SEI Virtual Schoolhouse

Virtual Learning Package 9: Agile & Measurement

Suzanne Miller
SEI Continuous Deployment of Capability Directorate

May 2020

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM20-0419

Meeting Conventions for Today

Please stay on mute for the lecture portion of the course module

If you are “in” the Skype meeting via web or app, please ask questions via the Chat window.

- A facilitator will collect the questions and either pass them to the instructor if something immediate, or organize them for the Q&A portion of the course module

Those on dial in will enter questions via email to David Walbeck -- dtwalbeck@sei.cmu.edu

Instructor will call for participation and discussion at various points. Please remember to come off mute before talking.

When you are done talking, before going back on mute, please say “Over” so others know you are finished.

The lecture part of this session is being recorded. Recording will be turned off during discussions.

We’re at the Tip of the Iceberg…Again!

Topics the SEI will address in this module include:

- **Why is Measurement a Challenge for Large Scale Agile Programs?**
- **Key Concepts for Measuring Large Complex Programs Using Agile and Lean Development**

What is **NOT** in this presentation:

- **A one size fits all measurement strategy for all Agile programs**
- **An Agile measurement strategy customized specifically for your program**

Why is Measuring Agile Challenging in Large Complex Programs?

Different Delivery Modes/Cadence Mean Different Measurement Strategies

Date-based Milestones Emphasizing Completion of Activities

Increment-Based Demos
Definition of Done
Continuous Integration & Test

***Flow-based** Measures Emphasizing Sustainable Completion of Product Elements*


Team Increment

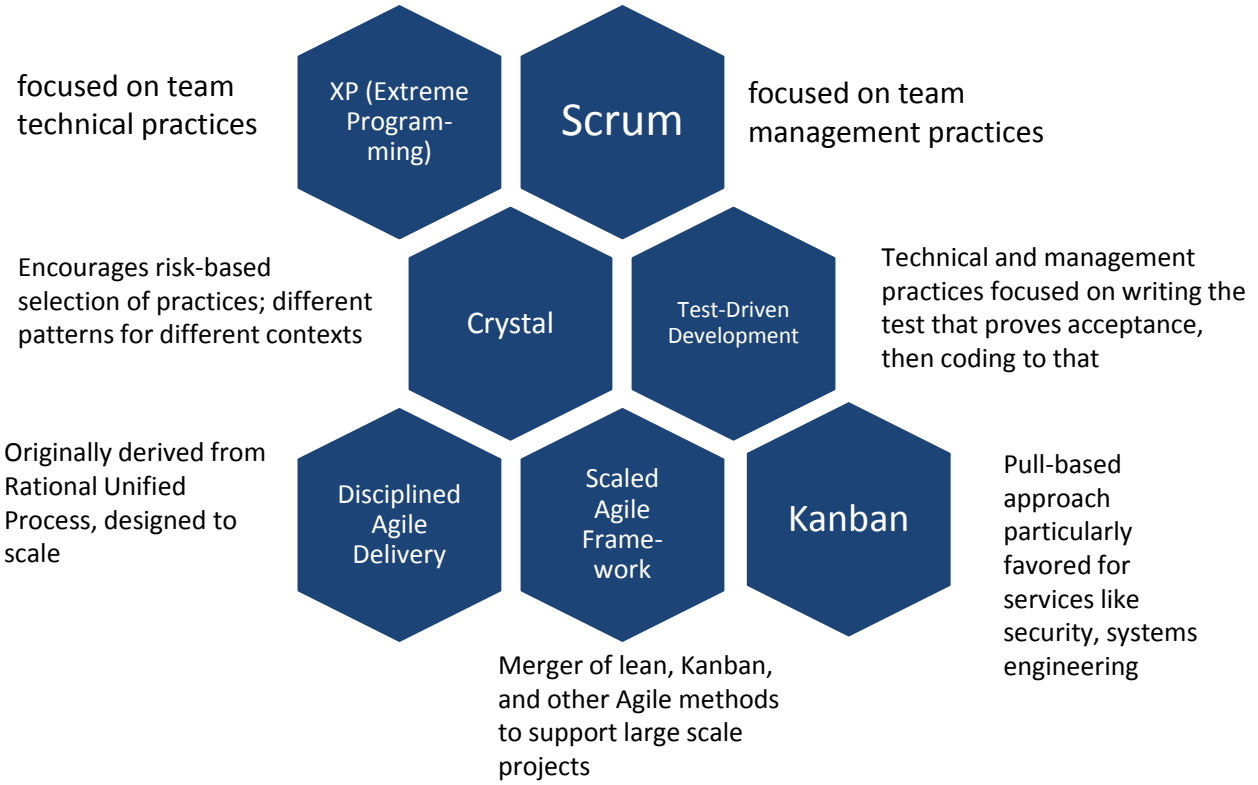

System Increment


Solution Increment

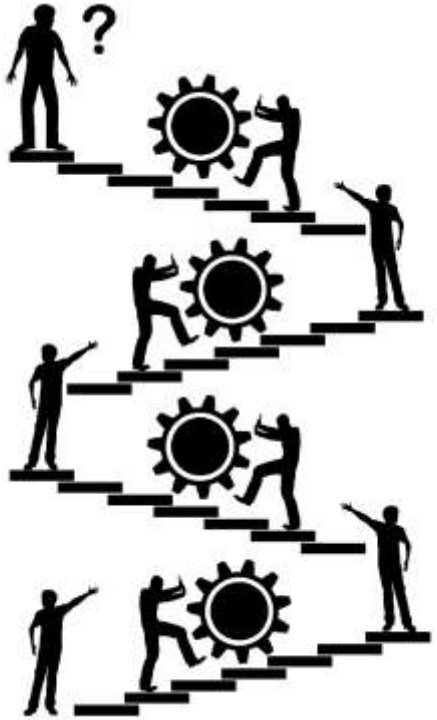

Release/ Deployment to Stakeholders

Different Agile Methods Leverage Different Measurement Strategies

Agile is principles-based, which means there isn't only "one way" to do it correctly, so a "one size fits all" measurement approach doesn't work, especially when a contractor is where development is occurring



Barriers to Effective Measurement in Agile Settings (some of these are shared with other life cycle approaches)



Metrics often focus exclusively on:

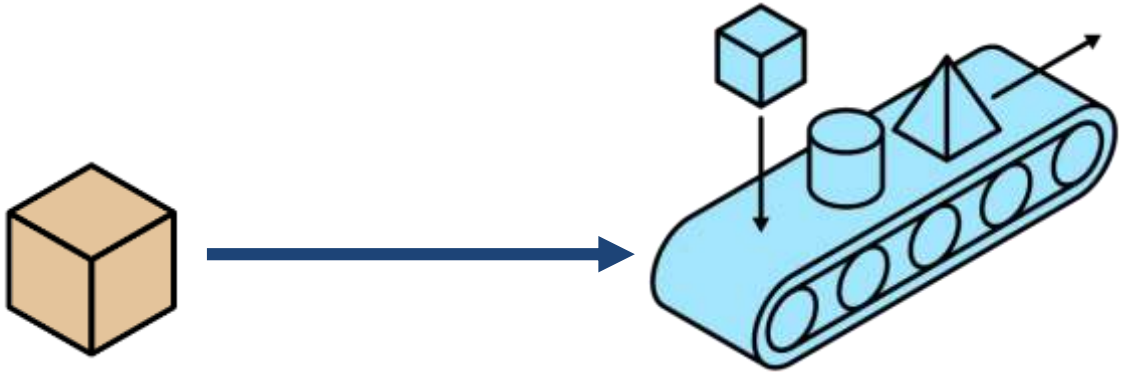
- Appeasing an authority role
- Demonstrating competence
- Validating the chosen path

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.

When deviations from the initial estimate are uniformly viewed in a negative light, exploiting options for better outcomes may be infeasible.

What Do You Measure When Moving Away from Buying a Box to Buying Continuous Delivery of Capability?



