Measuring the Efficacy of Agile Practice

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

William Nichols, September 17, 2014

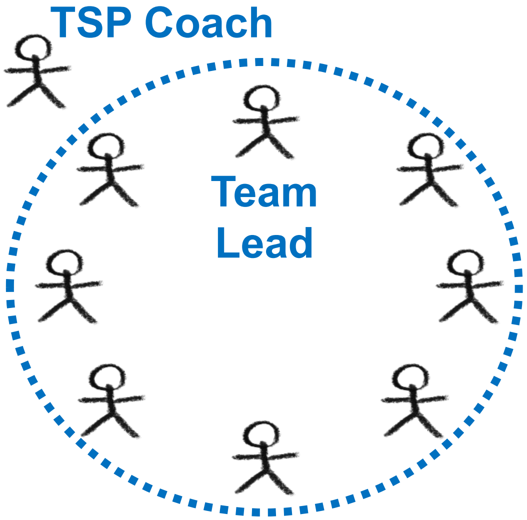
Document Markings

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0001650

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

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 traditional knowledge is ridiculous hokum." - Bill James

Where do we start?
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

How we use measurement

First-Order Measurement

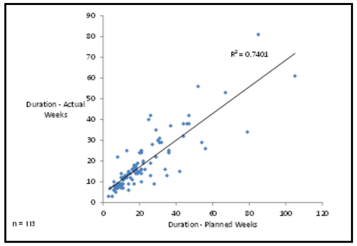
What *seems* to be happening?
Tends to be **qualitative** and fast.

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.



Concept:
Best practices should show themselves

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

Common Agile practices

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)

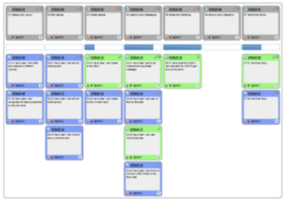
Implementation Practices (XP – Pair Programming, Coding Standards, Testing, 40-Hour Week; Scrum – team choice)

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

Our research questions included

How do selected factors affect throughput?

How can reliably we measure this?



How do selected factors affect quality?

How can we reliably measure this?

What practices can we infer from the data?

How will we know what they are really doing

Let’s make questions more specific

How did specific practices associate with throughput and quality?

How big are the development teams?

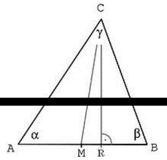
Does team size affect throughput? (for example is 7+/- 2 preferred)

Does team size affect quality?

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

Approach

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



Triangulate

- Compare with 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.

The data we planned to use

Initially we wanted to study the relative effectiveness of agile practices

- Use transaction data to measure process and results

- Use surveys to determine the practices that were used

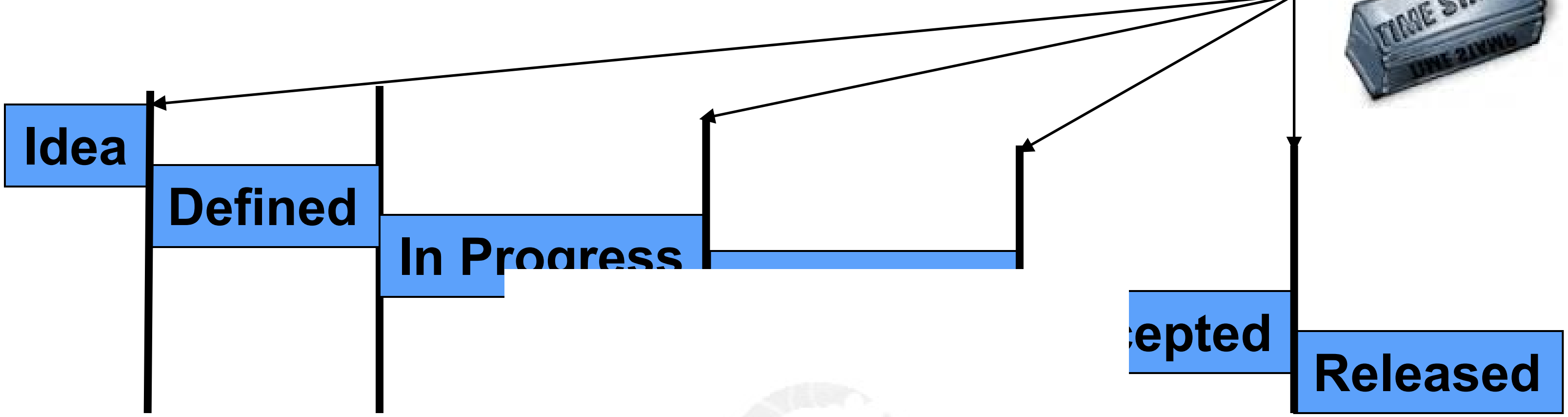
  Pair programming, refactoring, daily SCRUM, continuous integration,cross functional team, automated builds, compute throughput, burndown, burnup, and so forth

We did not obtain enough data on practices because of poor survey return rates.

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

We focused on practices and conditions we would observe.

Transaction data from the Agile tool

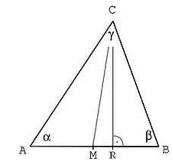


Project ID
Workflow State
Person
Stories
Event Date/Time stam
Story Points
Bugs
Blocked

Quarter
Half Years
Years

Dedicated team size
% dedicated work
Full Time Equivalent
WIP

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



Measurement

Work Product Size

Time on Task

Defects

Resource Availability

Schedule

Five direct measures

Team and team member data

Estimated during planning

Measured while working

Evaluated weekly or when a

- task is completed
- process phase is completed
- component is completed
- cycle is completed
- project is completed

Agile data we examined

We performed statistical analyses on two kinds of data, few of the expected predictor response had practical correlations.

