ECE444: Software Engineering

Metrics and Measurement 2

Shurui Zhou

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

Administrivia

- No paper review assignment this week
- Milestone 3
  - Group report 2%
  - Individual reflection 1%
  - Peer review 2%
- Please directly send me emails instead of message on Quercus

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**.

Maintainability?

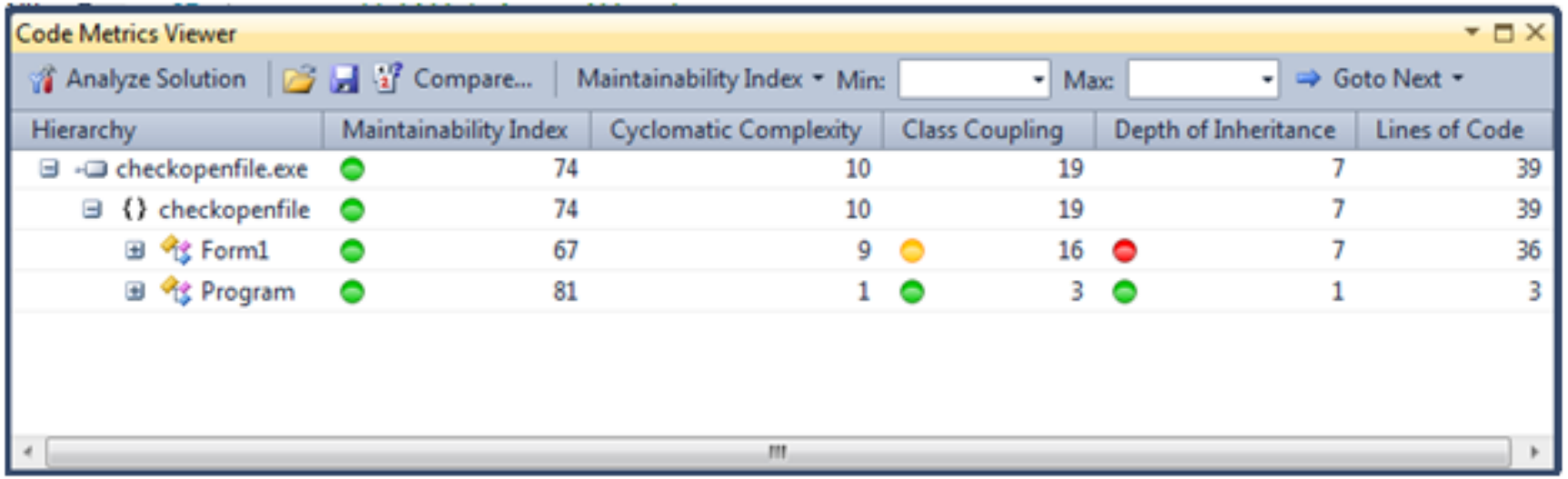
Maintainability

- 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?

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

Maintainability Index (Visual Studio since 2007)

= 171

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

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

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

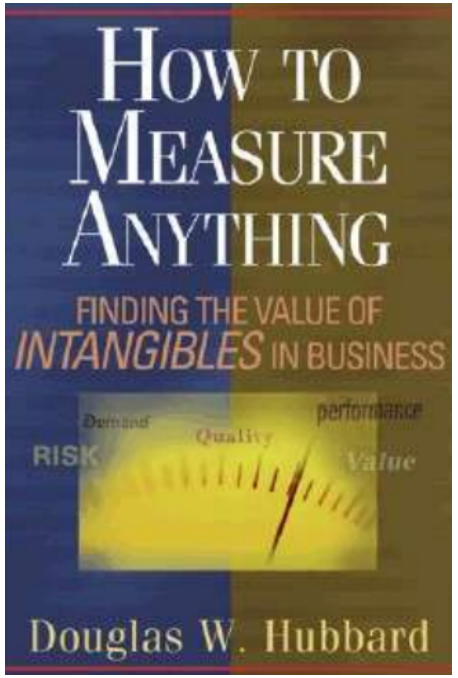
http://avandeursen.com/2014/08/29/think-twice-before-using-the-maintainability-index/

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

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.

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

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

What software qualities do we care about? (examples)

What software qualities do we care about? (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

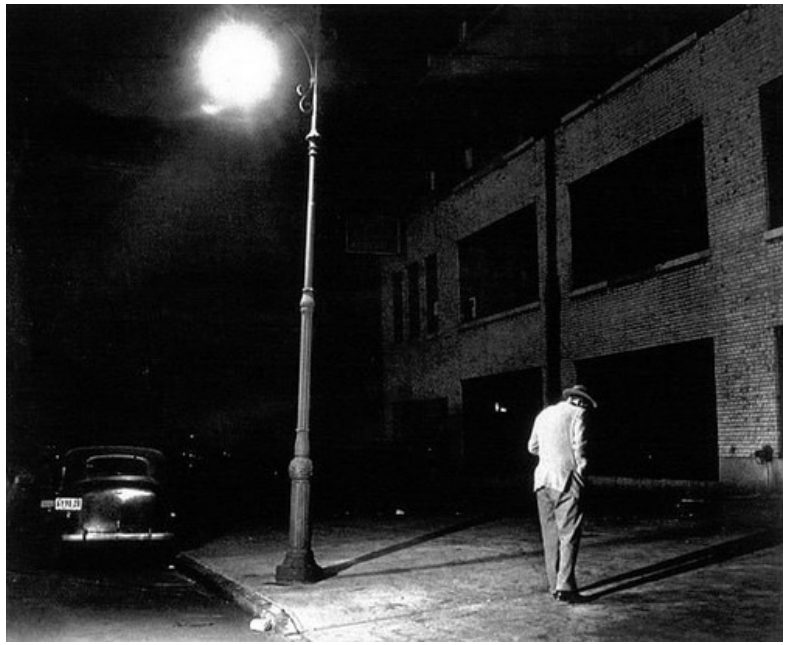
- If X is something we care about, then X, by definition, must be detectable.
  - How could we care about things like “quality,” “risk,” “security,” or “public image” if these things were totally undetectable, directly or indirectly?
  - If we have reason to care about some unknown quantity, it is because 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.

