SEI WEBINAR SERIES | Keeping you informed of the latest solutions

Copyright 2017 Carnegie Mellon University

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.

DM17-0593

Agile Metrics:

Three Secrets to Success

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Agile In Government

Bottom Line Up Front

1. Exercise Due Care

- The level of discipline and rigor applied must match the context served by the work
- Metrics give voice to things we want to hear about, we are responsible to choose
- Some very important things will lack high-resolution measures to inform us

2. Consider Systems' Perspectives

- A scrum team is its own system, and rich metrics to serve the team exist
- The enterprise consists of many other systems, which bring different perspectives
- Boundaries of generalizability exist among these systems

3. (Ruthlessly) Automate Basic Indicators and Analyses

- Wield tools in service of your needs, and do not limit the sphere of focus artificially
- Make metrics routine and boring – not episodic and authority-focused
- Tool chains and visualization techniques offer new opportunities

A Familiar Problem

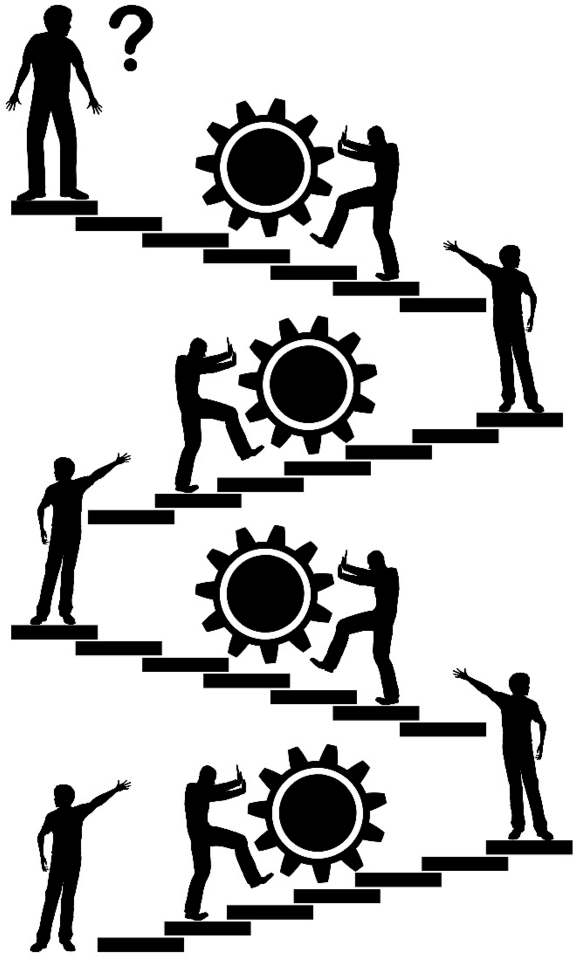
Data can shine a light on important things.

If we don’t focus on the right thing, we won’t get what we need.

Due Care is context-dependent, and should not be left up to the advocate of a particular methodology.

Multiple Intersecting Systems

Barriers to Automation



Metrics often focus exclusively on:

- Appeasing an authority role
- Demonstrating competence
- Validating the chosen path

This may engender trust concerns, and often conflicts with the concept of an empirical process – one where we learn from looking at facts that inform tactical/strategic options.

Polling Question #2

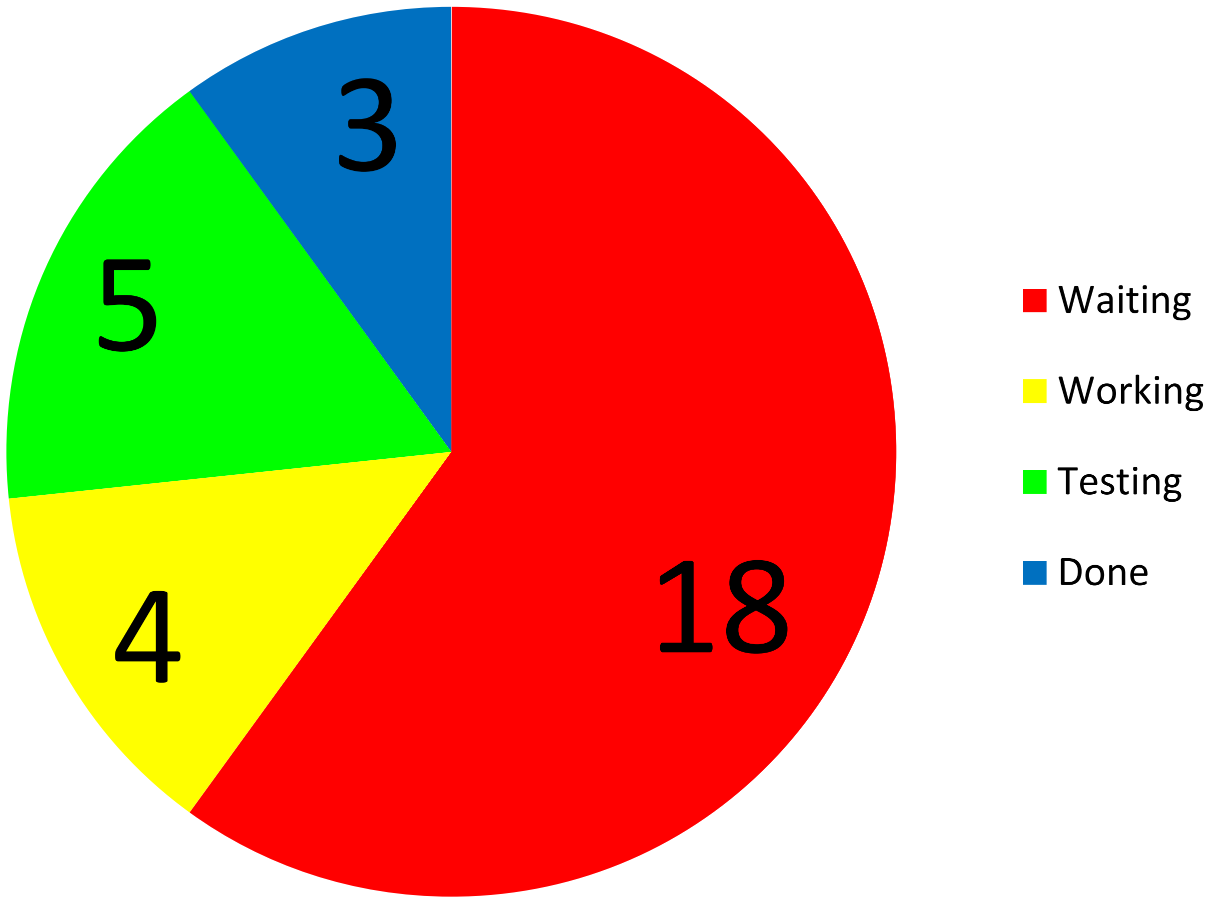
Your Role

1. Government employee working in a program office
2. Contractor working in a government program office
3. Employee of a firm serving a government program
4. Employee of a firm doing commercial work
5. Coach/Advisor/Consultant for government
6. Coach/Advisor/Consultant for industry
7. None of the above

Taking a Deterministic View

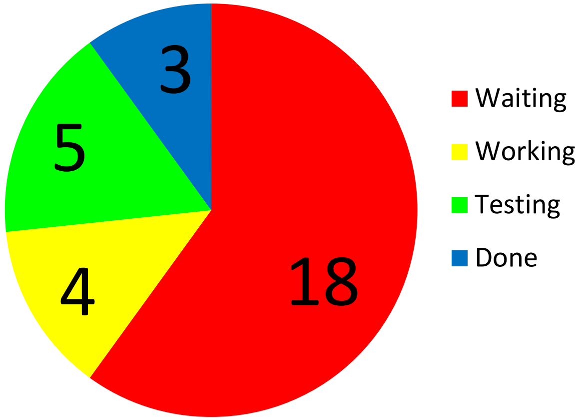
Three Numeric Examples

Basic Example



IT Modernization

IT Modernization Example



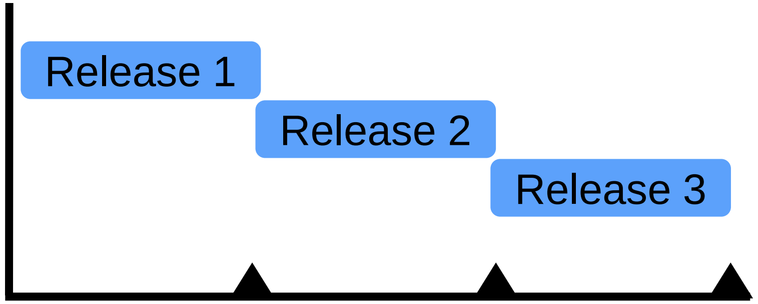
These are 30 *RICE* objects* that define the scope of work for one or our vendors.

They will be folded into a series of three releases, which will integrate the work of multiple vendors.

		Size Breakdown			*Planned Release*		
Object Type	*Count*	*L*	*M*	*S*	*R1*	*R2*	*R3*
Reports	3		2	1	1	1	1
Interfaces	4		4		2	2	
Conversions	3	1	1	1	3		
Enhancements	20	12	5	3	2	6	12

* note: CEMLI might be more familiar for those in this domain. RICE was chosen for the sake of brevity…

Managing Three Planned Releases



Common Focus for Metrics

- Size
- Effort
- Quality

Goal:

- Predict release performance

Questions:

- Is the work larger/smaller than estimated?
- Is the work taking more/less effort than we estimated?
- Will the quality of the delivered products be acceptable?