Predictors included

- FTE (full time equivalent staff month)
- Production Rate: story throughput per month
- Defect Rate: defects per FTE month
- Responsiveness : duration time through states (lead time)
- Coefficients of variation
- Team Size derived from who entered transactions
- Team Member % Dedicated derived from transaction counts
- WIP (Work in Progress) per FTE

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

Some data/expectation problems

CAUTION

Local context matters

Different Teams used the tool differently.

Not every team collected defects.

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

One project had two groups

Team A “groomed stories”

Team B implemented stories

We could not tell them apart from the data

Teams use different tools, work in different domains, and so forth

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

We based analysis and 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.

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

Predictability: Coefficient of Variation of user-story Throughput

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

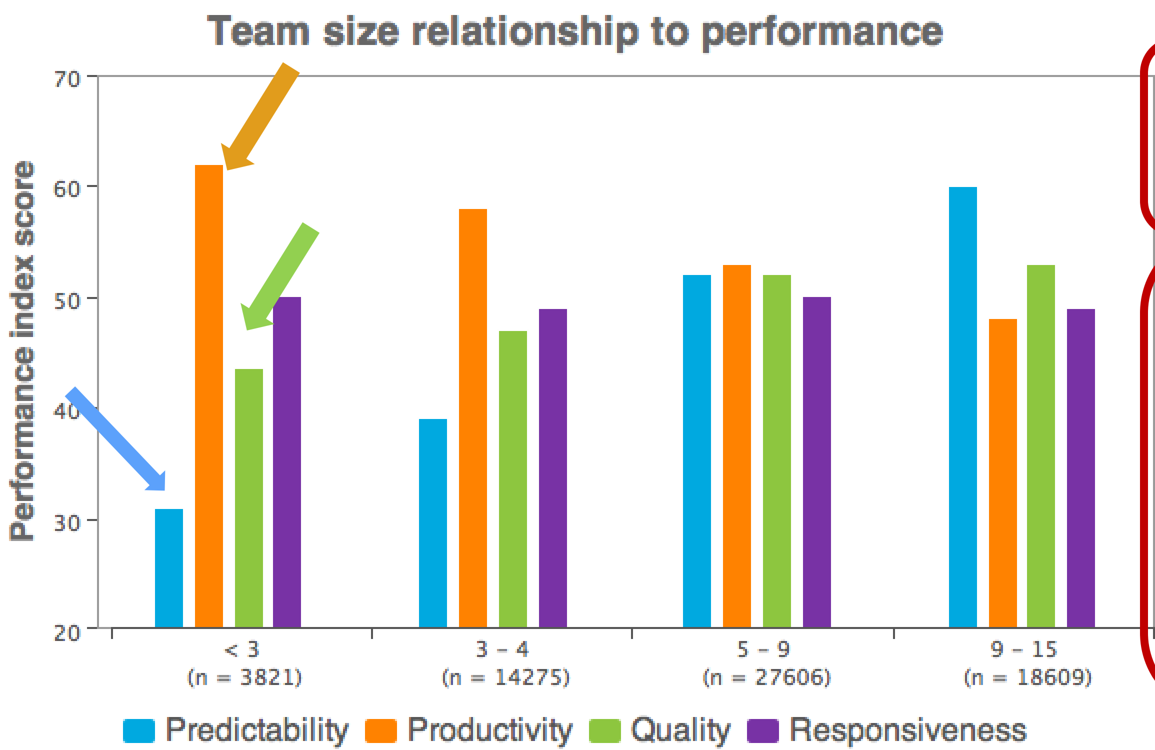
Mixing data is hard to interpret

CAUTION

Combining derived 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.



Why?

Smallest teams

40% less Predictability?

17% lower Quality?

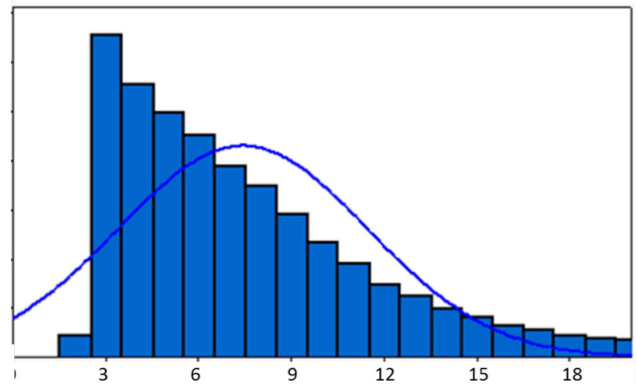
(measured by FTE)

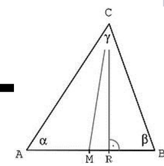
17% more Productive?

(measured by Stories)

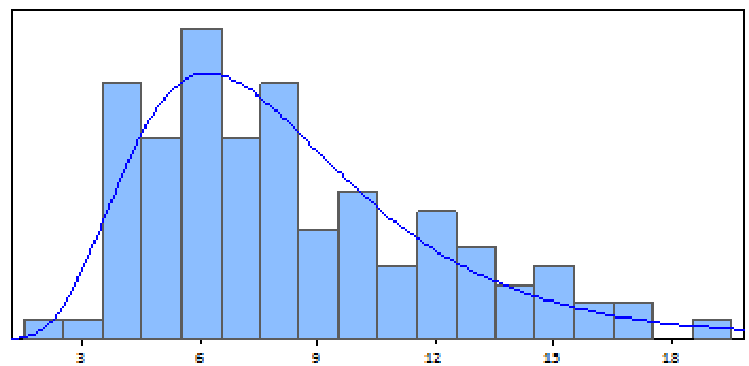
Team Size, (do they use 7 +/- 2?)