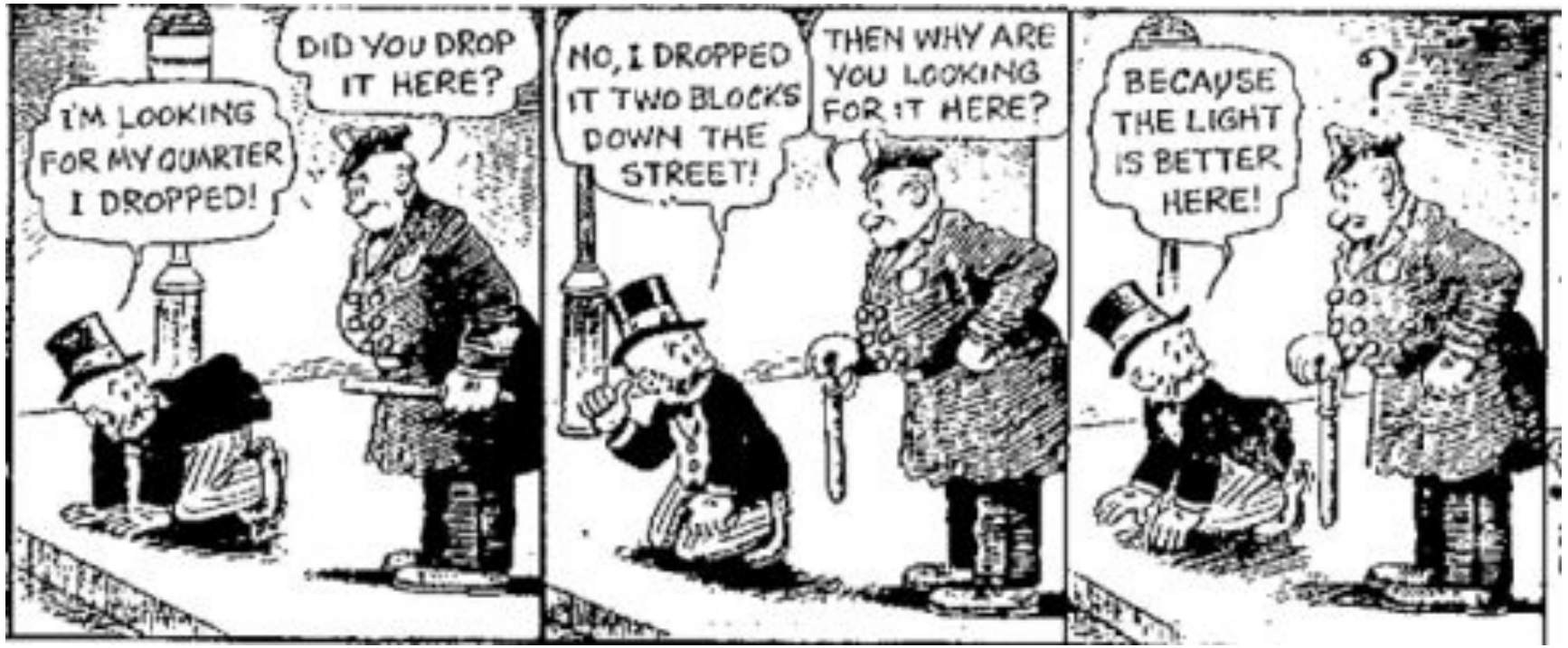
D. Hubbard, How to Measure Anything, 2010

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.

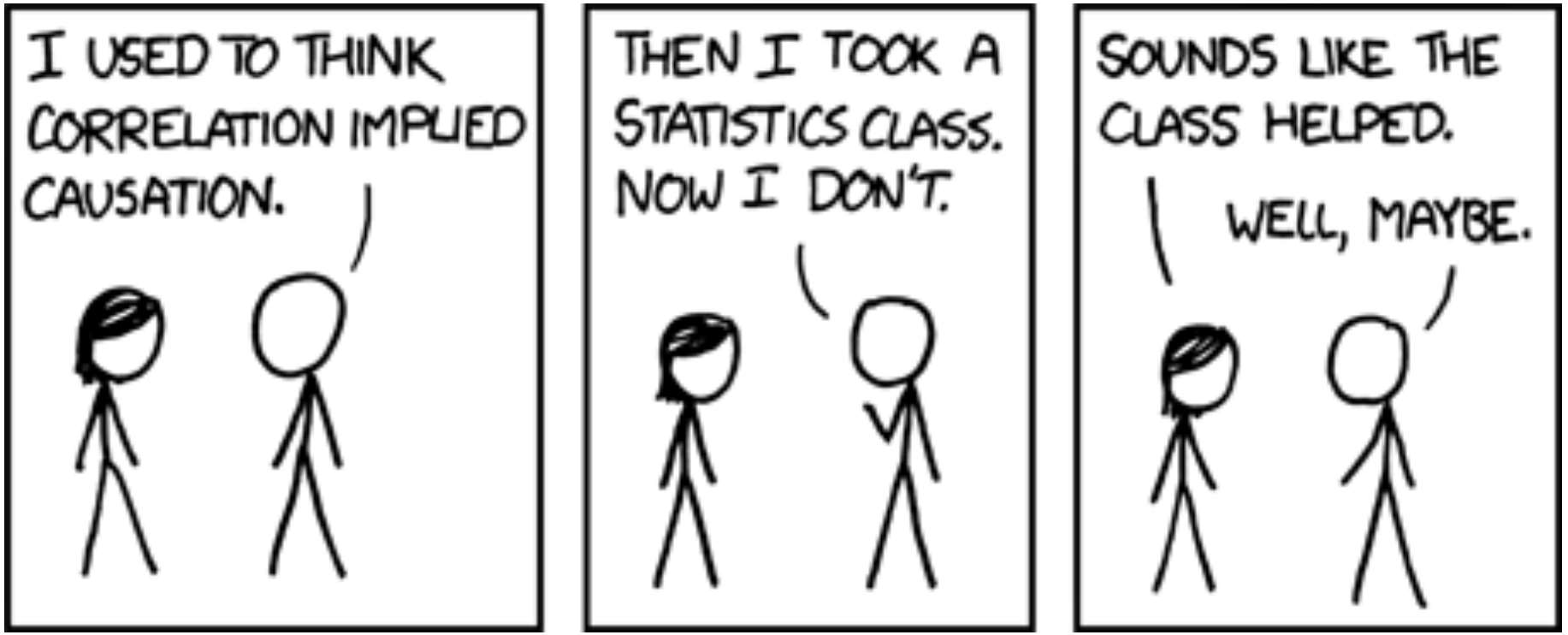
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1000457

