uncertainty based on one or more observations.
- Measurement is the empirical, objective assignment of numbers, according to a rule derived from a model or theory, to attributes of objects or events with the intent of describing them.

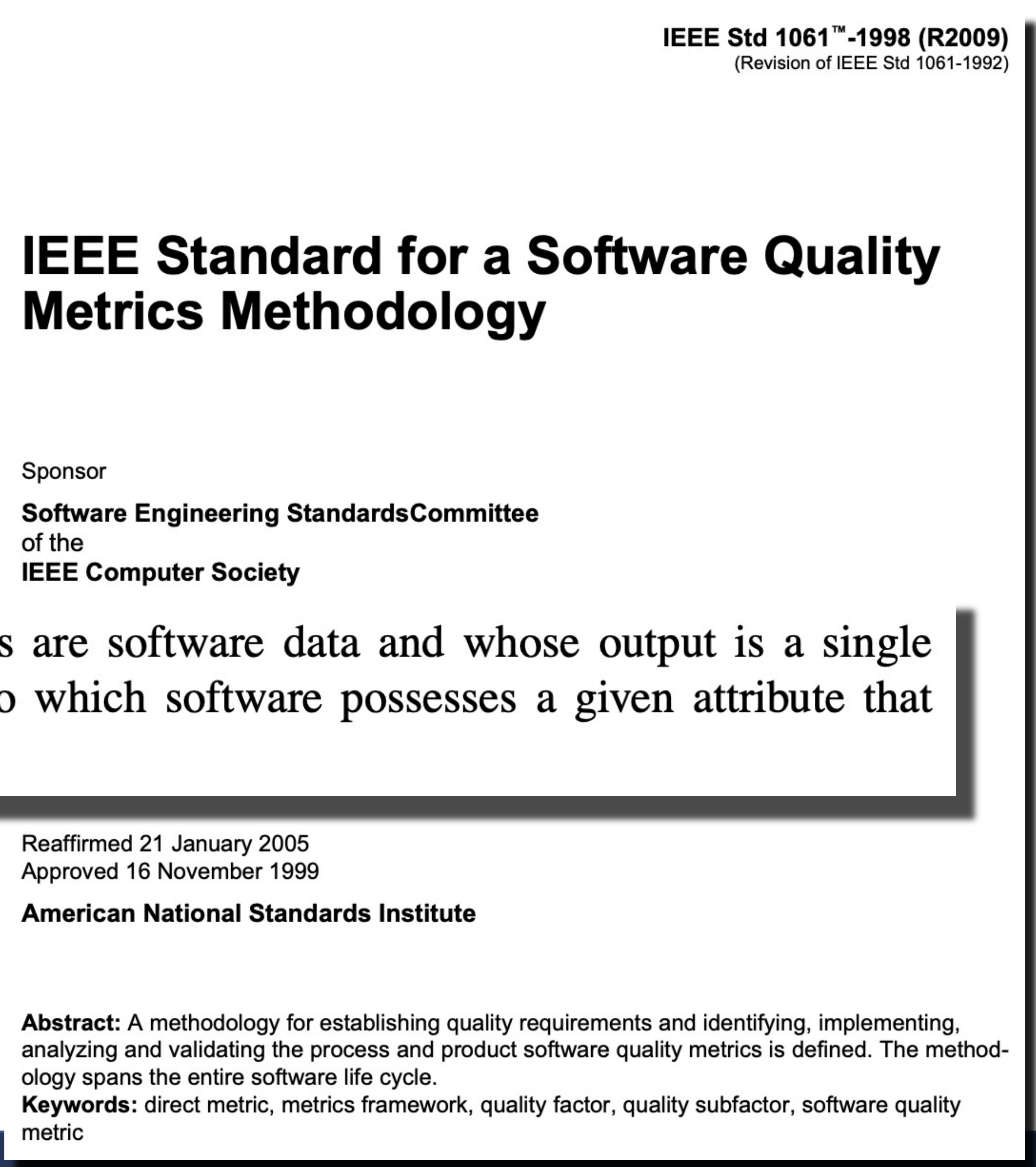




Software Engineering Metrics: What Do They Measure and How Do We Know?

Cem Kaner, *Senior Member, IEEE*, and Walter P. Bond

Software Quality Metric



IEEE Std 1061™-1998 (R2009)
(Revision of IEEE Std 1061-1992)

IEEE Standard for a Software Quality Metrics Methodology

Sponsor

Software Engineering StandardsCommittee
of the
IEEE Computer Society

Reaffirmed 21 January 2005
Approved 16 November 1999

American National Standards Institute

Abstract: A methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product software quality metrics is defined. The methodology spans the entire software life cycle.
Keywords: direct metric, metrics framework, quality factor, quality subfactor, software quality metric

2.24 software quality metric: A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=749159

What software qualities do we care about? (examples)

What software qualities did you pick for your app? (examples)

- Scalability
- Security
- Extensibility
- Documentation
- Performance
- Consistency
- Portability
- Installability
- Maintainability
- Functionality (e.g., data integrity)
- Availability
- Ease of use

What **process qualities** do we care about? (examples)

What **process qualities** do we care about? (examples)

- On-time release
- Development speed
- Meeting efficiency
- Conformance to processes
- Time spent on rework
- Reliability of predictions
- Fairness in decision making

- Measure time, costs, actions, resources, and quality of work packages; compare with predictions
- Use information from issue trackers, communication networks, team structures, etc…

Everything is measurable

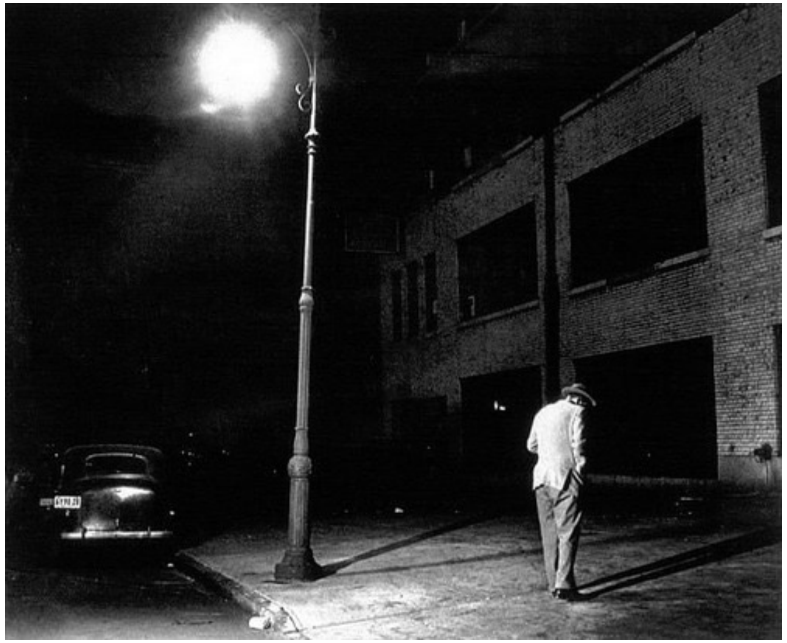
- If X is something we care about, then X, by definition, must be detectable.
  - How could we care about things like “quality,” “risk,” “security,” or “public image” if these things were totally undetectable, directly or indirectly?
  - If we have reason to care about some unknown quantity, it is because we think it corresponds to desirable or undesirable results in some way.
- If X is detectable, then it must be detectable in some amount.
  - If you can observe a thing at all, you can observe more of it or less of it
- If we can observe it in some amount, then it must be measurable.

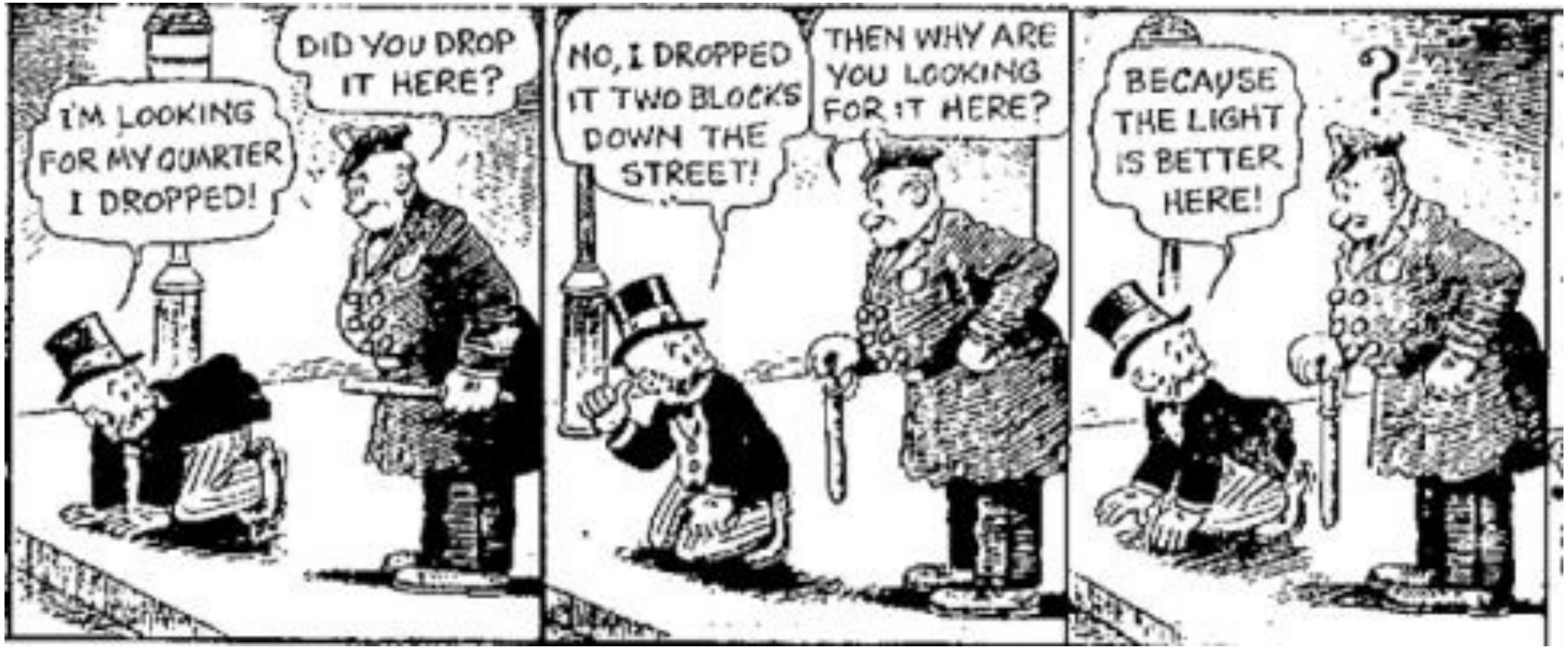
D. Hubbard, How to Measure Anything, 2010

Questions to consider.

