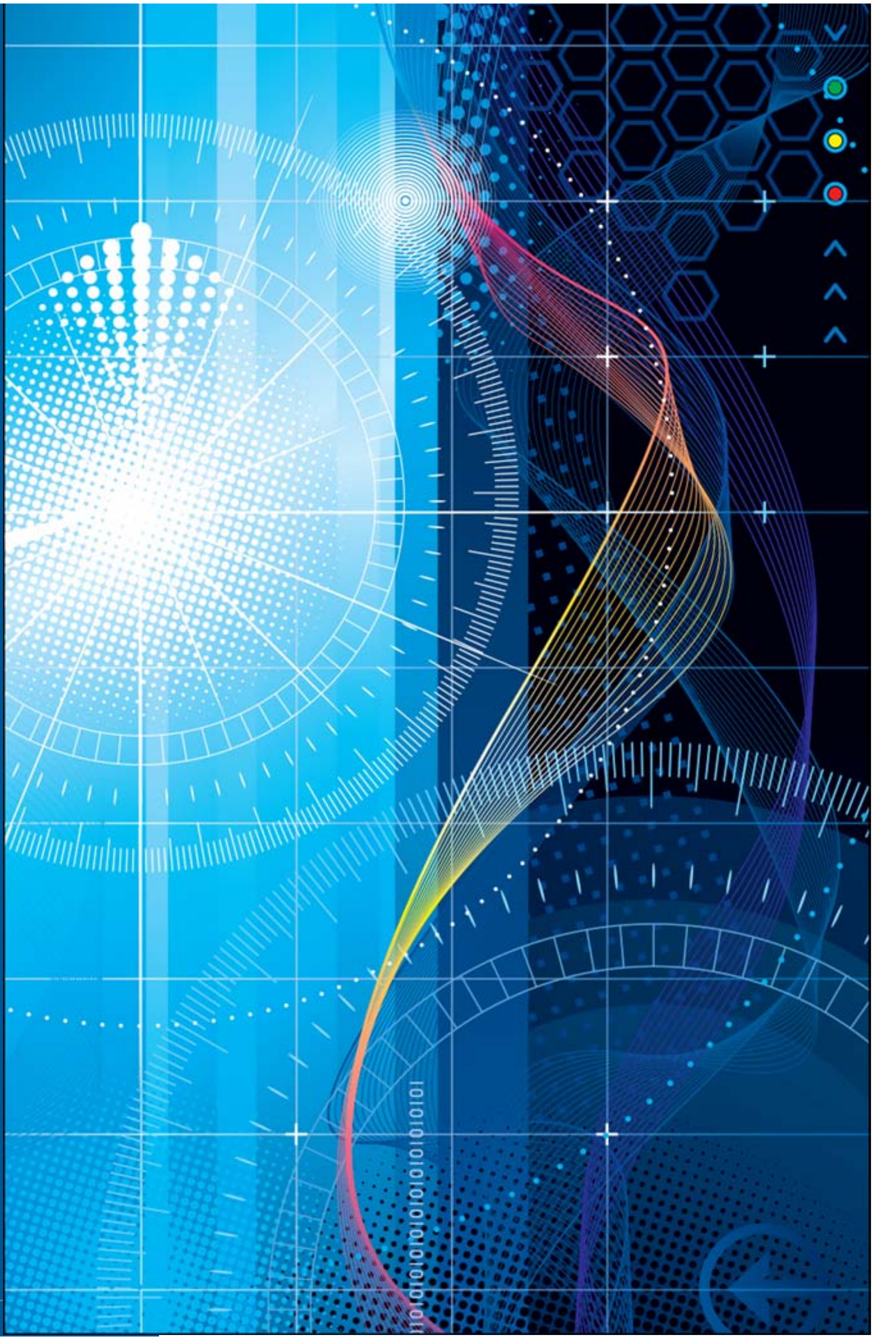
Measuring Agile Effectiveness: Why this was harder than we thought

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

William Nichols, Sept 2017



Software Engineering Institute | Carnegie Mellon University

Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

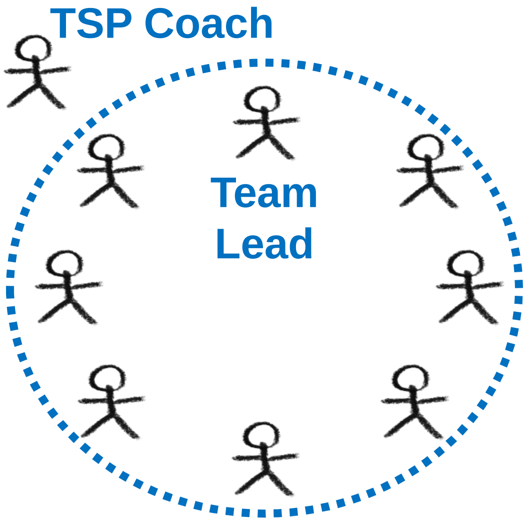
The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM17-0609

Who are we?

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA
A non-profit FFRDC federally funded research and deveolopment corporation



Team Software Process
Combines team communication, quality practices, measurement, planning, tracking, training, and coaching to build both Products and Teams.

http://war-on-ice.com/

A young statistician who played for me was hired as the analytics consultant for a professional hockey team.

Jobs like that didn’t exist when I was his age.

Sports Analytics was invented, more or less

Bill James was a night watchman at a pork and beans factory who liked to pass his time pouring over baseball statistics. He was part of a movement that dragged the statistical analysis of baseball out of the 1880s and into the 20th century.

His greatest skill was not statistical

Analysis, but his skepticism and

Insight.

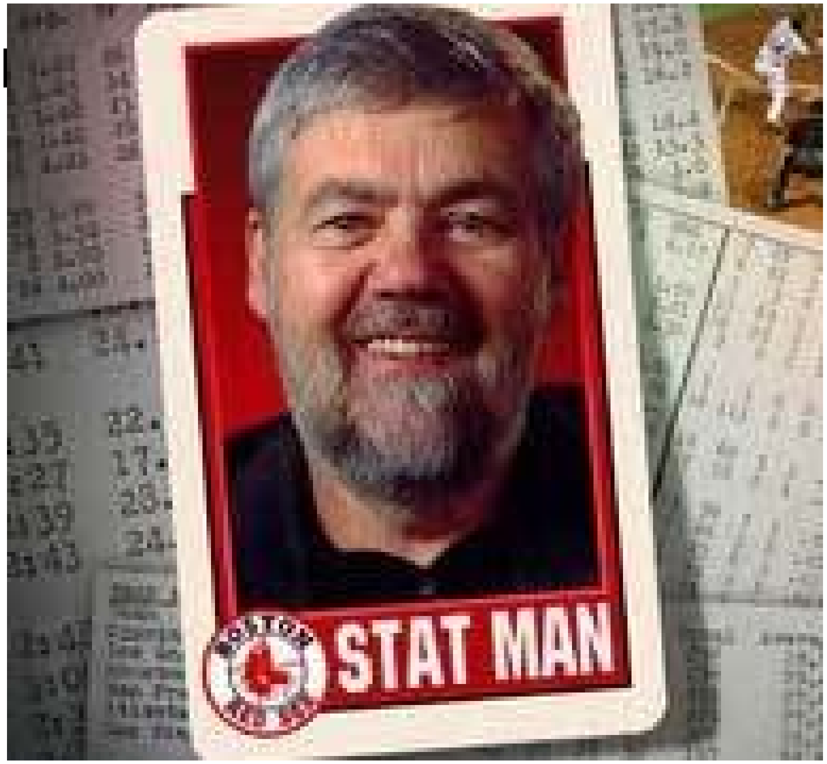
Analytics didn’t go mainstream

at the professional level until the

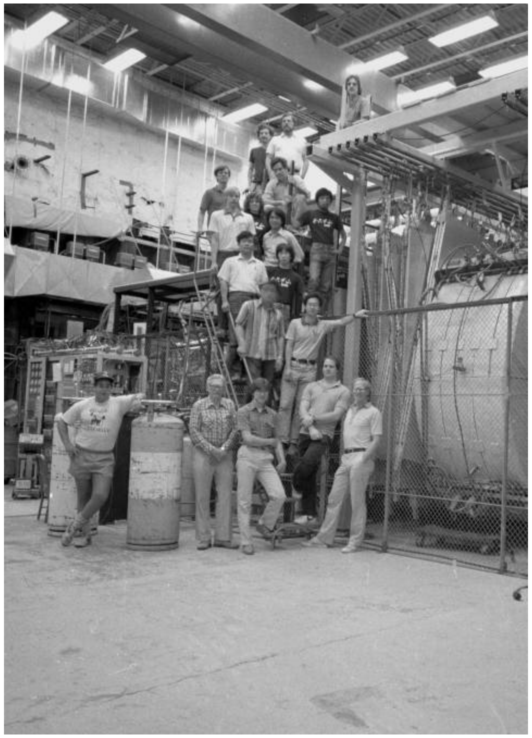
21st century!

He now works for the

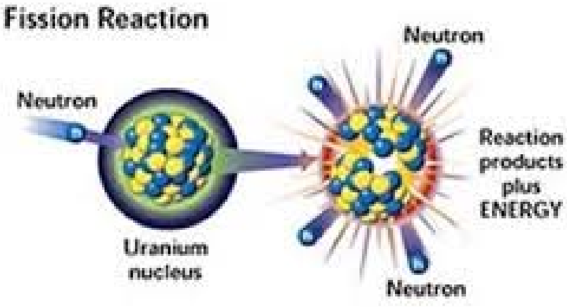
Boston Red Sox.



Who am I?