BUYING a Box

*If product is built as a single large batch, measurement focuses on **surrogates (like document production)** for progress toward the ultimate product.*

BUYING an ongoing delivery stream of capabilities

If we are delivering multiple small releases of our product, measurement can shift to measure the attributes of the product itself instead of surrogates like documents as the primary measures (measure Value, Quality)

Buying vs Building

There are things we expect the Builder to measure

There are things that senior management expects the Buyer to measure

Are the sources of measures for both the same?

- Ideally, yes
- Often, though, the buyer doesn't have direct access to the source of "build" measures

Is the use of measures the same for both Builder and Buyer?

- Builder uses measures (hopefully) as leading indicators that are used to make corrections in the process
- Buyer uses measures (hopefully) as leading indicators that are used to evaluate risk and confidence in the progress of the solution toward its goals

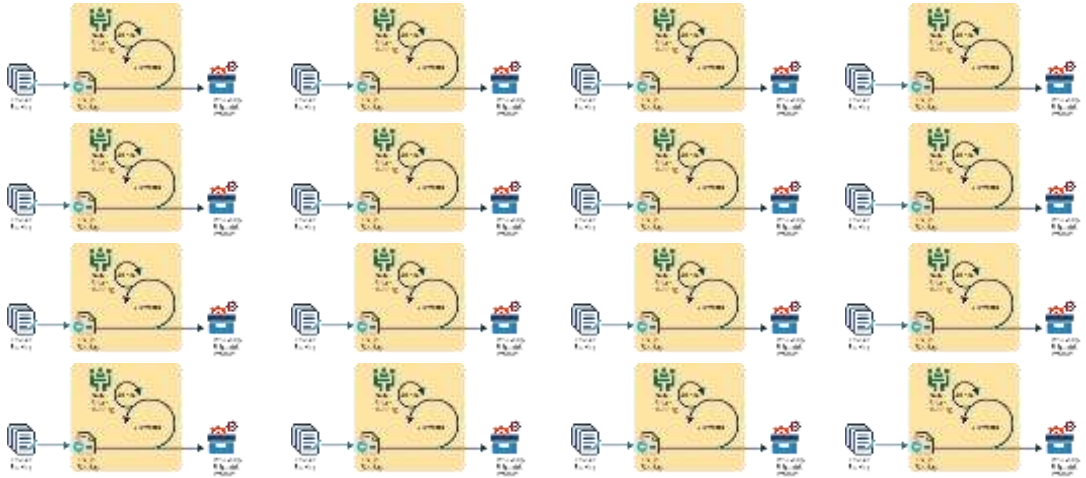
An opportunity in Agile settings is to bring the Buyer and Builder perspectives into closer alignment, IF collaborative behaviors are successful

Key Concepts for Measuring Large Complex Programs Using Agile and Lean Development

Understand Who Your Measurement Audience Is—People Who are Monitoring Progress ABOVE the Team Level

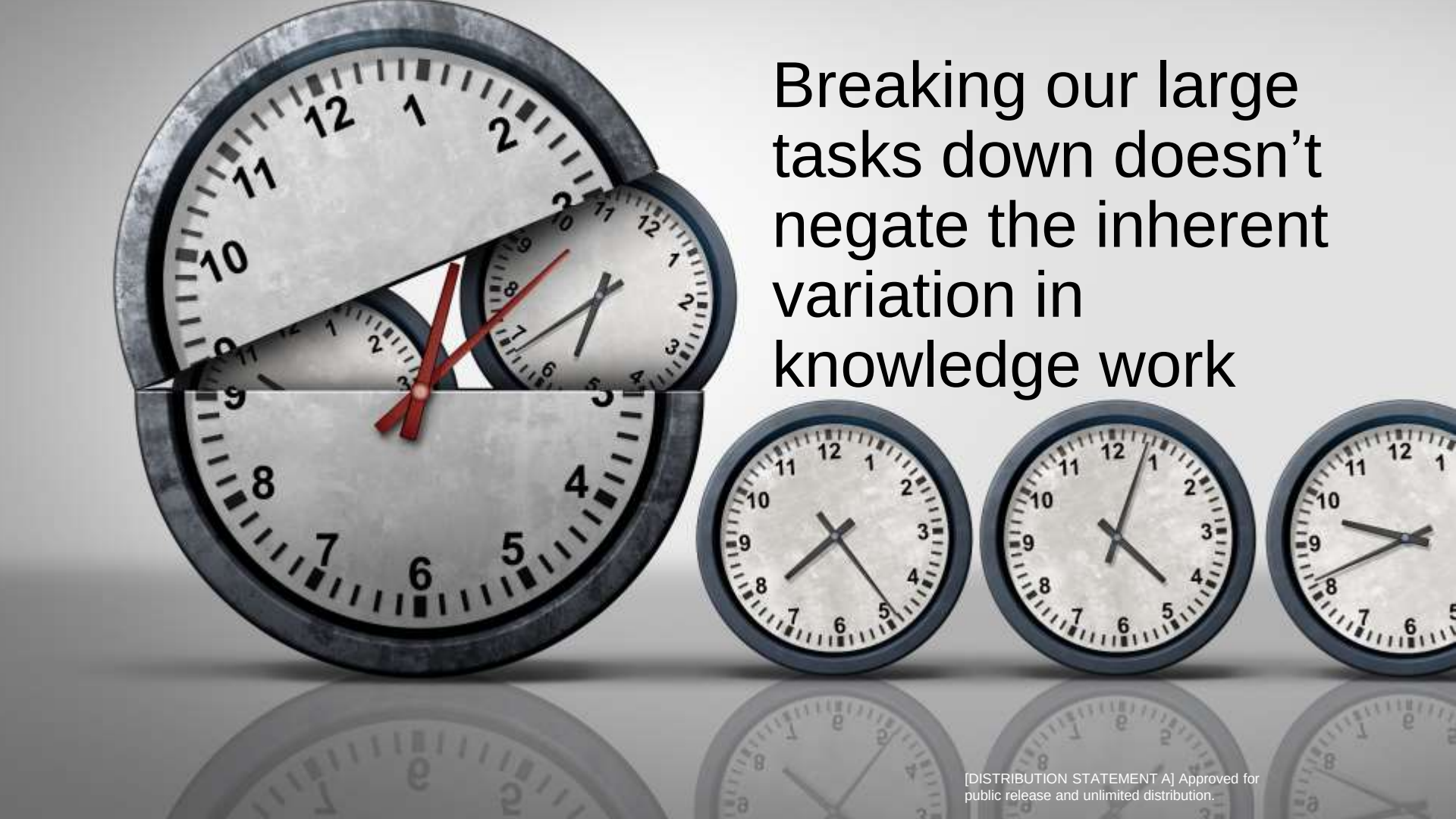
Geared to needs of external stakeholders

Intended to serve needs of the team - typically not shared out-side the team



Where do most "Agile" measures focus? TEAM level

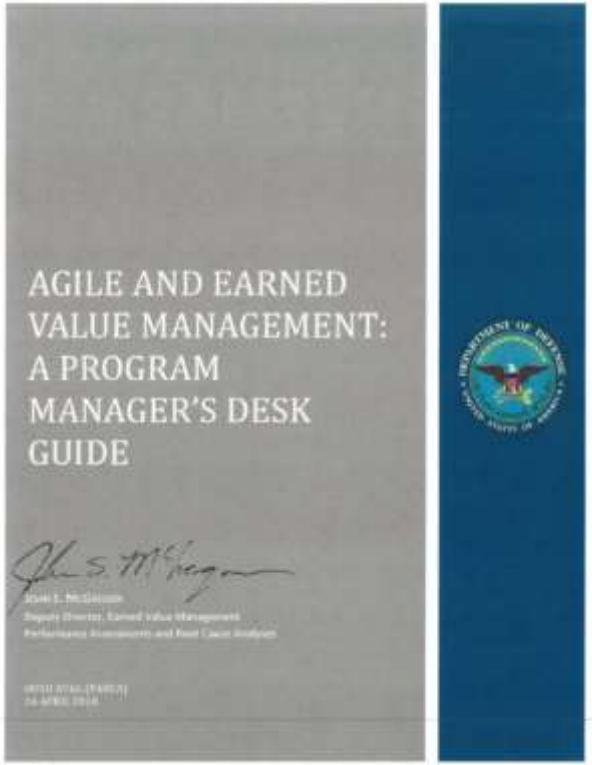
A perfect plan is NOT the goal!