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

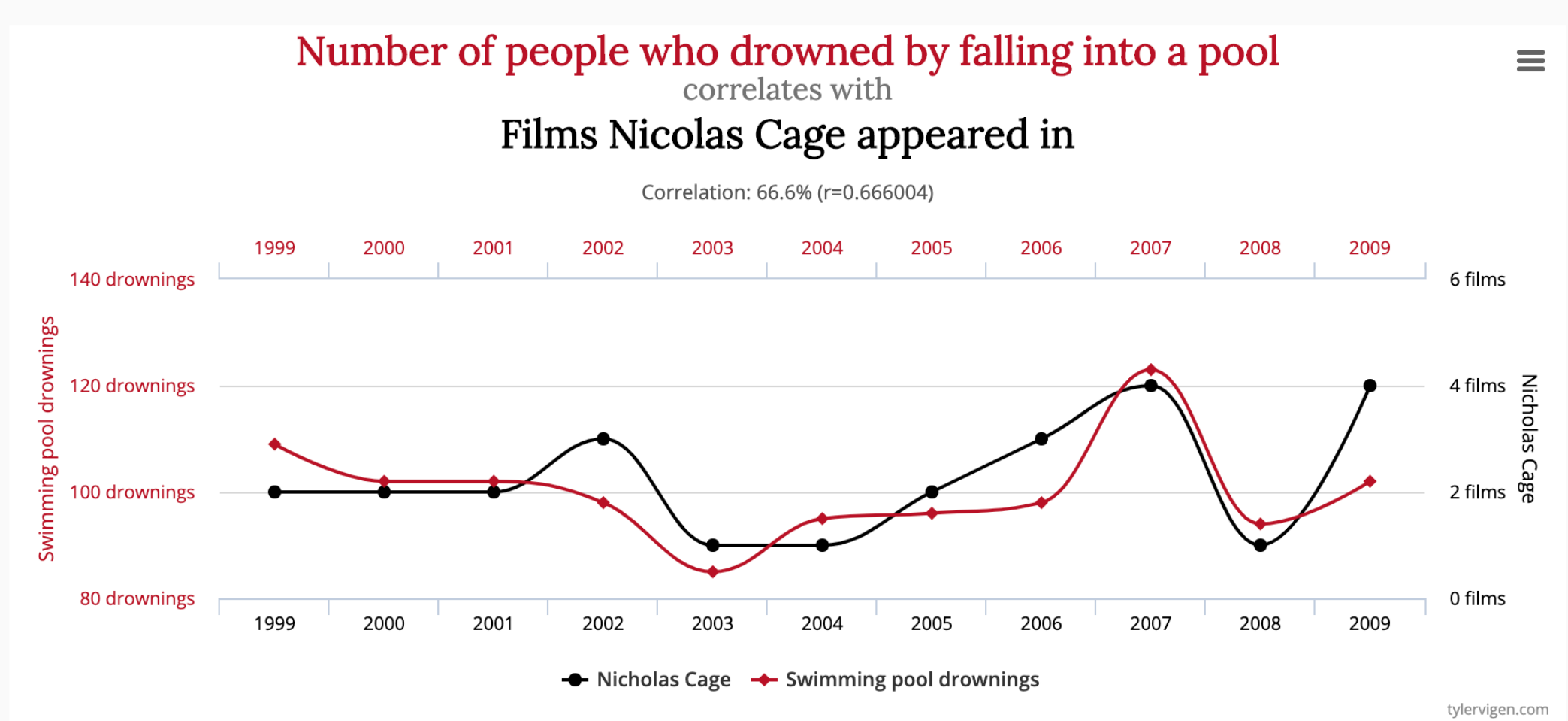
Correlation

- 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?