Project X
DUSEL
Main Injector


Fission Reaction
Neutron
Neutron
Reaction products plus ENERGY
Uranium nucleus
Neutron

E=mc²

Some Guiding Philosophy

"True ideas are those that we can assimilate, validate, corroborate, and verify. False ideas are those we cannot." - William James

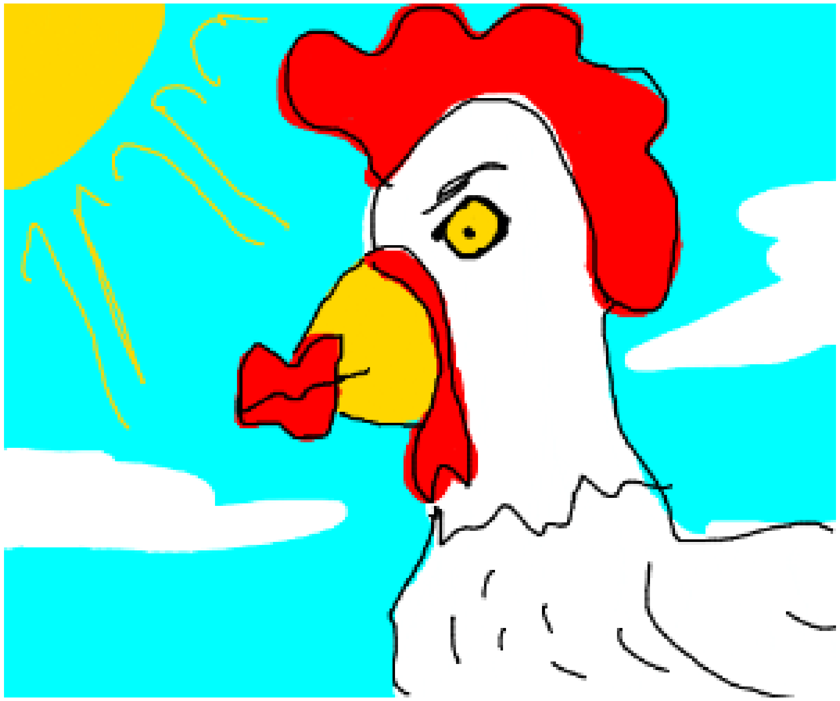
"*If you care to think about it at all you have to realize, as soon as you acquire a taste for independent thought, that a great portion of the sport's traditional knowledge is ridiculous hokum."* - Bill James

"I'm skeptical of any conclusion until it's proven to be so. I have a very healthy distrust of other people's opinions." - Dr. Rany Jazayerli

"Science is the organized skepticism in the reliability of expert opinion." – Richard Feynman

Philosophers worry about unrealistic problems, such as…

What if chickens had lips?

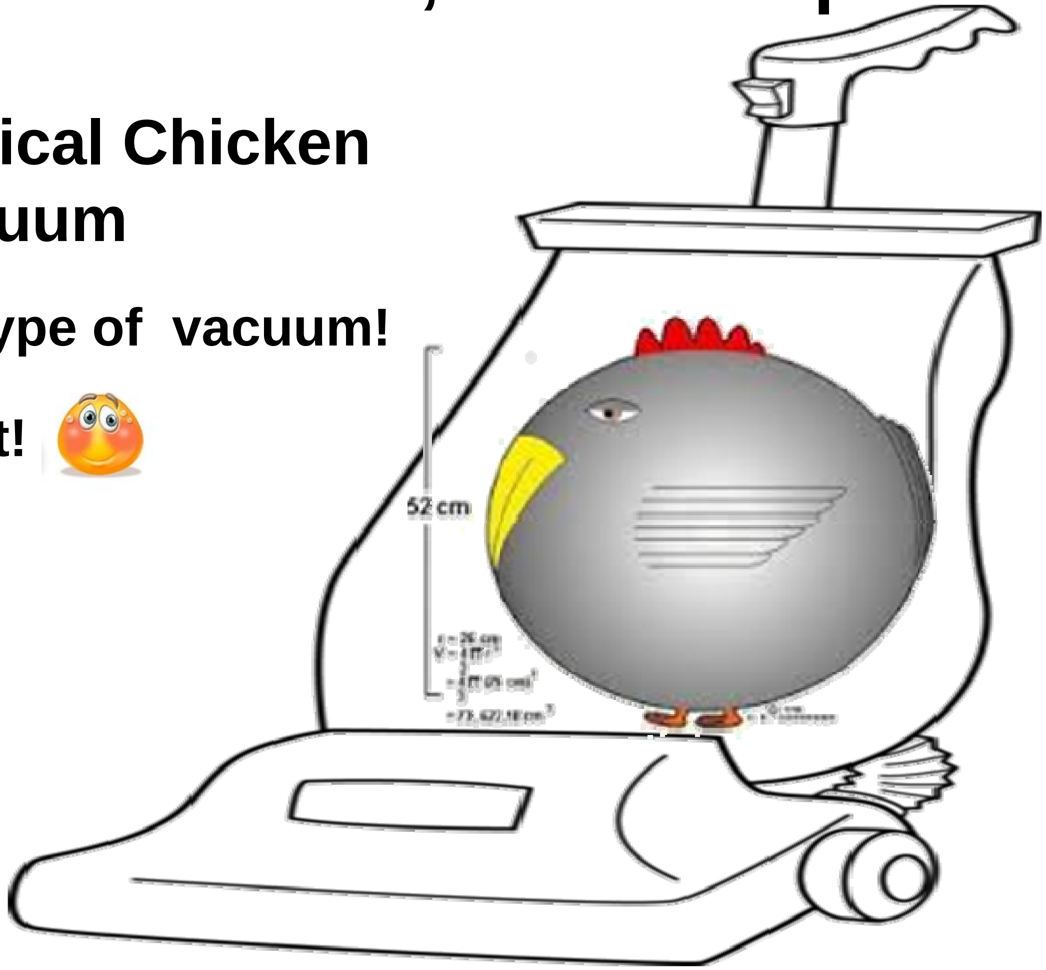


Physicists solve real problems, ignoring unimportant detail, for examaple

A Spherical Chicken in a Vacuum

Wrong type of vacuum!

Context!

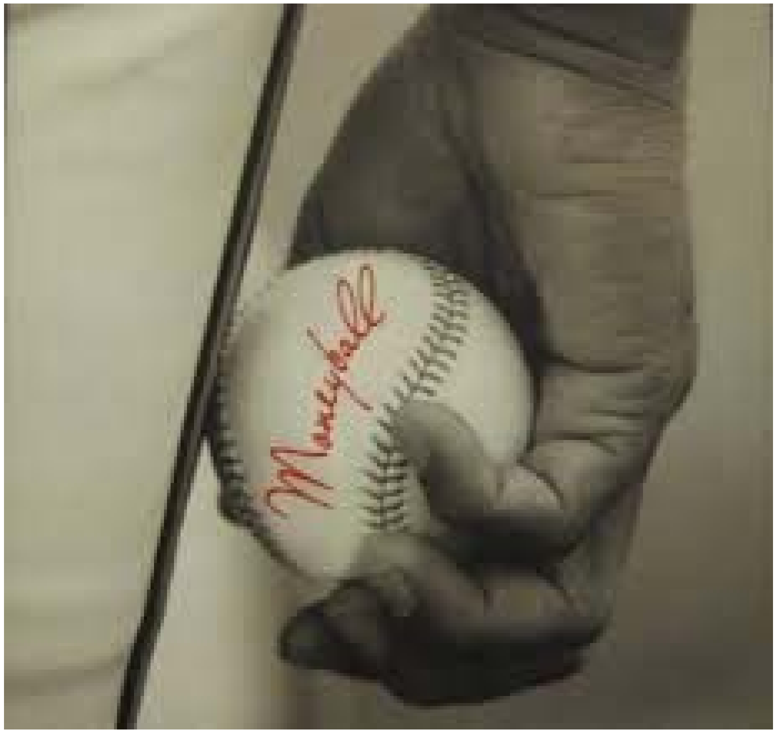


Engineers use the chicken to do useful things

Software engineers automate the work

Use data to find the value

Measuring Agile Efficacy



Where to start with a study of Agile?
The state of the practice

The Problem

The results of applying many software development methods are unpredictable.

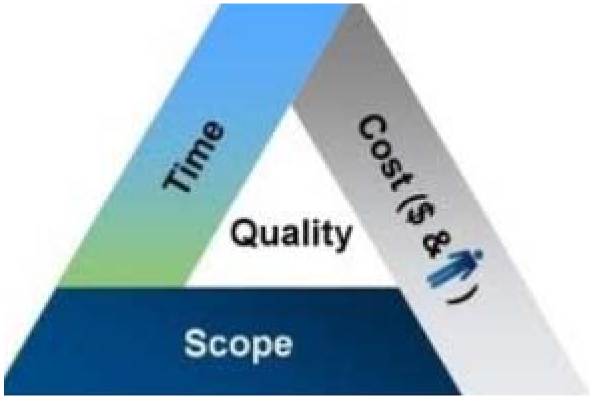
Decision making about method selection is based on suppositions, opinions, and fads.

What We Need

We need to set aside perceptions and market-speak and transform software engineering into an engineering discipline.

Decisions should be based on fair and unbiased analysis of information.

Measurement Can Help



Measurement

In physical science, the first essential step in the direction of learning any subject is to find principles of numerical reckoning and practicable methods for measuring some quality connected with it.

I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be.

William Thompson, Lord Kelvin, Lecture to the Institution of Civil Engineers, 3 May 1883

Repeatability and replicability remain foundations of good science.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution

How do you know if these are “Best Practices?”

The best software engineering methods share characteristics:

- Self-managed, self-organizing teams
- Team-focused and team-owned/team-executed practices for
  - planning, estimating, and commitment
  - measurement and tracking
  - quality management, defect management, and quality assurance
- Attention to good design
- Stakeholder involvement throughout development
- Embedded qualitative and quantitative feedback mechanisms
- Incremental and iterative development

- Architecture-led, iterative, incremental development strategy
- Rational management leadership style

Best Practices should show themselves.

Common Agile features

Planning Events (XP – Planning Game; Scrum – Sprint Planning Meeting, Daily Scrum)

Short iterations (XP – Small Releases; Scrum – Sprints)

Continuous Integration (XP – explicit practice; Scrum – assumed)

Code Ownership (XP – Collective Ownership; Scrum – team choice)

Design Practices (XP – Metaphor, Simple Design, Refactoring; Scrum – Backlog creation, grooming, and sprint story selection)

Requirements Management (XP – On-Site Customer; Scrum – Product Owner, Product & Sprint Backlogs)

Implementation Practices (XP – Pair Programming, Coding Standards, Testing, 40-Hour Week; Scrum – team choice)

If it makes a difference, that difference should be measurable in some way.