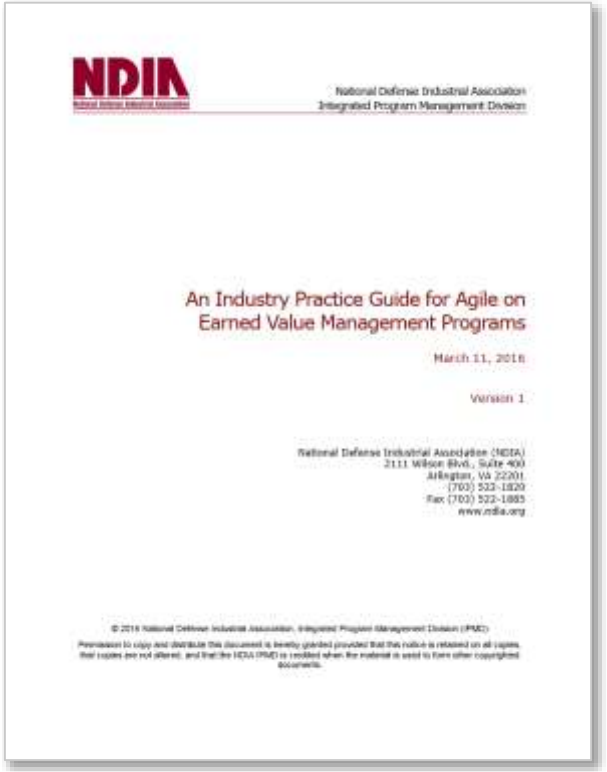


Breaking our large tasks down doesn’t negate the inherent variation in knowledge work

Are we delivering enough important capabilities to match the level of investment?

Resources: Earned Value Management for Programs Using Agile





Capability-Based Work Breakdown Structure is an Enabler for Agile Progress Measurement using EVM

1.1	Prime Mission Subsystem
1.1.1	Computer Software Configuration Item A
1.1.1.1	CSCI Requirements Analysis
1.1.1.2	CSCI Design
1.1.1.3	CSCI Code and Unit Test
1.1.1.4	CSCI Integration and Test
1.1.2	Computer Software Configuration Item B
1.1.3	High level Integration, Assembly, Test, and Checkout
1.1.4	…

FIGURE 1. SW Development MIL-STD-881D Appendix K WBS breakout

1.1	Prime Mission Subsystem
1.1.1	Capability A
1.1.1.1	Feature A1
1.1.1.2	Feature A2
1.1.1.3	Feature A3
1.1.1.4	Feature A4
1.1.2	Capability B
1.1.3	High level Integration, Assembly, Test, and Checkout
1.1.4	…

FIGURE 2. Possible Agile SW Development MIL-STD-881D WBS breakout. Not prescriptive.

Figures 1&2 from Page 4, PARCA Desk Guide

The WBS reported for EVM can (and traditionally does) align to a workflow-based waterfall development oriented hierarchy as found in MIL-STD-881C* (see Figure 1).

However, an outcome-based Agile structure that focuses on customer driven deliverables (see Figure 2) is also acceptable. Both WBS hierarchies are product-based and supported by the DoD EVMSIG and MIL-STD- 881C.

*DEPARTMENT OF DEFENSE STANDARD PRACTICE WORK BREAKDOWN STRUCTURES FOR DEFENSE MATERIEL ITEMS," October 2011

Quantifying Value Returned for Investment Made

There are no easy answers to questions like:

- Is the Global Positioning System (GPS) worth the cost?
- What should the MSRP be on the next aircraft carrier?
- Are fighter jets more expensive than they “should” be?

Earned Value Management Systems aspire to:

- Serve as a Leading Indicator of end point deliveries
- Offer a reporting level to engage in meaningful conversation
- Build an economic framework not limited to measures of time and money

Anti-Patterns Related to EVM That We Have Observed

Unreasonable emphasis on story point estimation accuracy, especially above the team level

- Raising the stakes of expressing an opinion about the technical solution
- Potentially locks people into a design based on committed effort, not user-value
- Making judgments on scaled up small team relative estimation is fraught with confounding factors

Very expensive earned value management approaches

- Tracking work packages with completion times of 2 or 3 days
- Reporting weekly or fortnightly earned value projections and associated variances
  - A three month cadence of EVM reporting is typically more useful

How Do We Achieve Value?

Are we delivering capabilities that satisfy the demands of the operational environment?

Quality

Quality of Software-Intensive Systems

Do we evaluate restaurants by the number of people who get sick from eating there?

- A job candidate by the number of times they've been fired from previous positions?
- A car by the number of times it leaves you stranded on the side of the road?
- A service provider by the number of times they treat you rudely?

Why are defect counts a primary indictor of software quality?

- ***Don't we deserve better?***

What we Want from Software-Intensive Systems

What the system DOES:

- A new channel enables communication with ## more platforms in theatre
- A new sensor broadens our aperture to encompass ##% more signals in our AOR

What the system CONSUMES:

- Re-hosting the platform led to ##% reduction in support costs
- Implementing a ‘wizard tool’ in the UI cut the number of operators required by 50%

What the system PRODUCES:

- Virtualization techniques create ##% more bandwidth in the uplink
- The Deep Learning Algorithm reduced false-positives by ##%

We Still Need to Have Confidence in our Build Quality

Some waterfall processes have adopted practices related to “phase containment”:

- When defects are identified, analysis of when the defect was introduced (reqmts, design, build, integration rework….) is used as an indicator of process steps that need improvement
- Ideal is to ‘catch’ defects as close to the injection point as possible, when they are cheaper to fix

How does this translate into an Agile setting?

- Each 2 week iteration (ideally) is a complete set of life cycle phases
  - How much does it matter if I found a defect on day 1 of a 4 day story execution or day 3?
- Mindset of “phase containment” shifts to a mindset of “iteration containment”
  - Same concept – catch defects close to when injected, but focus around “phase” is not useful
  - We are integrating more often (ideally) so we can catch integration-visible defects in a timeframe that allows easier fixes at much reduced cost
  - ***THIS IS ONE REASON SO MUCH ATTENTION IS BEING PAID TO DEVOPS AND AUTOMATED TEST***

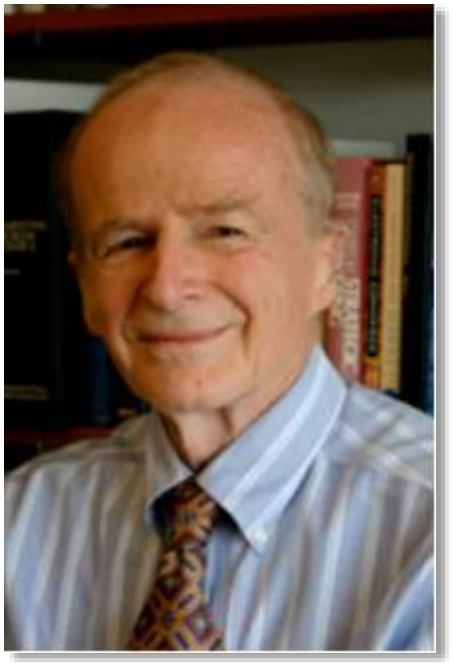
Flow

Are we delivering capabilities fast enough to keep pace with the user's operational needs?