- What properties do we care about, and how do we measure it?
- What is being measured? Does it (to what degree) capture the thing you care about? What are its limitations?
- How should it be incorporated into process? Check in gate? Once a month? Etc.
- What are potentially negative side effects or incentives?

Measurement is Difficult





I'M LOOKING FOR MY QUARTER I DROPPED!
DID YOU DROP IT HERE?
NO, I DROPPED IT TWO BLOCKS DOWN THE STREET!
THEN WHY ARE YOU LOOKING FOR IT HERE?
BECAUSE THE LIGHT IS BETTER HERE!
?

The streetlight effect

- A known observational bias.
- People tend to look for something only where it’s easiest to do so.
  - If you drop your keys at night, you’ll tend to look for it under streetlights.

What could possibly go wrong?

- Bad statistics: A basic misunderstanding of measurement theory and what is being measured.
- Bad decisions: The incorrect use of measurement data, leading to unintended side effects.
- Bad incentives: Disregard for the human factors, or how the cultural change of taking measurements will affect people.

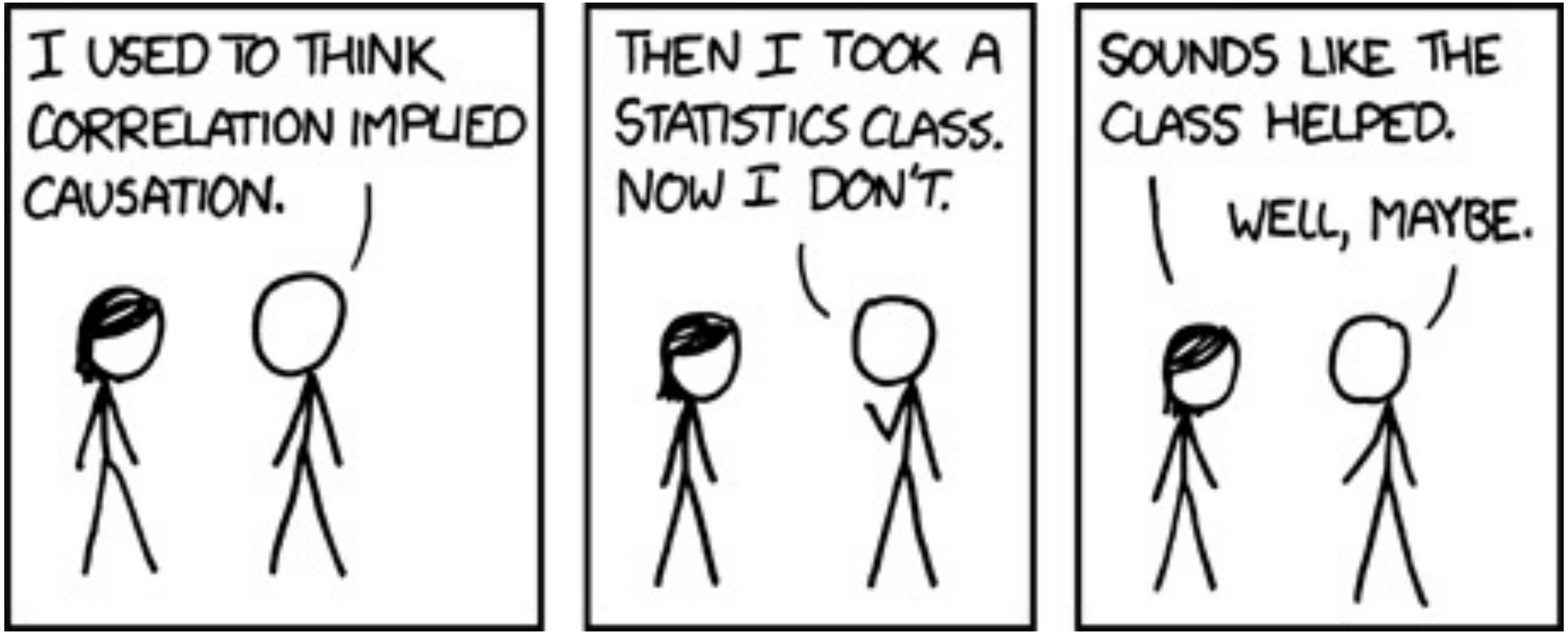
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1000457

Measurements validity

- Construct – Are we measuring what we intended to measure?
- Predictive – The extent to which the measurement can be used to explain some other characteristic of the entity being measured
- External validity – Concerns the generalization of the findings to contexts and environments, other than the one studied

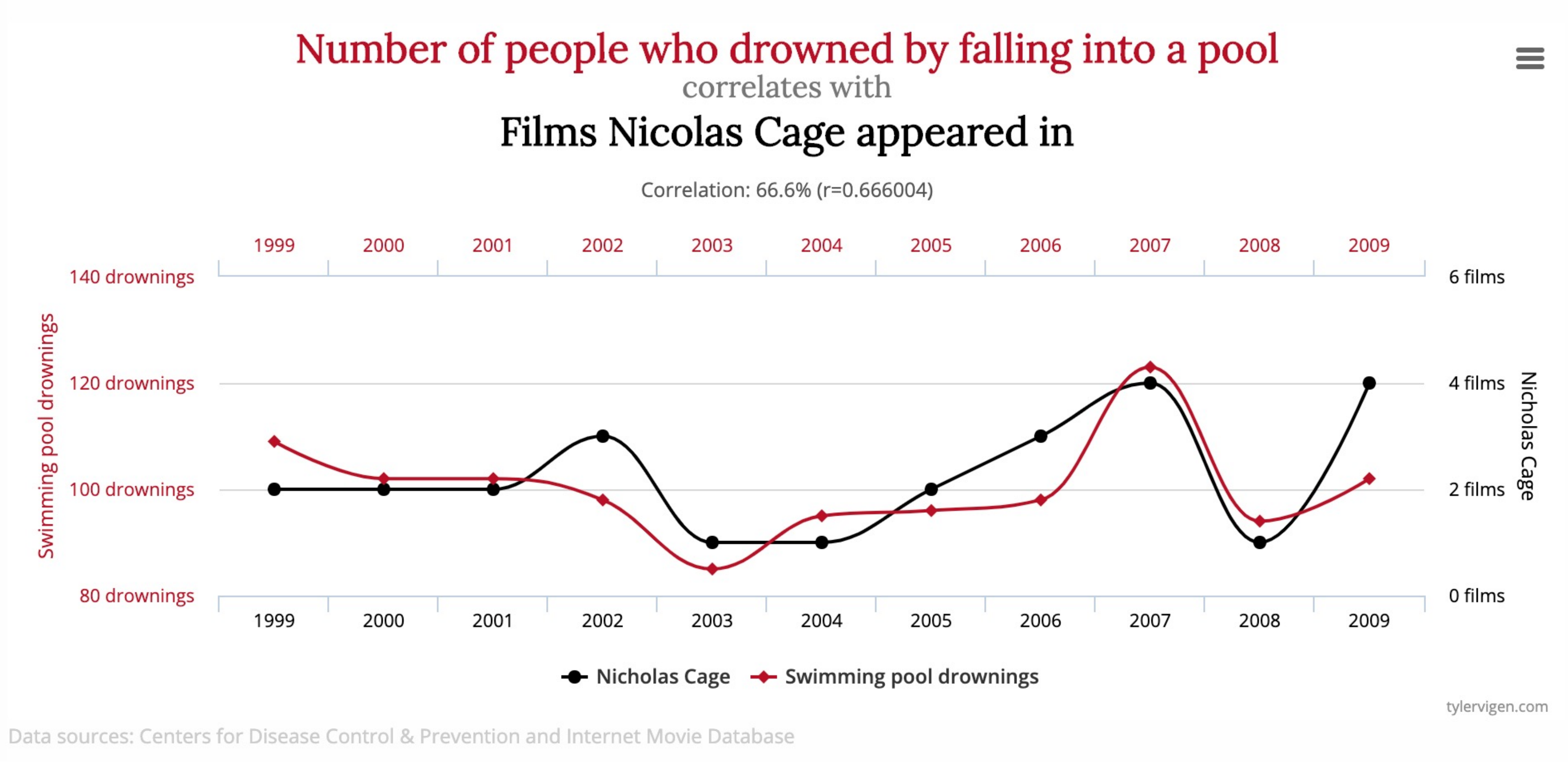
Correlation

- Independent variable X and dependent variable Y
- Influence of X on Y, e.g.
  - Influence of file size on error rate
  - Influence of comments on understandability
  - Influence of GUI on usability (speed)
  - Influence of heap size on performance
  - Influence of #abstract methods on #test cases
- Comparing two or more metrics
  - All metrics need to be well defined separately
- Statistical relationship?