What are the research questions?

What can we say about productivity?

How can we reliably measure this?

What can we say about quality?

How can we reliably measure this?

What practices are they using?

How will we know?

The data we planned to use

Initially we wanted to study the effectiveness of Agile practices.

- Use transaction data to measure process and results.
- Use surveys of practices used.

  Pair programming, refactoring, daily SCRUM, continuous integration, cross functional team, automated builds, compute throughput, burndown, burnup, and so forth.

The study became more about learning how to collect and use this type of data.

Surveys had a horrible return rate and were useless.

Lesson: rely on observable behavior, where possible.

Let’s ask some specific questions

How big are the development teams?

What are they producing?

How many defects are delivered to the customer?

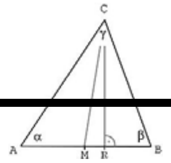
Does team size affect throughput?

Does team size affect quality?

What other factors (dedicated staff, load factors) influence productivity and quality?

Approach is observational, not experimental

Use data gathered from Agile teams using a popular data collection and project management tool.



Triangulate

We wanted a baseline for comparison.

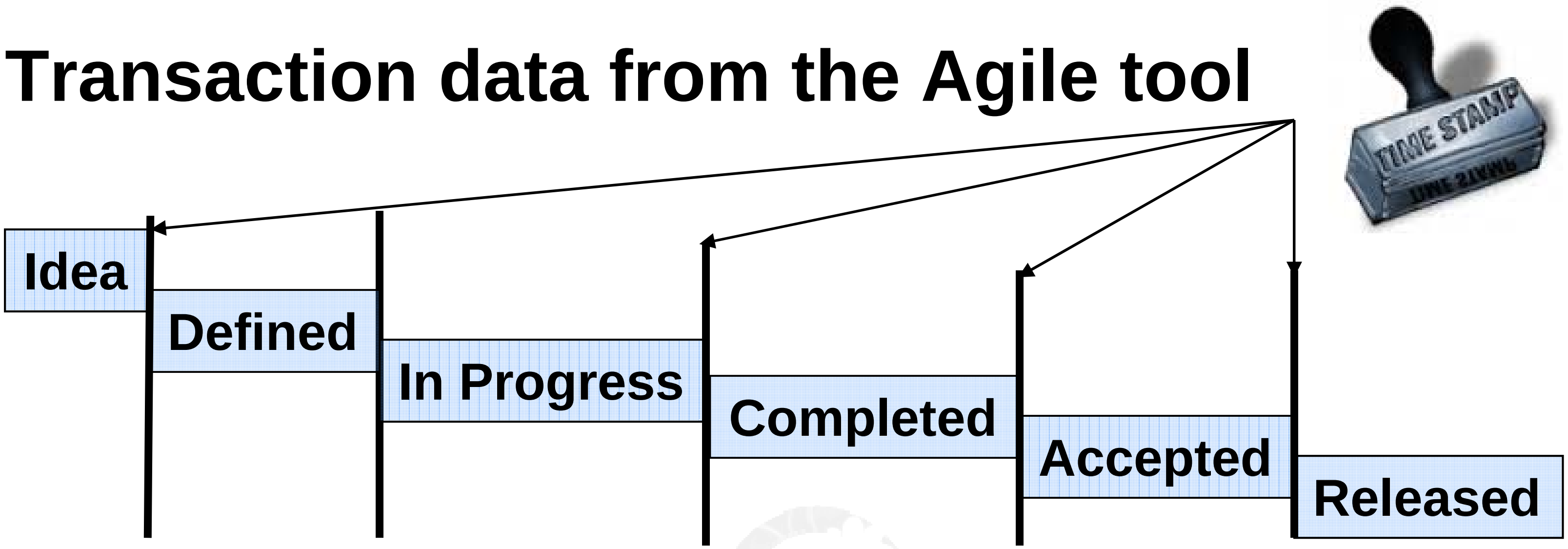
- Used our TSP data. Are they consistent? Do they seem to tell the same story?
- Take deeper dives using TSP effort, defect, and size data.

Estimate productivity (throughput), quality, and effort.

Describe team sizes, defect rates, and effort variability.

Look for correlations between practices (dedicated team members, team size) and throughput, quality, and efficiency.

Transaction data from the Agile tool



Project ID

Workflow State

Person

Stories

Event Date/Time Stamp

Story Points

Bugs

Blocked

Quarter

Half Years

Years

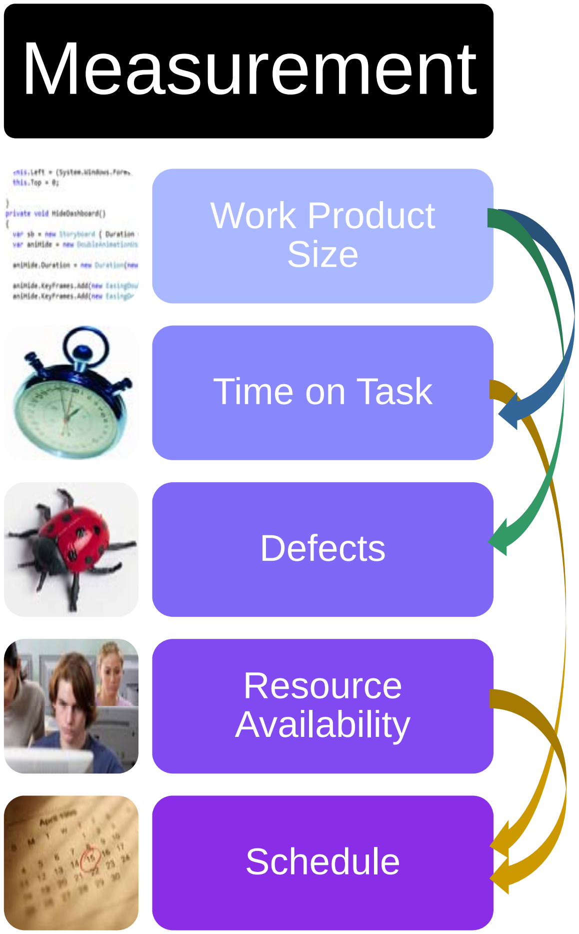
Dedicated team size

% dedicated work

Full Time Equivalent

WIP

Team Software Process (TSP$_{SM}$) measurement framework



Five direct measures

Team and team member data

Estimated during planning

Measured while working

Evaluated weekly or when a

- task is started/completed
- process activity is completed
- component is completed
- cycle is completed
- project is completed

Derived data we examined

We performed statistical analyses on two kinds of data. Few of the expected predictor responses had practical correlations.

Predictors included:

- **Productivity** defined in terms of story throughput
- **Defect Density** and relative defect density
- Median # Stories/Transaction
- **Responsiveness** in terms of time through states (lead time)
- **COV** – Coefficients of variation
- **Team Size** derived from the people who entered transactions
- **Team Member % Dedicated** derived from transaction counts
- **WIP** (Work in Progress) per Full Time Equivalent (FTE)

ANOVA: many had statistical significance, but none provided predictability (small R^2).

Some data/expectation problems

⚠ CAUTION

Local context of the data recording matters a lot!

Different teams used the tool differently.

Not every team collected defects, or collected defects at the same stage.

Defects typically were recorded when found, not attributed to an origin.

For some, a defect was “a first class story!” (counted as production twice)

One project had two groups:

Team A “groomed stories”

Team B implemented stories

(We could not tell their relationship from the data only.)

We reduced the data to a few large companies with many teams to increase the homogeneity of the data.

We based conclusions on

Effort: Full time equivalent staff (FTEs)

Quality: Released defect density (Defects divided by FTE)

Responsiveness: Average time-in-process for user stories or defects

Productivity: Number of user-story transactions recorded by the team (in the appropriate time period), divided by FTE

Predictability: Coefficient of Variation of user-story Productivity

If a team had not recorded any defects within the previous year, then the team was assumed to be not recording defects and was not included in the averaging.

Mixing data can be problematic

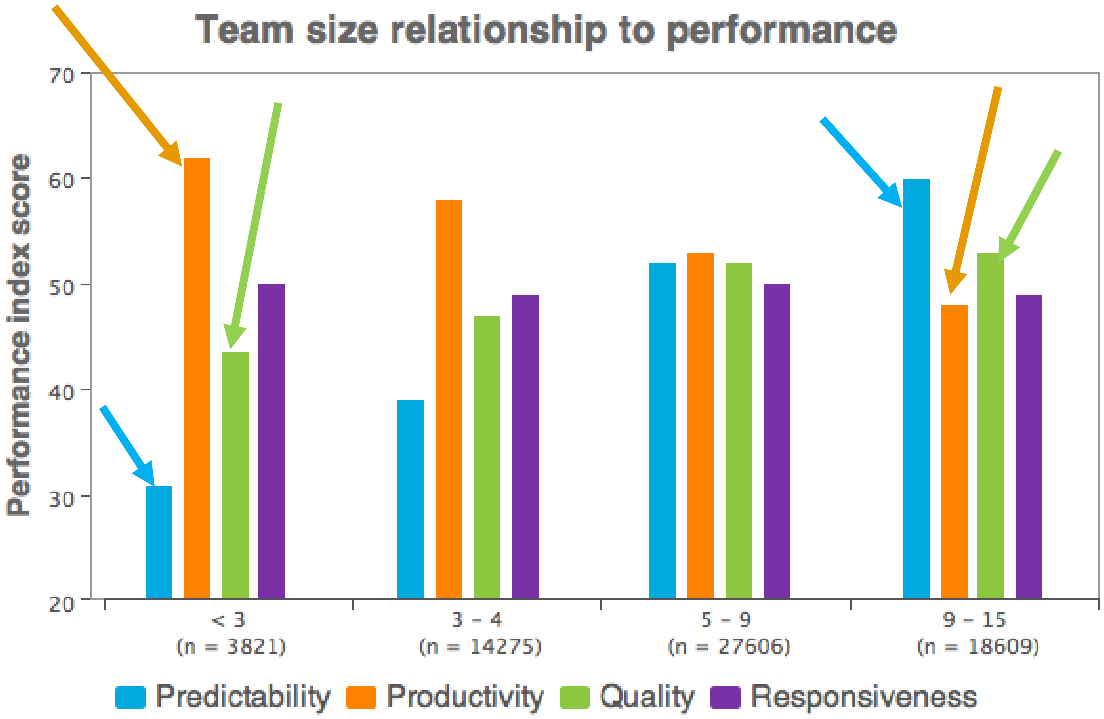
⚠ CAUTION

Combining derived proxy measures can have unexpected results.