Little's Law

…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…

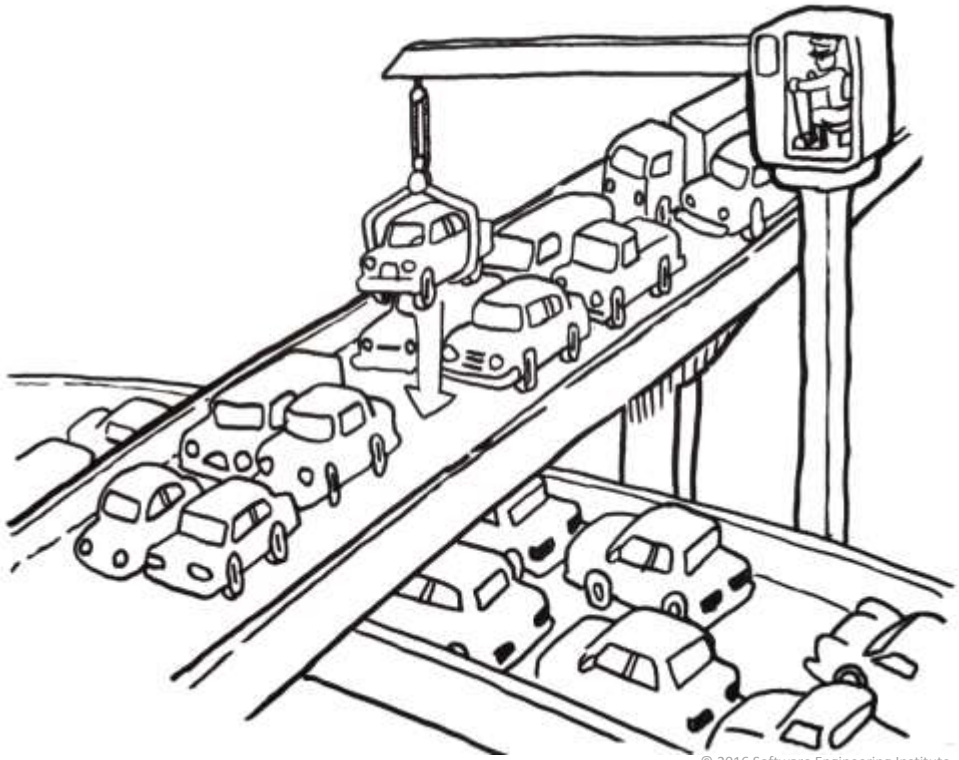
$$L = \lambda W$$



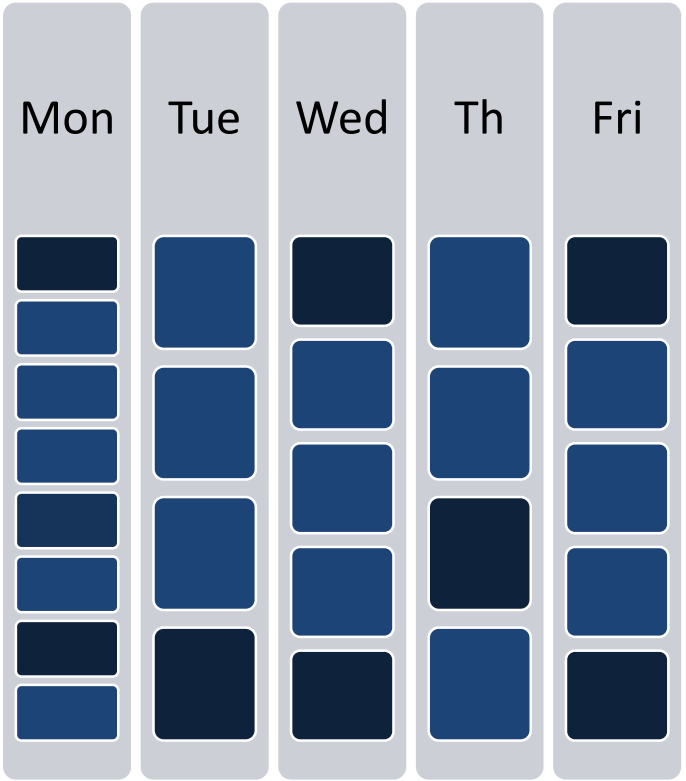
http://mitsloan.mit.edu/faculty-and-research/faculty-directory/detail/?id=41432

Everyone Knows Little's Law

Maximum Utilization Not the Way to Achieve Flow



Flow vs Utilization is a Dichotomy that is Seen on Both Contractor and Acquisition Side



100% Utilization:

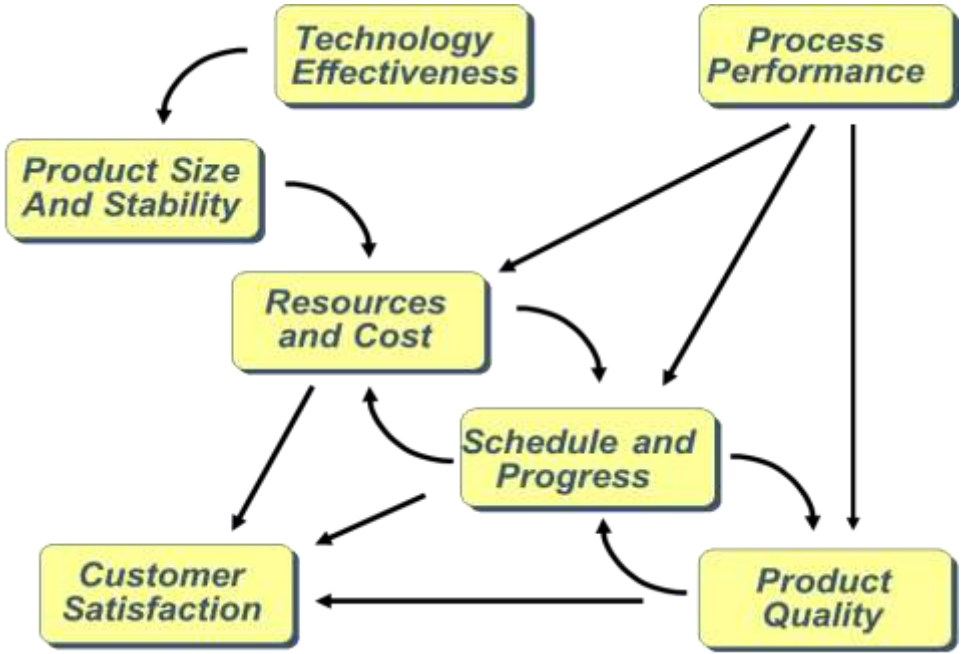
- Magnifies the impact of variation
- Maximizes task-switching overhead
- Assures slower overall progress

Change is inevitable, plan to learn

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

Measurement Considerations at Scale

What Doesn't Change: Measurement Challenge is the Multi-Variable Nature of Causality



Graphic Adapted from: *Practical Software & Systems Measurement*
http://www.psmsc.com/

Model performance & tradeoffs that drive tactical choices.

Understanding cause-and-effect.

A pre-requisite to acting on progress measures.

Agile & Lean thinking drives to new areas of focus.

The Defense Innovation Board Recommendations

Four Software Types

A. commercial (“off-the-shelf”) software with no DoD-specific customization required;

B. commercial software with DoD-specific customization needed;

C. custom software running on commodity hardware (in data centers or in the field);

D. custom software running on custom hardware (e.g. embedded software).

Four Types of Software Metrics

- Deployment Rate Metrics
- Response Rate Metrics
- Code Quality Metrics
- Program Management, Assessment, and Estimation Metrics

Welcome to your world!