I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

http://xkcd.com/552/

- For causation
  - Provide a theory (from domain knowledge, independent of data)
  - Show correlation
  - Demonstrate ability to predict new cases (replicate/validate)

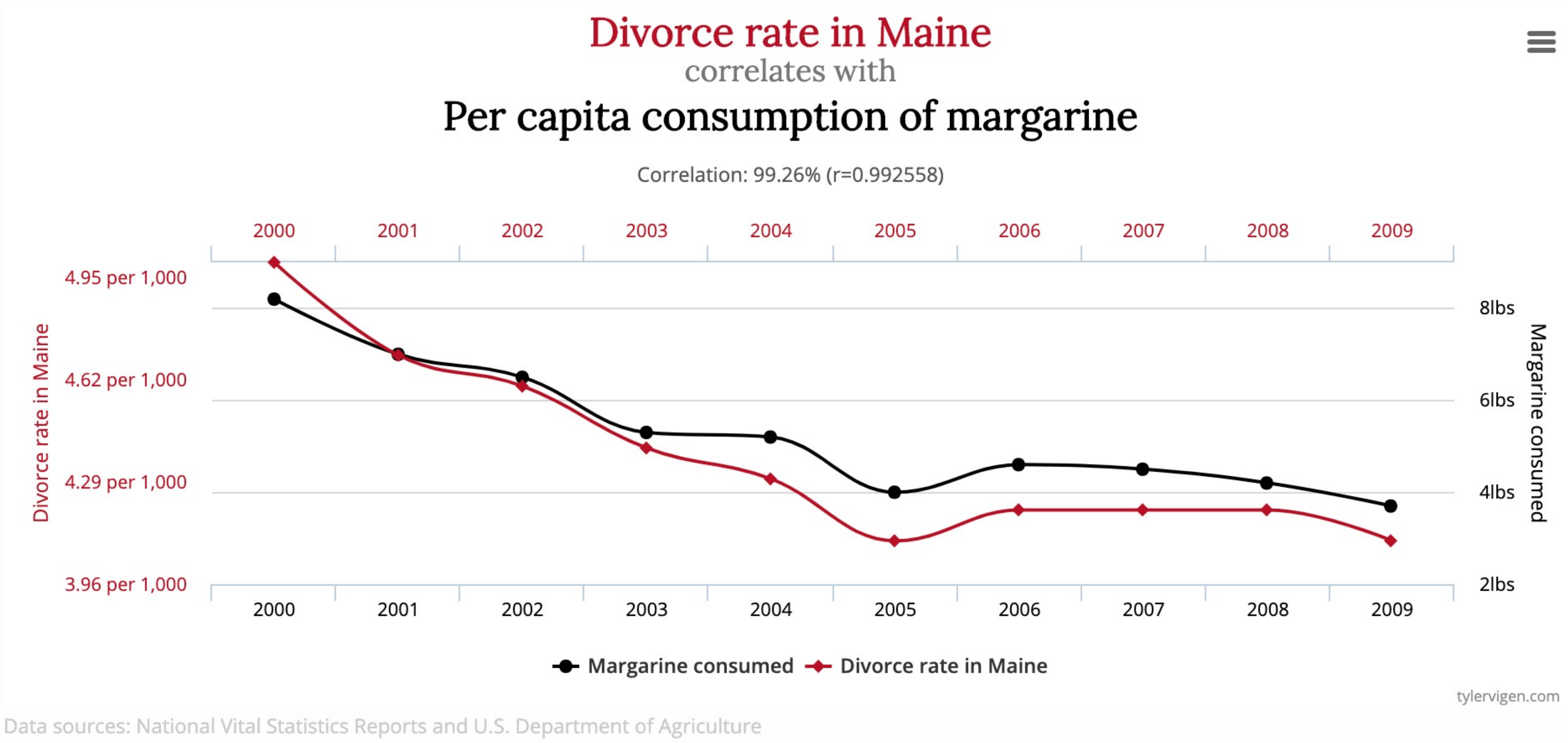


Number of people who drowned by falling into a pool
correlates with
Films Nicolas Cage appeared in

Correlation: 66.6% (r=0.666004)

Data sources: Centers for Disease Control & Prevention and Internet Movie Database

http://www.tylervigen.com/spurious-correlations



Divorce rate in Maine

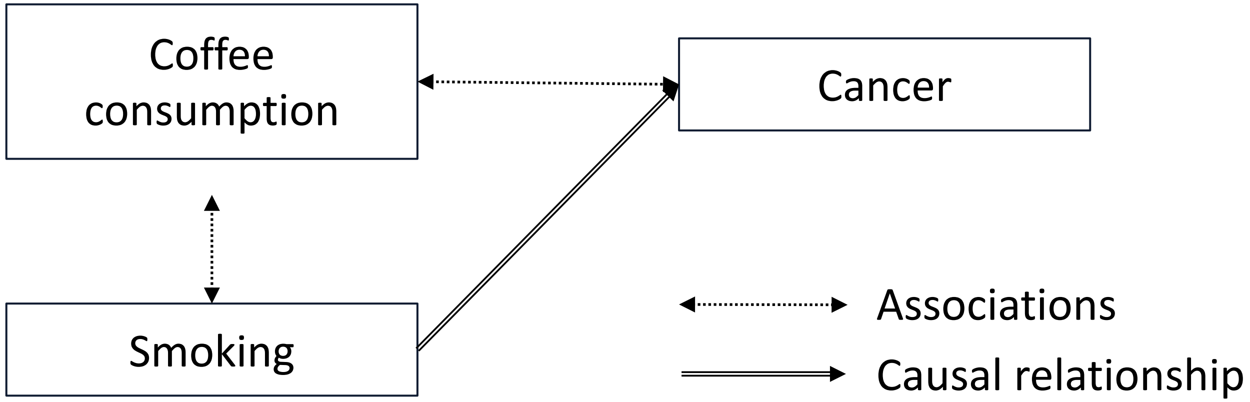
correlates with

Per capita consumption of margarine

Correlation: 99.26% (r=0.992558)

Data sources: National Vital Statistics Reports and U.S. Department of Agriculture

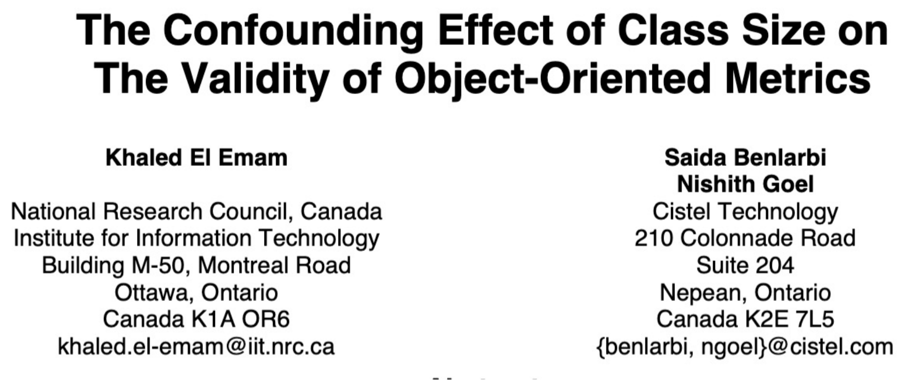
Confounding variables



- If you look only at the coffee consumption → cancer relationship, you can get very misleading results
- Smoking is a confounder

Confounding variables

- "Only 4, out of 24 commonly used object-oriented metrics, were actually useful in predicting the quality of a software module when the effect of the module size was accounted for."

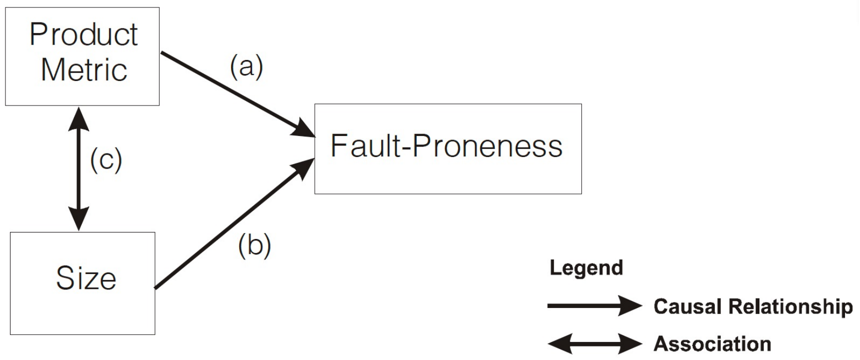


The Confounding Effect of Class Size on The Validity of Object-Oriented Metrics

Khaled El Emam

National Research Council, Canada
Institute for Information Technology
Building M-50, Montreal Road
Ottawa, Ontario
Canada K1A OR6
khaled.el-emam@iit.nrc.ca

Saida Benlarbi
Nishith Goel
Cistel Technology
210 Colonnade Road
Suite 204
Nepean, Ontario
Canada K2E 7L5
{benlarbi, ngoel}@cistel.com



The McNamara fallacy

Quickie!

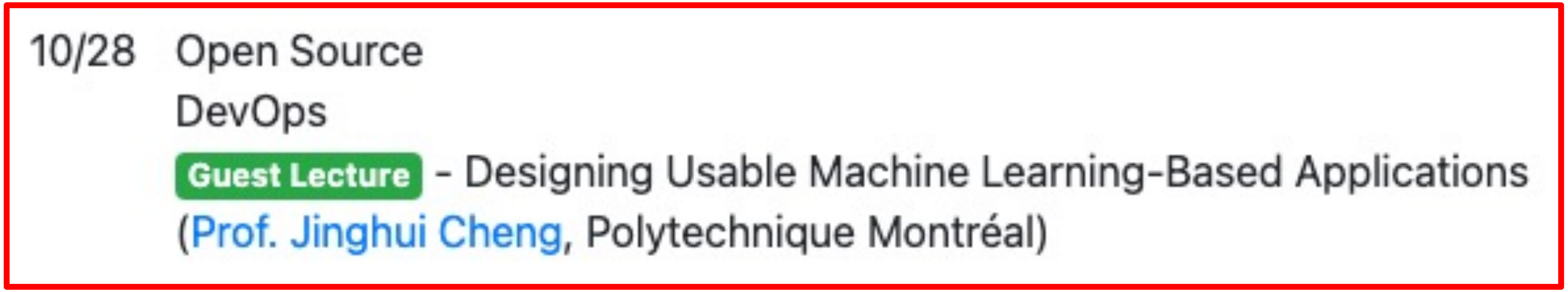
THE McNAMARA FALLACY

The McNamara Fallacy

- Measure whatever can be easily measured.
- Disregard that which cannot be measured easily.
- Presume that which cannot be measured easily is not important.
- Presume that which cannot be measured easily does not exist.

— Daniel Yankelovich, "Corporate Priorities: A continuing study of the new demands on business" (1972).