Metrics:

- Estimated vs. actual effort
- Planned vs. delivered products
- Estimated vs. actual size of products
- Defect counts and profiles
- Measures of performance

Understanding Benefit of IT Modernization

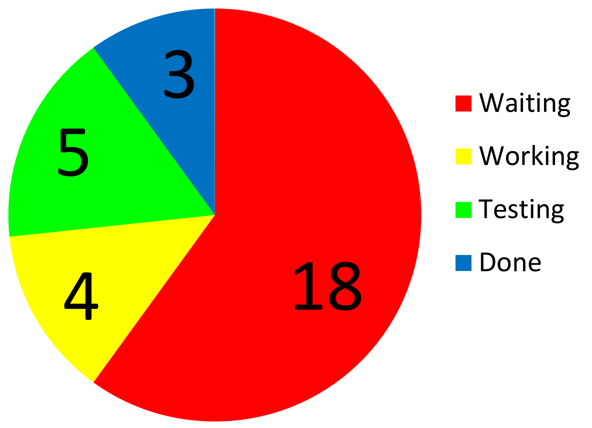
What combination of choices leads to improvements in things like:

- Amount of exception-handling
- Users finding the correct path through the system on the first try
- User migration to a new system

Can we iterate and experiment with functional changes as well as technological changes, to improve performance of the IT-enabled service?

Sustaining Embedded Systems

Sustaining Embedded Systems Example



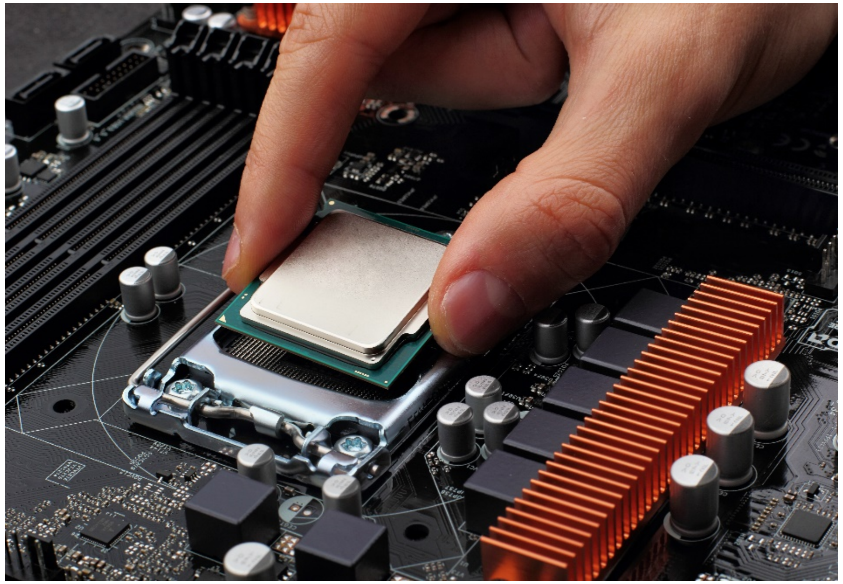
These are 30 ***Must-Fix Defects*** which limit the operational utility of the system in the field today.

There is a strategy for patching the fielded system based on logical groupings of the defects.

Sample of Fields in the Defect Database

Name	Description
FindActivity	lifecycle or mission activity that uncovered the defect
FindDateTime	date and time when the defect was discovered
TestID	If found in test, the ID# of the test that exposed the defect
FeatureBlocked	user capability that does not function due to the defect
SysComponent	configuration item or other component containing the defect

Fixing Fielded Defects



Common Focus for Metrics

- Cycle Time per Fix
- System Availability/Function
- Quality

Goal:

- Timely resolution of known defects

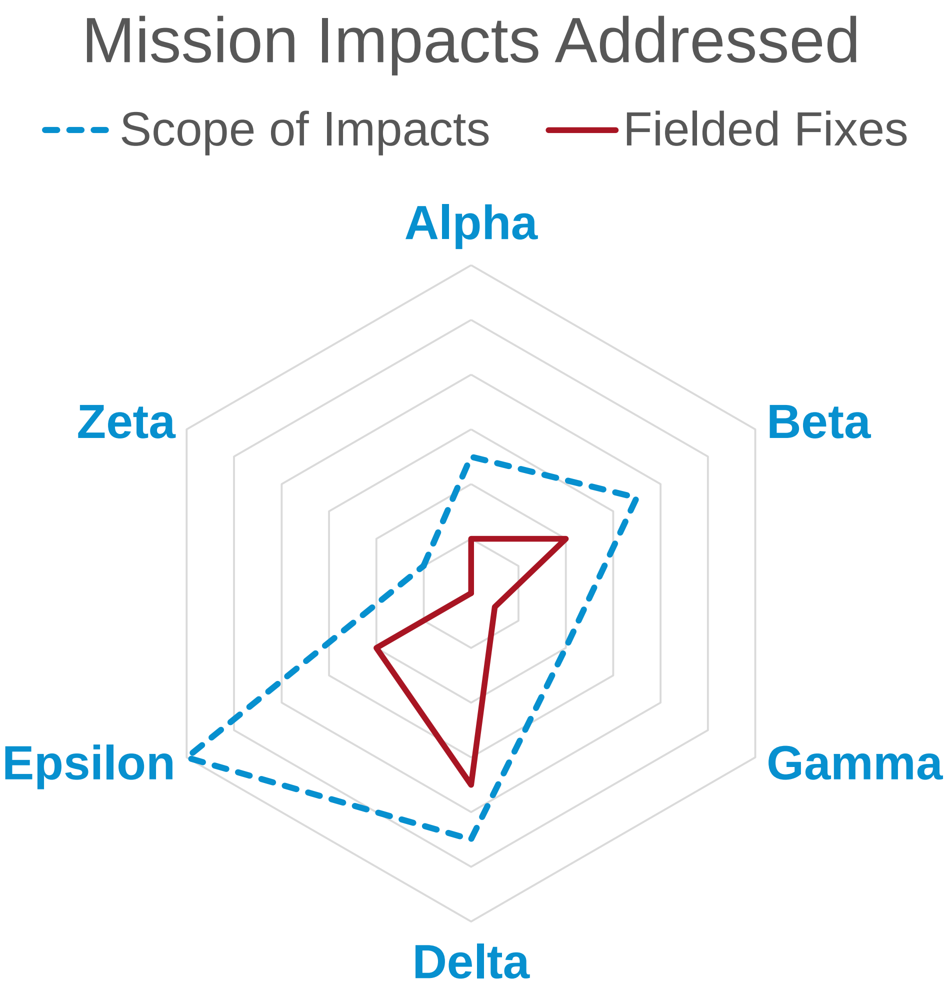
Questions:

- How many defects remain to fix?
- How many defects have been fixed?
- How many fixes have been deployed?
- How many fixes had to be redone?
- How fast are we fixing things?
- What functionality remains blocked?

Metrics:

- Tally of defects remaining/fixed
- Number of fixes per month
- First pass fix rate
- System down time
- Revenue/mission loss due to quality

Enabling Mission Threads with DR Fixes



The impact of fixing defects is charted for six (6) mission threads.

Looking at the area inside the blue dotted line:

- Epsilon has the greatest number of DR impacts
- Zeta has the lowest

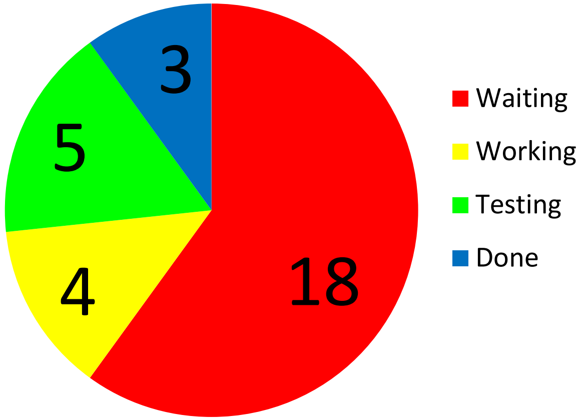
Looking at the area inside the red line:

- Fielded fixes have benefitted Delta the most
- Zeta the least

DR = Deficiency Report

R&D Pathfinder Projects

R&D Pathfinder Projects Example



These are 30 ***requirements*** to meet in order to establish a proof of concept for a new product offering.

A prototype satisfying most, if not all, of the requirements will be used to assess the potential market for the concept.

ID#	Priority	Requirement Text	Success Criteria
1	H	… text statements	… text statements
2	H	… text statements	… text statements
3	M	… text statements	… text statements
…	…	…	…
30	L	… text statements	… text statements

Building a Proof of Concept

Common Focus for Metrics

- Requirements Satisfaction
- Test Cases Passed/Failed
- Technical Performance Attributes

Goal:

- Effective demonstration of capability

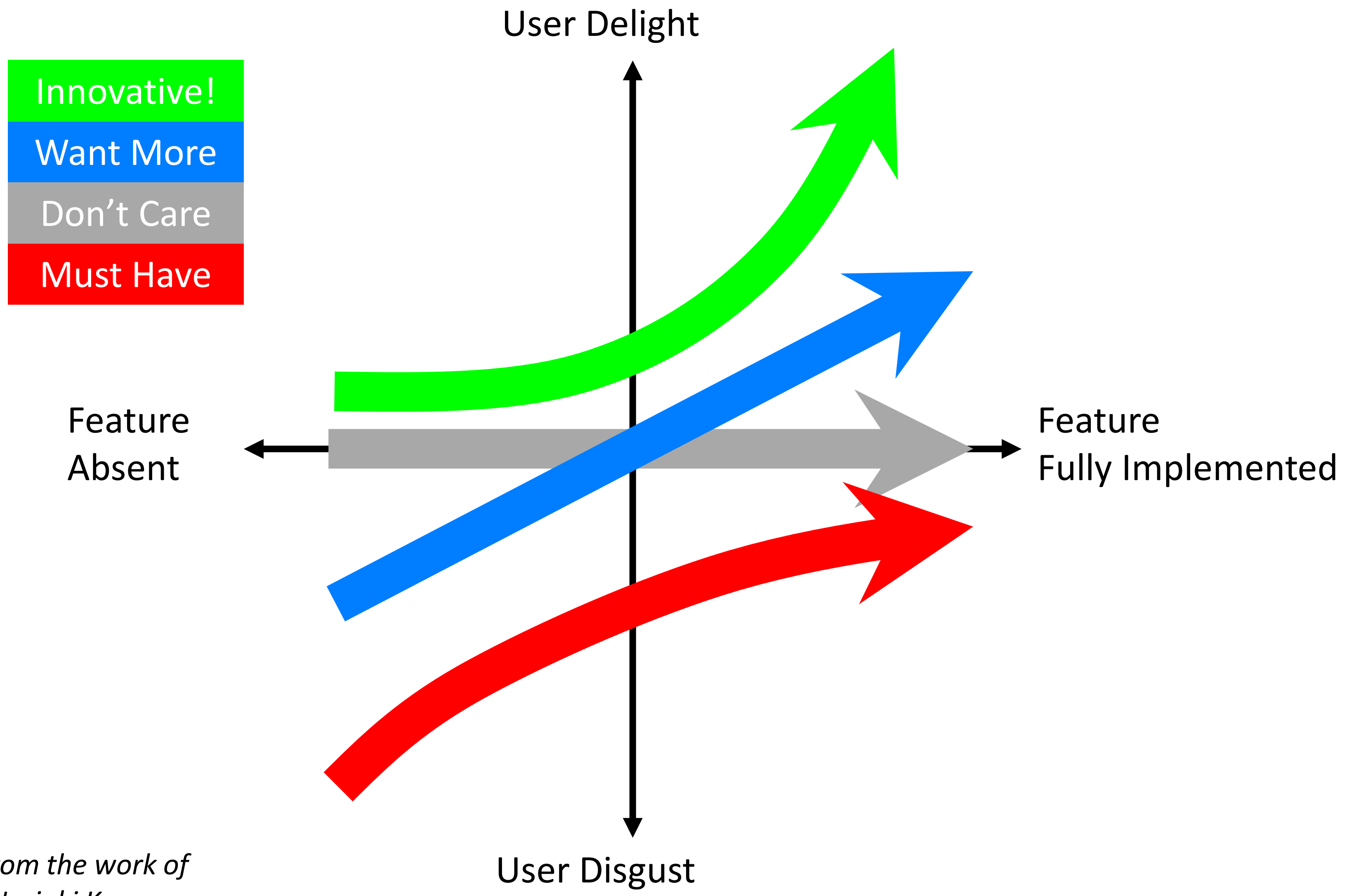
Questions:

- Is each requirement achievable?
- Which are the most challenging?
- How confident can we be about production feasibility?
- What are the bases for estimating total lifecycle cost for this product?

Metrics:

- Count (or %) of objectives achieved
- Number of business case questions answered
- Effort expended

Understanding User Value with KANO Analysis*



** Adapted from the work of Professor Noriaki Kano*

Polling Question #3

Which of the examples is the best match for your context?

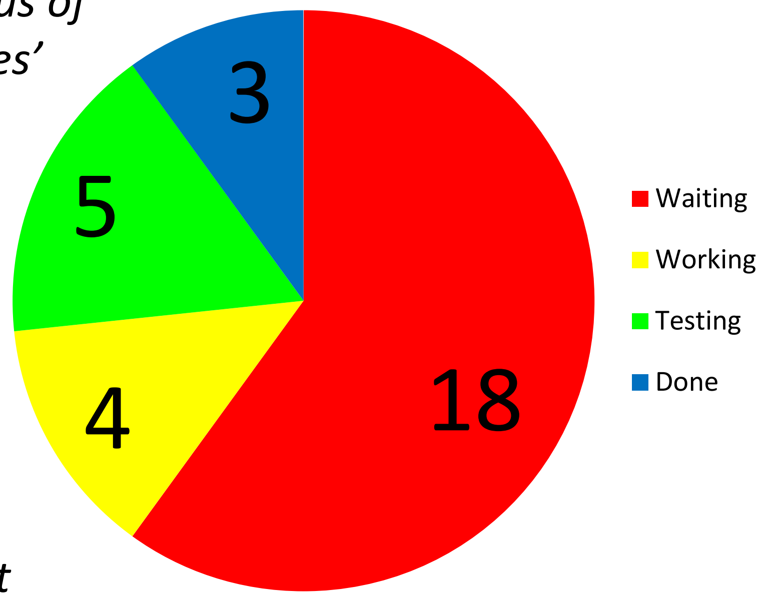
1. IT Modernization
2. Sustaining Embedded Systems
3. R&D Pathfinder Projects
4. More than one of the above
5. None of the above

Flow Metrics Examples

Cumulative Flow Diagram

Constructing a Cumulative Flow Diagram[1]

Here we have a Pie Chart showing the status of 30 ‘work packages’

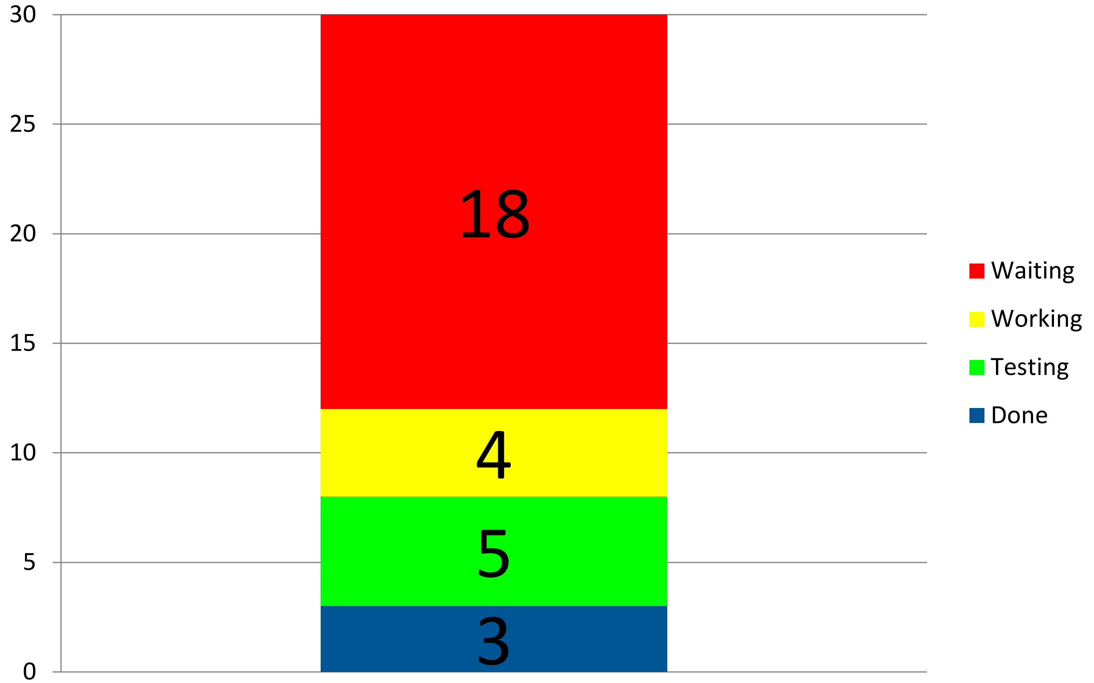


This is a snapshot for a single point in time.

Constructing a Cumulative Flow Diagram$_2$

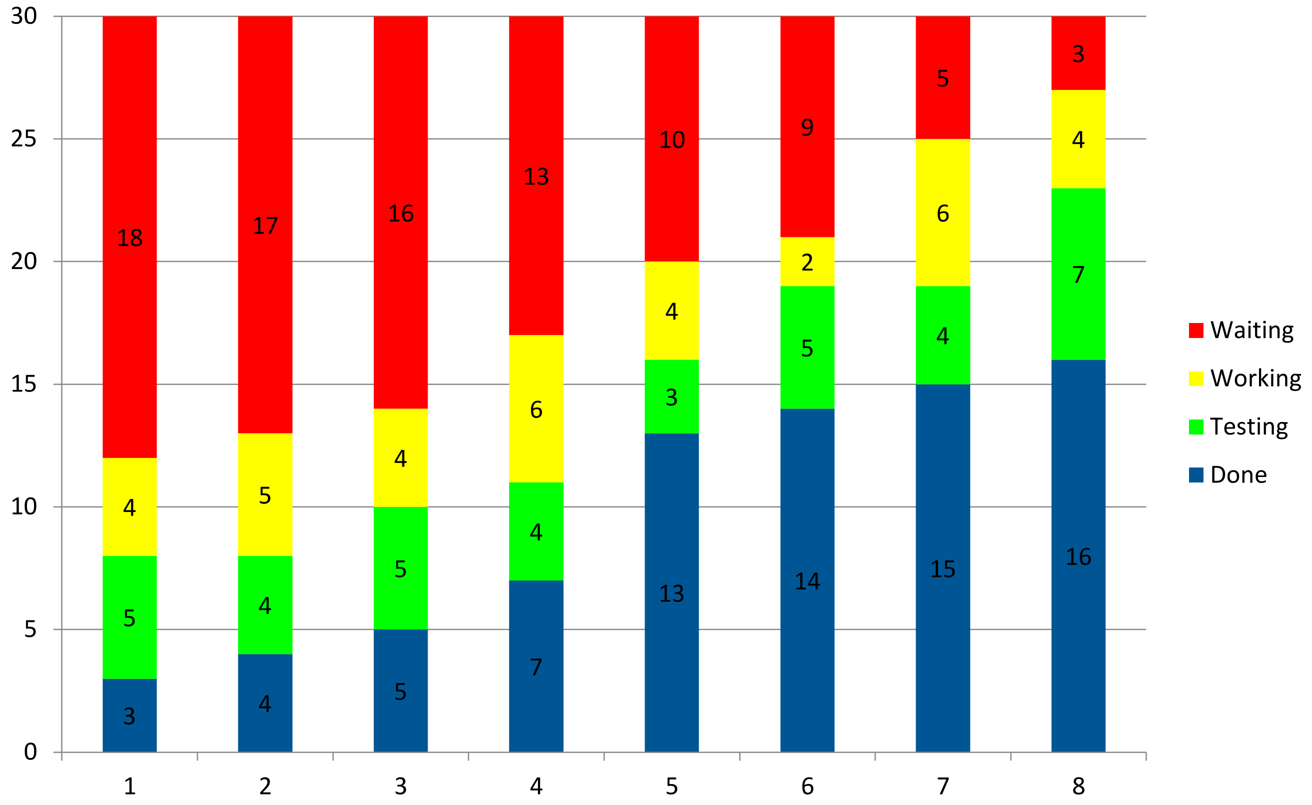
Same data, but presented in a stacked column chart

For a single point in time.



