Mark Kasunic
Senior Member of the Technical Staff
Software Engineering Institute

Mark Kasunic is a senior member of the technical staff at the Software Engineering Institute (SEI) at Carnegie Mellon University. He is currently a member of the Team Software Process Initiative within the Software Solutions Division. Since joining the SEI in 1994, his work has focused on transitioning performance improvement technologies into practice through applied research, course development, coaching, and training. His current research and development interests include data quality assessment and improvement, project performance measurement, and practical measurement and analysis approaches that help individuals and teams improve their technical performance. Mark has an extensive list of technical publications and conference presentations addressing software engineering and measurement. Before joining the SEI, Mark was an engineer and manager at Boeing in Seattle. He has a Masters Degree in Systems Engineering and is a senior member of IEEE. Mark is a certified TSP Mentor Coach and a certified Scrum Master.
THE LIFE OF A SOFTWARE ENGINEER.

CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.

MUCH LATER...

OH MY. I'VE DONE IT AGAIN, HAVEN'T I?
The Sobering State of Software Engineering

39% of software projects are successful

43% of software projects cost more, take longer, or do less

18% of software projects failed

Another Data Point

In a survey of 166 IT leaders:

- **89%** of projects do not regularly meet their budget
- **59%** projects are typically delivered late
- **33%** state that rework is at least 25% of their budget

2014 IT Leadership Survey - Blueprint Software Systems Inc.
When Measurement Benefits the Measured

Kasunic & Nichols, April 23, 2014

PROBLEM?
Can Measurement Help?

CEO’s have a lower opinion of software groups than of other technical groups due to consistently optimistic estimates, schedule delays, cost overruns, poor quality when delivered, and outright failures. Software is much worse in all of these.

Better measures of projects … will improve the professional status of the software community and perhaps lead to CEO’s having more respect for software groups than they have today.

Capers Jones
InfoQ Interview
March 30, 2014
http://www.infoq.com/articles/Jones-measuring-agile-adoption
The World Without Measurement

Science?

Engineering?

Medicine?
We All Measure ….

Driving?

Clothing Size?

Cooking?

Getting to Work on Time?
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.
First-Order Measurement

Second-Order Measurement

Third-Order Measurement
The State of the Practice

An Issue
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 in Your Work Life

The Team

Individual Software Engineer

The Organization

Software Community at Large
Measurement & the Individual Software Engineer

You
Stellar athletes understand that they must set specific goals to reach their potential.

Measurement provides the necessary feedback that drives athletes to achieve world-class excellence.
Can software engineers leverage goal-setting and measurement in the same way?
Do you?
What type of measures do you *typically* use to estimate the duration of your work for a schedule?

☐ First order: Qualitative - based on what I think I’ve done before.

☐ Second order: Quantitative - based on quantitative data from previous project(s).

☐ Third order: Statistical - based on statistical patterns of data from my previous projects.

☐ None of the above.
Should You Be Using Measurement?

Yes. And it needs to go beyond first order measurement.

Measurement is needed to manage your work.
Managing the Work

But isn’t it the managers job to manage?
The Evolution of the Management Approach

Body Management
People as oxen.

Task Management
People as machines.

Knowledge Management
People as individuals.

Frederick Taylor

Peter Drucker
Evolution of the Worker

Hunter-Gatherer

Farmer & Artisan

Industrial Revolution Worker

Technology Professional
Taylorism – Scientific Management

For years, the basic power equation in organizations was simple and effective:

Knowledge held by a few (the managers), plus iron discipline over the many (the workers).

The worker was viewed as an instrument, a bundle of muscles programmed through instruction.
The Birth of the Knowledge Worker

- New data processing age was born during 2\textsuperscript{nd} half of 20\textsuperscript{th} century
- Work became asynchronous and non-linear
- Nature of knowledge work demanded significant control by the worker (instead of the manager)
What Differentiates Knowledge Work?

**Manual work**
Consists of converting materials from one form to another.
The work output is tangible.

**Knowledge work**
The work is done in the head.
The work can’t be seen.
A Shift In the Locus of Control

This new breed of worker has a new job: converting knowledge into actions that convert information from one form to another.

Because the behaviors of a knowledge worker are primarily private, supervisors cannot supervise.

Due do the nature of knowledge work, it is the worker that has almost total authority in matching methods to the varying job tasks and situations that they encounter.

However, with this reality, there is also a shift in responsibility …
Managing the Work

But isn’t it the managers job to manage?

No.
Management provides goals.
Knowledge workers manage their work.
Controlling Your Own Destiny

To control the way they work, software engineers must plan their projects.

For management to trust these plans, the engineers must make accurate plans.

To make accurate plans, they must have data.

To have data, they must measure their performance.
You Need Data To Manage Yourself

GOOD

Metrics are good.