Discussion: Measuring Usability



10/28 Open Source

DevOps

Guest Lecture - Designing Usable Machine Learning-Based Applications

(Prof. Jinghui Cheng, Polytechnique Montréal)

Discussion: Usability

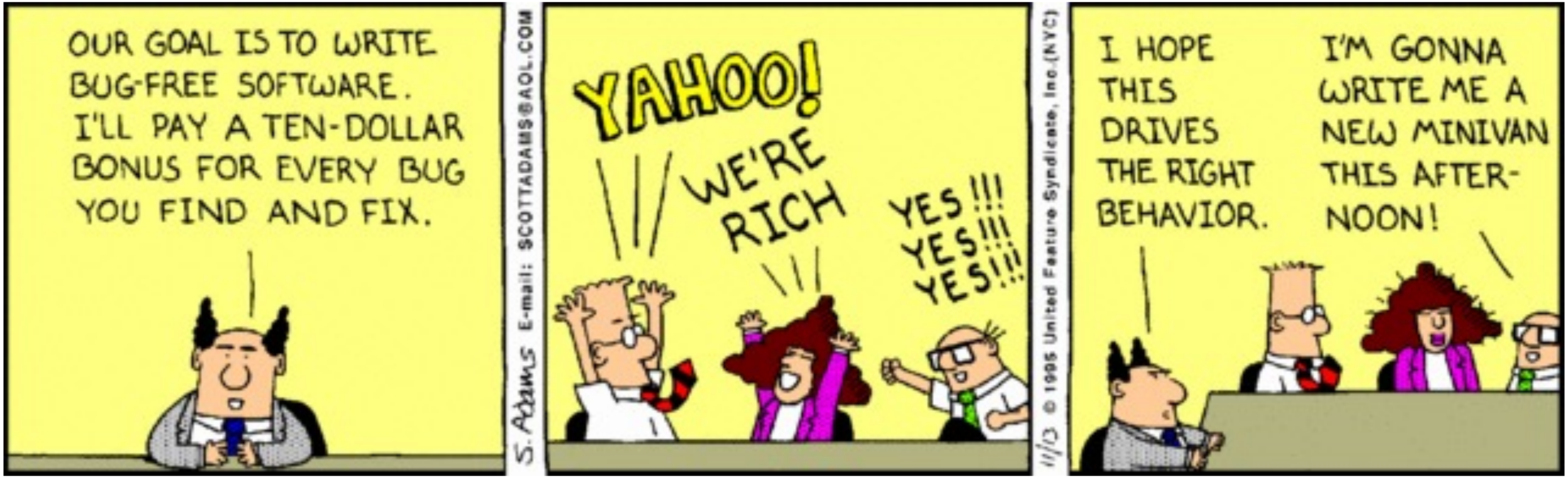
- Users can see directly how well this attribute of the system is worked out.
- One of the critical problems of usability is too much interaction or too many actions necessary to accomplish a task.
- Examples of important indicators for this attribute are:
  - List of supported devices, OS versions, screen resolutions, and browsers and their versions.
  - Elements that accelerate user interaction, such as “hotkeys,” “lists of suggestions,” and so on.
  - The average time a user needs to perform individual actions.
  - Support of accessibility for people with disabilities.

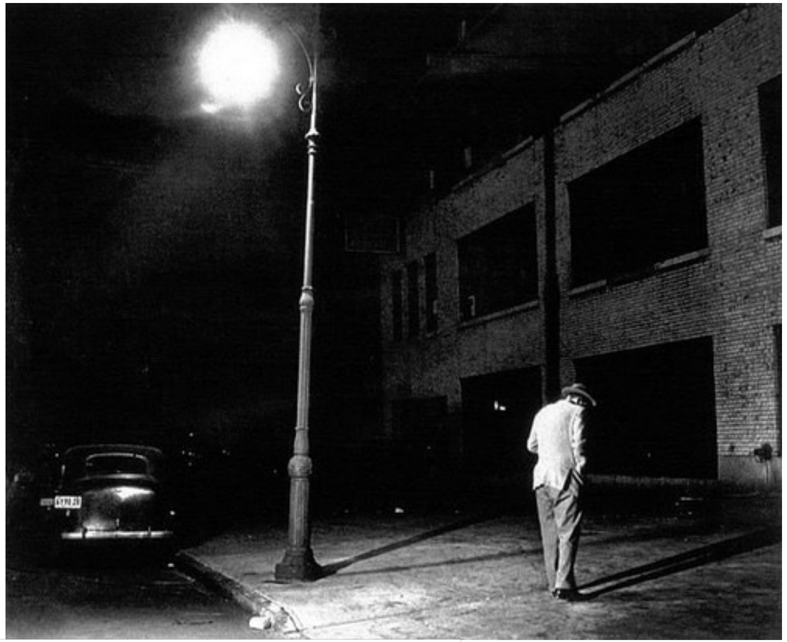
Measurement strategies

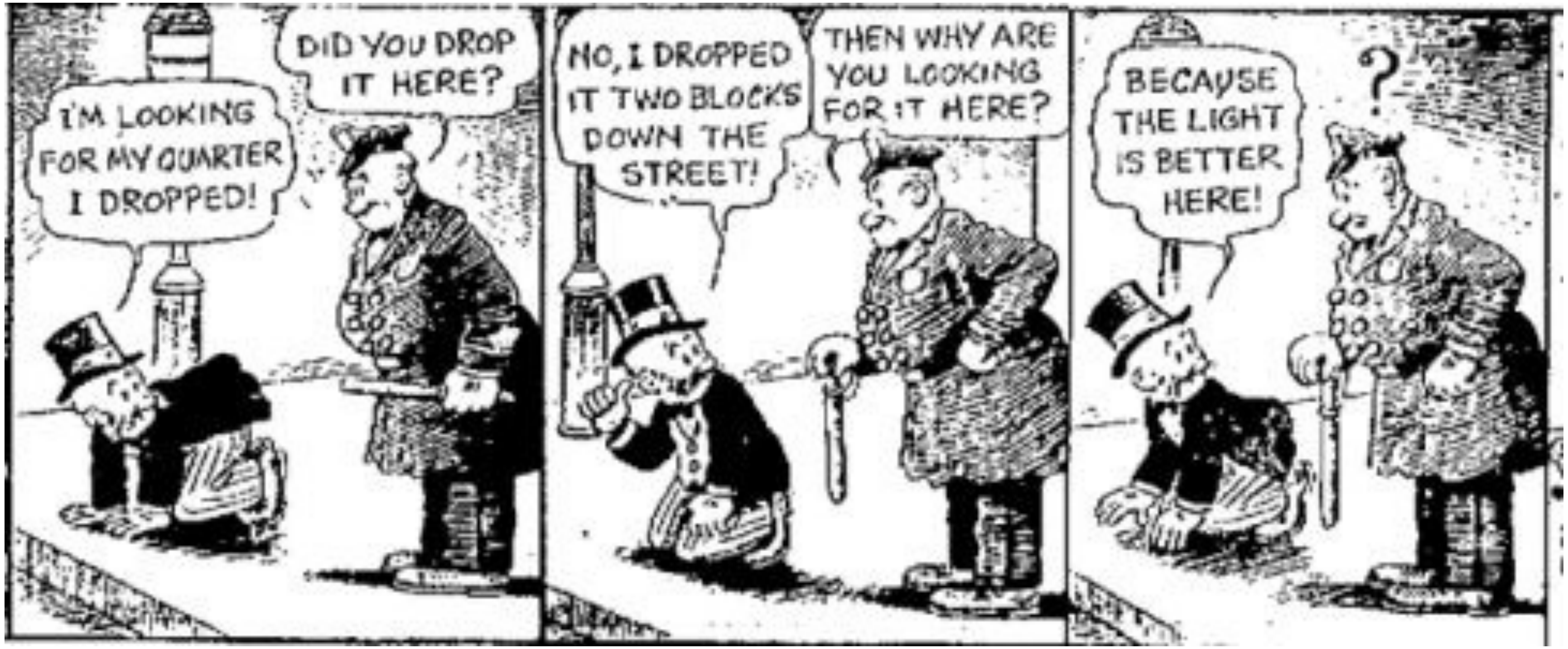
- Automated measures on code repositories
- Use or collect process data
- Instrument program (e.g., in-field crash reports)
- Surveys, interviews, controlled experiments, expert judgment
- Statistical analysis of sample

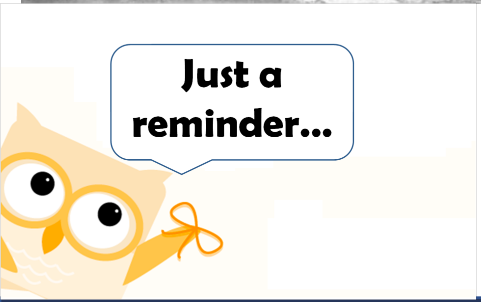
Metrics and Incentives

Goodhart’s law: “When a measure becomes a target, it ceases to be a good measure.”


OUR GOAL IS TO WRITE BUG-FREE SOFTWARE. I'LL PAY A TEN-DOLLAR BONUS FOR EVERY BUG YOU FIND AND FIX.
S. Adams E-mail: SCOTTADAMS@AOL.COM
YAHOO!
WE'RE RICH
YES!!! YES!!! YES!!!
11/13 © 1995 United Feature Syndicate, Inc.(NYC)
I HOPE THIS DRIVES THE RIGHT BEHAVIOR.
I'M GONNA WRITE ME A NEW MINIVAN THIS AFTER-NOON!




I'M LOOKING FOR MY QUARTER I DROPPED!
DID YOU DROP IT HERE?
NO, I DROPPED IT TWO BLOCKS DOWN THE STREET!
THEN WHY ARE YOU LOOKING FOR IT HERE?
BECAUSE THE LIGHT IS BETTER HERE!
?