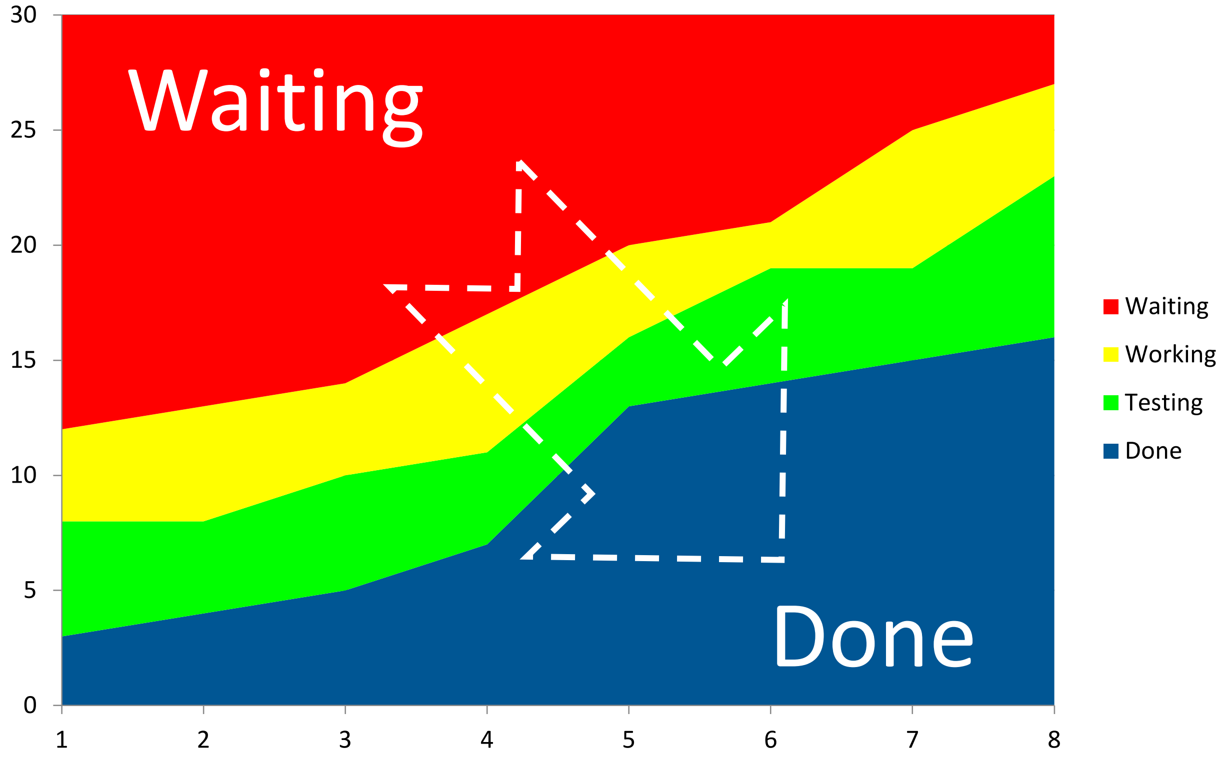
Constructing a Cumulative Flow Diagram$_3$

… adding the next 7 times



Constructing a Cumulative Flow Diagram$_4$

… now we are looking at the flow from "Waiting" to "Done"…
This view starts to show patterns a little easier…

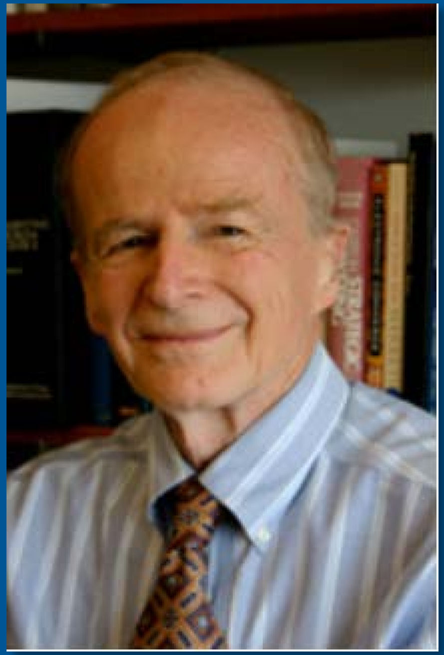


Theoretical Basis

Little’s Law

$$L = \lambda W$$

…the long-term average number L of customers in a stationary system is equal to the long-term average effective arrival rate λ multiplied by the average time W that a customer spends in the system…



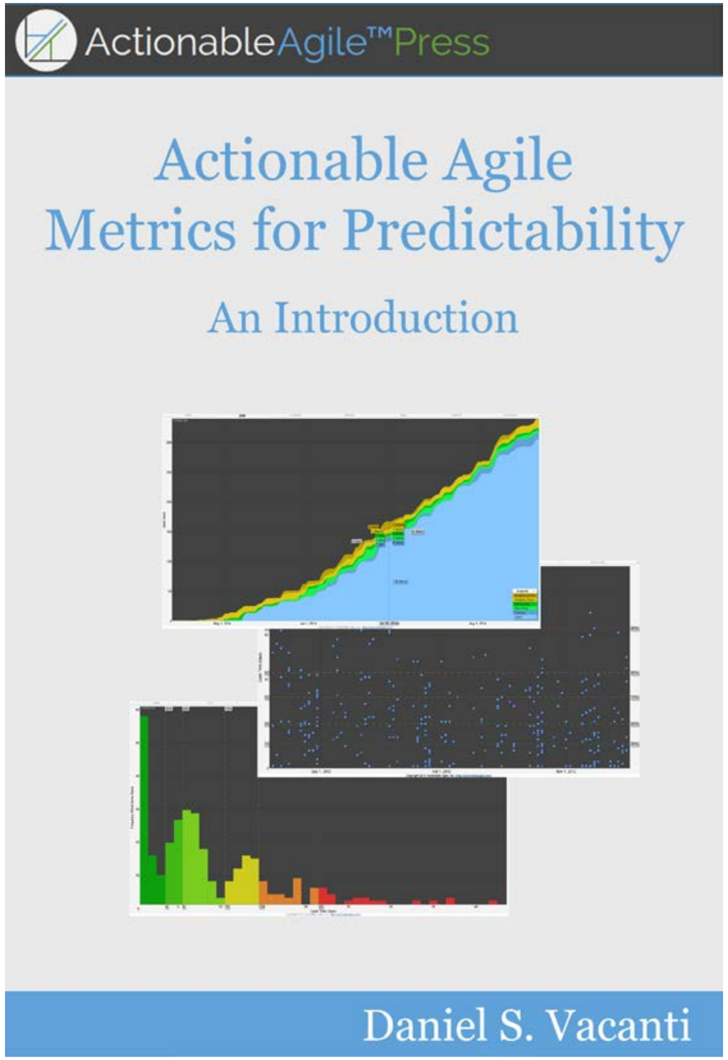
http://mitsloan.mit.edu/faculty-and-research/faculty-directory/detail/?id=41432

Little's Law in Agile Metrics

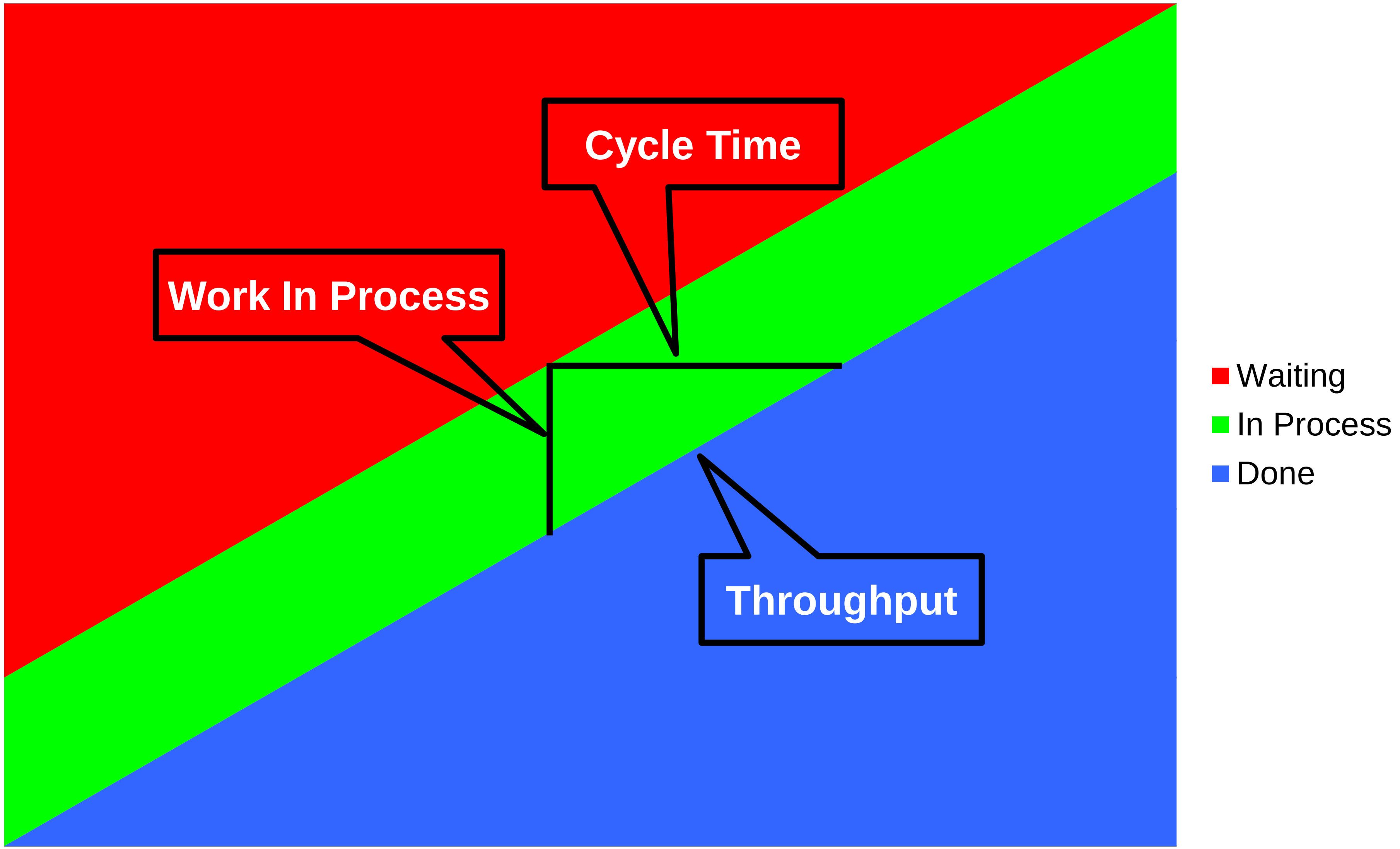
Three Metrics Emphasized*:

1. **Work In Progress** (the number of items that we are working on at any given time),
2. **Cycle Time** (how long it takes each of those items to get through our process), and
3. **Throughput** (how many of those items complete per unit of time).