Given the following data, do small teams really have lower quality?

The combination assumes random and unbiased correlation between effort and product size.

What a quality and production normalized to?



Smallest teams

40% less Predictability?

17% lower Quality?
(measured by FTE)

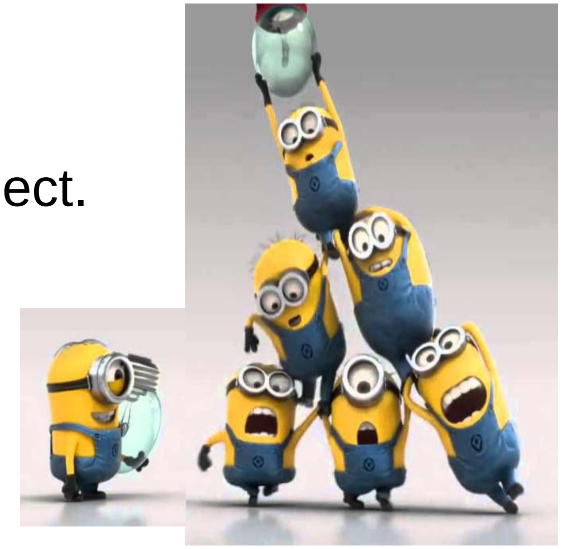
17% more Productive?
(measured by Stories)

How many software engineers does it take to change a light bulb?

7 +/- 2 because that’s the optimum team size.

2 because one resigns half way through the project.

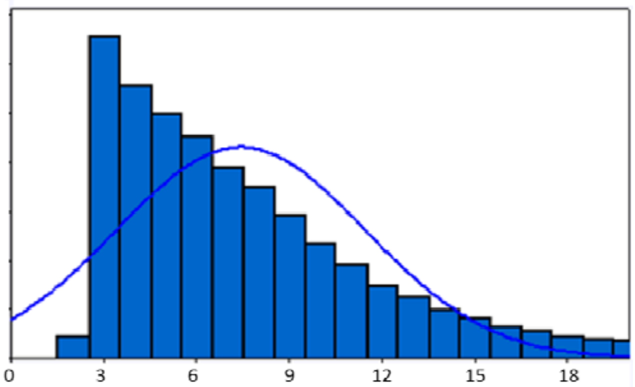
0: it’s a hardware problem!

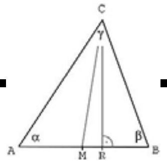


Team Size (Do they use 7 +/- 2?)

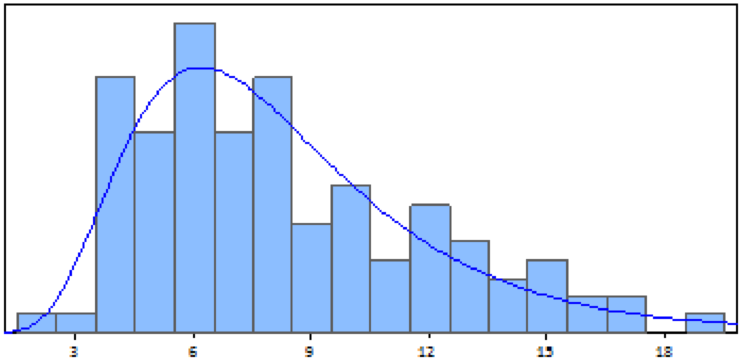
Agile Project Team sizes
[FTE]

How do they behave?





What about the comparison?
TSP_{SM} Project Team sizes
[People]



Does team size affect throughput?

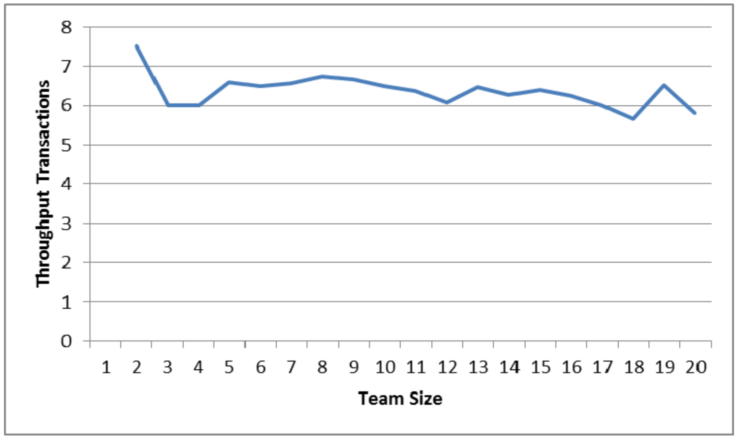
?

Measure: Story transactions per FTE

NOT weighted by story point

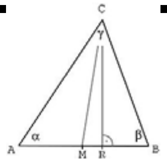
No evidence story points contain useful information

Slightly higher for very small teams

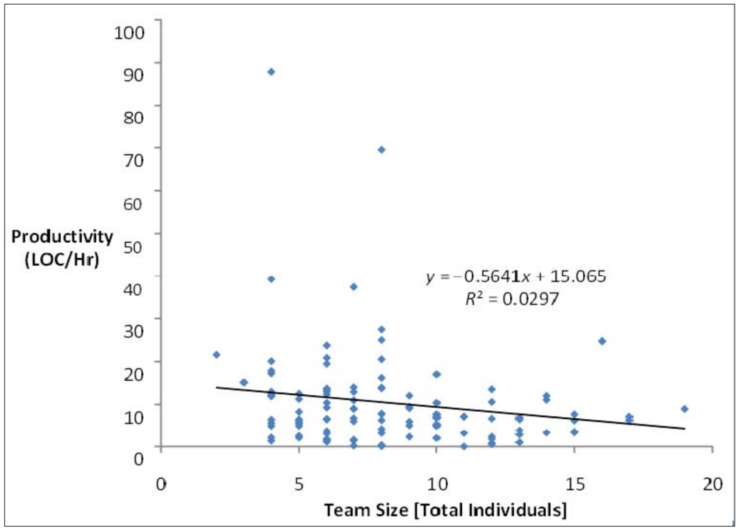


Compare to TSP LOC/Hr

Note VERY weak correlation



Teams must have some way to cope with growing team size.



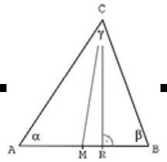
Do Team Size and Dedicated Staff affect quality?

BUSTED ?

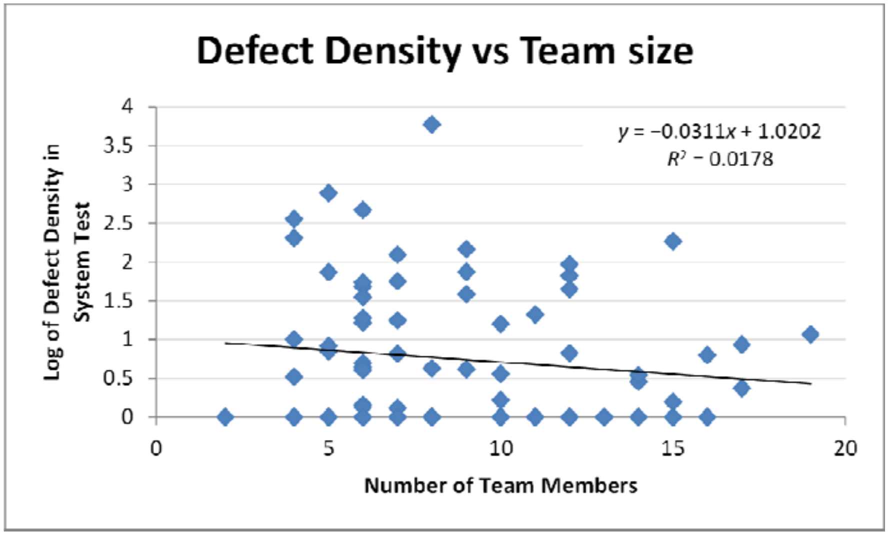
Although there is some hint that dedicated rather than part time staff may improve quality,

the **median** values are all close to zero so do not show.

Everything visible is an outlier.



TSP$_{SM}$ shows a very small effect on team size with almost zero correlation.