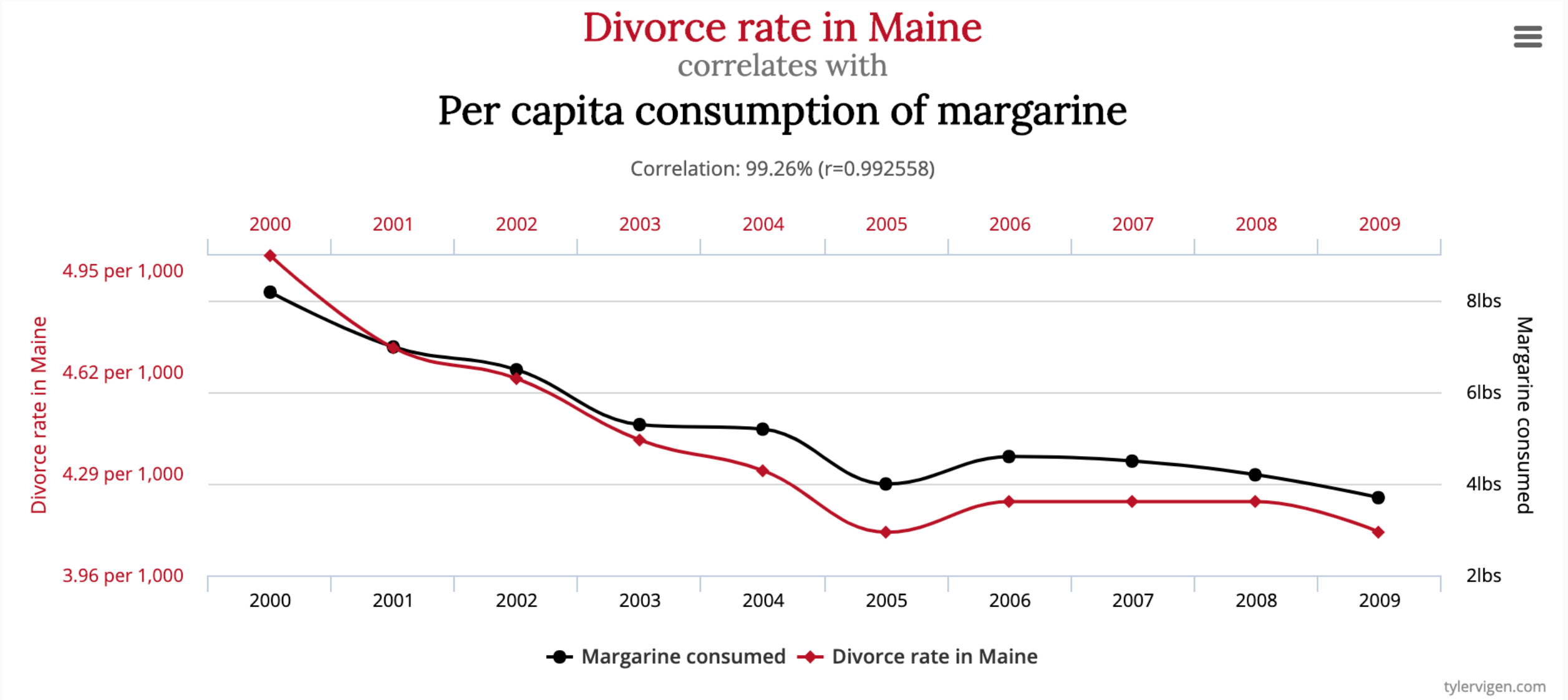


http://xkcd.com/552/

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

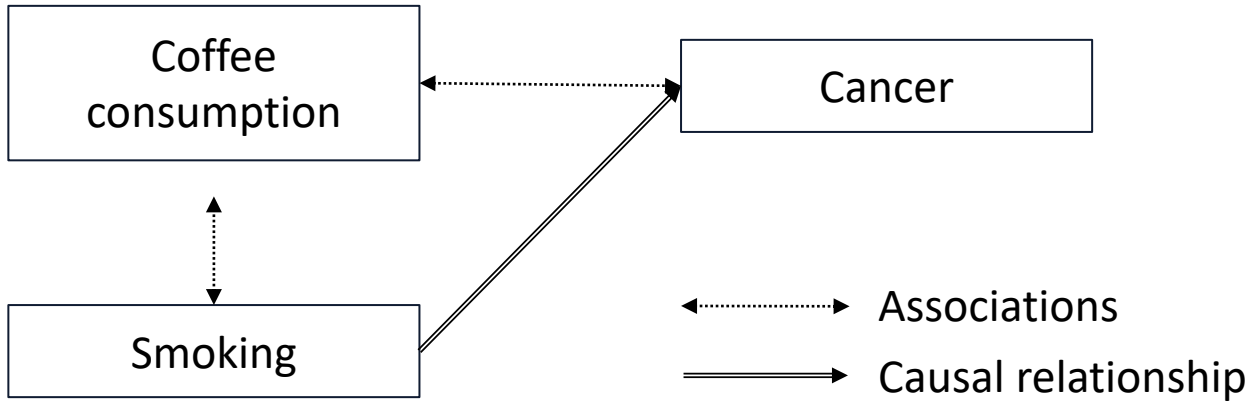


http://www.tylervigen.com/spurious-correlations



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

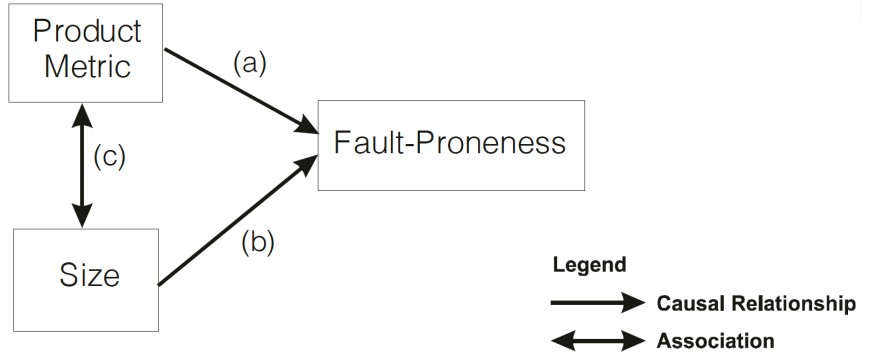
- “Only 4, out of 24 commonly used object-oriented metrics, were actually useful in predicting the quality of a software module when the effect of the module size was accounted for.”

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

The McNamara Fallacy

- There seems to be a general misunderstanding to the effect that a mathematical model cannot be undertaken until every constant and functional relationship is known to high accuracy. This often leads to the omission of admittedly highly significant factors (most of the “intangibles” influences on decisions) because these are unmeasured or unmeasurable. To omit such variables is equivalent to saying that they have zero effect... Probably the only value known to be wrong…
  - J. W. Forrester, Industrial Dynamics, The MIT Press, 1961

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

Discussion: Usability

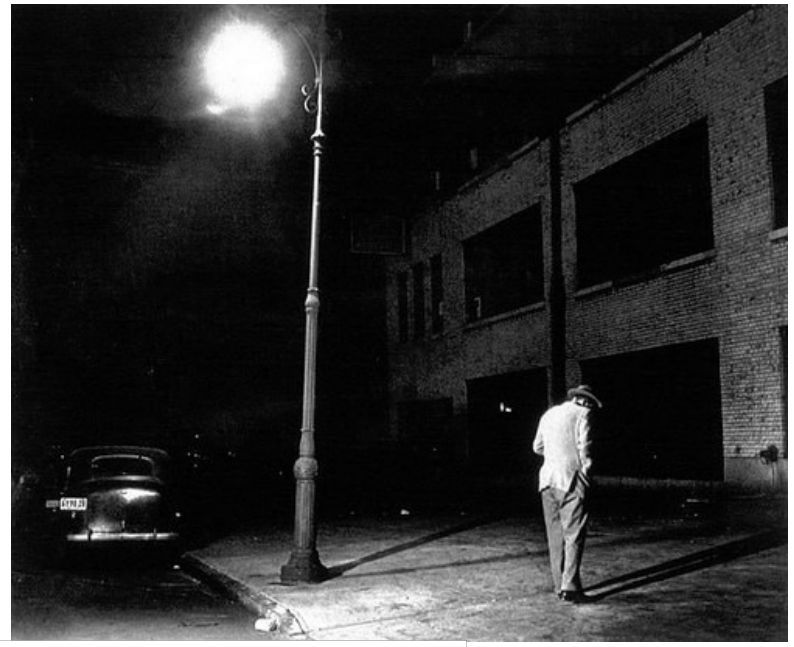
- Users can see directly how well this attribute of the system is worked out.
- One of the critical problems of usability is too much interaction or too many actions necessary to accomplish a task.
- Examples of important indicators for this attribute are:
  - List of supported devices, OS versions, 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.