Defense Innovation Board Metrics for Software Development

Version 0.9, last modified 9 Jul 2018

Software is increasingly critical to the mission of the Department of Defense (DoD), but DoD software is plagued by poor quality and slow delivery. The current state of practice within DoD is [illegible] complexity is often estimated based on number of source lines of code (SLOC), and [illegible] measured in terms of programmer productivity. While both of these quantities [illegible], they are not necessarily predictive of cost, schedule, or performance. They [illegible]ect as measurements of program success, defined broadly as delivering [illegible] and value to users. Measuring the health of software development activities [illegible]s using these obsolete metrics is irrelevant at best and, at worst, can be [illegible] alternative, we believe the following measures are useful for DoD to track [illegible]ftware programs and drive improvement in cost, schedule, and performance.

#	Metric	Target value (by software type)[i]				Typical DoD values for SW
		COTS[ii] apps	Custom -ized SW[iii]	COTS HW/OS[iv]	Real-time HW/SW[v]	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)[vi]	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing[vii]	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<8 hr	<1 day	N/A	?
6	Automated test coverage of specs / code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code available to DoD for inspection/rebuild	N/A	100%	100%	100%	0%
10	Complexity metrics	#/type of specs structure of code #/type of platforms		# programmers #/skill level of teams #/type deployments		Partial/ manual tracking
11	Development plan/environment metrics					
12	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

These are Dramatically Different from Today's Performance

Four Types of Software Metrics

- Deployment Rate Metrics
- Response Rate Metrics
- Code Quality Metrics
- Program Management, Assessment, and Estimation Metrics

These are not as focused on determining VALUE (other than time to capability as value) as they are on supporting measurement of QUALITY and FLOW

#	Metric	Target value (by software type)[i]				Typical DoD values for SW
		COTS[iii] apps	Custom -ized SW[iii]	COTS HW/OS[iv]	Real-time HW/SW[v]	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)[vi]	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing[vii]	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
6	Automated test coverage of specs / code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code available to DoD for inspection/rebuild	N/A	100%	100%	100%	0%
10	Complexity metrics	#/type of specs structure of code #/type of platforms		# programmers #/skill level of teams #/type deployments		Partial/ manual tracking
11	Development plan/environment metrics					
12	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

Some DIB Metrics Help to Understand ***Quality***

7	Number of bugs caught in testing vs field use
8	Change failure rate (rollback deployed code)

Defense Innovation Board Metrics for Software Development

Version 0.9, last modified 9 Jul 2018

Software is increasingly critical to the mission of the Department of Defense (DoD), but DoD software is plagued by poor quality and slow delivery. The current state of practice within DoD is that software complexity is often estimated based on number of source lines of code (SLOC), and rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance. They are especially suspect as measurements of program success, defined broadly as delivering needed functionality and value to users. Measuring the health of software development activities within DoD programs using these obsolete metrics is irrelevant at best and, at worst, can be misleading. As an alternative, we believe the following measures are useful for DoD to track performance for software programs and drive improvement in cost, schedule, and performance.

#	Metric	Target value (by software type)[i]				Typical DoD values for SW
		COTS[ii] apps	Custom -ized SW[iii]	COTS HW/OS[iv]	Real-time HW/SW[v]	
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)[vi]	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing[vii]	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
6	Automated test coverage of specs / code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code available to DoD for inspection/rebuild	N/A	100%	100%	100%	0%
10	Complexity metrics	#/type of specs structure of code #/type of platforms		# programmers #/skill level of teams #/type deployments		Partial/ manual tracking
11	Development plan/environment metrics					
12	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

https://media.defense.gov/2018/Jul/10/2001940937/-1/-1/0/DIB_METRICS_FOR_SOFTWARE_DEVELOPMENT_V0.9_2018.07.10.PDF

Some DIB Metrics Help to Understand ***Flow***

#	Metric
1	Time from program launch to deployment of simplest useful functionality
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)[vi]
3	Time from code committed to code in use

Defense Innovation Board Metrics for Software Development

Version 0.9, last modified 9 Jul 2018

Software is increasingly critical to the mission of the Department of Defense (DoD), but DoD software is plagued by poor quality and slow delivery. The current state of practice within DoD is that software complexity is often estimated based on number of source lines of code (SLOC), and rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance. They are especially suspect as measurements of program success, defined broadly as delivering needed functionality and value to users. Measuring the health of software development activities within DoD programs using these obsolete metrics is irrelevant at best and, at worst, can be misleading. As an alternative, we believe the following measures are useful for DoD to track performance for software programs and drive improvement in cost, schedule, and performance.

#	Metric	Target value (by software type)[i] COTS[ii] apps	Custom -ized SW[iii]	COTS HW/OS[iv]	Real-time HW/SW[v]	Typical DoD values for SW
1	Time from program launch to deployment of simplest useful functionality	<1 mo	<3 mo	<6 mo	<1 yr	3-5 yrs
2	Time to field high priority fcn (spec → ops) or fix newly found security hole (find → ops)[vi]	N/A <1 wk	<1 mo <1 wk	<3 mo <1 wk	<3 mo <1 wk	1-5 yrs 1-18 m
3	Time from code committed to code in use	<1 wk	<1 hr	<1 da	<1 mo	1-18 m
4	Time req'd for full regression test (automat'd) and cybersecurity audit/penetration testing[vii]	N/A <1 mo	<1 da <1 mo	<1 da <1 mo	<1 wk <3 mo	2 yrs 2 yrs
5	Time required to restore service after outage	<1 hr	<6 hr	<1 day	N/A	?
6	Automated test coverage of specs / code	N/A	>90%	>90%	100%	?
7	Number of bugs caught in testing vs field use	N/A	>75%	>75%	>90%	?
8	Change failure rate (rollback deployed code)	<1%	<5%	<10%	<1%	?
9	% code available to DoD for inspection/rebuild	N/A	100%	100%	100%	0%
10	Complexity metrics	#/type of specs structure of code #/type of platforms		# programmers #/skill level of teams #/type deployments		Partial/ manual tracking
11	Development plan/environment metrics					
12	"Nunn-McCurdy" threshold (for any metric)	1.1X	1.25X	1.5X	1.5X each effort	1.25X Total $

https://media.defense.gov/2018/Jul/10/2001940937/-1/-1/0/DIB_METRICS_FOR_SOFTWARE_DEVELOPMENT_V0.9_2018.07.10.PDF

Due Care Obligations

Searching under the bright street light for the keys which were dropped in the dark alley around the corner is futile.

Convenient data don't always yield useful information.

Secondary analysis may overshadow the original purpose, changing consequences to data providers unfairly.

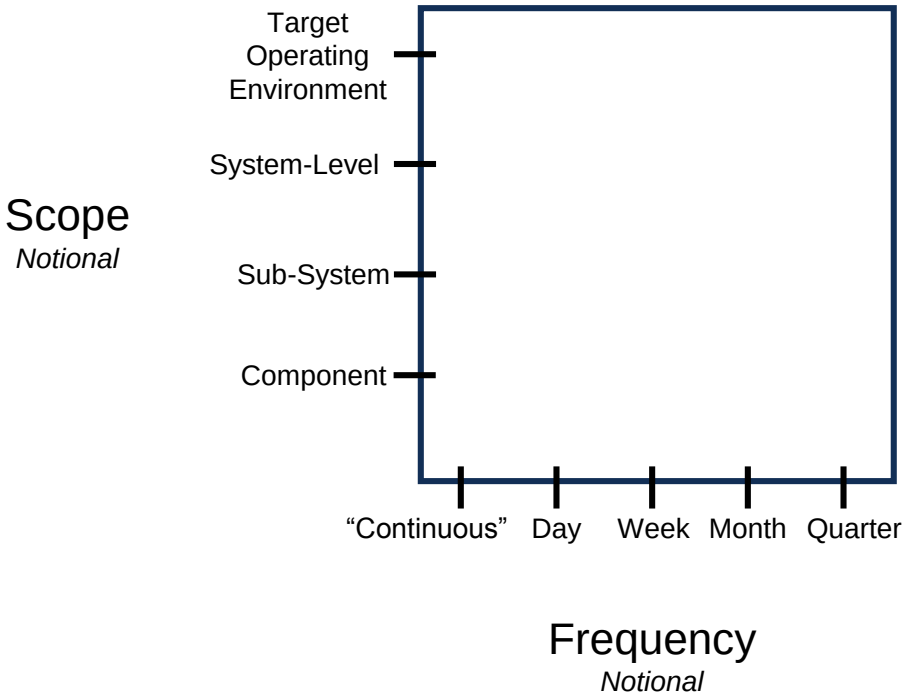
The Important Outcome may be Difficult to Measure