Agile Project Team sizes
[FTE]





TSP$_{SM}$ Project Team sizes
[People]



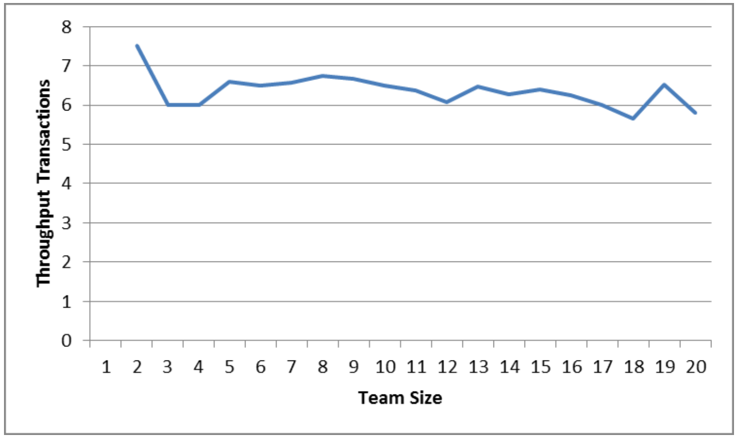
Does team size affect throughput?

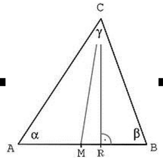
?

Measure: Story transactions per FTE

NOT weighted by story point

Slightly higher for very small teams

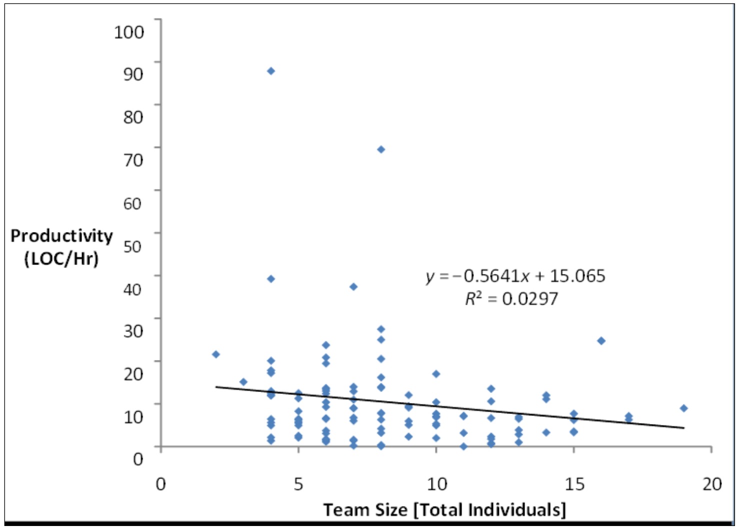




TSP LOC/Hr

Note VERY weak correlation

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



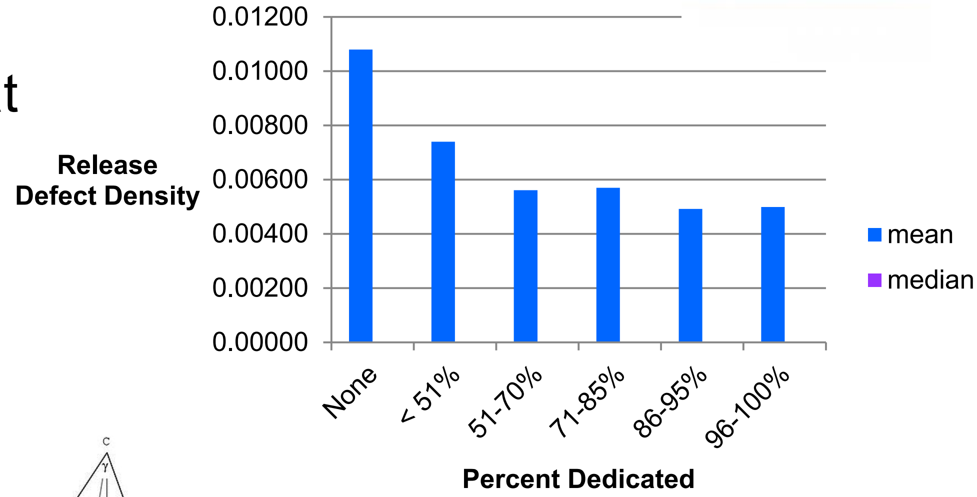
Do Team Size and Dedicated Staff affect quality?

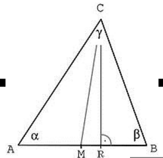
?

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

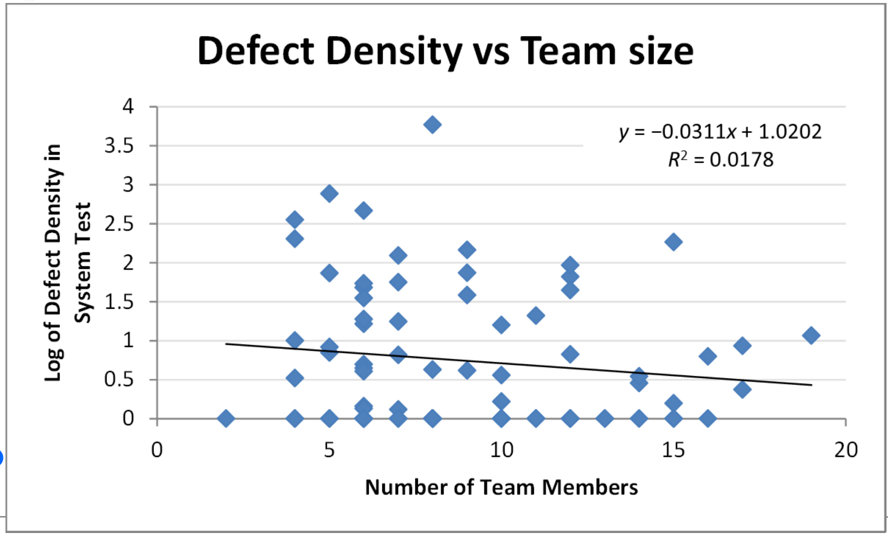
The median values are all close to zero so do not show.