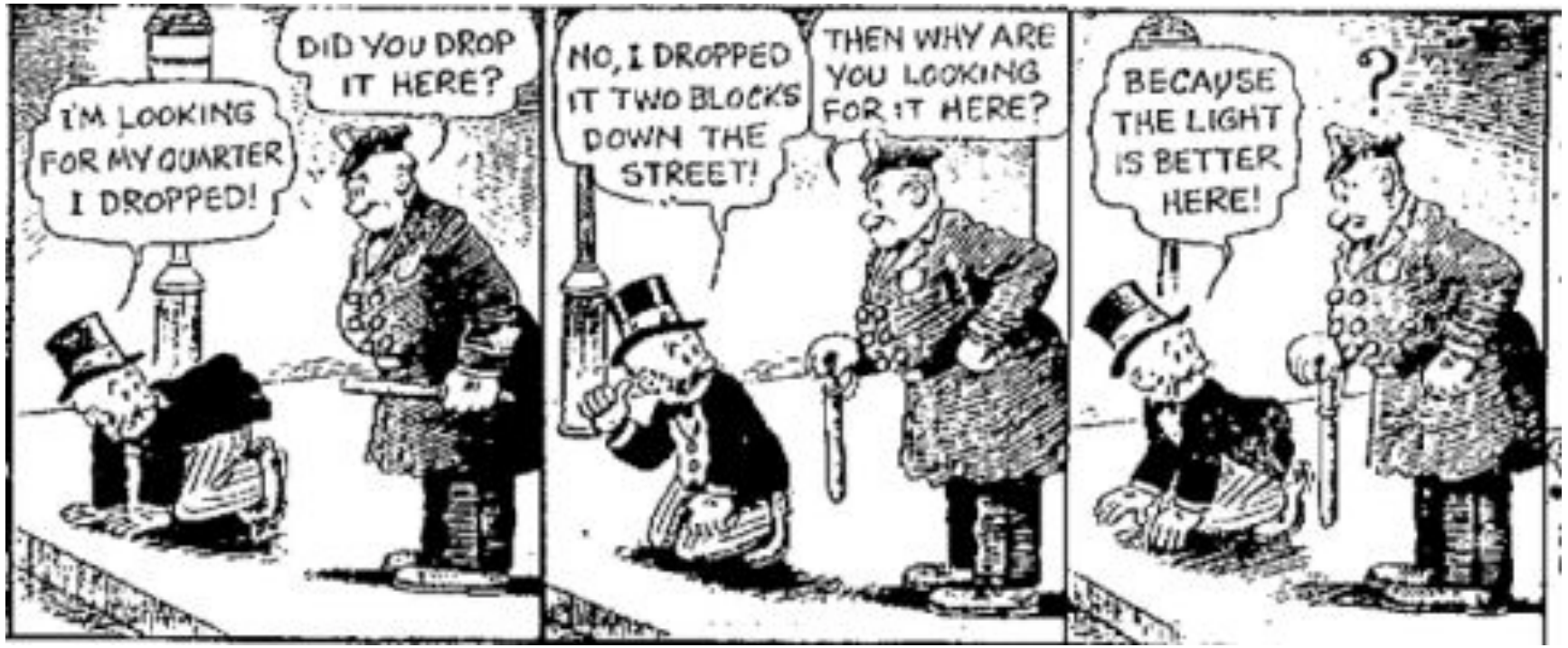
Measurement strategies

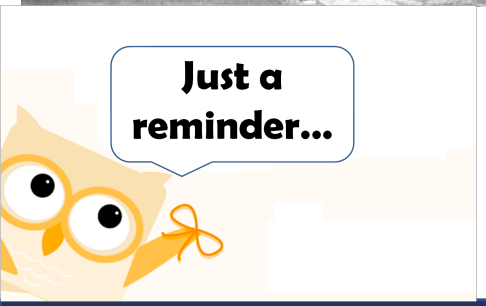
- Automated measures on code repositories
- Use or collect process data
- Instrument program (e.g., in-field crash reports)
- Surveys, interviews, controlled experiments, expert judgment
- Statistical analysis of sample

Metrics and Incentives

Goodhart's law: "When a measure becomes a target, it ceases to be a good measure."



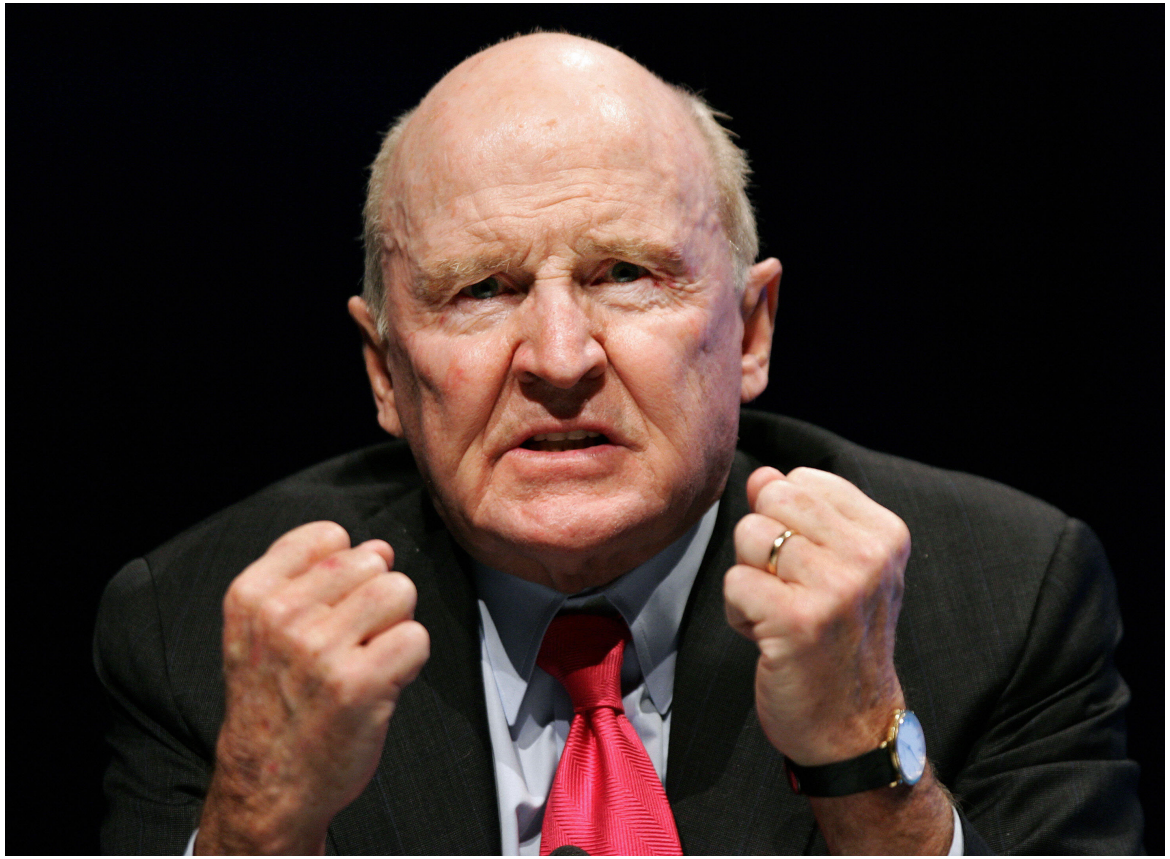

I'M LOOKING FOR MY QUARTER I DROPPED!
DID YOU DROP IT HERE?
NO, I DROPPED IT TWO BLOCKS DOWN THE STREET!
THEN WHY ARE YOU LOOKING FOR IT HERE?
BECAUSE THE LIGHT IS BETTER HERE!
?


Just a reminder...