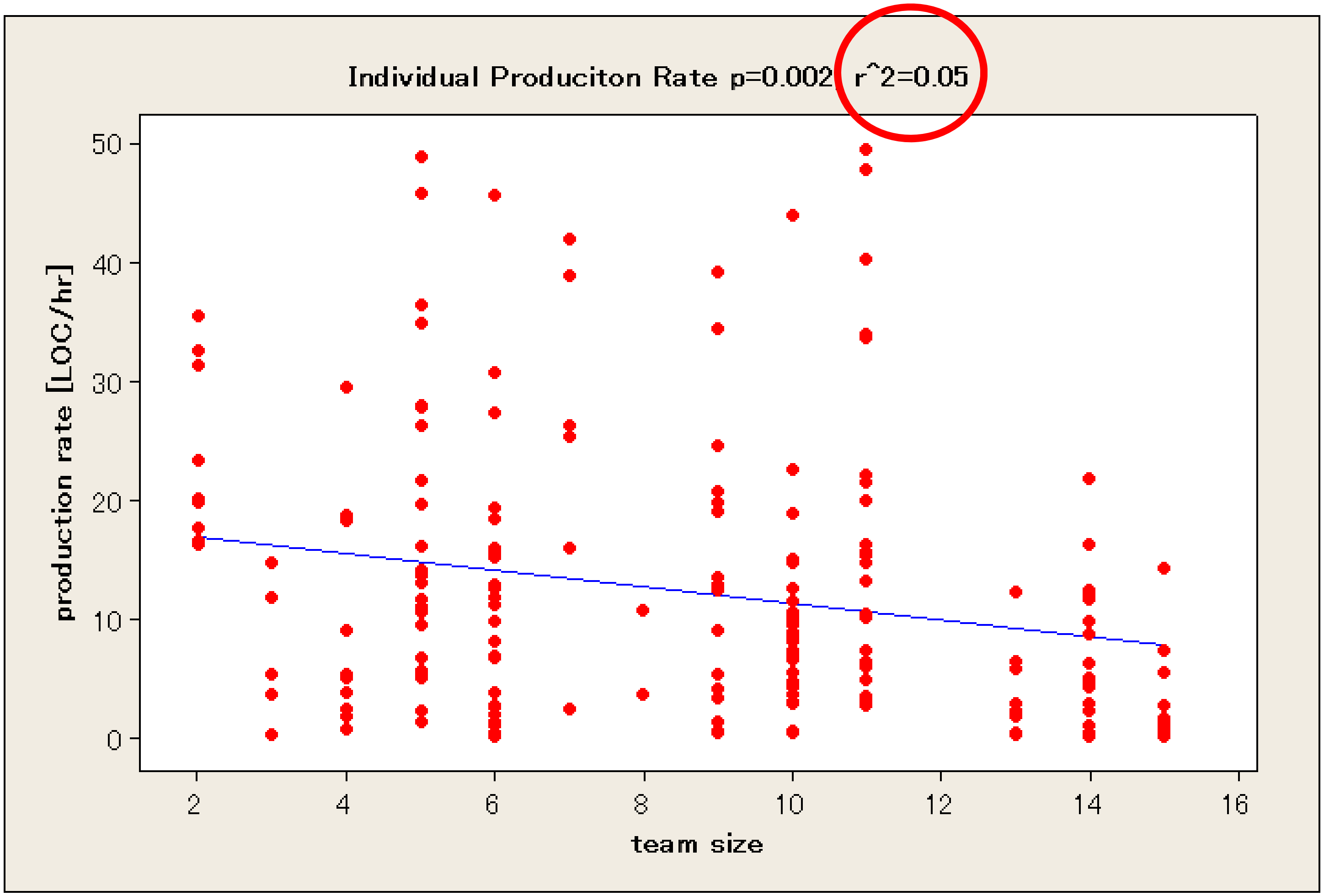


Is small team rate biased by small sample variability? How could we tell?

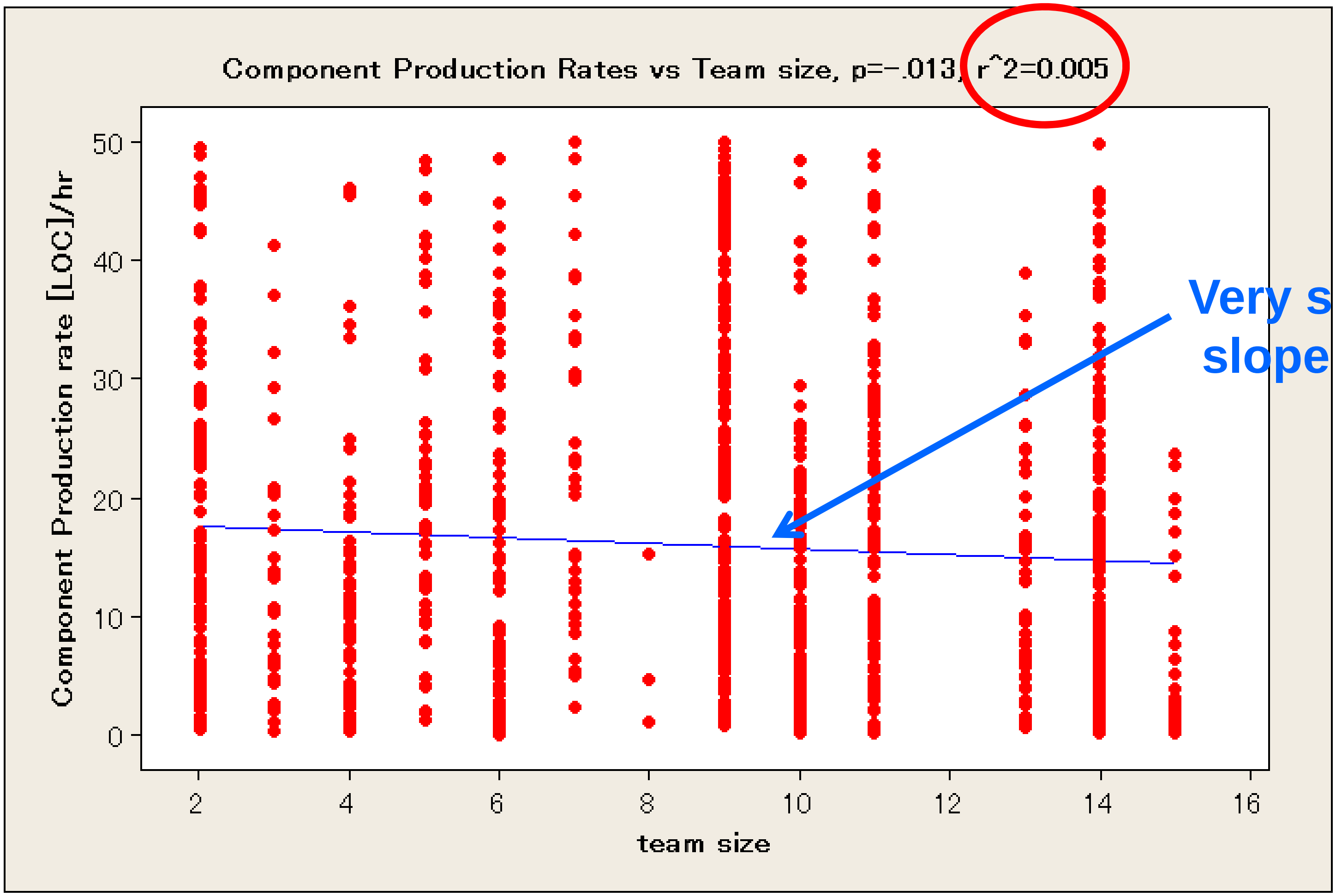
Dig deeper

Apply some statistical analyses at team, individual, and component levels.

Correlate developer code rate with team size (newer TSP data)



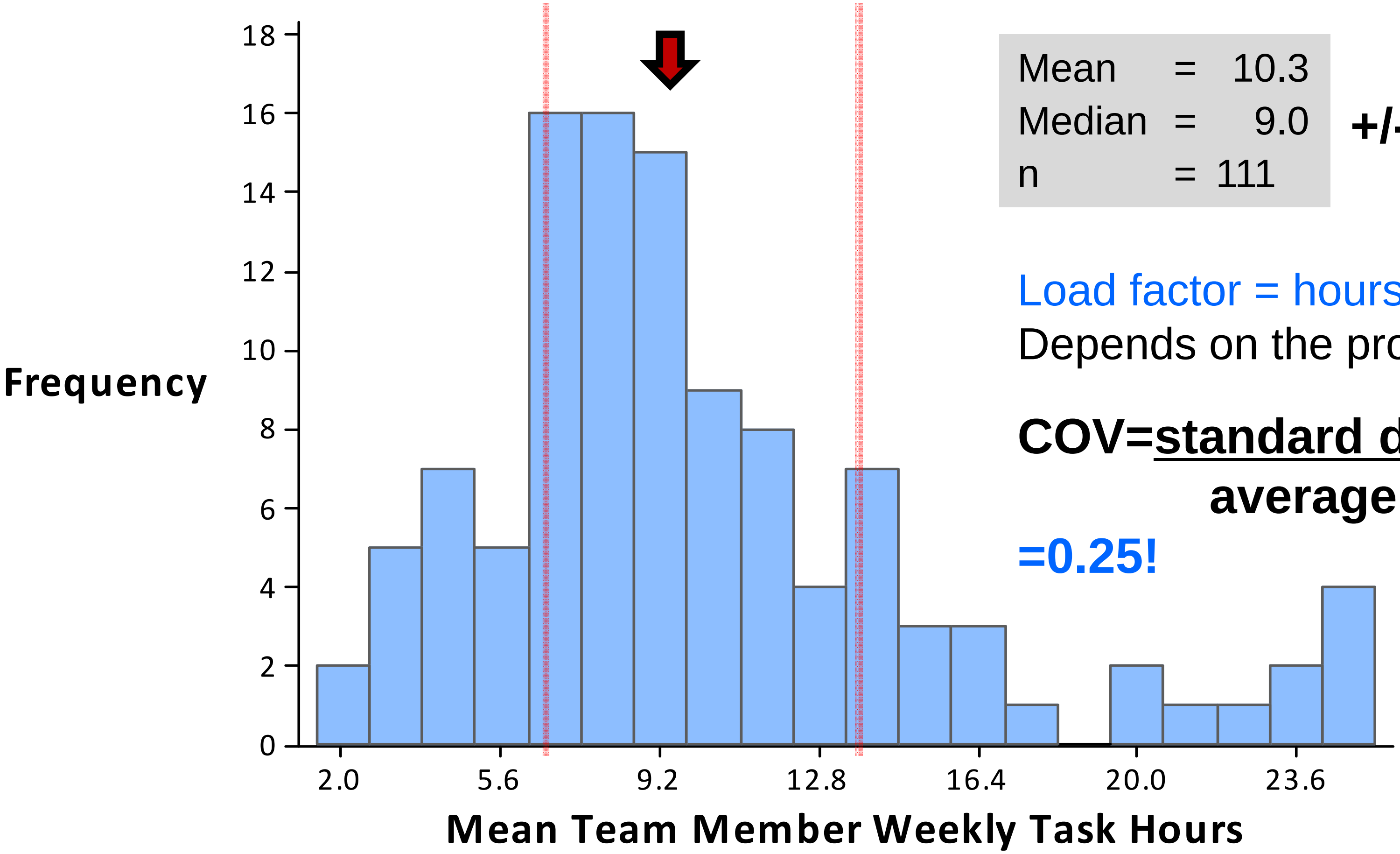
Correlate component code rate with team size (newer TSP data)



Very shallow slope

Deeper still – direct hours

Project level mean team member direct hours per week



Mean	=	10.3
Median	=	9.0
n	=	111

+/- 2.5

Load factor = hours/40

Depends on the project

$$\textbf{COV}=\frac{\textbf{standard deviation}}{\textbf{average hours}}$$

=0.25!