But …
This Is A Data Collection Fallacy

- **GOOD** → Metrics are good.
- **BETTER** → More would be better.
- **BEST** → Most is best.
Only Four Basic Measures Needed

Software engineers only need to collect four basic measures to manage their schedule performance and the quality of their work.

- Time on Task
- Size
- Defects
- Schedule
Measures Are Estimated and Then Tracked

At the beginning of an effort, the work is planned and divided into a set of tasks or activities called *phases*. The basic measures are estimated.

- product size
- time-in-phase
- defects injected into a phase
- defects found in a phase
- task completion dates

During the project, these measures are collected in real time

- time-in-phase
- defect type injected in phase
- find/fix time for each defect
- task completion dates

When a product has been completed

- product size is measured
No one wants to be measured by others. And, that’s not what we’re talking about here.
Tool Support For Data Collection

Collecting the four core measures would be impractical in the absence of software support.

A number of tools are available that make it easy to collect this type of data.

Collecting Personal Data

Automatically entered into time log.

Click to enter defect into log.
Tracking Your Time

All activities that contribute to the value chain are listed as tasks in your plan. Work against any task in your plan is timed.

- When you begin work on a task, you start the timer in the tool.
- When you stop work on a task, you stop the timer.
- The tool calculates durations automatically.
- If your task is interrupted, you can stop and then restart your timer to resume.
- If you forget to use your timer, you estimate your time-on-task, and enter it manually.
Hi. I'm Joe.
How Am I Spending My Time?

Wow! What happened?
How Am I Spending My Time?

Before

<table>
<thead>
<tr>
<th>Design</th>
<th>Code</th>
<th>Test</th>
</tr>
</thead>
</table>

After

<table>
<thead>
<tr>
<th>Design</th>
<th>Code</th>
<th>Test</th>
</tr>
</thead>
</table>

Incorporate design before beginning code

Savings

Time is Money

Time is Money
Analysis of Data to Improve

Percent Defects Found

Review Rate (Lines of Code per Hour)

[Humphrey 2005, page 192]
Analysis of Data to Improve

Optimal review rate for this SW Engineer: Approx. 100 LOC/Hr.

[Humphrey 2005, page 192]
What About Quality Performance?

In the absence of measurement …

- No feedback loop
- No learning
- Mistakes will happen again
Defect Tracking

Most defects are discovered during personal reviews, inspections, and other quality control activities.
Whenever a defect is found, you open the defect log of the planning/tracking tool and record the following:

• the start time when defect was found
• defect type
• the process phase where the defect was injected
• the process phase where the defect was removed
• a brief description of the defect
• the stop time (when you have completed fixing the defect)
Fix Time by Defect Type

Fix Time (Min.)

- Function: 350 minutes
- Assignment: 20 minutes
- Interface: 5 minutes
- Syntax: 2.5 minutes
- Documentation: 2.5 minutes
- Packaging, Checking, Data, System, Environment: 0 minutes

Defect Type
Improving Review Practices

Use data to update review checklist.

Prevent errors from occurring again and again.
Using Measurement To Understand ...

... and, to get better

- Defects recorded by type and time-to-fix
- Closed-loop feedback
- Learn from mistakes so they don’t happen again

Closed-loop feedback & learning
Taking Responsibility

Others cannot manage how you estimate your work and how you manage the quality of our work.

Knowledge workers manage themselves with data.

Software engineers are knowledge workers.

Only four basic measures are need to manage your work.
Can software engineers leverage goal-setting and measurement the way that star athletes do?

Yes! Absolutely.
Bill Nichols joined the Software Engineering Institute (SEI) in 2006 as a senior member of the technical staff and serves as a Personal Software Process (PSP) instructor and Team Software Process (TSP) Mentor Coach with the TSP Initiative within the Software Solutions Division (SSD). Prior to joining the SEI, Dr. Nichols lead a software development team at the Bettis Laboratory near Pittsburgh, Pennsylvania, where he had been developing and maintaining nuclear engineering and scientific software for 14 years. His TSP publications include the the PSP and TSP Bodies of Knowledge, The TSP Coach Mentoring Program Guidebook, and various publications addressing software quality planning. Research publications include an algorithm for use in neutron diffusion programs, design and performance of a physics data acquisition system, and experimental results in particle physics. He has a doctorate in physics from Carnegie Mellon University.
Measurement On Your Team

The Team
Self-Managed Team of Knowledge Workers

Traditional Project
The PM plans, directs, and tracks the work.

Self-Managed Team
The Team directs, and tracks the work.
Self-Managed Teams Plan Their Work

Management provides the goals and constraints for the project.

The team then develops its plan for meeting management’s objectives.

If necessary, the team negotiates with management to arrive at a mutually agreeable outcome.

But it’s up to the team to manage their work! Not someone else.

The team must have data to manage the work, to meet their commitments.