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

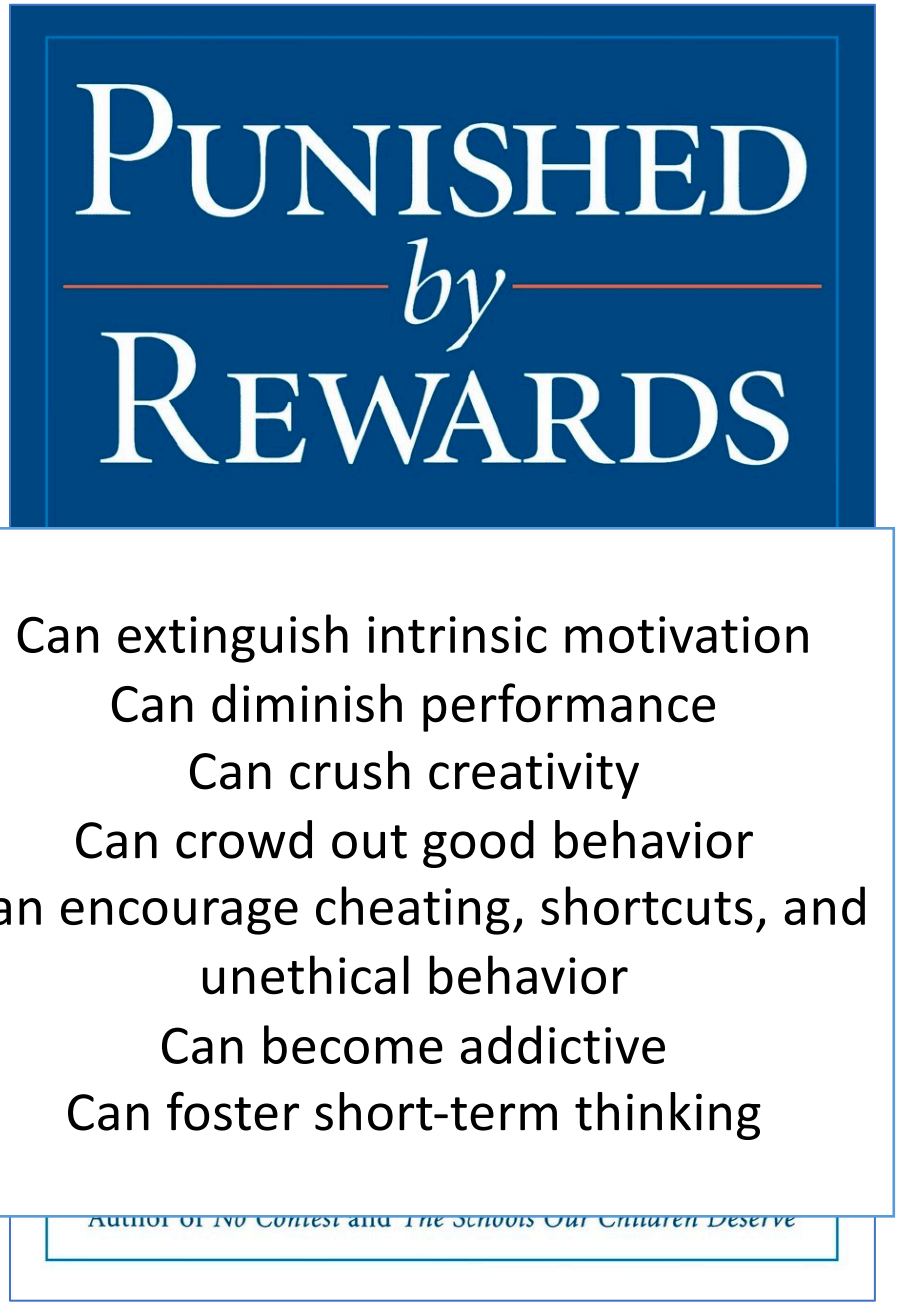


John Francis Welch Jr.

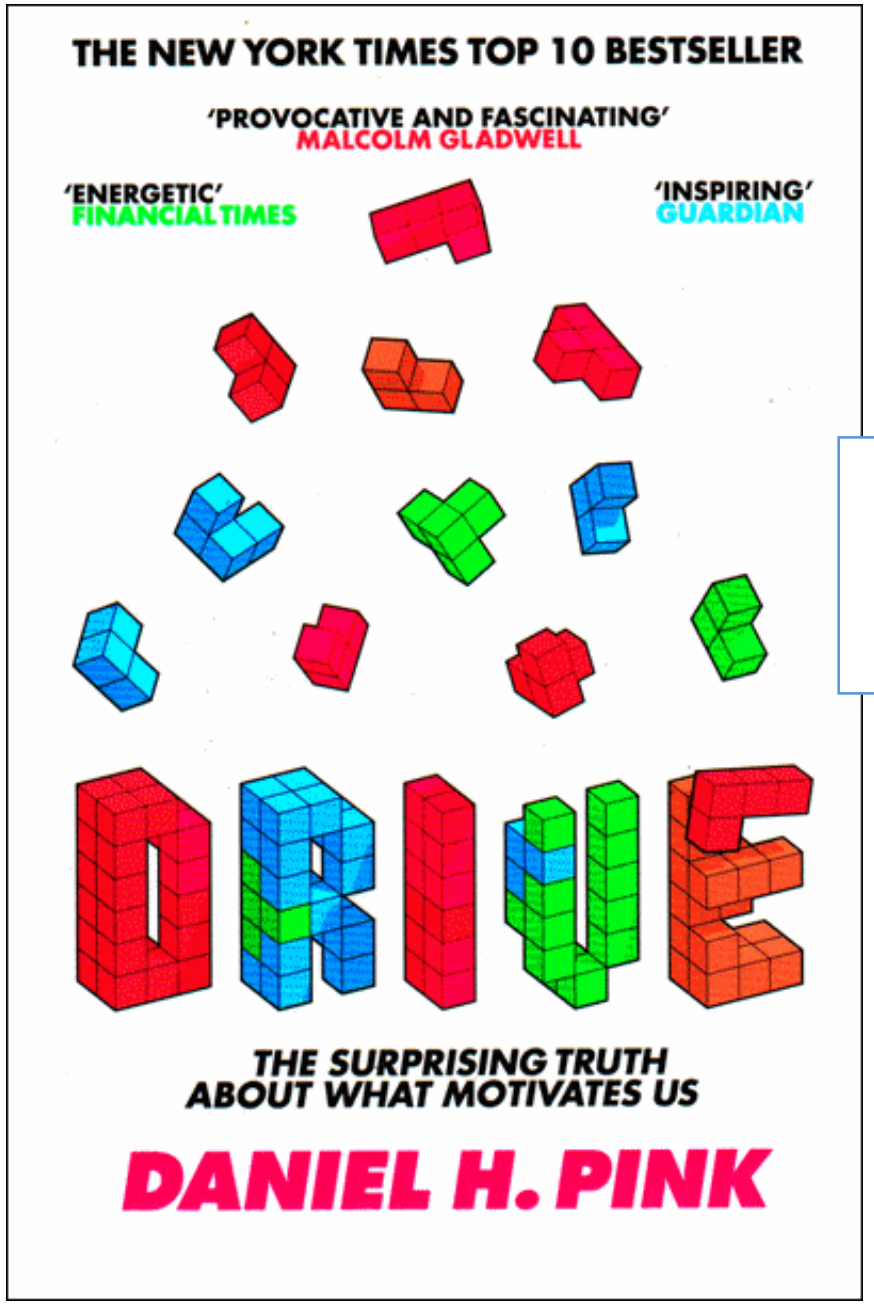
(November 19, 1935 – March 1, 2020) was an American business executive, chemical engineer, and writer. He was chairman and CEO of General Electric (GE) between 1981 and 2001.