What effect will that COV have on shorter sprints or smaller teams?

Did not always see what we expected

Data did not match our expectations. User's needs and objectives were not aligned with the type of studies we planned.

Lifecycle steps and sizes of items going through these steps were uncontrolled, highly variable, and very local.

We used data from TSP to validate some findings.

Based on Agile literature, we expected that Agile practices **would** significantly improve the responsiveness of the team (e.g. reduce time-in-process). The data did not support this. Was this an instrumentation problem?

Based on Agile literature, we expected that Agile practices **would not** have a major effect on quality. The data did not show a strong effect on quality. But how were they collecting quality data?

We also expected team size would negatively affect quality and productivity, but we saw almost no effect of team size on quality.

Summary of tentative findings

At least up to 20 or so, Team Size doesn't matter much for

- Productivity
- Defect density

This is surprising.

Source of variability?

Mean “load factor” is 2.67 from TSP data (averaged over an entire cycle).

Median “load factor” is closer to 3.0.

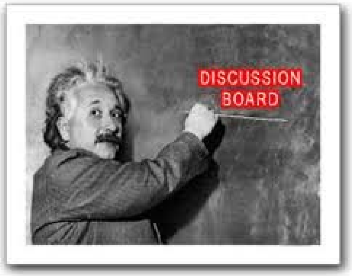
This is not surprising.

Load factor is not a constant.

Variability (COV) of “load factor” is high, standard deviation = 0.25.

This is surprising enough for further study.

Summary of some data lessons



Large volumes of **Transactional Data** are different from conventional.

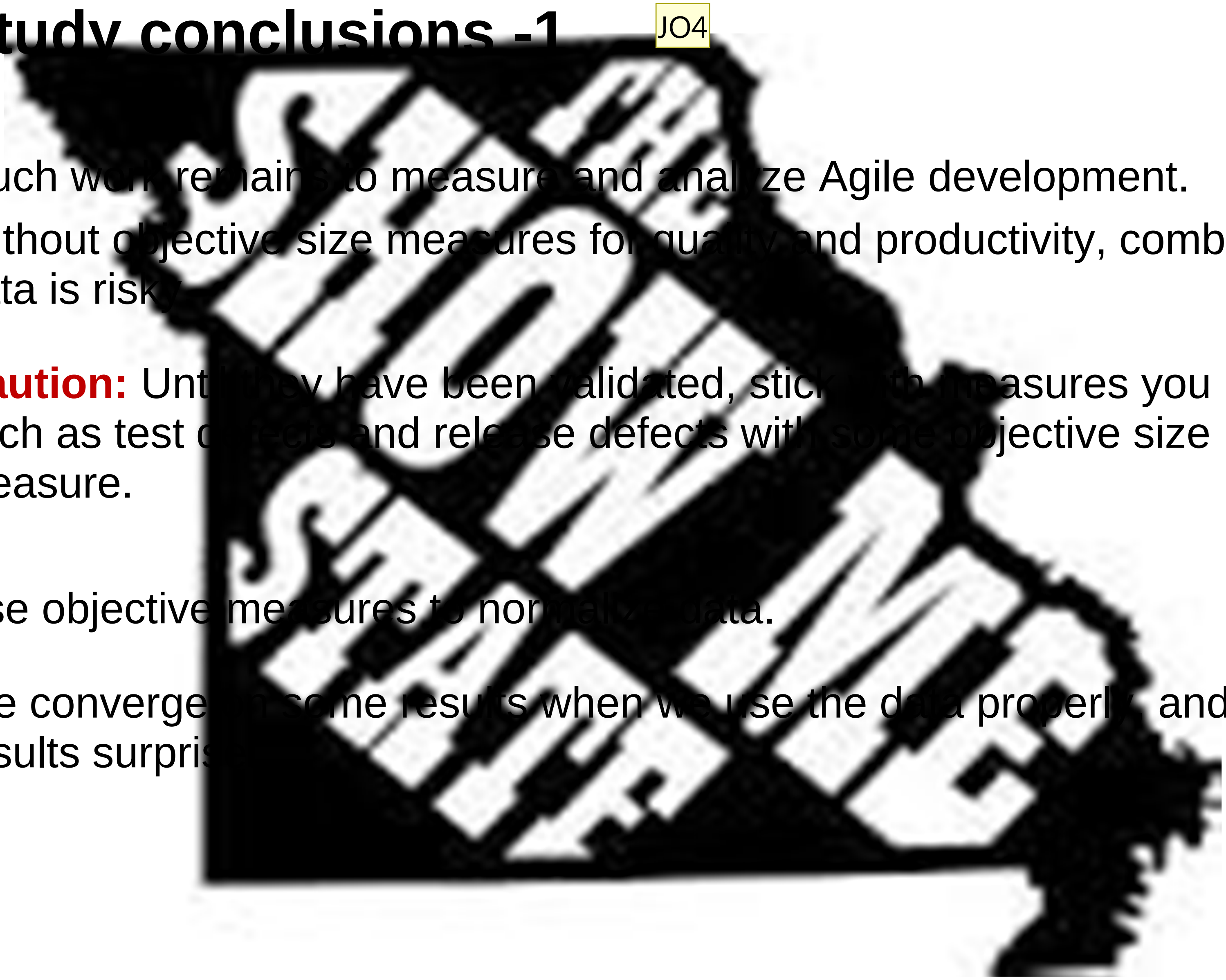
Transactional Data has some characteristics of **Big Data**.

Observational, un**C**ontrolled, needs **C**ontext, **A**dapted, **M**erged (from Kaiser Fung and William of Occam)

- Data are observational rather than designed; definitions may vary.
- There are no controls or quasi-controls to test causality.
- We adapted data collected by different people with different purpose.
- Get the context; for example, identify how users use the tool.
- Merged data exacerbates the issue of lack of definition and misaligned objectives.
- We must come to terms with variability of data early on; significance does not mean real predictable effects.

Study conclusions -1

JO4



Much work remains to measure and analyze Agile development.

Without objective size measures for quality and productivity, combining data is risky.

Caution: Until they have been validated, stick with measures you know, such as test defects and release defects with some objective size measure.

Use objective measures to normalize data.

We converge on some results when we use the data properly, and the results surprise.

Slide 38

JO4 These are conclusions, but after a key note I would want to see some bold actions embedded in the closing. Next year is the 50th anniversary of the NATO conference on software engineering. Software was characterized as late, over budget, and buggy. What's changed. Certainly there should be a call to do something about measurement. Going back to your Lord Kelvin slide, where would engineering be without standard, objective, direct measures of the product and process. Imagine the world without square feet (nice visual joke there too!)

James Over, 8/31/2017

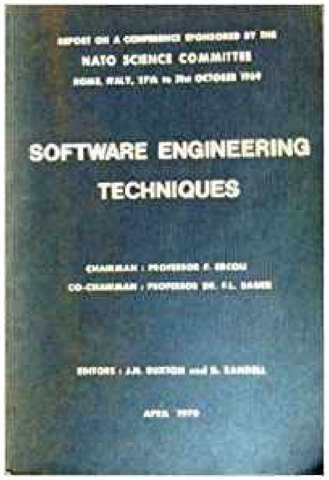
Study conclusions -2

Much common wisdom might not apply to you. **How will you know?**

- How many developers should there be on a team?
- How long should a sprint be?
- **Show me!** Measure **and** think for yourself!

“I'm skeptical of any conclusion until it's proven to be so. I have a very healthy distrust of other people's opinions.” - Dr. Rany Jazayerli

Where do we go from here?



REPORT ON A CONFERENCE SPONSORED BY THE
NATO SCIENCE COMMITTEE
ROME, ITALY, 27th to 31st OCTOBER 1969

SOFTWARE ENGINEERING
TECHNIQUES

CHAIRMAN : PROFESSOR P. ERCOLI
CO-CHAIRMAN : PROFESSOR DR. F.L. BAUER

EDITORS : J.N. BUXTON and B. RANDELL

APRIL 1970

Next year is the 50th anniversary of the NATO conference on software engineering.

Software was characterized as late, over budget, and buggy.

What has changed?

Now, software is often late, over budget, and buggy… and insecure.

Call to Action

It’s now up to **you**.

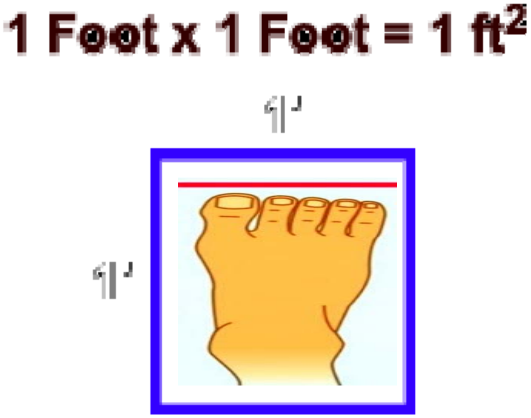
Don’t let others fool you and don’t fool yourselves!

Science needs measurement.

If you don’t know how to measure it, find a way!

You might have to invent it.