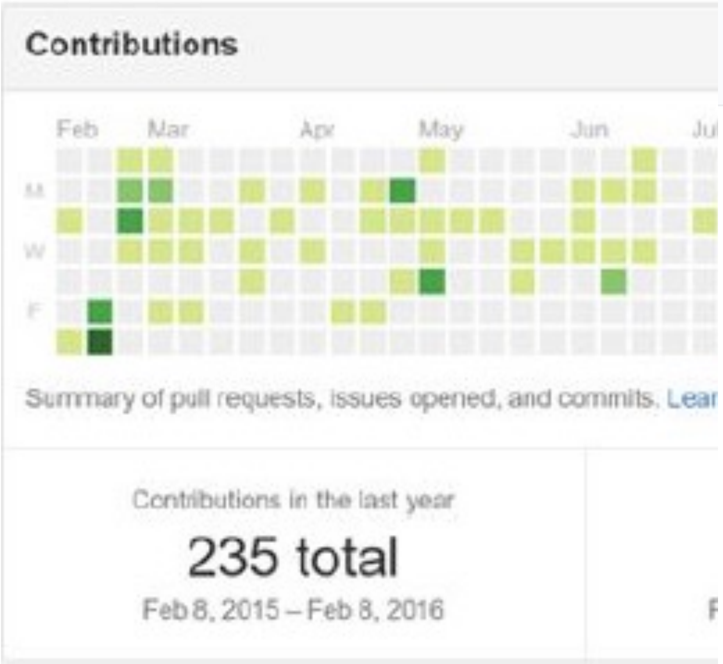

Just a reminder...

> “Measuring programming progress by lines of code is like measuring aircraft building progress by weight.”

“In IBM there's a religion in software that says you have to count K-LOCs, ... How big a project is it? ... And IBM wanted to sort of make it the religion about how we got paid. How much money we made off OS 2, how much they did. How many K-LOCs did you do? And we kept trying to convince them - hey, if we have - a developer's got a good idea and he can get something done in 4K-LOCs instead of 20K-LOCs, should we make less money? Because he's made something smaller and faster, less KLOC.”

--- Steve Ballmer

https://www.pbs.org/nerds/part2.html



Contributions

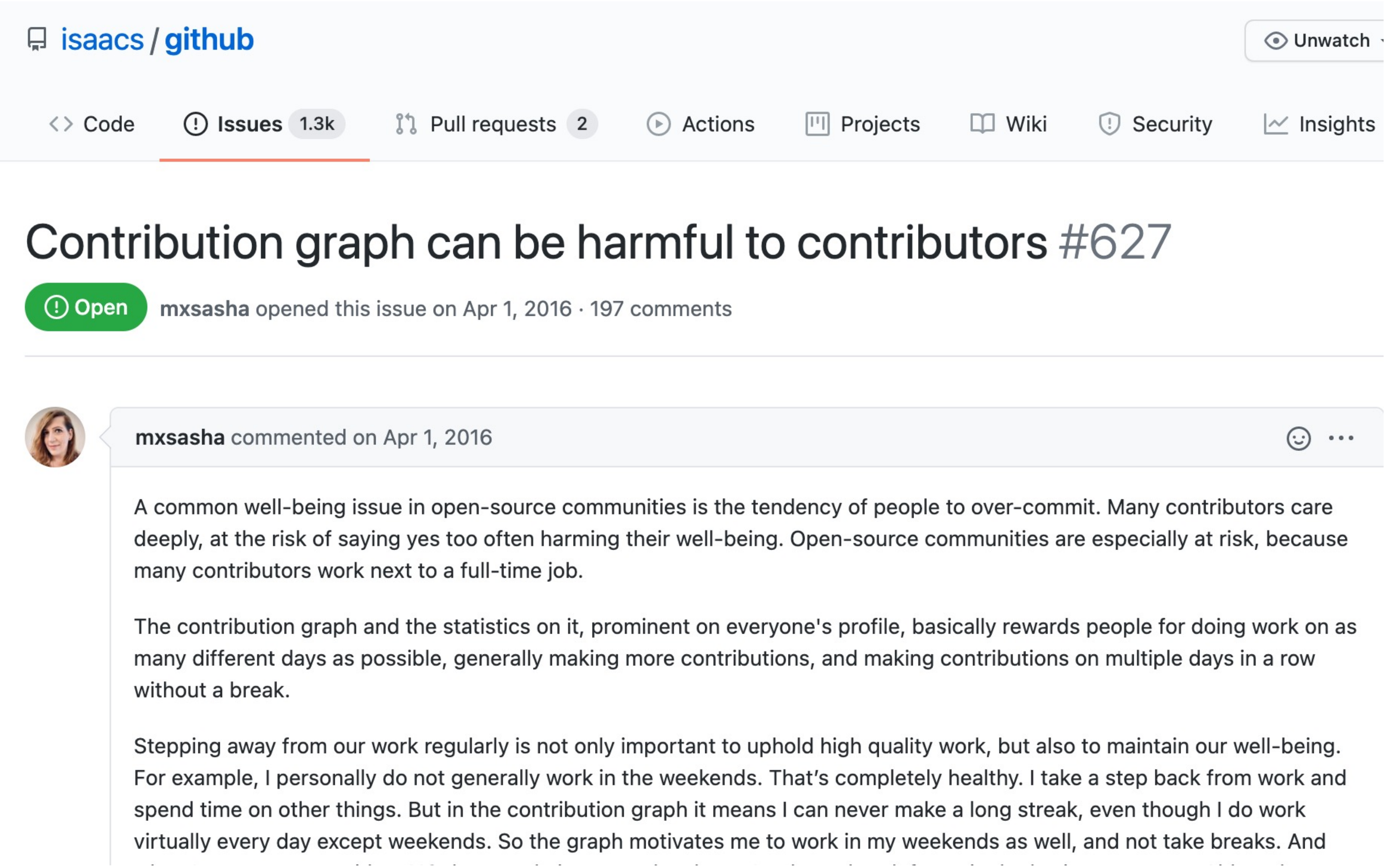
Summary of pull requests, issues opened, and commits. Lear

Contributions in the last year

235 total

Feb 8, 2015 – Feb 8, 2016

F



isaacs / github

Contribution graph can be harmful to contributors #627

Open mxsasha opened this issue on Apr 1, 2016 · 197 comments

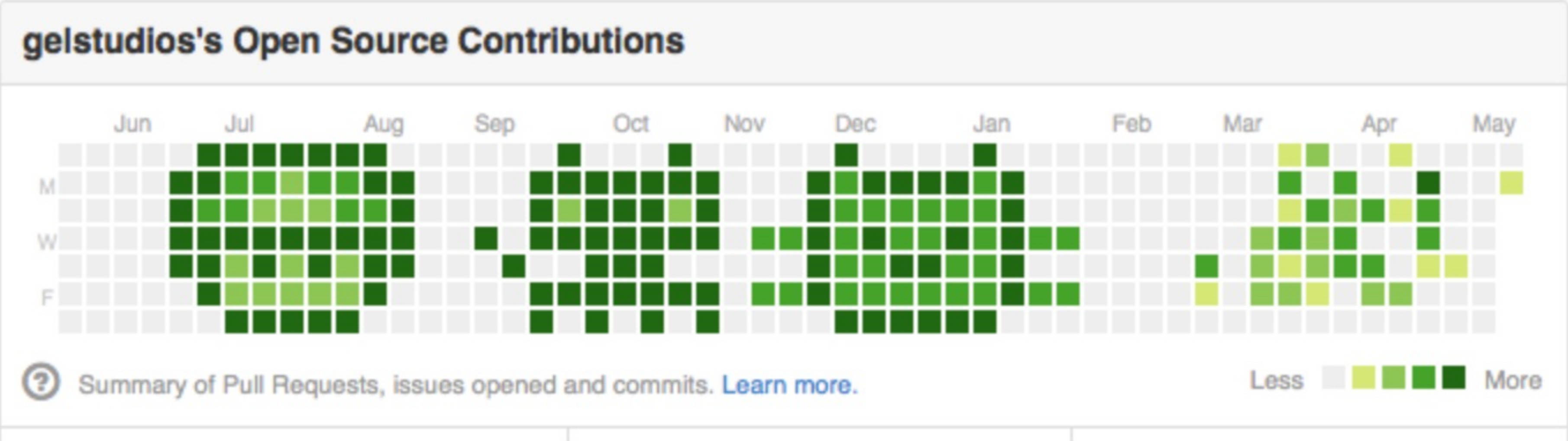
mxsasha commented on Apr 1, 2016

A common well-being issue in open-source communities is the tendency of people to over-commit. Many contributors care deeply, at the risk of saying yes too often harming their well-being. Open-source communities are especially at risk, because many contributors work next to a full-time job.

The contribution graph and the statistics on it, prominent on everyone's profile, basically rewards people for doing work on as many different days as possible, generally making more contributions, and making contributions on multiple days in a row without a break.

Stepping away from our work regularly is not only important to uphold high quality work, but also to maintain our well-being. For example, I personally do not generally work in the weekends. That's completely healthy. I take a step back from work and spend time on other things. But in the contribution graph it means I can never make a long streak, even though I do work virtually every day except weekends. So the graph motivates me to work in my weekends as well, and not take breaks. And

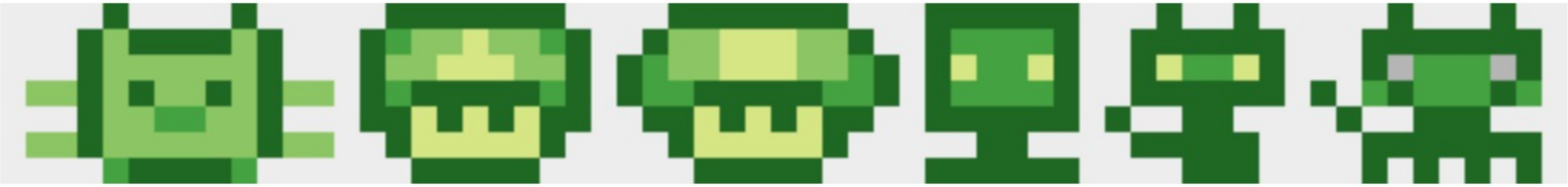
Contributing graphs considered h
https://www.hanselman.com/



gelstudios's Open Source Contributions

Summary of Pull Requests, issues opened and commits. Learn more.