Everything visible is an outlier





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

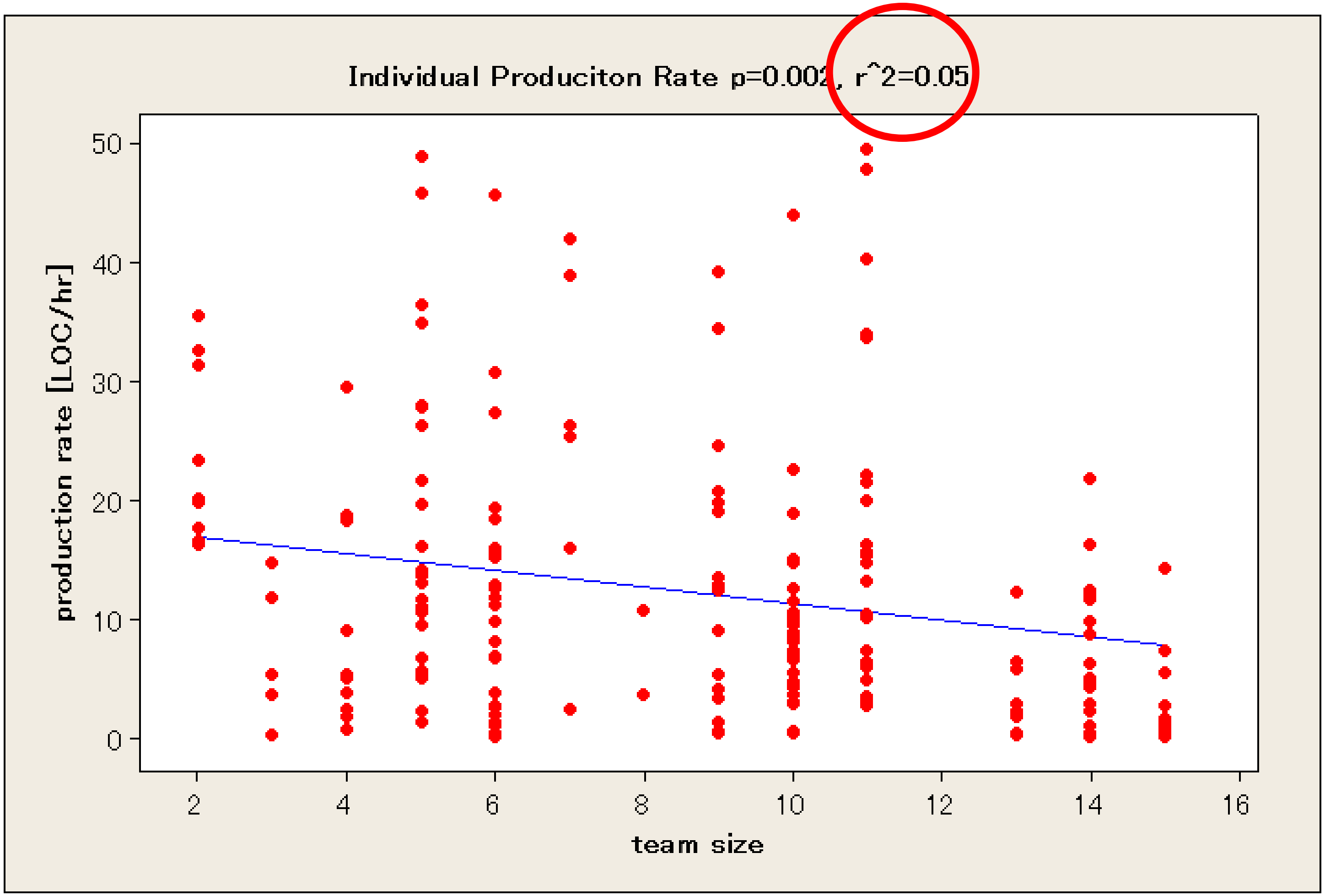


Is small team rate biased by small sample variability? How could we tell?