Test Start Date Fidelity	vs.	Growth in Accumulated Test Results

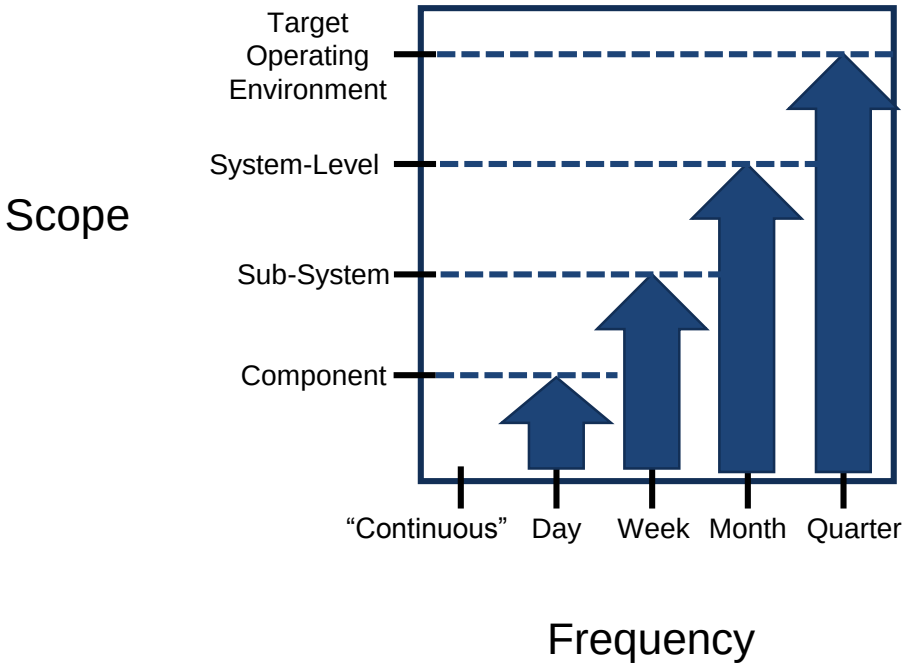
Yes, starting test at the planned date is a good thing, but

- Counter-productive trade-offs can always help us meet a date
- The completion of test (and analysis) is ultimately what matters

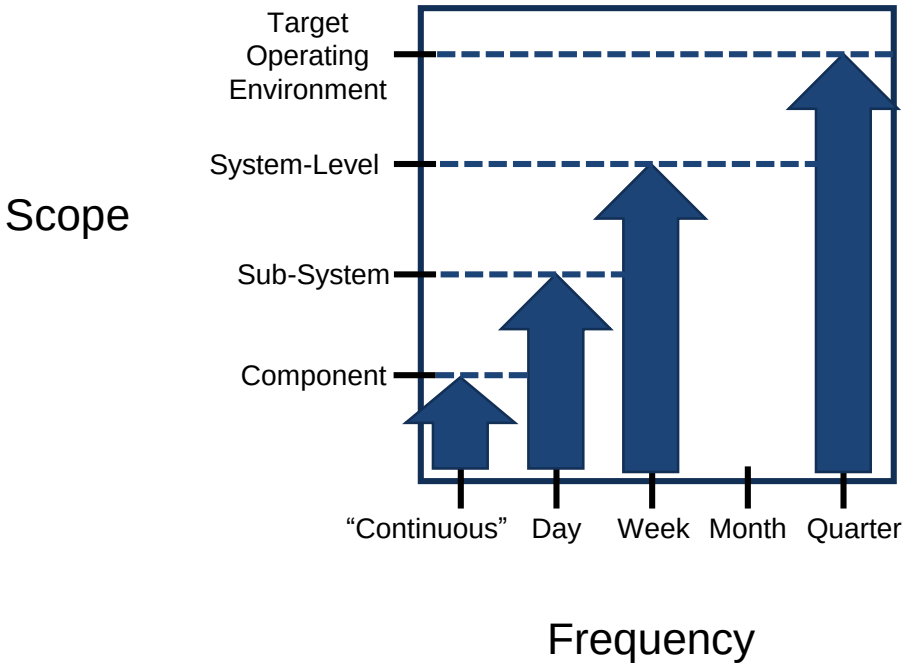
Scope and Frequency of Integration & Test



Scope and Frequency of Integration & Test



Scope and Frequency of Integration & Test



Measuring the “Shift Left” of Testing

Questions:

- How frequently are we integrating and testing?
- At what system scope are we doing that?
- How quickly are we getting to a system-level build?
- How frequently does the baseline change at each level?
- Are some system components not getting integrated and tested early/frequently?
- Where has automation changed things most?
- What is our coverage of specifications/requirements/logic/code?

Competing Business Models Can Interfere

Resource Utilization Rate vs. Delivery Rate

Yes, we need to assure people have enough work to do, but

- Finishing work is more important than being busy
- There is a point of severely diminishing returns

Focus on Bottlenecks by Reducing Batch Size

Speeding up flow by balancing utilization

- Some refer to this as “slowing down so we can speed up”

Manufacturing examples focus on structural bottlenecks, which are less transient

- Knowledge work includes fluctuation in product complexity over time

Adding costly resources to a chokepoint in the process is not the only solution

- Changing batch size can help mitigate bottlenecks

Reinertsen’s Batch Size First Principle:

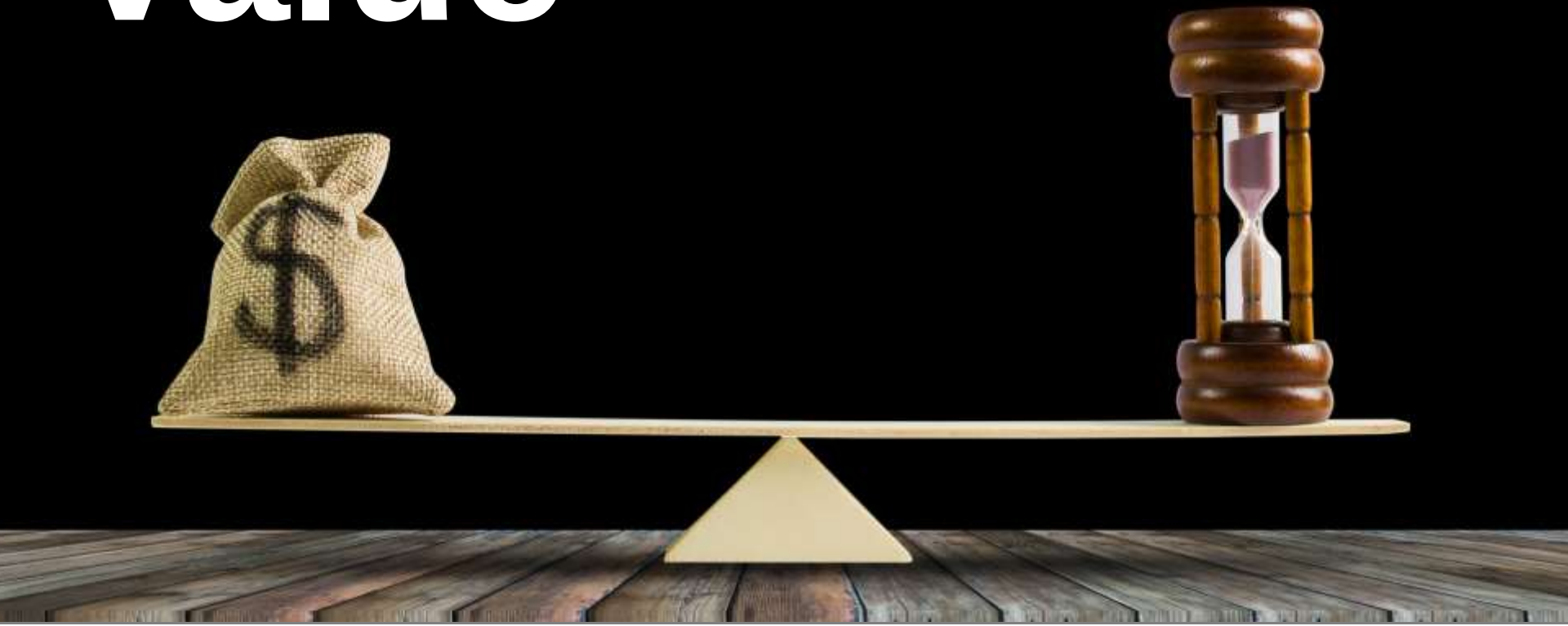
Reduce batch size before you attack bottlenecks*