Pixel Art



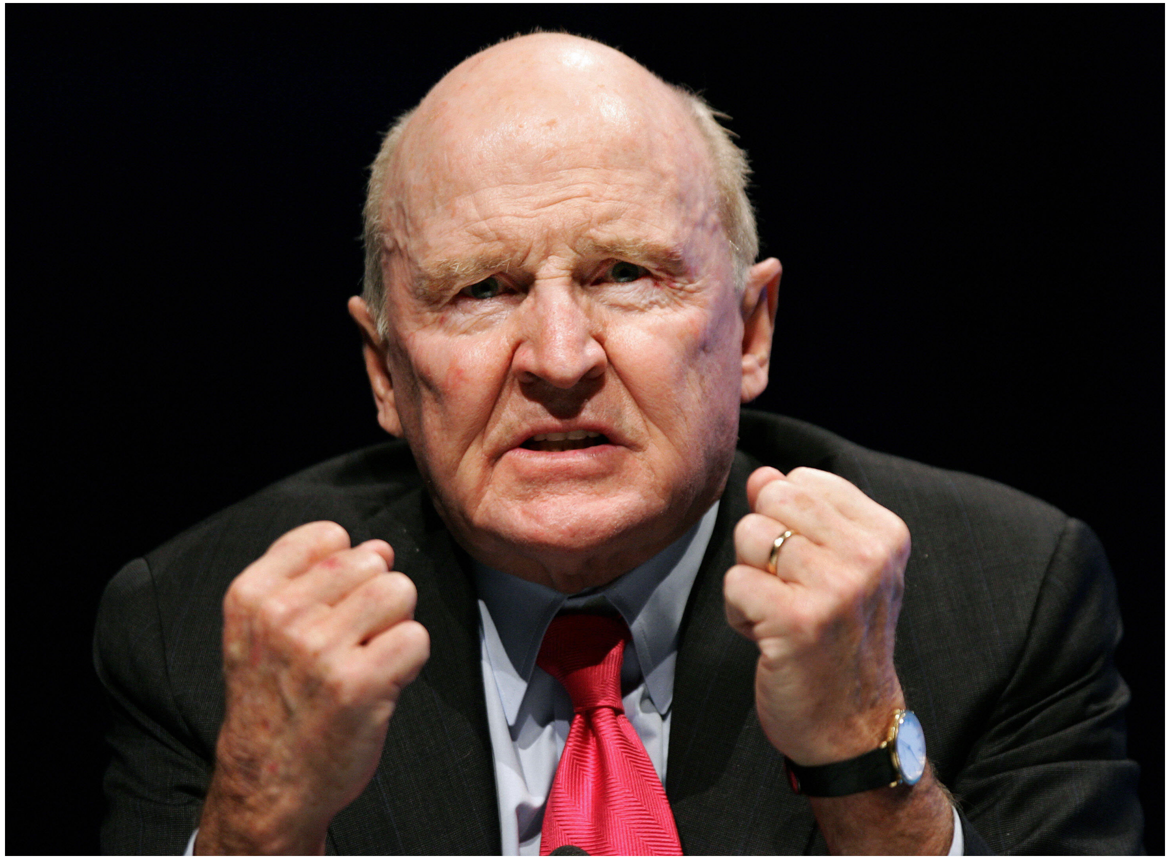
Included "art" from left to right: kitty, oneup, oneup2, hackerschool, octocat, octocat2

https://github.com/gelstudios/gitfiti

Productivity Metrics

- Lines of code per day?
  - Industry average 10-50 lines/day
  - Debugging + rework ca. 50% of time
- Function/object/application points per month
- Bugs fixed?
- Milestones reached?

Stack Ranking

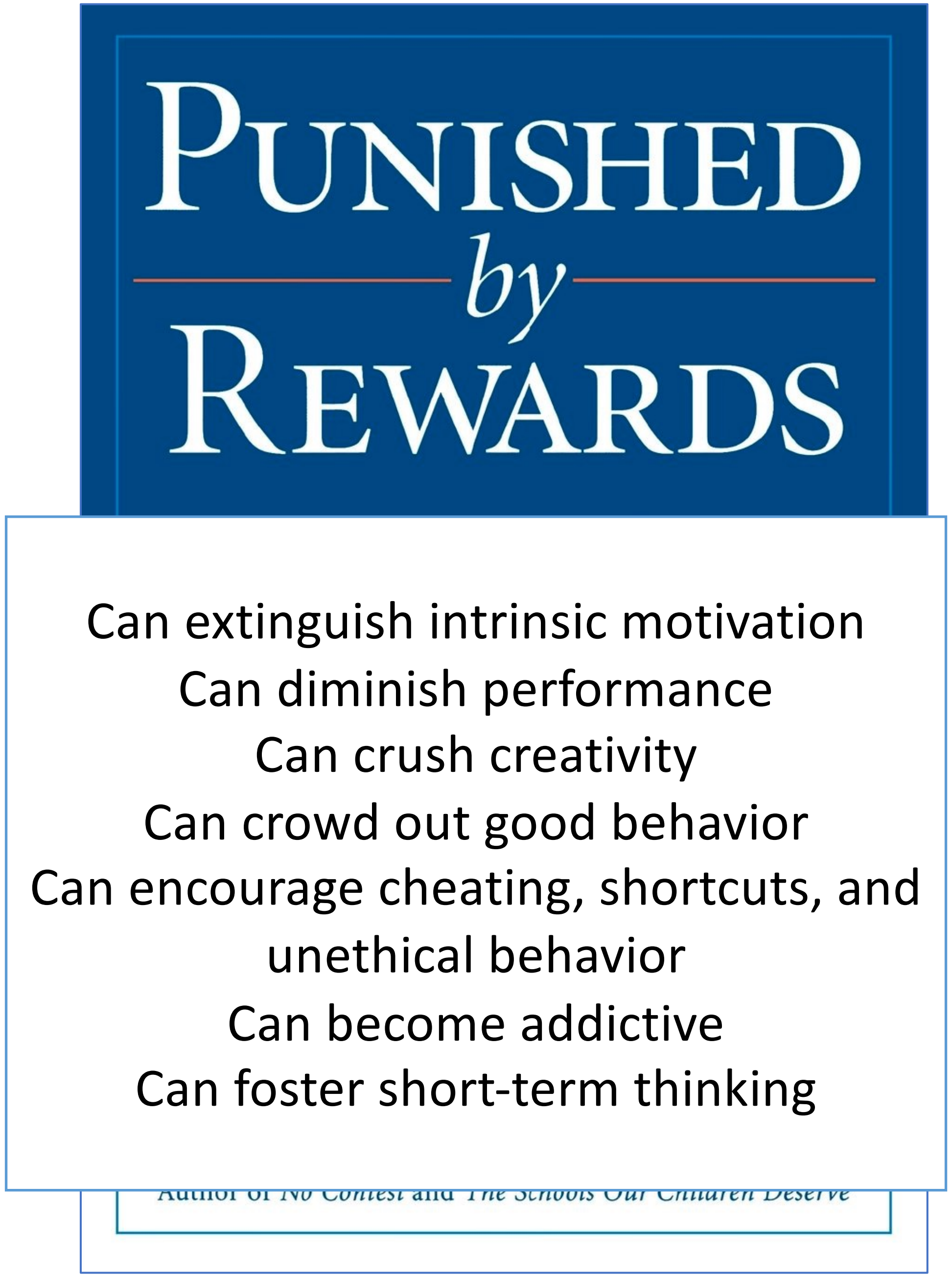


John Francis Welch Jr.

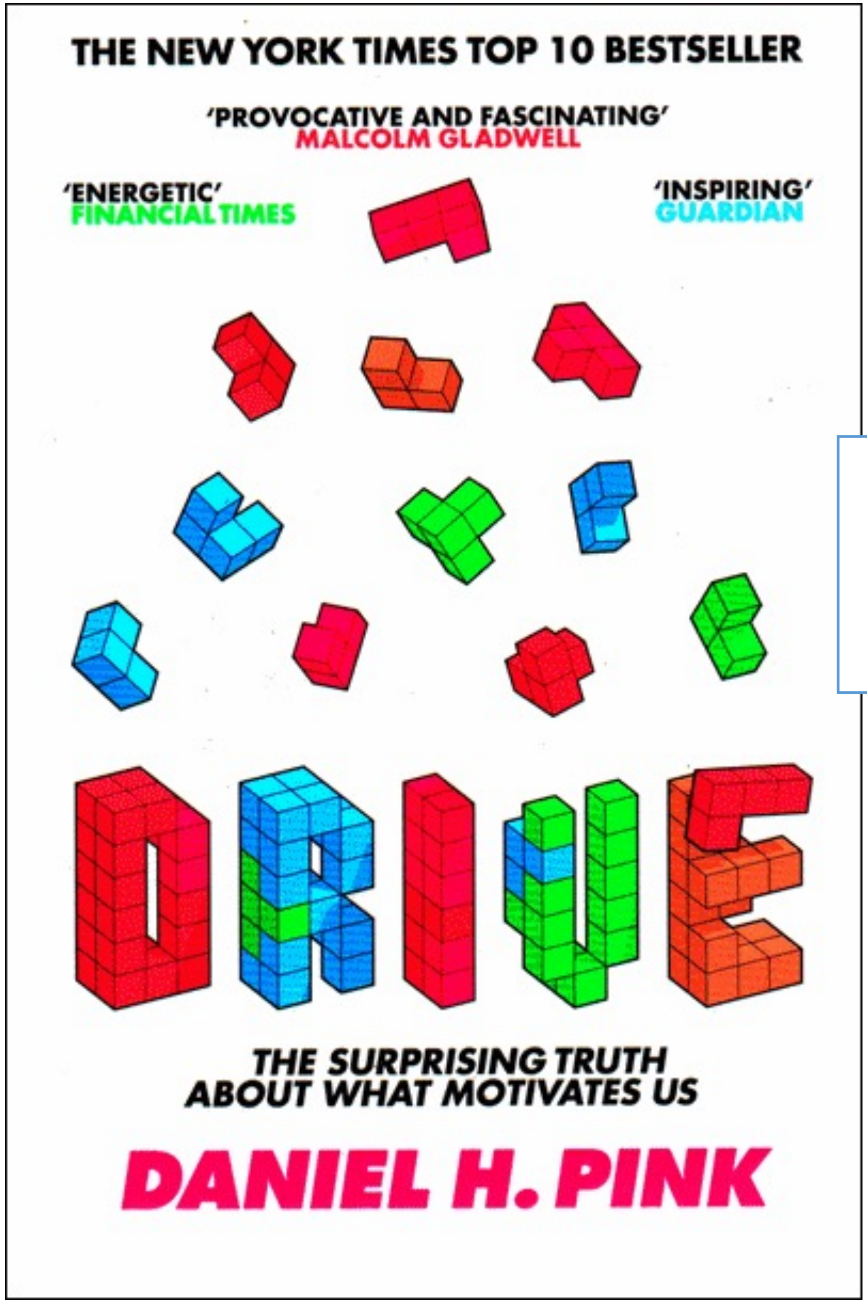
(November 19, 1935 – March 1, 2020) was an American business executive, chemical engineer, and writer. He was chairman and CEO of General Electric (GE) between 1981 and 2001.