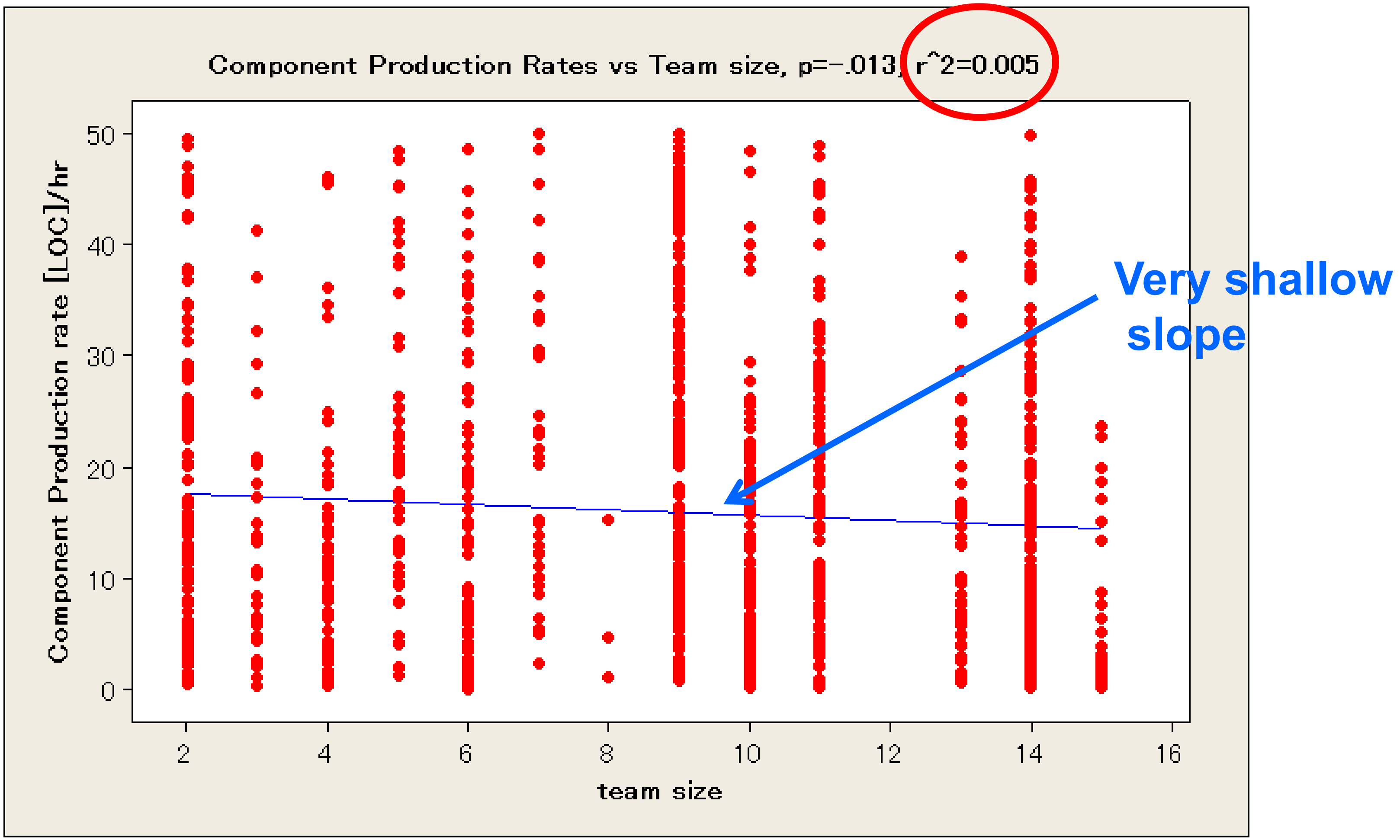
Dig deeper

Apply some statistical analyses at Team, Individual, and components

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

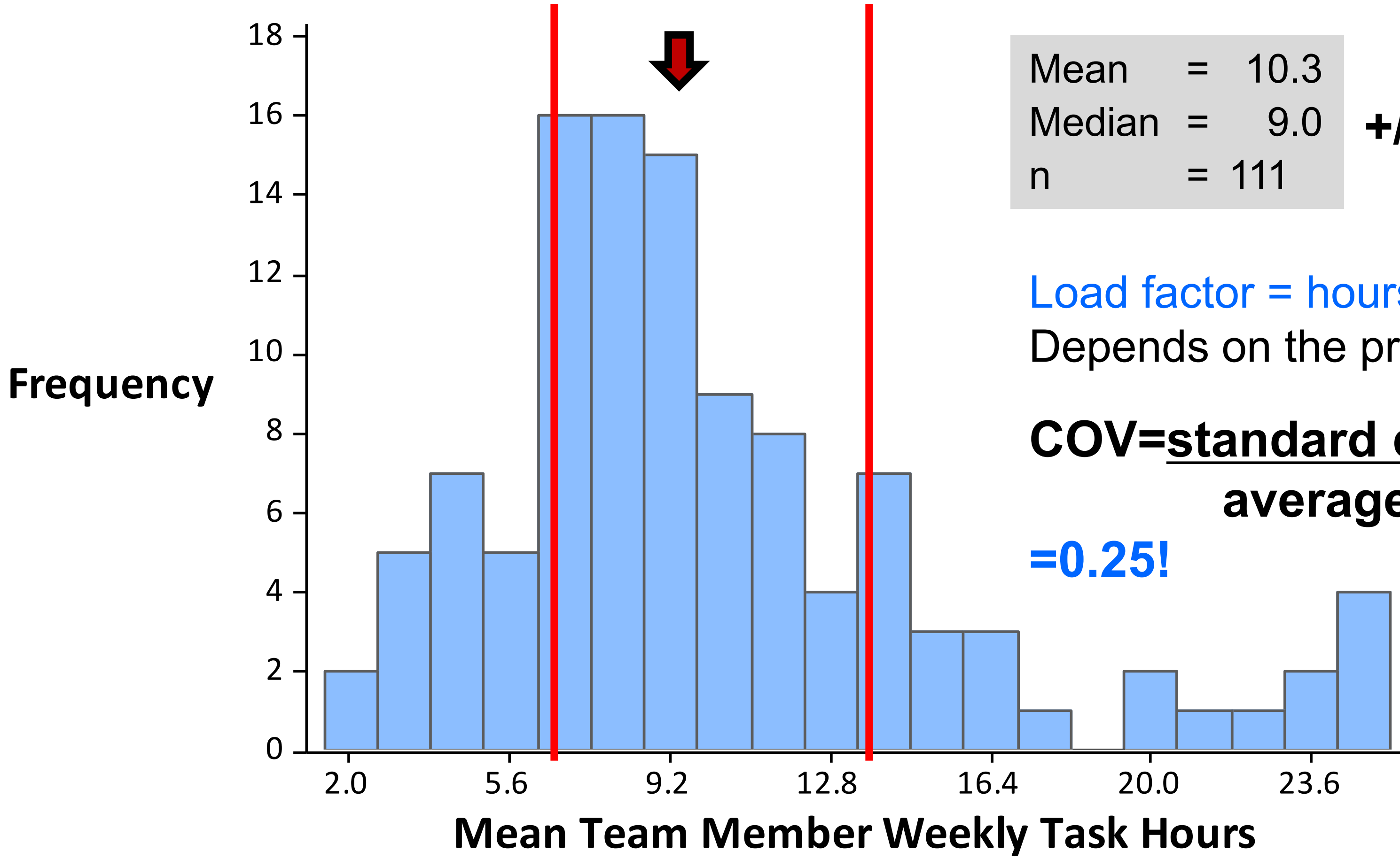


Correlate component code rate for component with team size (newer TSP data)



Deeper still, Direct Hours

mean Team Member Weekly Direct Hours per Week



Mean	=	10.3
Median	=	9.0
n	=	111

+/- 2.5

Load factor = hours/40

Depends on the project

COV=<u>standard deviation</u>

average hours

=0.25!