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



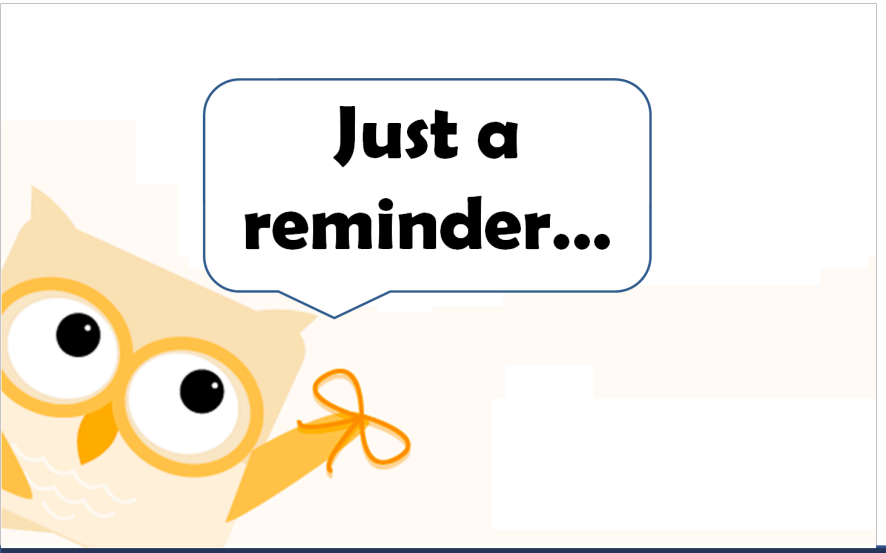
Autonomy
Mastery
Purpose

Temptation of Software Metrics

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



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

IEEE Standard for a Software Quality Metrics Methodology

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

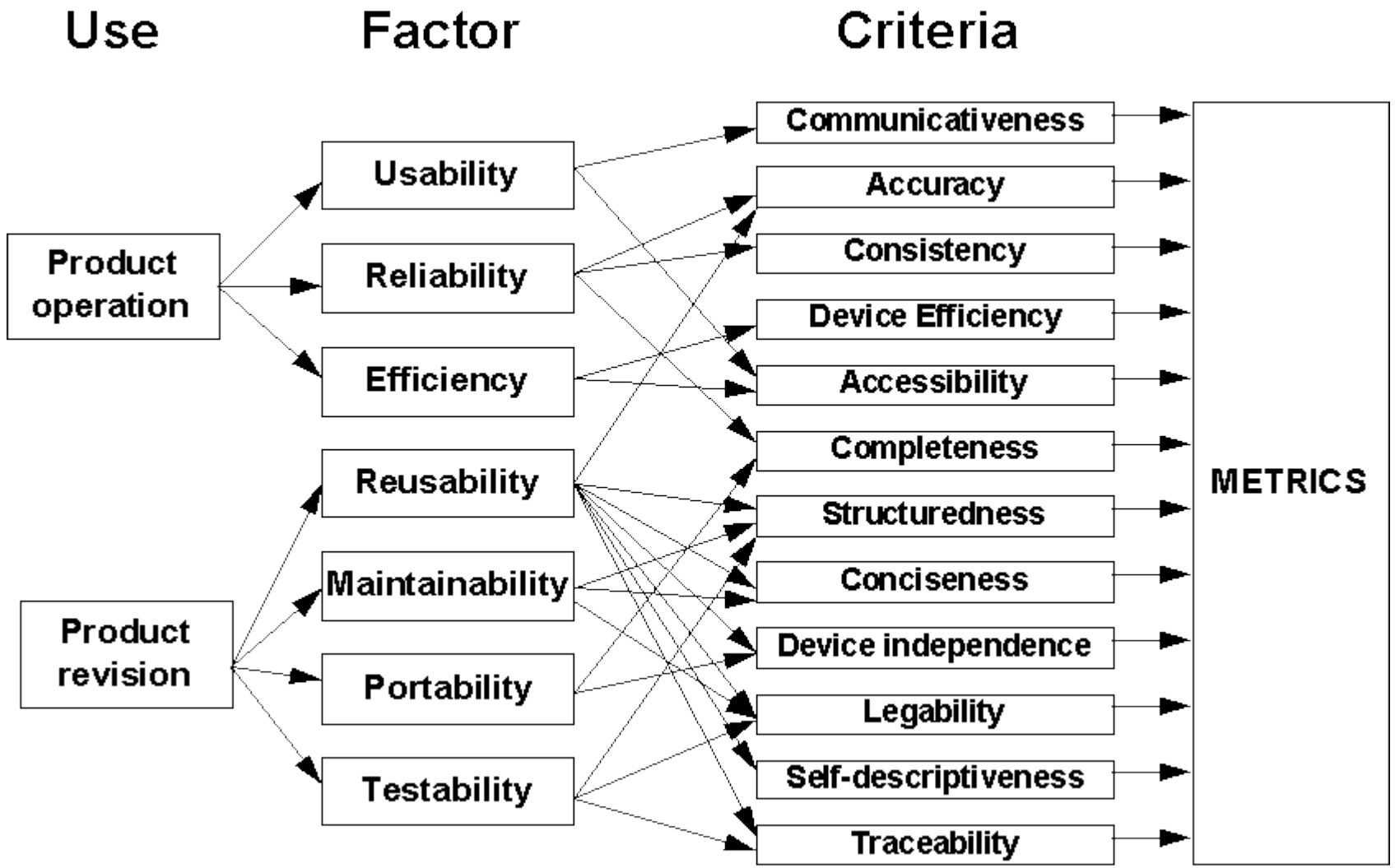
Travis CI	Build status
Coveralls	Test coverage
CodeClimate	Coverage & static analysis
CodeCov	Test coverage
Circle CI	Build status
AppVeyor	Build status
BitHound	Static analysis & dep. mgmt
SauceLabs	Cross-browser testing
Inch CI	Documentation