Incentivizing Productivity

- What happens when developer bonuses are based on
  - Lines of code per day
  - Amount of documentation written
  - Low number of reported bugs in their code
  - Low number of open bugs in their code
  - High number of fixed bugs
  - Accuracy of time estimates



Can extinguish intrinsic motivation
Can diminish performance
Can crush creativity
Can crowd out good behavior
Can encourage cheating, shortcuts, and unethical behavior
Can become addictive
Can foster short-term thinking

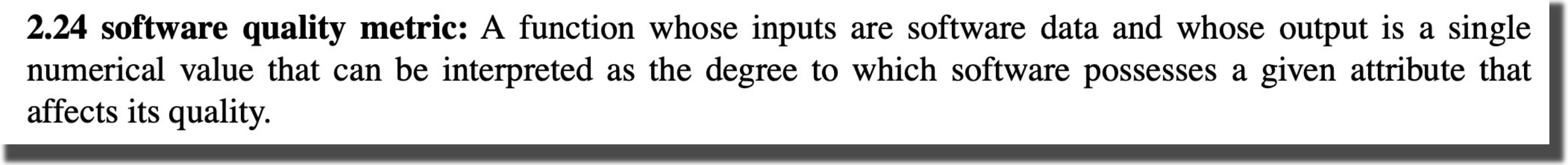

THE NEW YORK TIMES TOP 10 BESTSELLER
'PROVOCATIVE AND FASCINATING'
MALCOLM GLADWELL
'ENERGETIC'
FINANCIAL TIMES
'INSPIRING'
GUARDIAN
DRIVE
THE SURPRISING TRUTH ABOUT WHAT MOTIVATES US
DANIEL H. PINK

Autonomy
Mastery
Purpose

Software Quality Metric

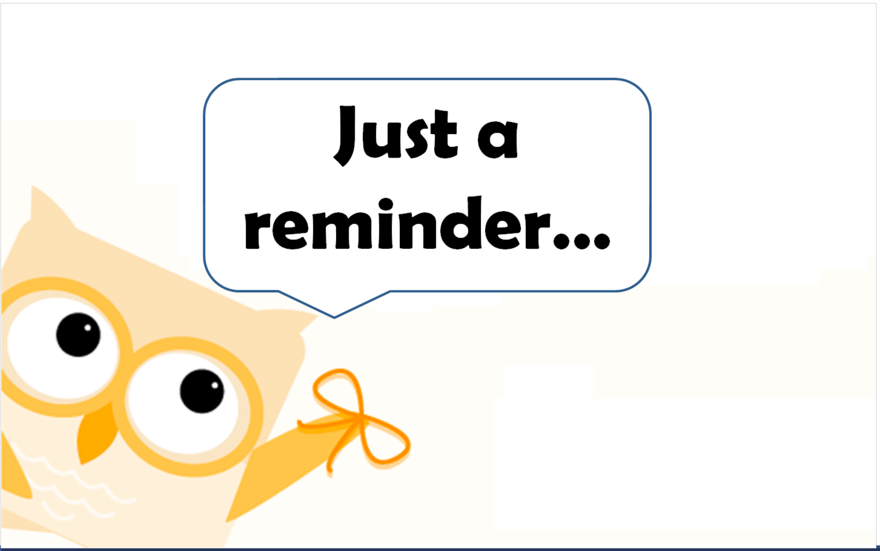
IEEE Std 1061™-1998 (R2009)
(Revision of IEEE Std 1061-1992)

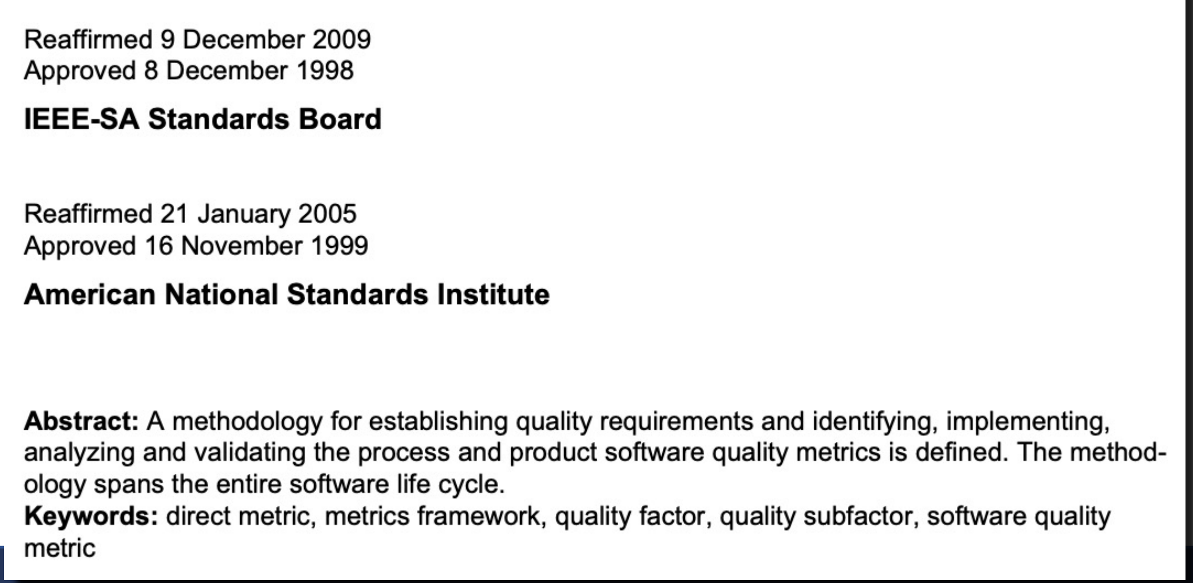
IEEE Standard for a Software Quality Metrics Methodology



2.24 software quality metric: A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software possesses a given attribute that affects its quality.

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=749159





Reaffirmed 9 December 2009
Approved 8 December 1998

IEEE-SA Standards Board

Reaffirmed 21 January 2005
Approved 16 November 1999

American National Standards Institute

Abstract: A methodology for establishing quality requirements and identifying, implementing, analyzing and validating the process and product software quality metrics is defined. The methodology spans the entire software life cycle.
Keywords: direct metric, metrics framework, quality factor, quality subfactor, software quality metric

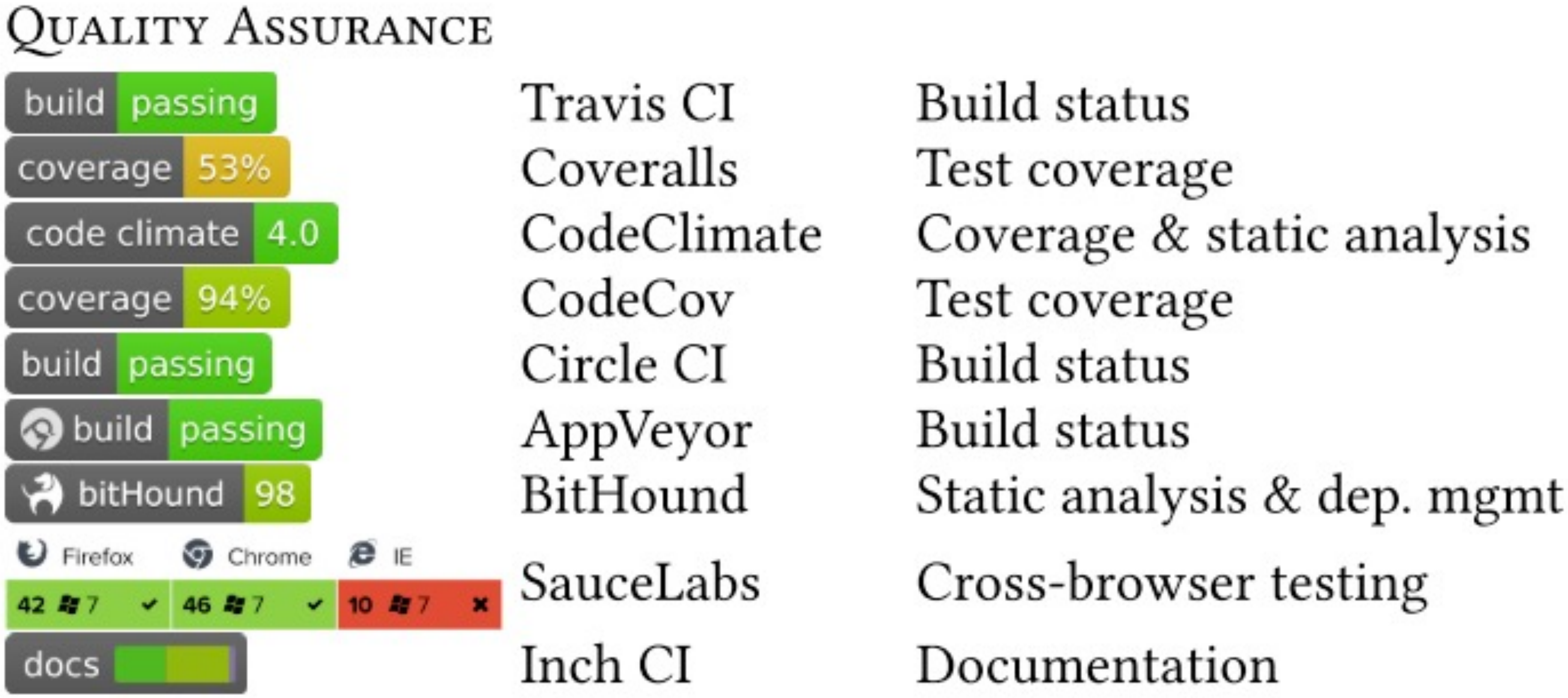
Software Quality Metrics

- IEEE 1061 definition: “A software quality metric is a function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which software processes a given attribute that affects its quality.”
- Metrics have been proposed for many quality attributes; may define own metrics