Did not always see what we expected

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

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

We used data from Team Software Process to validate some findings.

Based on Agile Literature, we expected that Agile practices **would** significantly improve the responsiveness of the team (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 Results

A least not up to 20 or so Team size doesn't matter much with

- Throughput or,
- Defect density

  This is surprising!

What is the 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, and shows other factors are important

Load factor is not a constant

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

This is surprising and suggests need for 3rd order measurement

Summary of some data cautions

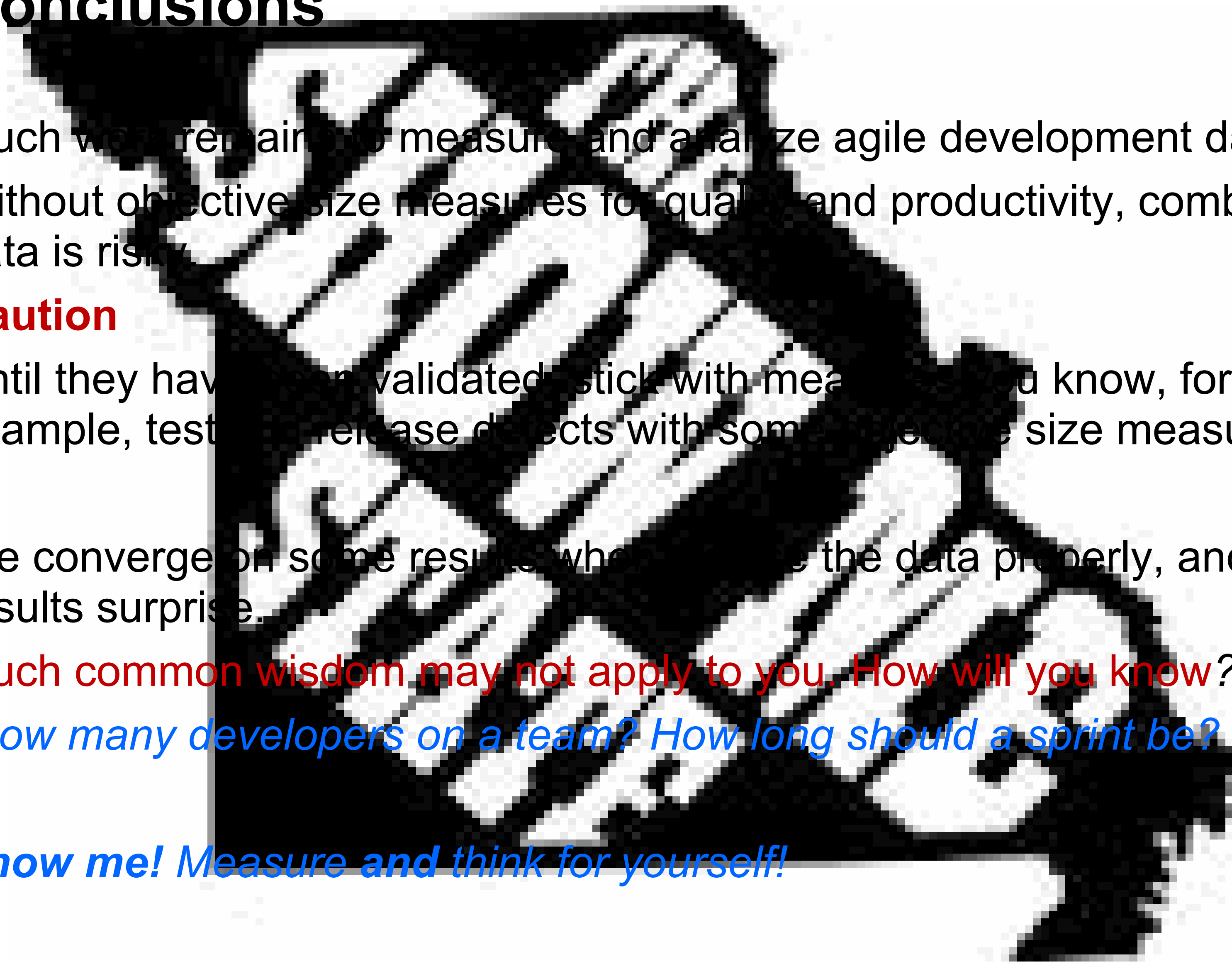
Large volumes of Transactional Data are different than conventional

It has some characteristics of **Big Data**

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

1. Data is observational rather than designed, definitions may vary
2. No controls or quasi-controls to test causality
3. Adapted data collected by different people with different purposes
4. Get the context, for example, identify how users use the tool
5. Created and collected by someone else for a different purpose
6. Merged data exacerbates the issue of lack of definition and misaligned objectives
7. Come to terms with variability of data early on, significance does not mean real predictable effects

Conclusions



Much work remains to measure and analyze agile development data

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

Caution

Until they have been validated, stick with measures you know, for example, test and release defects with some objective size measure.

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

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

How many developers on a team? How long should a sprint be?