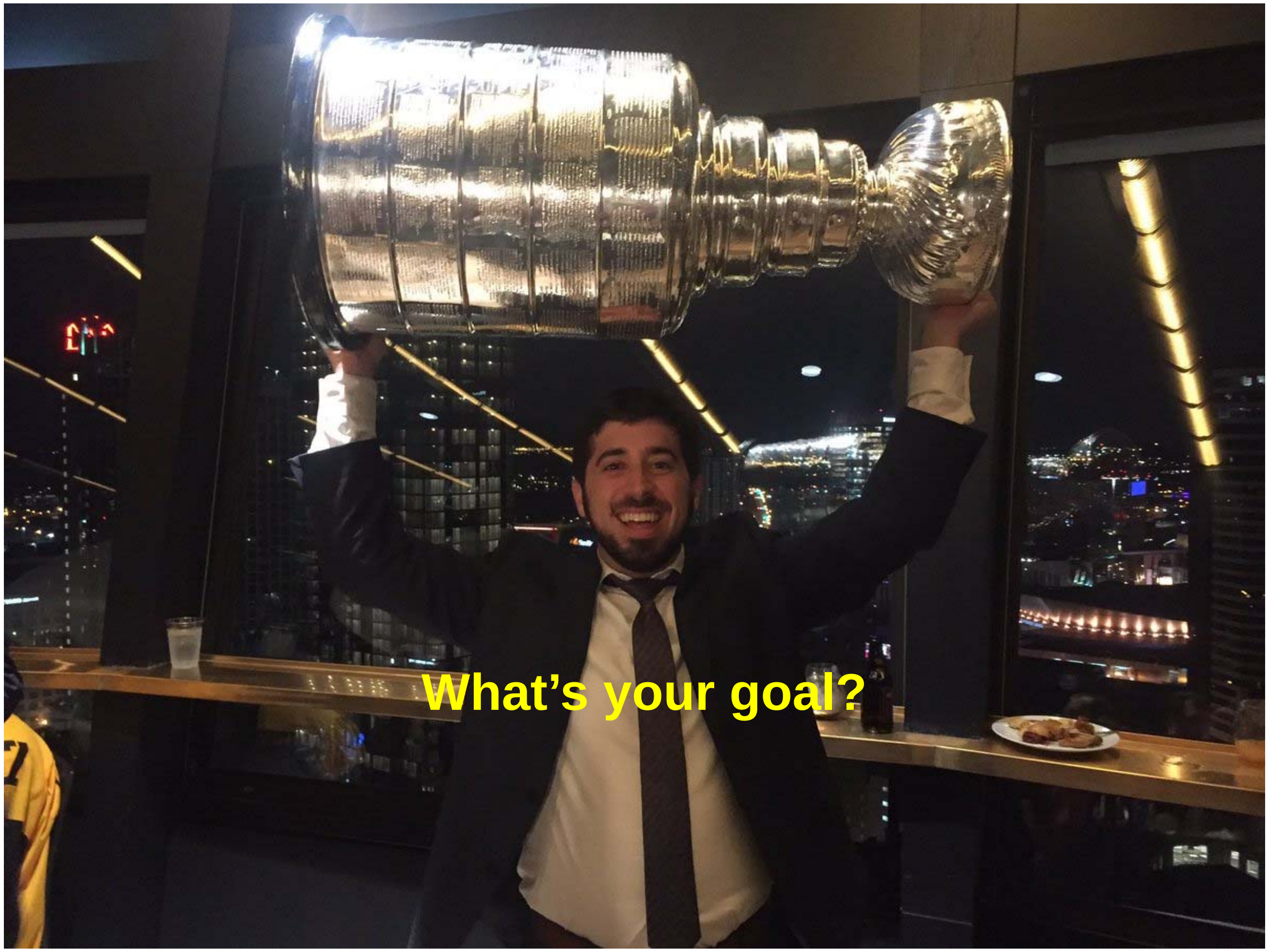
Where would engineering be without coherent standard measures?



Remember Sam, the sports statistician?

He had a really good summer…

Can you guess who he works for?



What's your goal?

What are *you* going to do?

Efficacy of Agile LENS Team

William Nichols
wrn@sei.cmu.edu
412-268-1727

with
Mark Kasunic
James McCurley
Will Hayes
Sarah Sheard

And special thanks to
Larry Maccherone

Common Agile Features

Planning Events (XP – Planning Game; Scrum – Sprint Planning Meeting, Daily Scrum)

Short iterations (XP – Small Releases; Scrum – Sprints)

Continuous Integration (XP – explicit practice; Scrum – assumed)

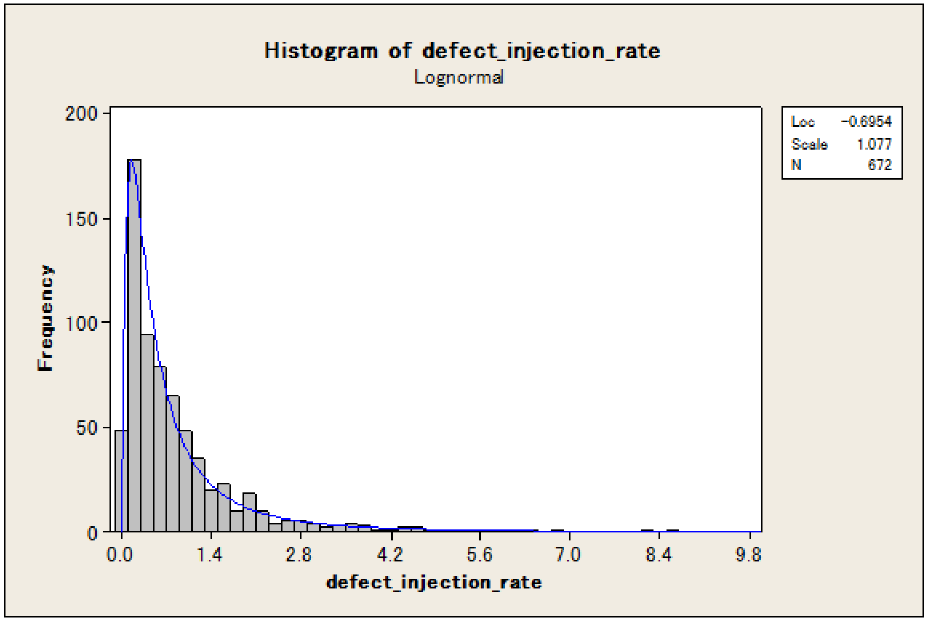
Code Ownership (XP – Collective Ownership; Scrum – team choice)

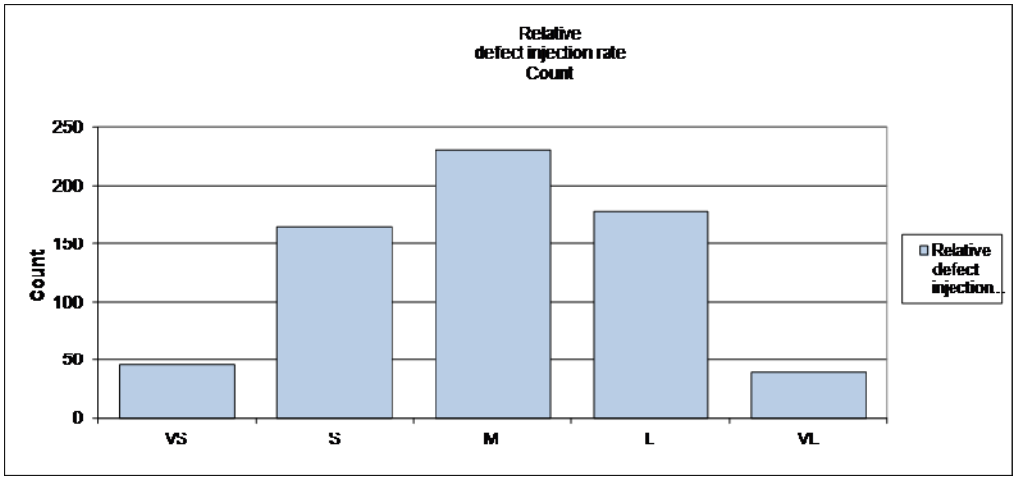
Design Practices (XP – Metaphor, Simple Design, Refactoring; Scrum – Backlog creation, grooming, and sprint story selection)

Requirements Management (XP – On-Site Customer; Scrum – Product Owner, Product & Sprint Backlogs)

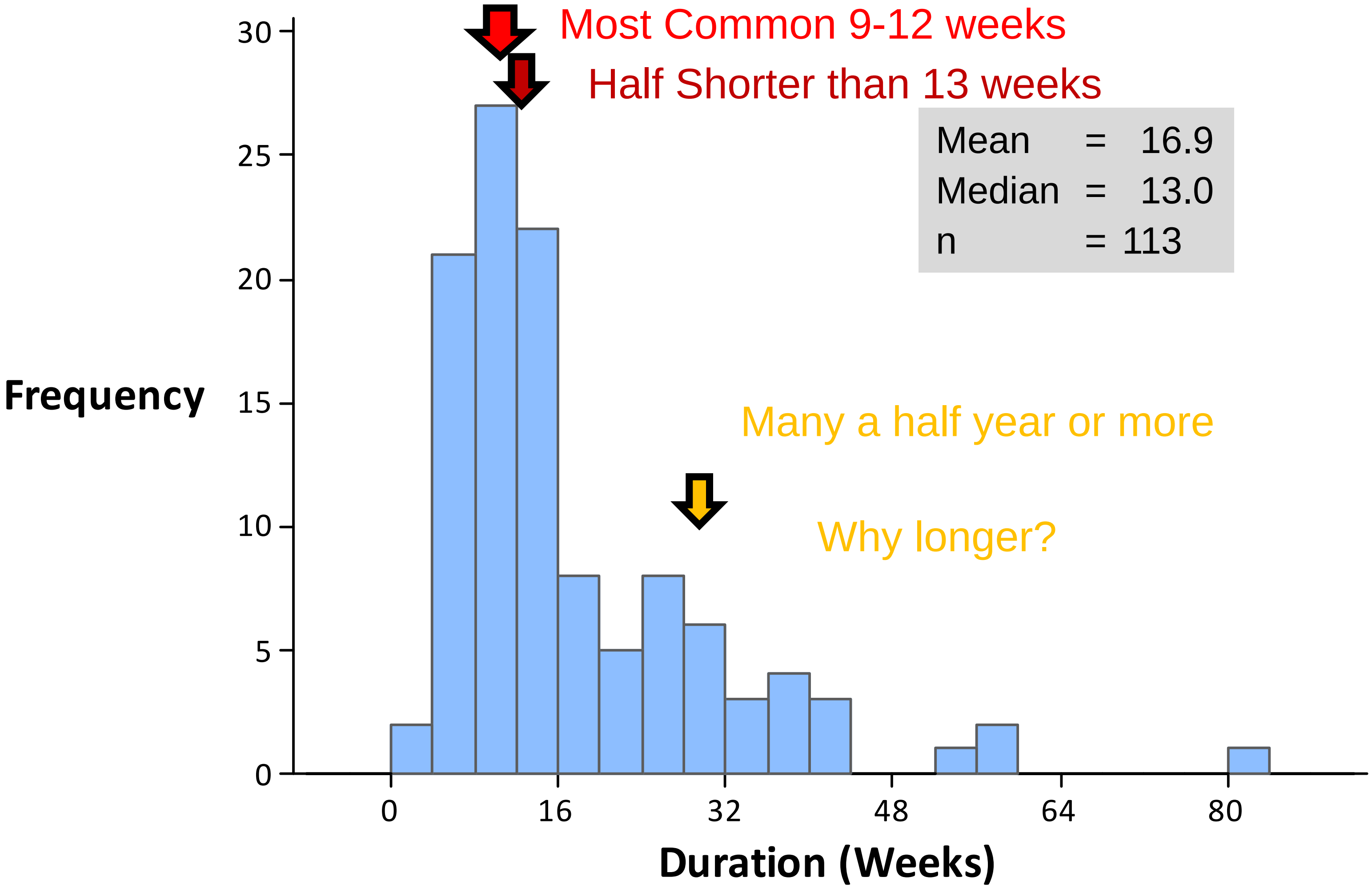
Implementation Practices (XP – Pair Programming, Coding Standards, Testing, 40-Hour Week; Scrum – team choice)

Defect Injection, the most predictable process?