> *Goldratt suggested letting the slowest Boy Scout, Herbie, walk at the head of the line. In my years as a scout leader, we did not use this logical and conceptually appealing approach. You see, this approach contains the embedded assumption that the hiking group can only hike in one large batch*

* *Principle B21: Page 133: Principles of Product Development Flow: Donald G. Reinertsen*

Selected Aspects of Measuring Value, Quality, and Flow

Value



Product Utility and Total Cost of Ownership

System performance priorities drive design tradeoffs

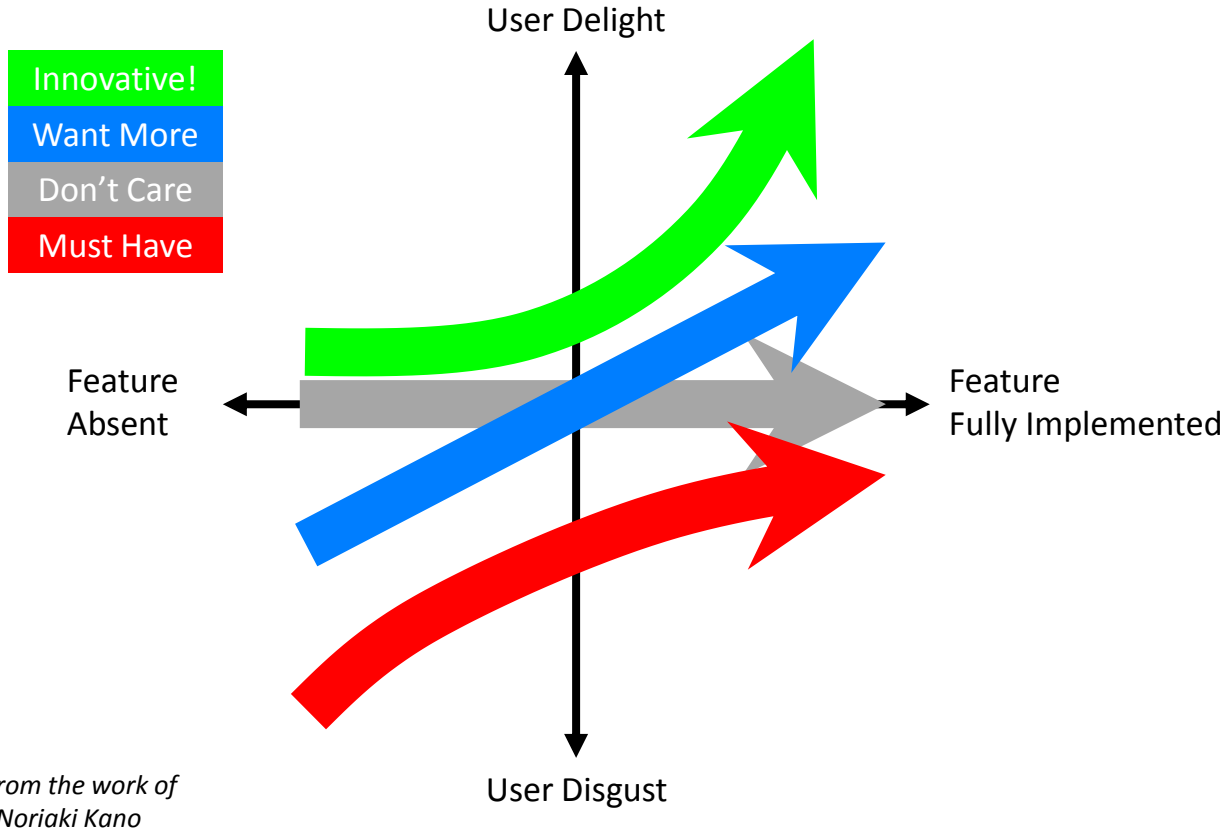
- Differentiate *must haves*, *satisfiers*, and *game changers*

Utility functions for system attributes drive user preference

- Fleet operations drive a variety of performance needs
- Operators are one of many important system users
- Strategic and tactical considerations apply

Acquisition performance is an element of system performance

Understanding User Value with KANO Analysis*



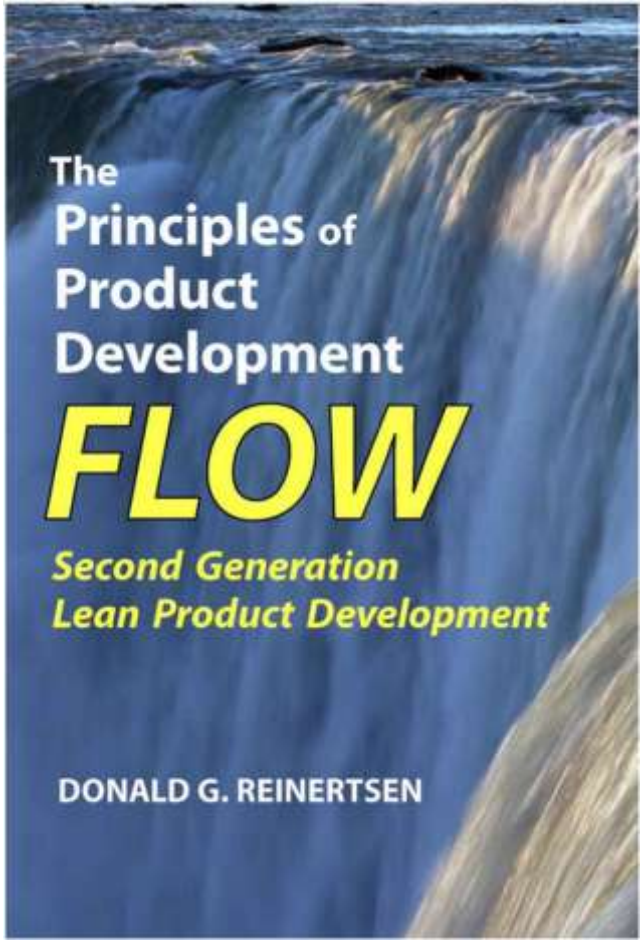
** Adapted from the work of Professor Noriaki Kano*