Show me! *Measure* ***and*** *think for yourself!*

Efficacy of Agile Practices 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)

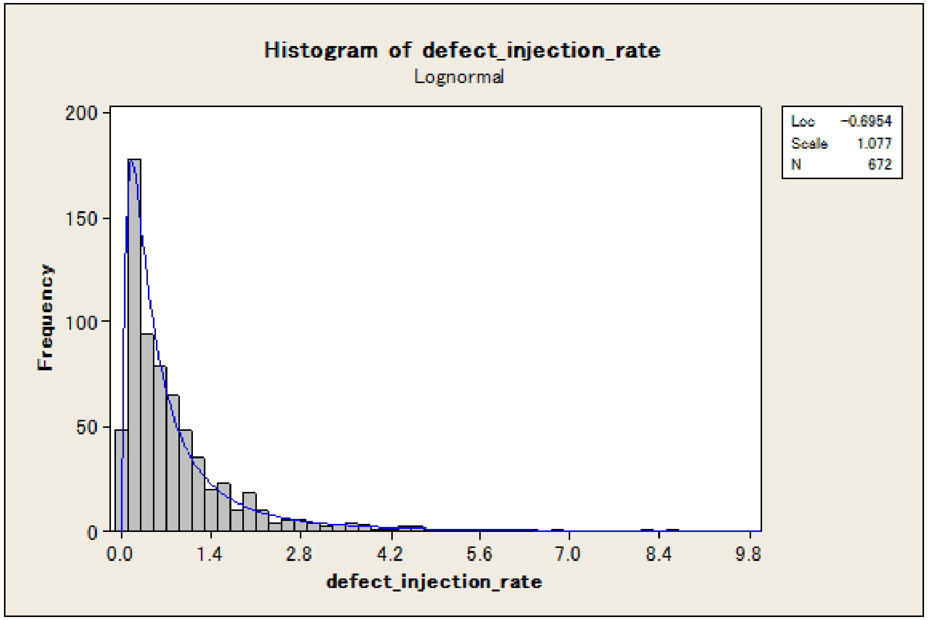
Code Ownership (XP – Collective Ownership; Scrum – team choice)

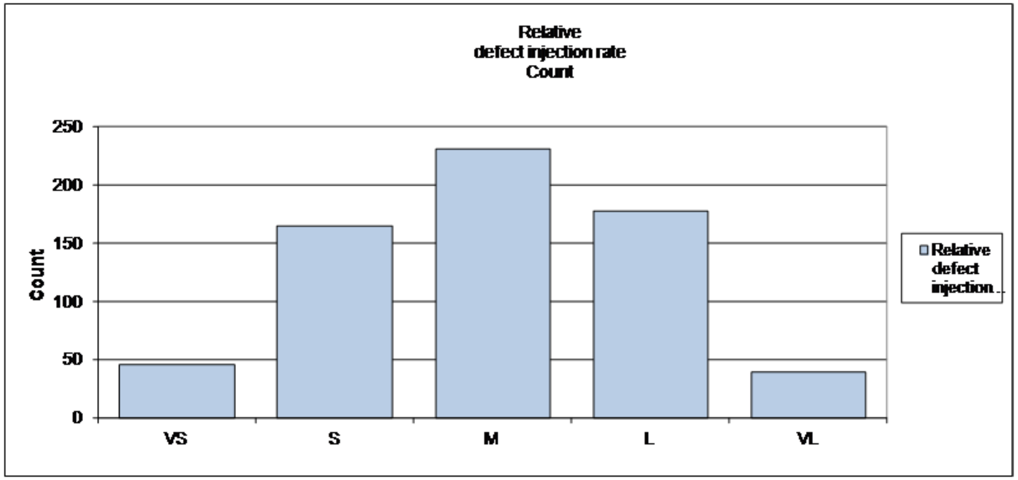
Design Practices (XP – Metaphor, Simple Design, Refactoring; Scrum – Backlog creation, grooming, and sprint story selection)

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

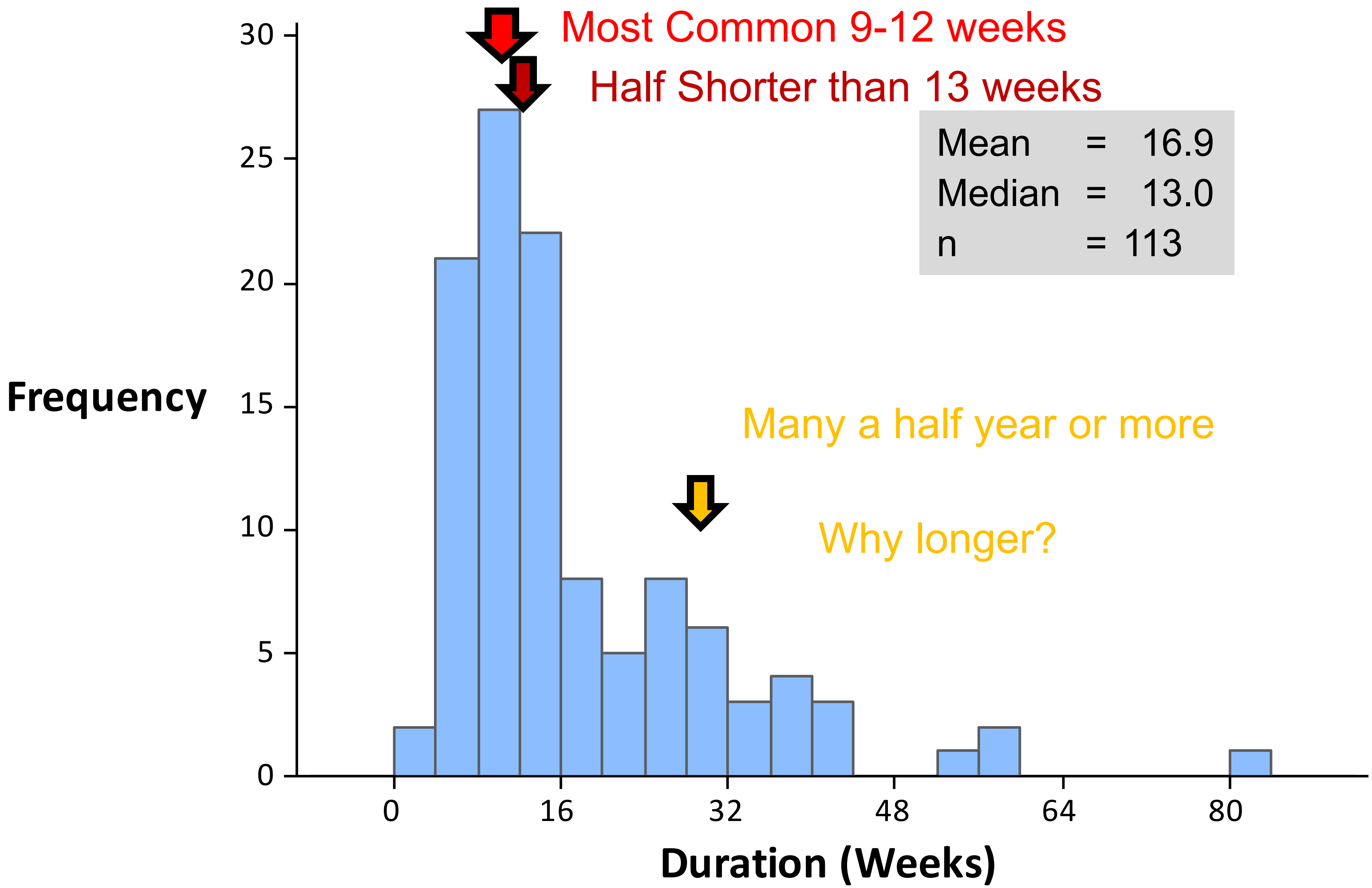
Implementation Practices (XP – Pair Programming, Coding Standards, Testing, 40-Hour Week; Scrum – team choice)