What were Project Durations? [Weeks]



Most Common 9-12 weeks

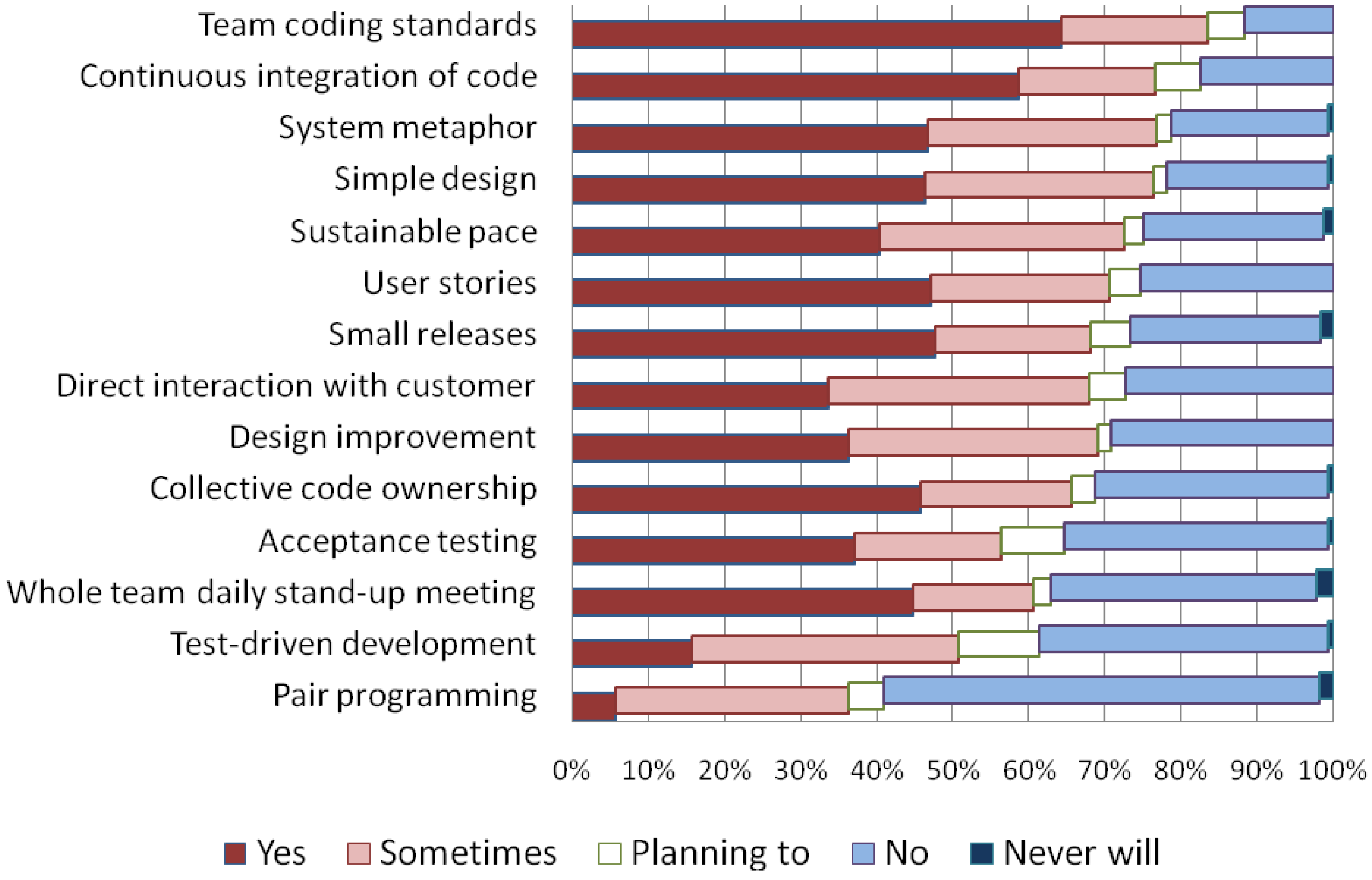
Half Shorter than 13 weeks

Mean = 16.9
Median = 13.0
n = 113

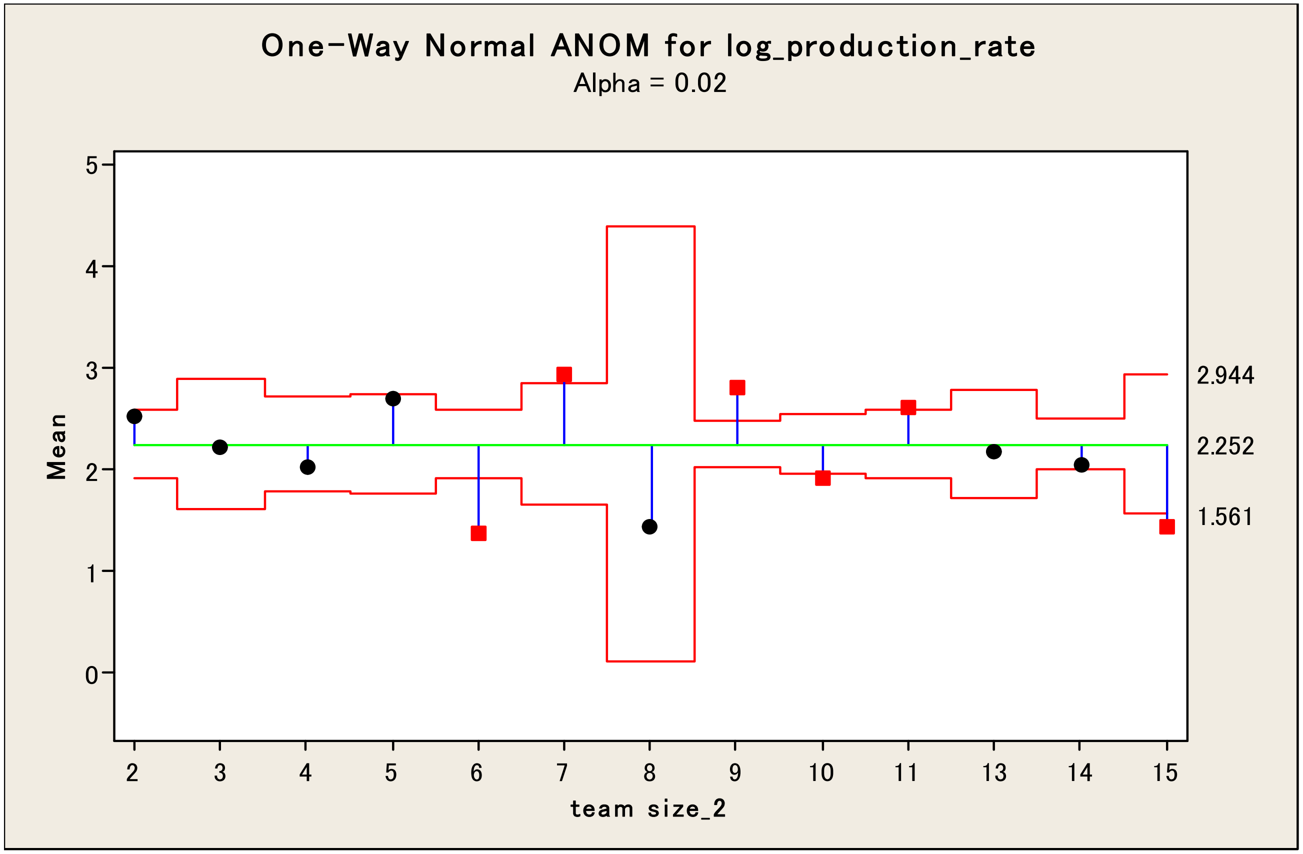
Many a half year or more

Why longer?

Microsoft – Agile Adoption by Practice



ANOMA, Team Production Rate by Team size some newer TSP data



It is our responsibility as scientists ... to teach how doubt is not to be feared but welcomed and discussed.

If you're doing an experiment, you should report everything that you think might make it invalid - not only what you think is right about it

Religion is a culture of faith; science is a culture of doubt.

It is better to have questions that don't have answers, rather than having answers that can't be questioned.

The statements of science are not of what is true and what is not true, but statements of what is known with different degrees of certainty.

Give me questions I can't answer not answers I can't question.

We must be careful not to believe things simply because we want them to be true. No one can fool you as easily as you can fool yourself!ly right, we can only be sure we are wrong!

How we use measurement

First-Order Measurement

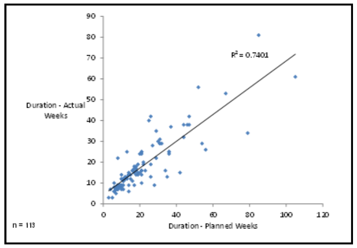
What *seems* to be happening?
Tends to be qualitative and fast.

Second-Order Measurement

What's *really* happening? And how is it changing?
It needs to be quantitative; subject to more refined models.

Third-Order Measurement

What *happens* in a more general and universal sense?
Needs to be precise with checks for validity; statistical variation must be characterized and interpreted appropriately.


Duration - Actual Weeks
R² = 0.7401
Duration - Planned Weeks
n = 113