QA badges on GitHub

Shields IO

https://shields.io/



Quality Assurance

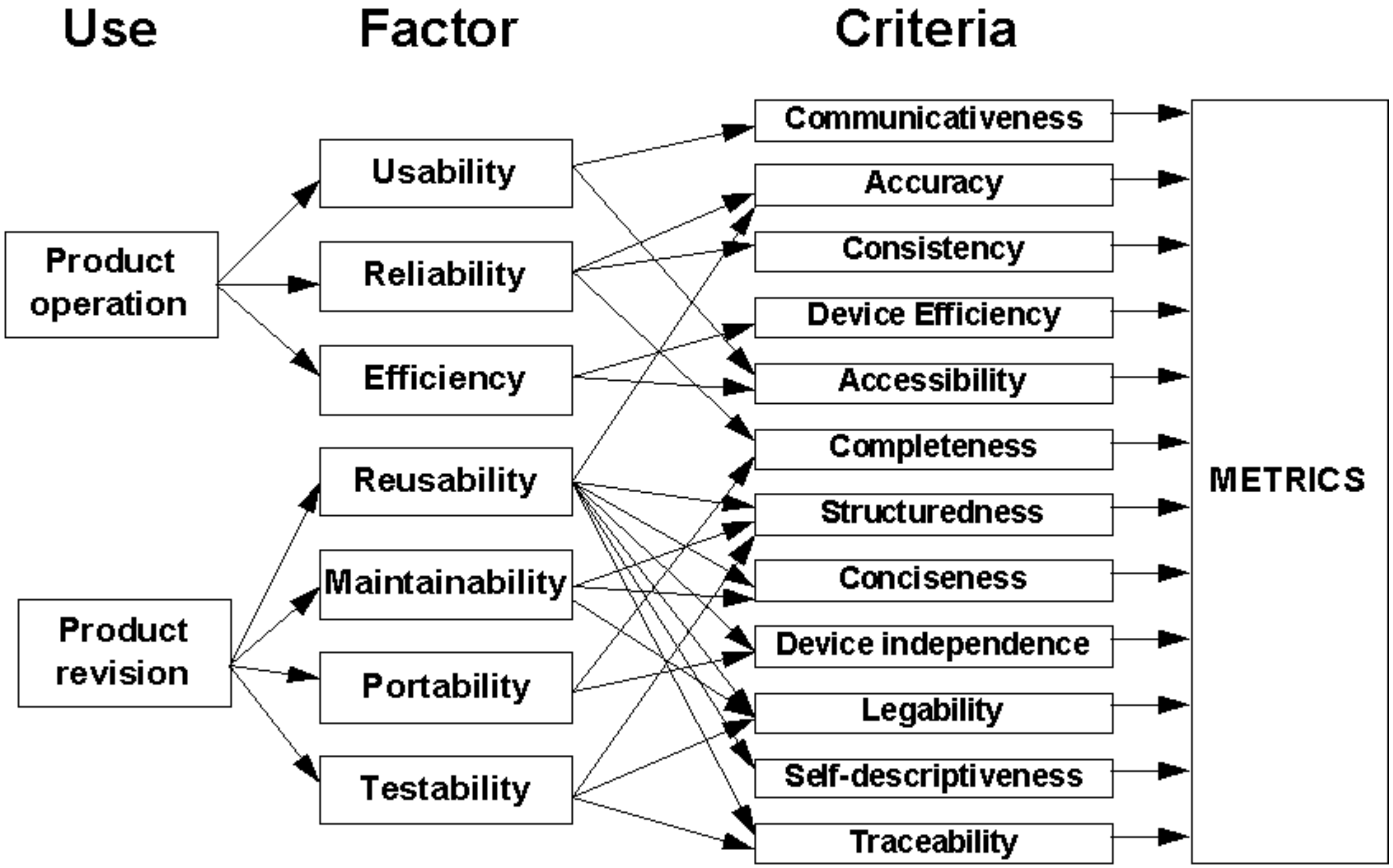
build passing	Travis CI	Build status
coverage 53%	Coveralls	Test coverage
code climate 4.0	CodeClimate	Coverage & static analysis
coverage 94%	CodeCov	Test coverage
build passing	Circle CI	Build status
build passing	AppVeyor	Build status
bitHound 98	BitHound	Static analysis & dep. mgmt
Firefox 42 7 ✓ Chrome 46 7 ✓ IE 10 7 ✗	SauceLabs	Cross-browser testing
docs	Inch CI	Documentation

Metrics of software quality, i.e., *design goals*

Functional correctness	Adherence of implementation to the specifications
Robustness	Ability to handle anomalous events
Flexibility	Ability to accommodate changes in specifications
Reusability	Ability to be reused in another application
Efficiency	Satisfaction of speed and storage requirements
Scalability	Ability to serve as the basis of a larger version of the application
Security	Level of consideration of application security

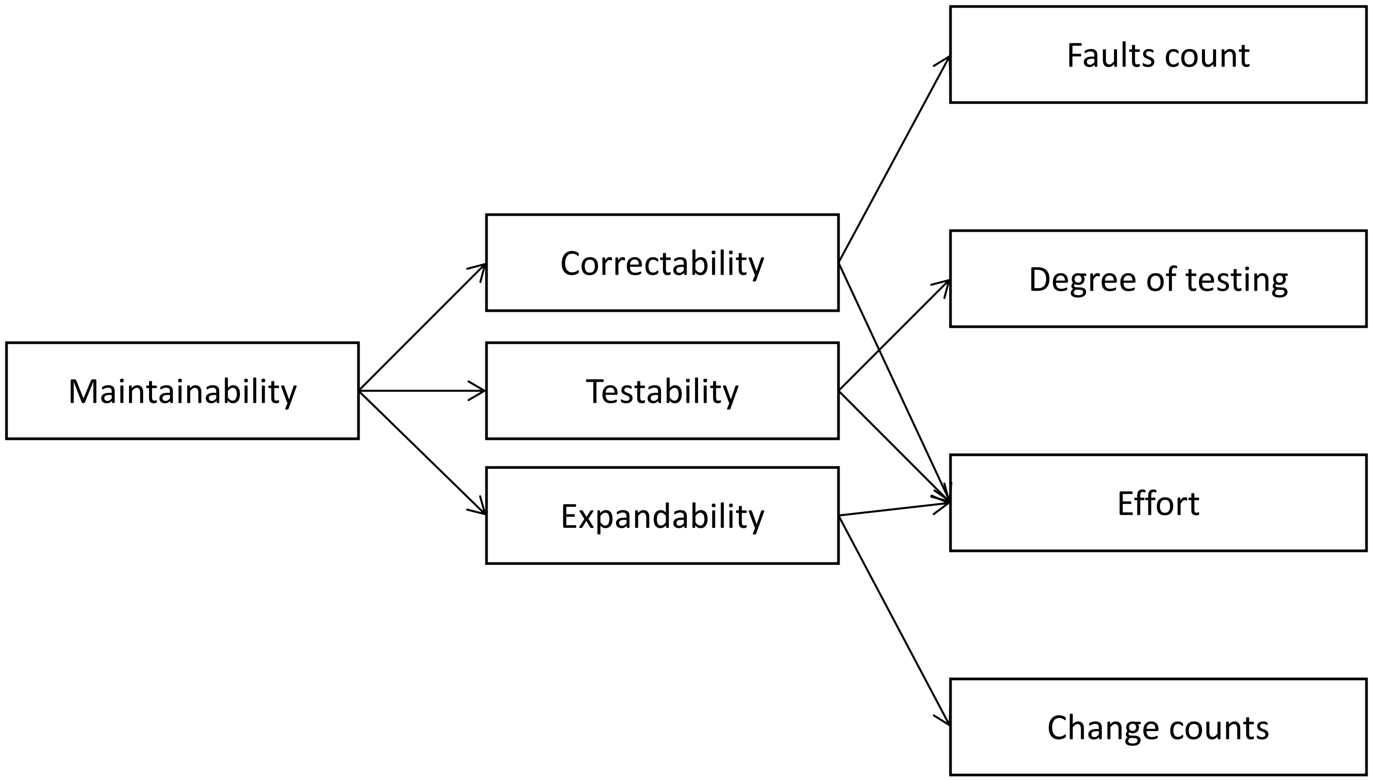
Source: Braude, Bernstein,
Software Engineering. Wiley 2011

External attributes: Measuring Quality



McCall model has 41 metrics to measure 23 quality criteria from 11 factors

Decomposition of Metrics



Closure time
Isolate/fix time
Fault rate

Statement coverage
Test plan completeness

Resource prediction
Effort expenditure

Change effort
Change size
Change rate

Object-Oriented Metrics

- Number of Methods per Class
- Depth of Inheritance Tree
- Number of Child Classes
- Coupling between Object Classes
- Calls to Methods in Unrelated Classes
- …

Other quality metrics?

- Comment density
- Test coverage
- Component balance (system breakdown optimality and component size uniformity)
- Code churn (number of lines added, removed, changed in a file)
- …

Warning

- Most software metrics are controversial
  - Usually only plausibility arguments, rarely rigorously validated
  - Cyclomatic complexity was repeatedly refuted and is still used
  - "Similar to the attempt of measuring the intelligence of a person in terms of the weight or circumference of the brain"
- Use carefully!
- Code size dominates many metrics
- Avoid claims about human factors (e.g., readability) and quality, unless validated
- Calibrate metrics in project history and other projects
- Metrics can be gamed; you get what you measure

Summary

- Measurement is difficult but important for decision making
- Software metrics are easy to measure but hard to interpret, validity often not established
- Many metrics exist, often composed; pick or design suitable metrics if needed
- Careful in use: monitoring vs incentives
- Strategies beyond metrics