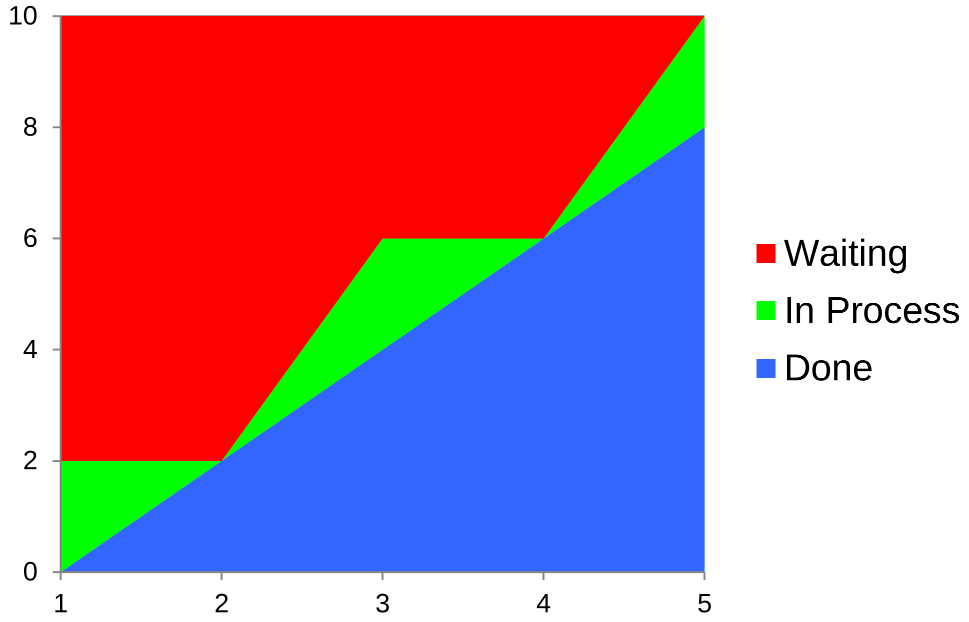
** Excerpted from page 13 of the book depicted on the right.*

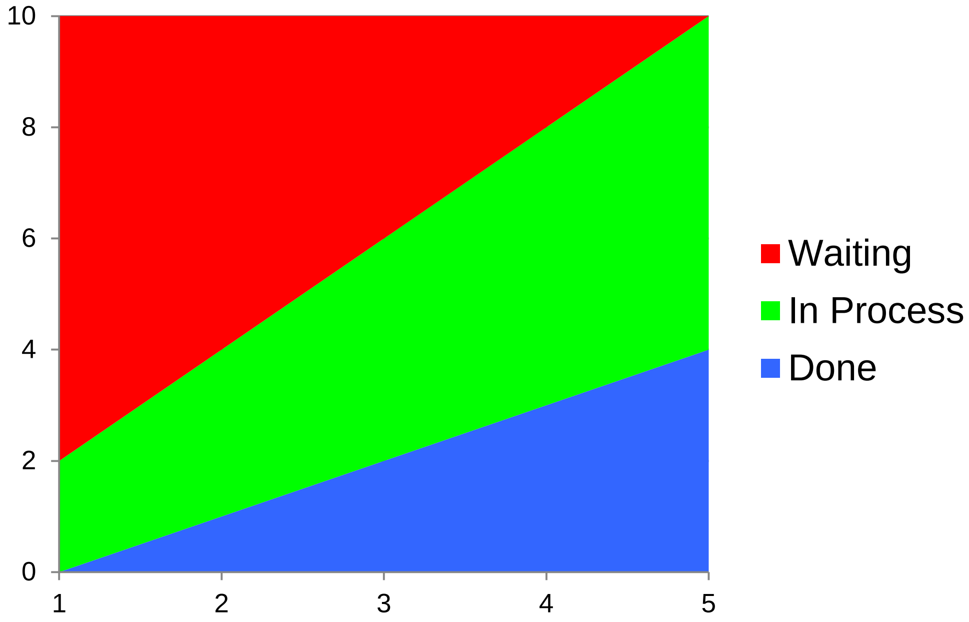


Utility of Little's Law

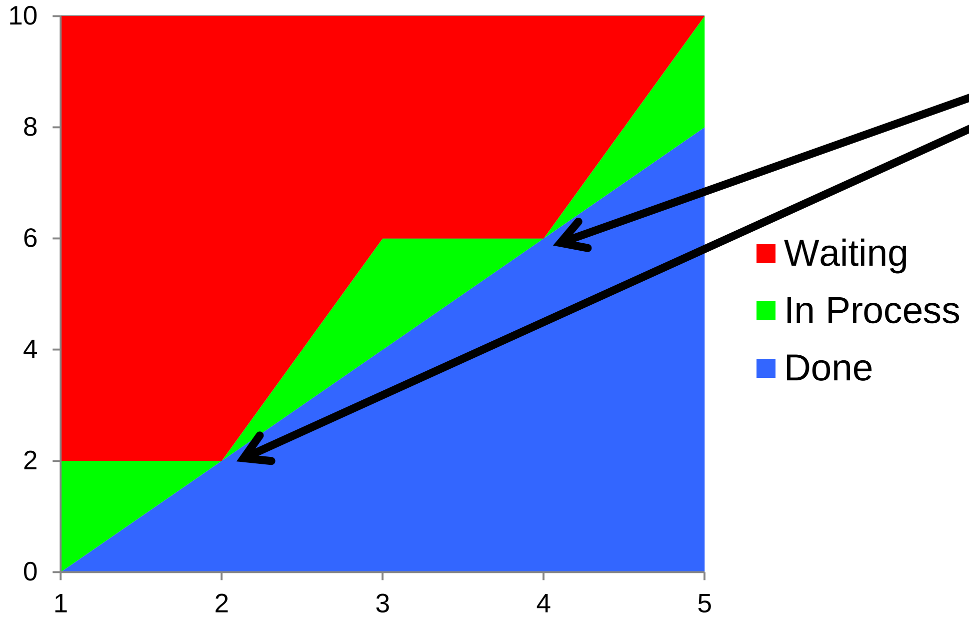


Exercise: What is Going on Here?





Exercise: What *MIGHT BE* Happening$_1$



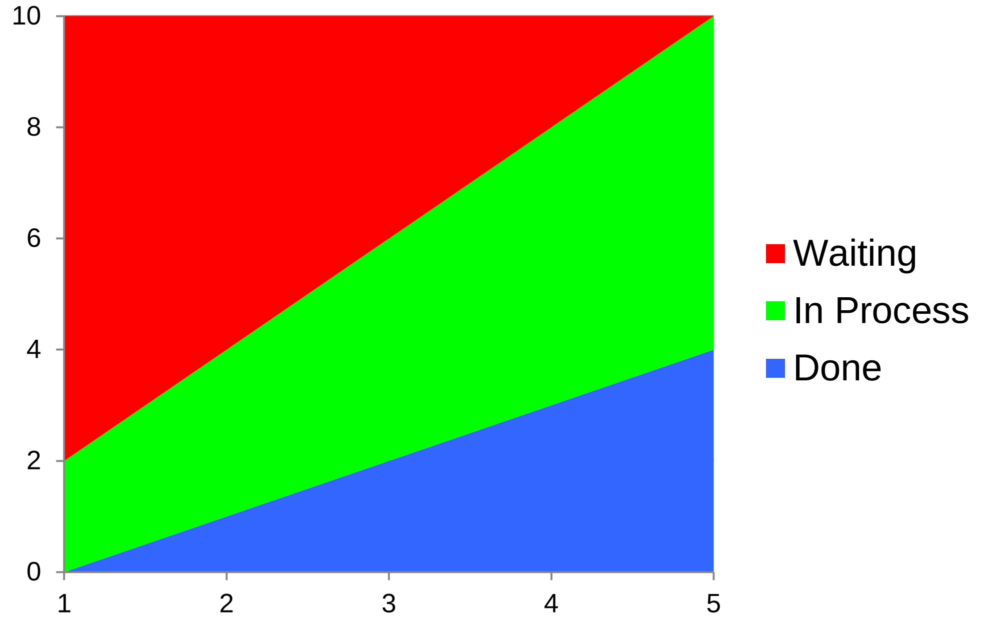
At time 2, and then again at time 4, the number of items “In Process” goes to zero.

- Have we lost the resource(s) performing the work due to rework demands from elsewhere?
- Is this intentional scheduling of work to occur only during time periods 1, 3, and 5?

Exercise: What *MIGHT BE* Happening$_2$

The number of items that are “In Process” is growing over time.

- The rate at which things enter “In Process” is greater than the rate at which things leave “In Process.”
- Are people moving onto new items without completing their work?
- Are new resources being added, who start new work at each time period?
- Are things moving into the “Done” state quickly enough?



Polling Question #4

Cumulative Flow Diagrams and Little’s Law – Your Opinion

1. Interested and would like to learn more
2. That’s enough information for me, thanks
3. Not sure how to answer right now…

Cumulative Flow Diagrams – Beyond Basics

Vacanti elaborates on Little's Law and "Flow Debt*" using CFDs.