Measurement helps teams manage their commitments.
Measurement To Benefit the Measured

Measurement feedback to the team

Personal Reviews
- Software code

Unit Tests
- Software code

Inspection
- Software code

System Test
- Software code

Team

date defect found

# defects
Measurement Used to Manage

- **Plan**: Objectives, Product Versions, Data
- **Execute & Measure**: Weekly Review
- **Refine & Measure**: Adjustments made to plan based on feedback

**Cycle**:
- **Execute**
- **Measure**
- **Weekly Review**

The data is used throughout to assess performance.

Adjustments are made to the plan based on measurement feedback.
Planning, Doing, and Learning

The Project

- **Cycle launch**
- **Postmortem or retrospective**
- **Cycle (a.k.a., iteration or sprint)**
- **Weekly objectives**
- **Two-three tasks per team member**

Release 1

Cycle 1

Cycle 2

Release 2

Cycle 3

Cycle 4

Cycle 5

Release n

Cycle 6

Cycle 7

Cycle n

© 2014 Carnegie Mellon University
Comparing Estimates to Actuals

For both schedule and quality …

Is the project on track?
Comparing Estimates to Actuals

For both schedule and quality …

Closed loop feedback

Is the project on track?
Comparing Estimates to Actuals

For both schedule and quality …

Closed loop feedback
Learning

Is the project on track?
Comparing Estimates to Actuals

For both schedule and quality …

Closed loop feedback
Learning
Performance improvement

Estimates

Actuals

Is the project on track?
Measurement That Benefits the Measured

- Project and Product Objectives
- Refined Objectives
- Postmortem (Retrospective) Lessons Learned
- Team Planning or Replanning
- Data
- Product Versions
- Plan
- Cycle
  - Execute
  - Measure
  - Weekly Review
  - Team Planning or Replanning
  - Postmortem (Retrospective) Lessons Learned
  - Data
  - Product Versions
  - Plan
  - Cycle
Again ... Only Four Basic Measures Needed

Software engineers only need to collect four basic measures to manage their schedule performance and the quality of their work.

- Time on Task
- Defects
- Size
- Schedule
Derived From the Four Basic Measures

Many other useful measures and indicators can be easily derived from the four basic measures including:

- estimation accuracy*
- prediction intervals*
- time in phase distribution
- defect injection distribution
- defect removal distribution
- productivity
- reuse percentage
- cost performance index
- planned value
- earned value
- predicted earned value
- defect density
- defect density by phase
- defect removal rate by phase
- defect removal leverage
- review rates
- process yield
- phase yield
- failure cost of quality (COQ)
- appraisal COQ
- appraisal/failure COQ ratio

* Both size and time
How Do You Know If It’s a Best Practice?

Organizations want a way to gauge their performance and to compare their performance with others in their industry.

Data on project performance is needed to provide evidence of what (exactly) constitutes a best practice.

How do you even know what a best practice is unless you measure and compare it to other practices?

Benchmarks provide a reference point for interpreting performance.
Benchmarking & Best Practices

You can’t benchmark without measurement!

We are here.  But we want to be here.

How were those projects able to achieve such great performance?

Performance Report

SW Engineer
Description of the Data

The data was submitted to the SEI between 2000 and 2012.

The source data is from 93 projects in the United States and 20 projects from Mexico.

This is data that has been aggregated at the **team** level at the time of a cycle postmortem (retrospective).

Only data from a project’s last postmortem is included.

Tests were conducted to ensure that extracted data represented unique projects.
How Long Were Project Durations? [Weeks]

Mean = 16.9
Median = 13.0
n = 113
What were Project Durations? [Weeks]

Most Common 9-12 weeks

Mean = 16.9
Median = 13.0
n = 113
What were Project Durations? [Weeks]

Most Common 9-12 weeks
Half Shorter than 13 weeks

Mean = 16.9
Median = 13.0
n = 113

Frequency

Duration (Weeks)
What were Project Durations? [Weeks]

- Most Common 9-12 weeks
- Half Shorter than 13 weeks
- Many a half year or more

Frequency

<table>
<thead>
<tr>
<th>Duration (Weeks)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>48</td>
<td>5</td>
</tr>
<tr>
<td>64</td>
<td>3</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
</tr>
</tbody>
</table>

Mean = 16.9
Median = 13.0
n = 113
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?
Projects come in a variety of sizes.

Mean = 11.7
Median = 6.6
n = 112
Projects come in a variety of sizes

- Mean = 11.7
- Median = 6.6
- n = 112

Size - Actual Added and Modified
Thousand Source Lines of Code [KLOC]
Size - Actual Added and Modified
Thousand Source Lines of Code [KLOC]

But how much code a team can write

Mean = 11.7
Median = 6.6
n = 112
Team Size

But how much code a team can write also depends upon team size.