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

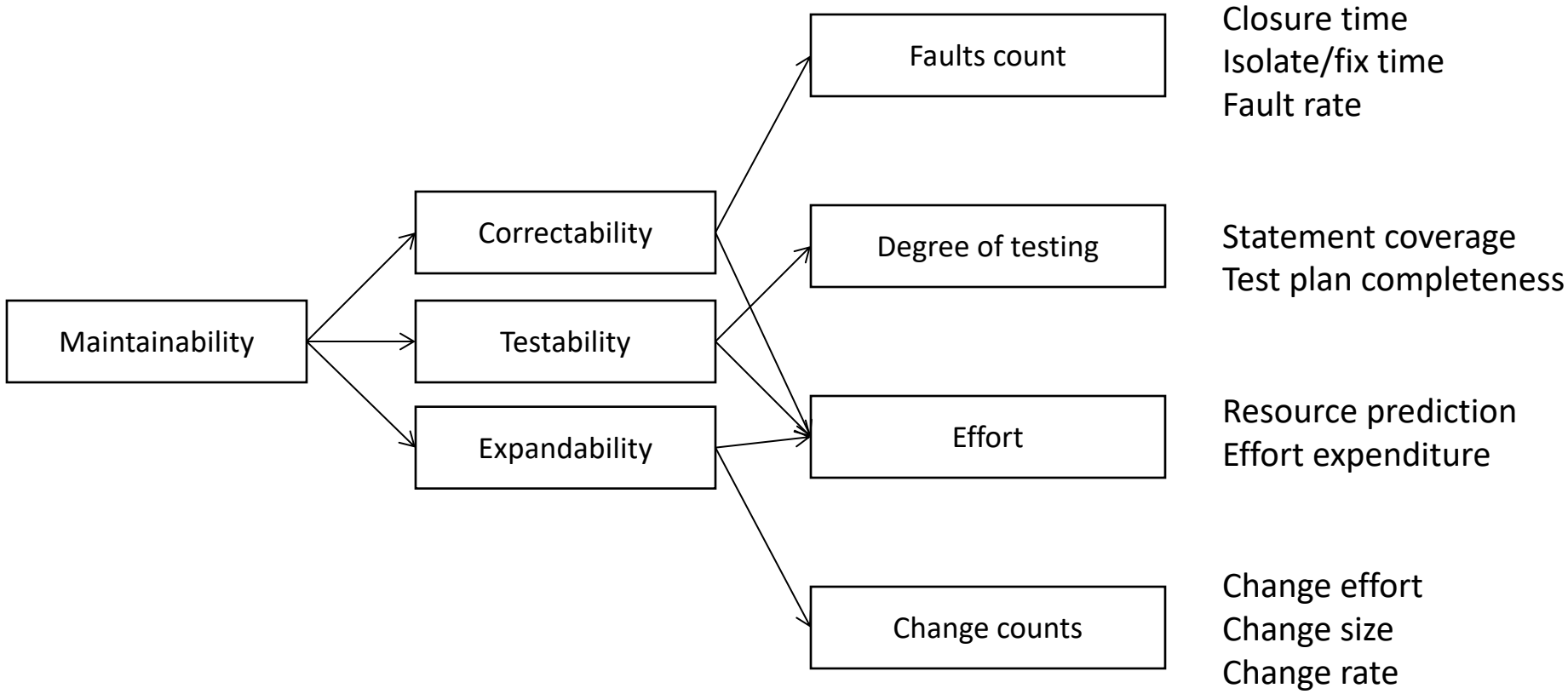
Source: Braude, Bernstein, Software Engineering. Wiley 2011

External attributes: Measuring Quality



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

Decomposition of Metrics



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

(Some) strategies

- Metrics tracked using tools and processes (process metrics like time, or code metrics like defects in a bug database).
- Expert assessment or human-subject experiments (controlled experiments, talk-aloud protocols).
- Mining software repositories, defect databases, especially for trend analysis or defect prediction.
  - Some success e.g., as reported by Microsoft Research
- Benchmarking (especially for performance).

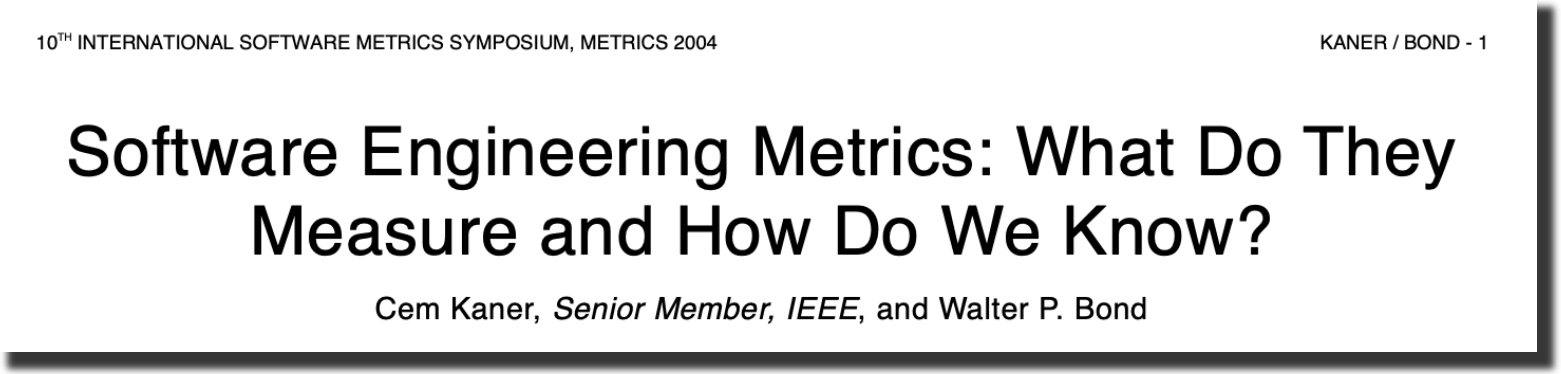
Factors in a successful measurement program

- Set solid measurement objectives and plans.
- Make measurement part of the process.
- Gain a thorough understanding of measurement.
- Focus on cultural issues.
- Create a safe environment to collect and report true data.
- Cultivate a predisposition to change.
- Develop a complementary suite of measures.

Carol A. Dekkers and Patricia A. McQuaid,
"The Dangers of Using Software Metrics to (Mis)Manage", 2002.

Kaner's questions when choosing a metric

1. What is the purpose of this measure?
2. What is the scope of this measure?
3. What attribute are you trying to measure?
4. What is the attribute's natural scale?
5. What is the attribute's natural variability?
6. What instrument are you using to measure the attribute, and what reading do you take from the instrument?
7. What is the instrument's natural scale?
8. What is the reading's natural variability (normally called measurement error)?
9. What is the attribute's relationship to the instrument?
10. What are the natural and foreseeable side effects of using this instrument?


10TH INTERNATIONAL SOFTWARE METRICS SYMPOSIUM, METRICS 2004

KANER / BOND - 1

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

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

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

Further Reading on Metrics

- Sommerville. Software Engineering. Edition 7/8, Sections 26.1, 27.5, and 28.3
- Hubbard. How to measure anything: Finding the value of intangibles in business. John Wiley & Sons, 2014. Chapter 3
- Kaner and Bond. Software Engineering Metrics: What Do They Measure and How Do We Know? METRICS 2004
- Fenton and Pfleeger. Software Metrics: A rigorous & practical approach. Thomson Publishing 1997