Hyman Minski popularized these terms for types of debtors:

- *Hedge,*
- *Speculative, and*
- *Ponzi.*

Patterns of flow can help you identify which category best describes the prevalent decision making style in your project.

Ever been on a project that was trying to do so many things that none of them ever got finished? Is that a Ponzi project?

* Page 144

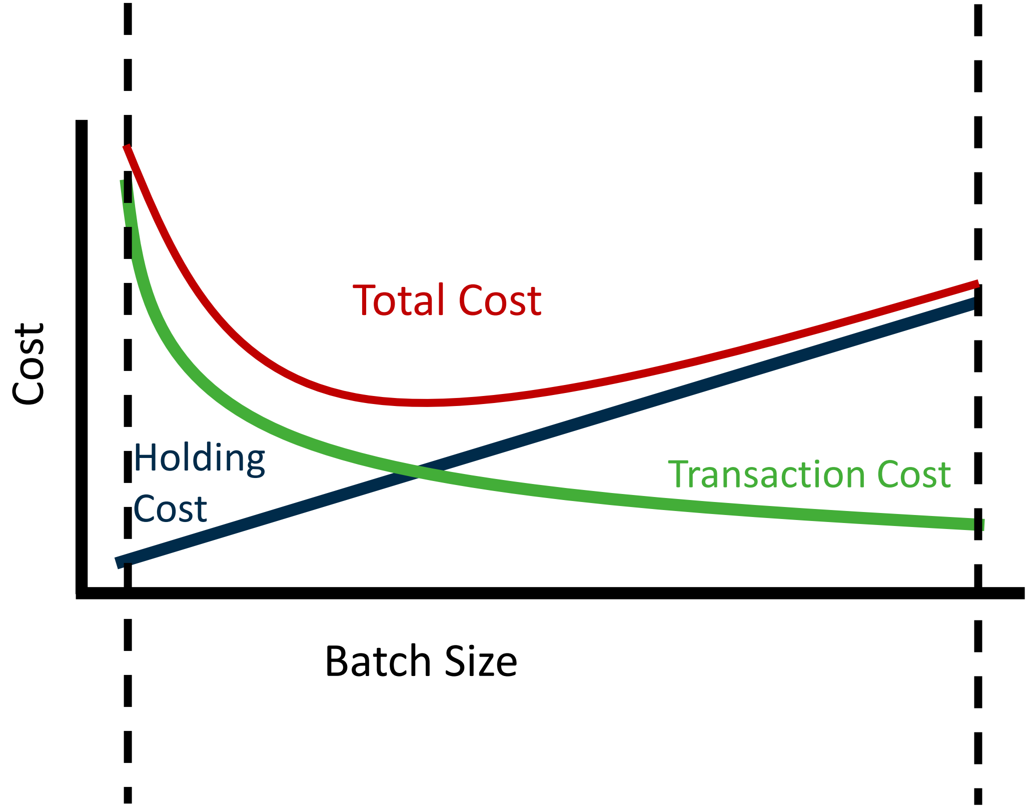
Influence on Modern Agile Practice

Lean Economics

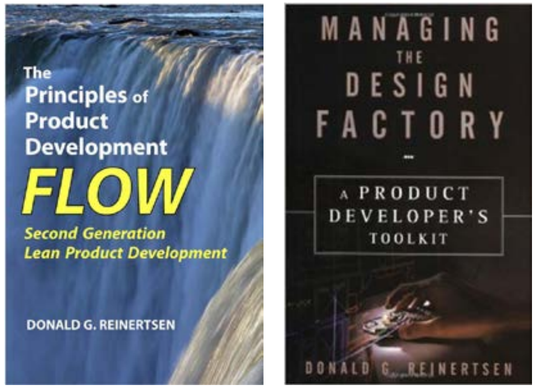
Economies of Batch Size

Specify, build test & ship a **SINGLE** requirement

Specify, then implement, then test & then ship **ALL** requirements



U-Curve optimization problem as described in *Principles of Product Development Flow*, by Don Reinertsen



Metrics for Flow-based Product Development

Queues

- Design-in-Process Inventory
- Queue Size
- Trends in Queue Size
- Cost of Queues
- Aging of Items in Queues

Batch Size

- Batch Size
- Trends in Batch Size
- Transaction Cost per Batch
- Trends in Transaction Cost

Cadence

- Processes Using Cadence
- Trends in Cadence

Capacity Utilization

- Capacity Utilization Rate

Feedback

- Feedback Speed
- Decision Cycle Time
- Aging of Problems

Flexibility

- Breadth of Skill Sets
- Number of Multipurpose Resources
- Number of Processes with Alternate Routes

Flow

- Efficiency of Flow
- DIP Turns

Page 235: Principles of Product Development Flow: Don Reinertsen

Polling Question #5

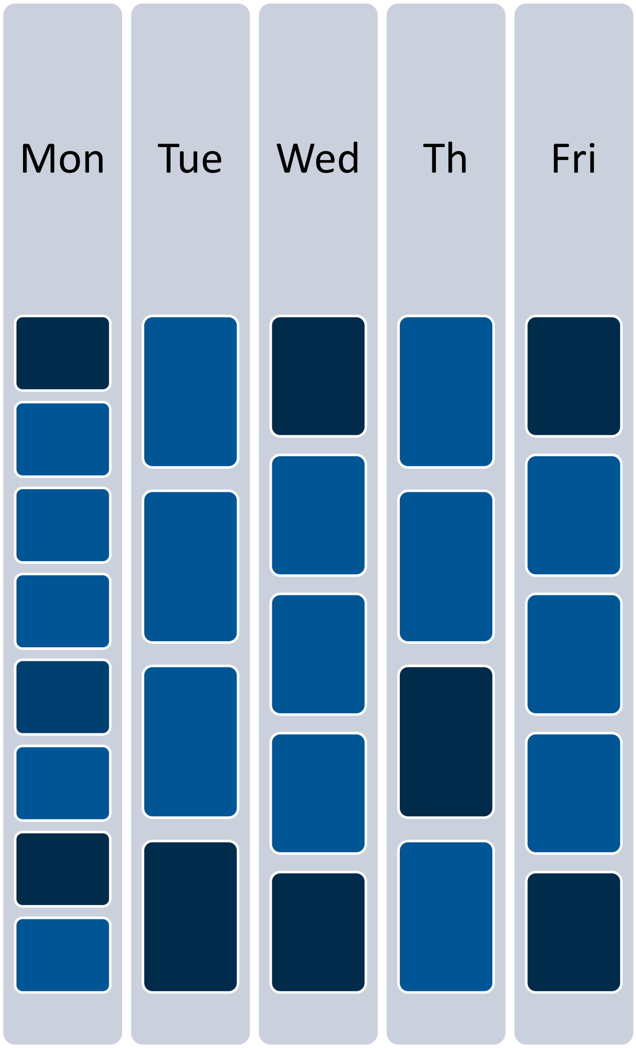
Experience with flow-based metrics?

1. Never heard of it before
2. Yes, I've read about it or seen it before
3. Yes, I have used them in my own work

Clash of Mind-Sets

Deterministic Plans for an Uncertain World

Value Flow: Utilization is the Wrong Goal



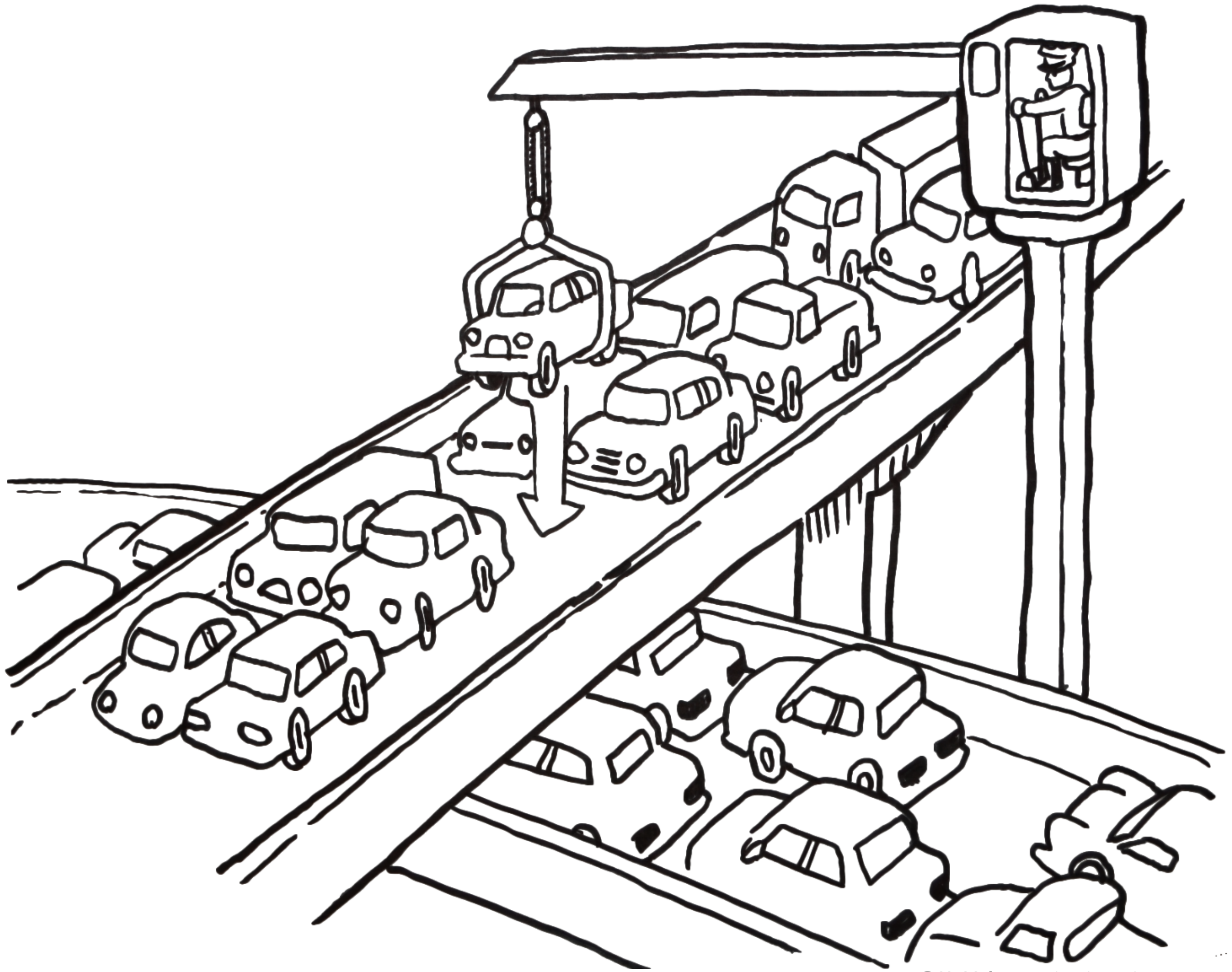
100% Utilization:

- Magnifies the impact of variation
- Maximizes task-switching overhead
- Assures slower overall progress

Change is inevitable, plan to learn

Multi-tasking is a myth we don't accurately comprehend

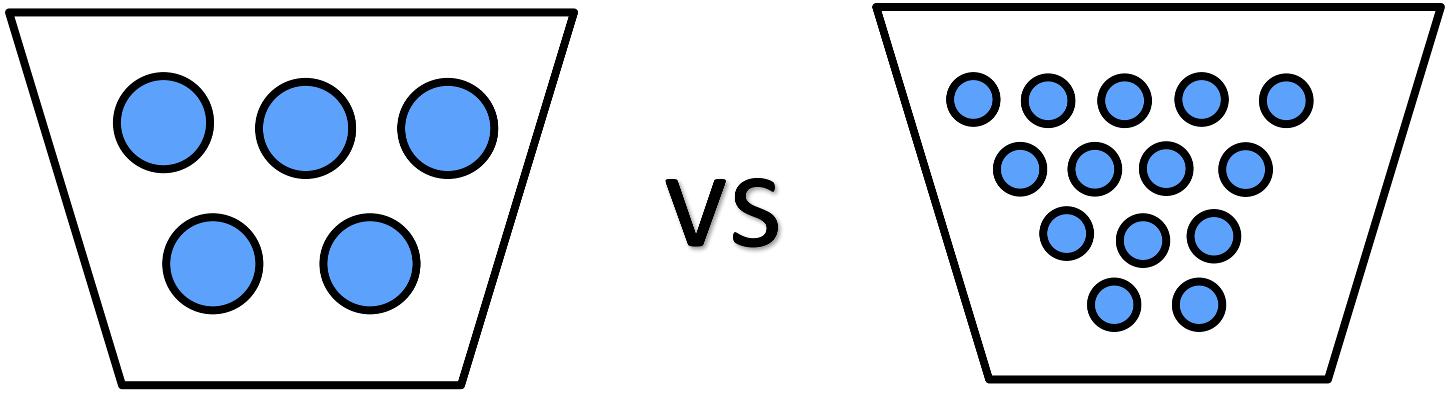
Maximum Utilization is Counterproductive



Diagnostic Metrics

Helping Teams Deliver

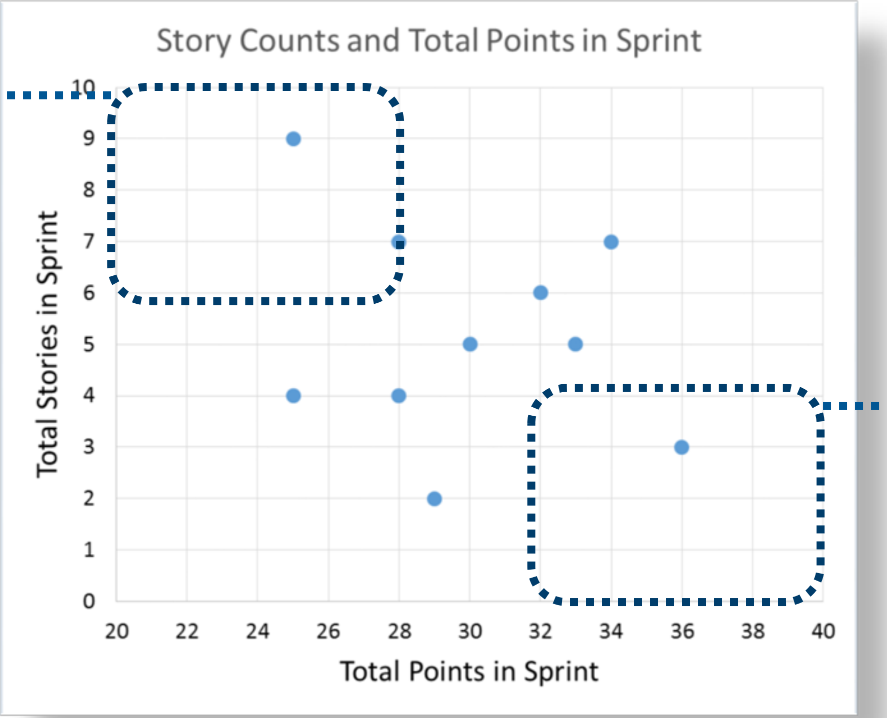
Batch Size Analysis – Story Size Focus



Splitting stories requires engineering judgment

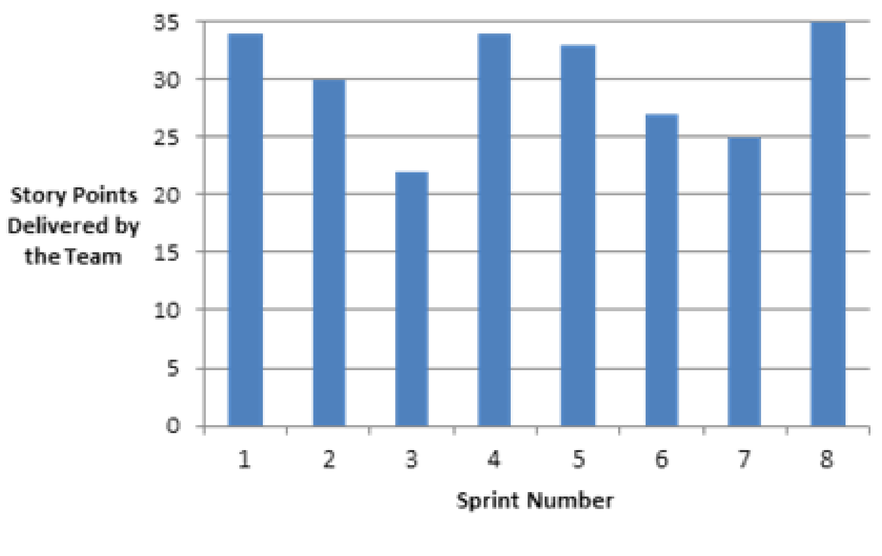
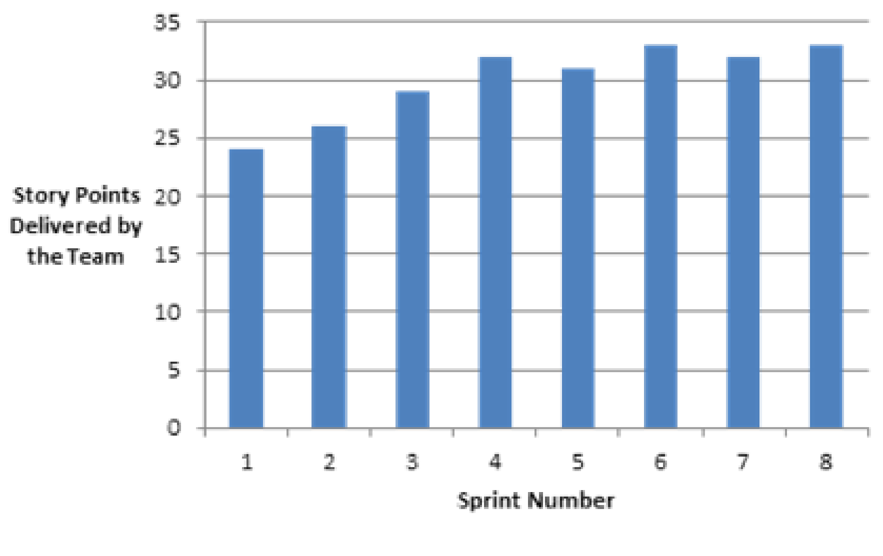
Potential Story Granularity Indicator?

Sprints with many small stories



Sprints with a few large stories

Coefficient of Variation – Analysis of Velocity

	
Average = 30	Average = 30
Standard Deviation = 4.84	Standard Deviation = 3.38
Coefficient of Variation = 16.13	Coefficient of Variation = 11.27