<table>
<thead>
<tr>
<th>Team Size</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
</tr>
</tbody>
</table>

Mean = 8.2
Median = 8.0
n = 111
Are these larger teams productive?

Mean = 8.2
Median = 8.0
n = 111
Are these larger teams productive? How would we measure?

Team Size

Frequency

<table>
<thead>
<tr>
<th>Team Size</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
</tr>
</tbody>
</table>

Mean = 8.2
Median = 8.0
n = 111

When Measurement Benefits the Measured
Kasunic & Nichols, April 23, 2014
© 2014 Carnegie Mellon University
Code Production Rate

Are these larger teams productive?
How would we measure?
Derived rate measure

Productivity (LOC/Hr)

Frequency

Mean = 10.3
Median = 7.1
n = 112

Mean = 10.3
Median = 7.1
n = 112
What is the relationship between team size?
And production rates?

n = 111

Productivity (LOC/Hr)

Team Size

R² = 0.0297
When Measurement Benefits the Measured

Kasunic & Nichols, April 23, 2014

© 2014 Carnegie Mellon University

Team Size vs. Productivity

Productivity (LOC/Hr)

Team Size

n = 111

A few very high rates with smaller team size

R² = 0.0297

n = 111
Team Size vs. Productivity

- Overall, weak relationship
- Slope is not a useful predictor

$k^2 = 0.0297$

$n = 111$
Team Size vs. Productivity

What else might matter?

Productivity (LOC/Hr)

n = 111

R² = 0.0297
What do you think is the average number of weekly task hours that teams are able to accomplish?
What do you think is the average number of *task hours* that a team member spends during a week?

- [ ] 35 to 40 hours
- [ ] 30 to 35 hours
- [ ] 25 to 30 hours
- [ ] 20 to 25 hours
- [ ] 15 to 20 hours
- [ ] less than 15 hours
Mean Team Member Weekly Task Hours

Mean = 10.3
Median = 9.0
n = 111
Mean Team Member Weekly Task Hours

Mean Team Member Weekly Task Hours

Mean = 10.3
Median = 9.0
n = 111

Depends on the project
A Lot
Plan Vs. Actual Hours

Can you plan the hours?

R\textsuperscript{2} = 0.952

\begin{align*}
\text{Actual Hours} \quad \text{Plan vs. Actual Hours} \\
\text{Planned Hours}
\end{align*}

n = 113
Plan Vs. Actual Hours

Can you plan the hours?

$R^2 = 0.952$

YES! If you use your data!

Actual Hours

Planned Hours

$n = 113$
Plan Vs. Actual Hours

Can you plan the hours?

YES! If you use your data!

Actual Hours are an important factor in total production.

Actual Hours vs. Planned Hours

\[ R^2 = 0.952 \]

Can you plan the hours?

YES! If you use your data!

Actual Hours are an important factor in total production.

n = 113
Let’s Looks at Some Quality-Based Profiles
Total Defects Injected Per KLOC

Mean = 54.7
Median = 31.5
n = 79
Total Defects Injected Per KLOC

To Error is human.

Mean = 54.7
Median = 31.5
n = 79
Defects are inevitable, and predictable.

To **error** is human.

- Mean = 54.7
- Median = 31.5
- n = 79
Total Defects Injected Per KLOC

To Error is human.

Mean = 54.7
Median = 31.5
n = 79

Defects are inevitable, and predictable.

Getting them out requires a plan.

Use your data!
Injection and Removal of Defects

Defect Injection phase

Intermediate product with defects

Defect removal phase

Requirements, design, code

Reviews, inspections, tests

Phase Yield (% defects removed)
Multiple defect removal filters are required to realize high-quality software.

- Personal reviews
- Inspections
- Tests
Defect Density - Summary
Defect Density - Summary

Remove Design Defects
Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects
Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Test Removes Other Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Test Removes Other Defects

Defect Density - Summary

Remove Design Defects

Remove Code Defects

Remove Code Defects

Test Removes Other Defects
Defect Removal Density – Median of Defects Per KLOC

How do you know if you are on track?

- DLD Review: 2.2
- Code Review: 5.2
- Code Inspection: 3.3
- Unit Test: 3.8
- Build/Integration Test: 0.7
- System Test: 0.15

Defects Per KLOC (Median)

Guidelines:
- < 5
- < 0.5
- 0.2
All project performance charts are available as a download with today’s webinar.

These include charts not presented in this webinar.
In God we trust ...
All others bring Data

W. Edwards Deming
Contact Information

Mark Kasunic
Senior Member of Technical Staff
Software Engineering & Acquisition Practices
Software Solutions Division
Telephone:  +1 412-268-5863
Email:  info@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone:  +1 412-268-5800
SEI Phone:  +1 412-268-5800
SEI Fax:  +1 412-268-6257