Defect Injection, the most predictable process?

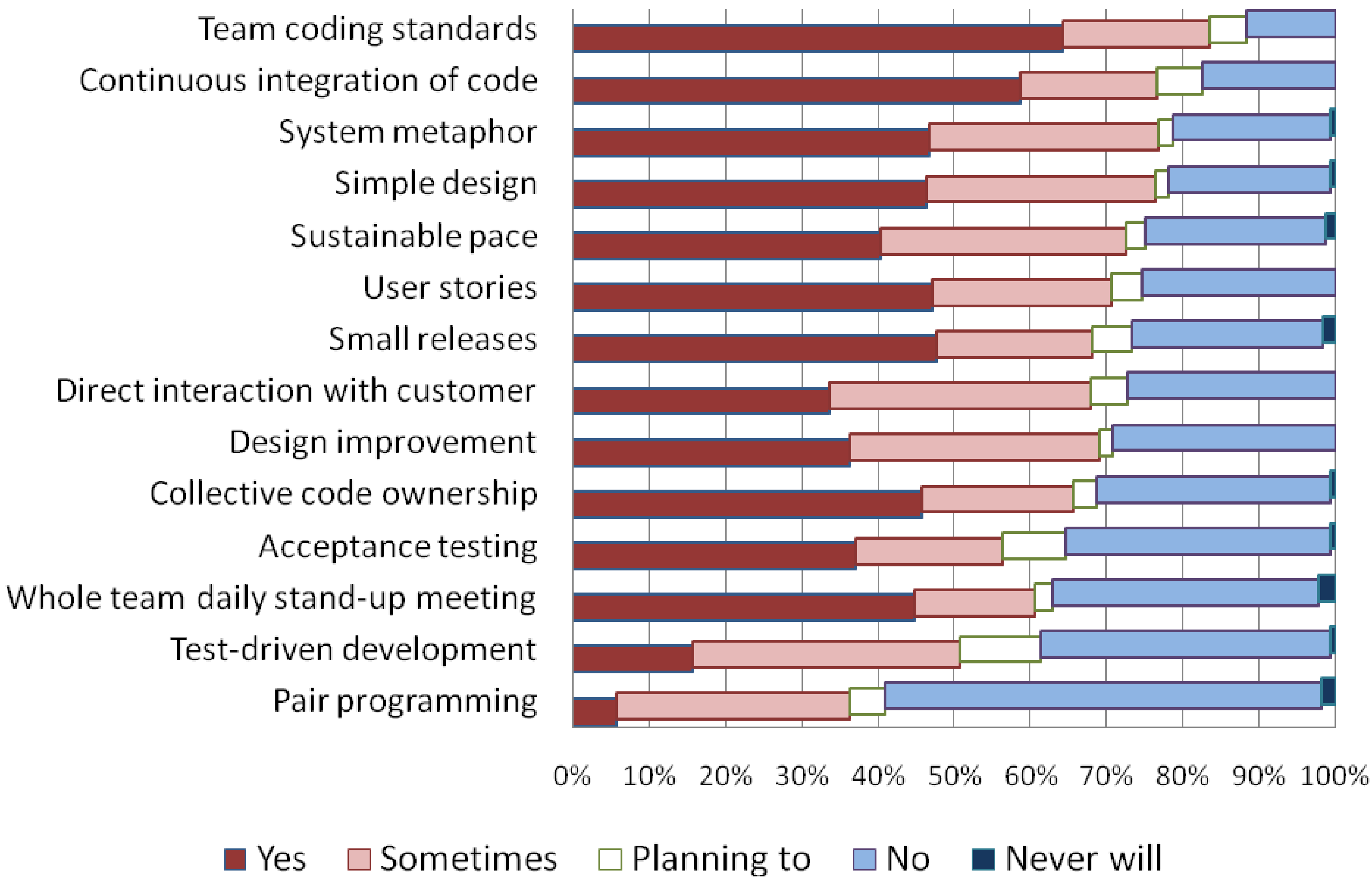




What were Project Durations? [Weeks]



Microsoft – Agile Adoption by Practice



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