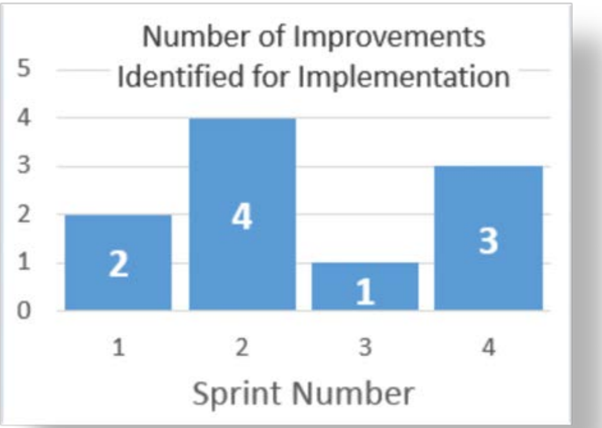
Diagnostic Metrics

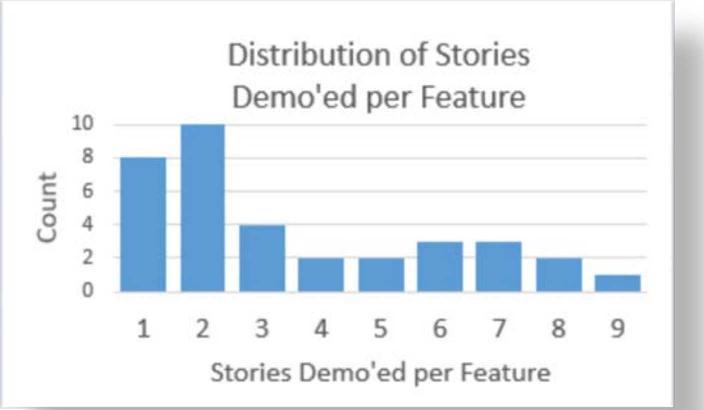
Understanding Program Performance

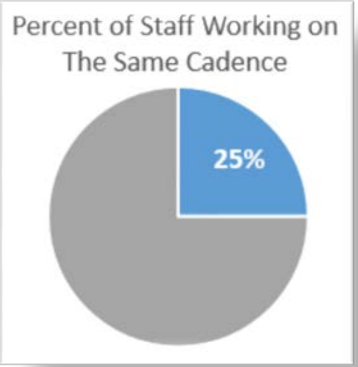
Indicator Examples$_1$

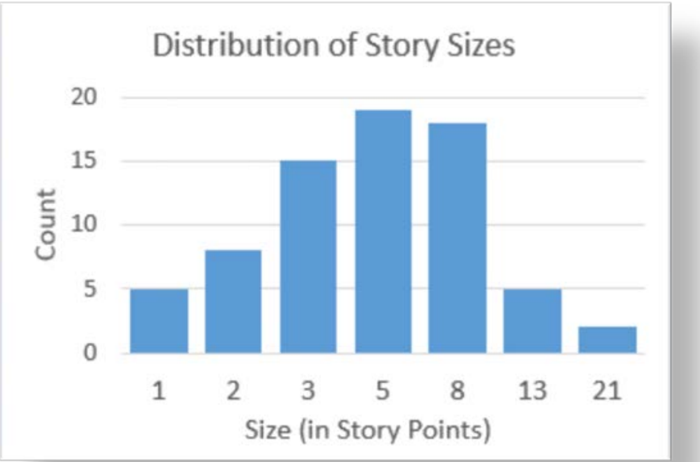
Essential Process Attributes

- Cadence
- Synchronization
- Short Learning Cycles
- Reduction in Batch Size
- Iterative and Incremental Delivery

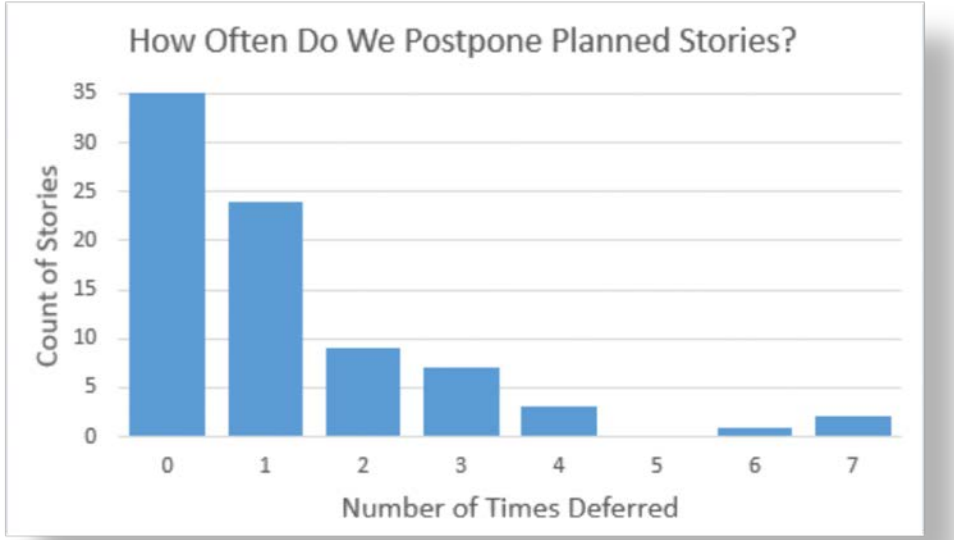


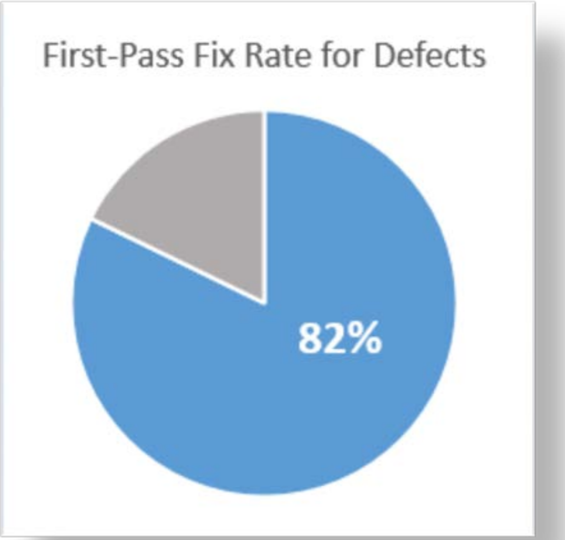


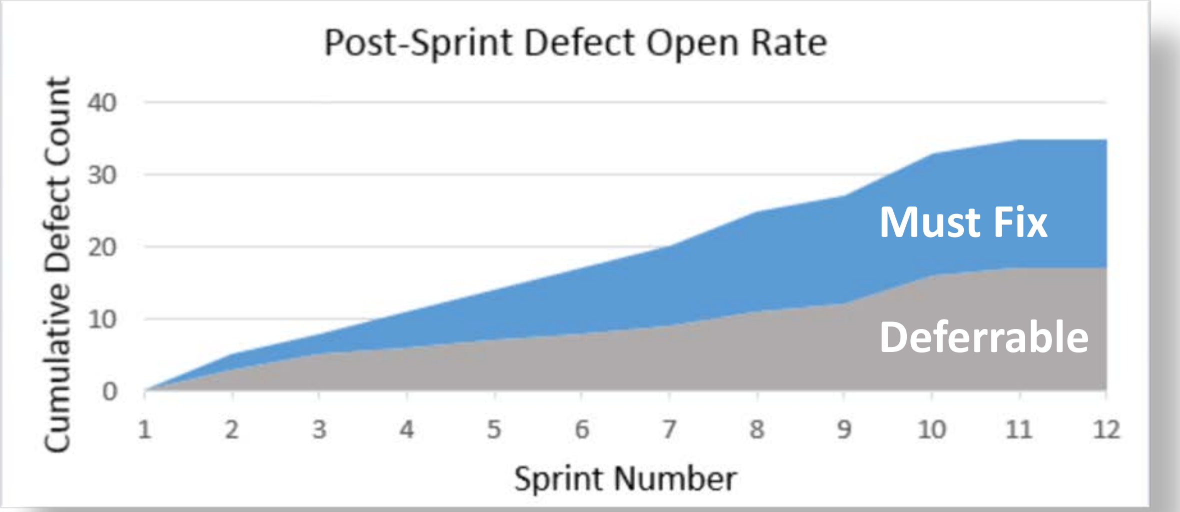




Indicator Examples$_2$





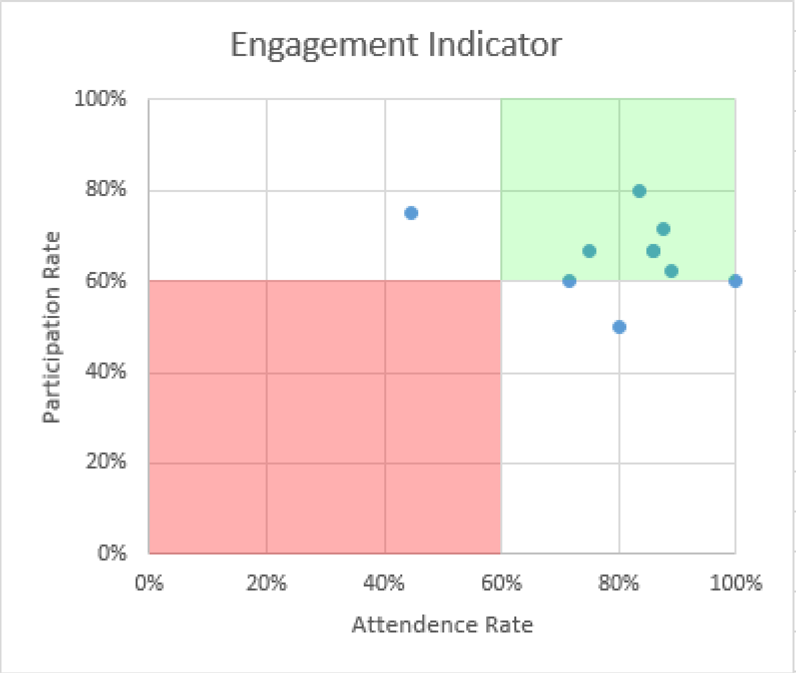


Adopting New Approaches

Assessing Engagement

Simple Indicator, Powerful Analysis

Ceremony	Expected	Attended	Engaged	%Attend	%Engage
1	5	5	3	100%	60%
2	7	6	4	86%	67%
3	8	7	5	88%	71%
4	4	3	2	75%	67%
5	9	4	3	44%	75%
6	6	5	4	83%	80%
7	9	8	5	89%	63%
8	7	5	3	71%	60%
9	7	6	4	86%	67%
10	5	4	2	80%	50%



Subset/aggregate data to look for trends across:

- Particular event types
  - Are ‘standups’ not working?
- Pockets of staff
  - Have we alienated ‘release managers’?

In Closing…

Bottom Line

1. Exercise Due Care

- The level of discipline and rigor applied must match the context served by the work
- Metrics give voice to things we want to hear about, we are responsible to choose
- Some very important things will lack high-resolution measures to inform us

2. Consider Systems' Perspectives

- A scrum team is its own system, and rich metrics to serve the team exist
- The enterprise consists of many other systems, which bring different perspectives
- Boundaries of generalizability exist among these systems

3. (Ruthlessly) Automate Basic Indicators and Analyses

- Wield tools in service of your needs, and do not limit the sphere of focus artificially
- Make metrics routine and boring – not episodic and authority-focused
- Tool chains and visualization techniques offer new opportunities