Cost of Delay

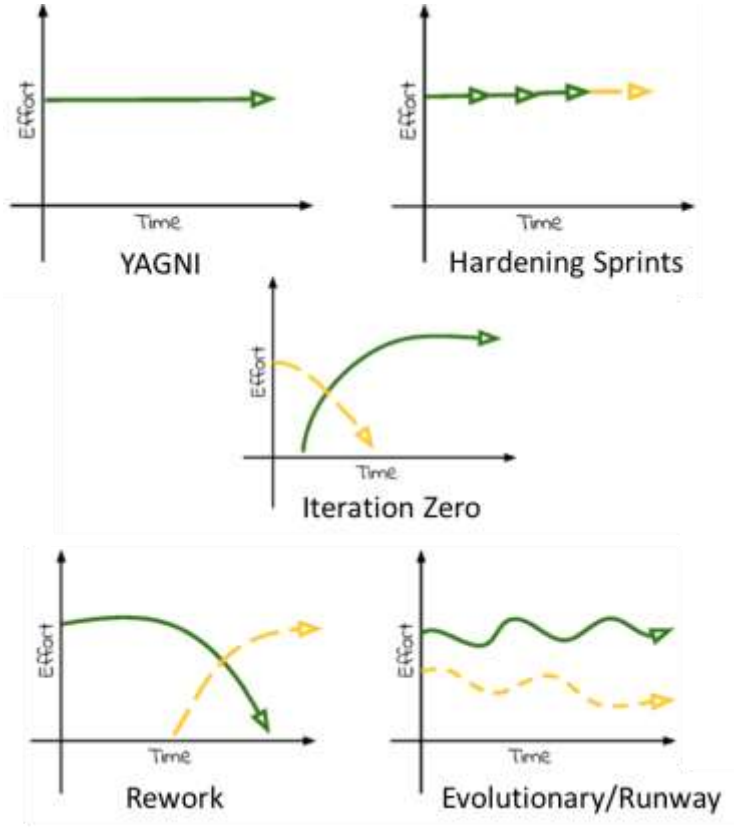
Reinertsen urges a focus on cost of delay as a surrogate for value, and proposed an approach to sequencing based on “Weighted Shortest Job First.”

Scaled Agile Framework (SAFe) elaborated the operational concepts as shown below:

$$\frac{\text{User-Business Value} + \text{Time Criticality} + \text{Risk Reduction and/or Opportunity Enablement}}{\text{Job Duration or Size}}$$



Architecture Always Matters – Especially at Scale



Quality

Examples of User-Centric Quality Outcomes

“This feature I’m working on will save the user 2 hours every day, when it’s deployed.”

“The next release of the system will reduce support costs by $1M annually, because we will be able to eliminate the three most common exception-handling challenges.”

“This new communications protocol will allow us to share data with 3 more legacy platforms.”

“Early user testing shows that the new interface will eliminate an average of 3 mouse clicks per standard query – mouse clicks that require users to confirm data they have no authority to change. This will reduce both time to complete tasks, and user frustration.”

Focus on Speed-to-Quality

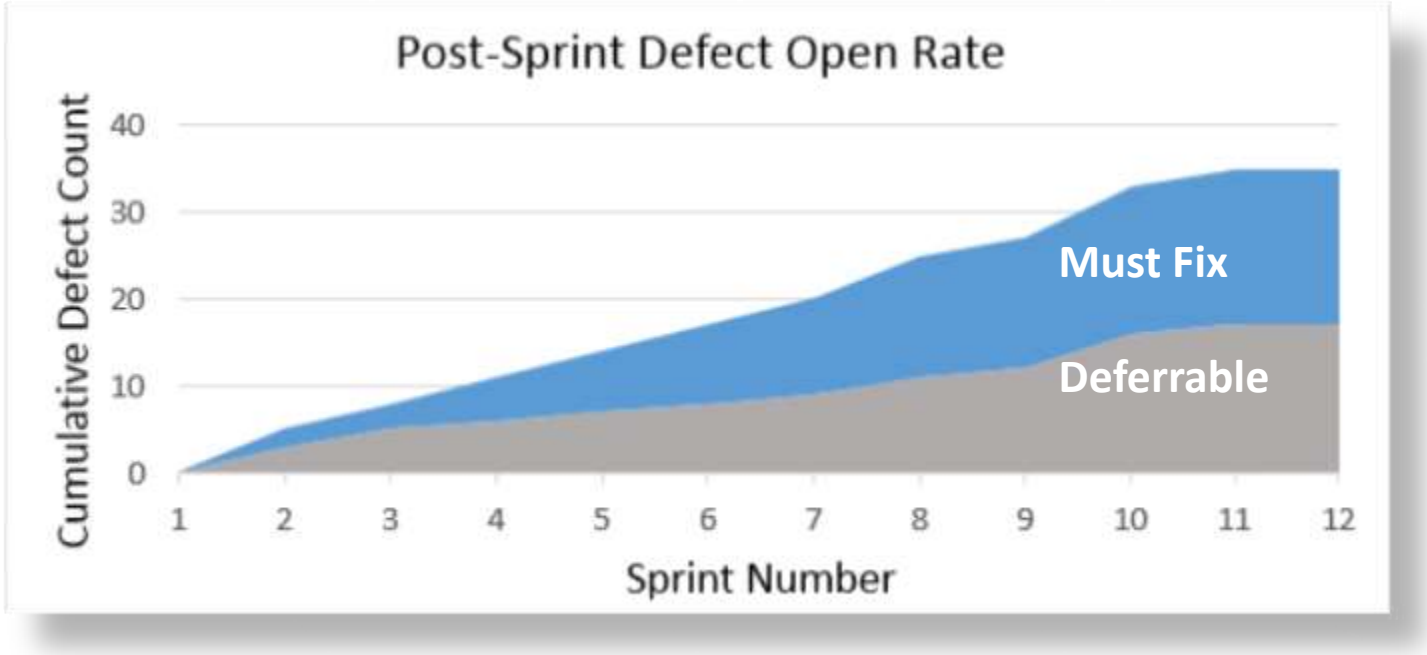
Quality Management Systems (QMS) have long been a focus

- Defining processes to discover and remediate defects earlier
- Quantifying process performance as a measure of risk
- Disciplined application of proven methods

Understanding the efficiency and effectiveness of the QMS is valuable

- Baselines aid in forecasting and planning
- Monitoring performance aids risk management
- Defensible basis for quantifying improvement

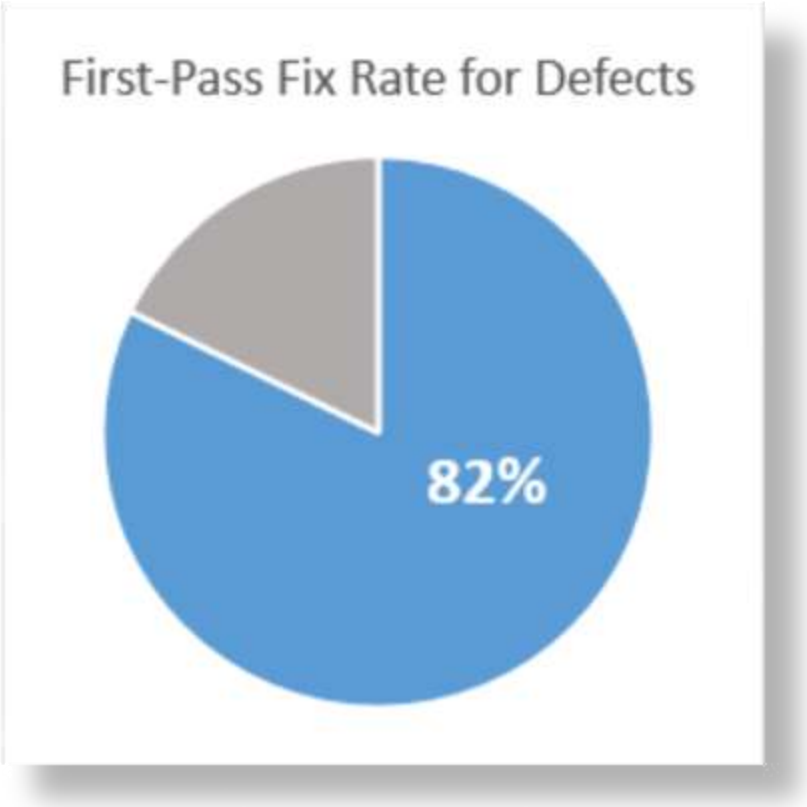
Speed to Quality – Understanding Rework Levels



Speed to Quality – Analyzing Rework Cycles

How often do we fix defects completely on the first attempt?

- Avoid introducing new defects during rework activities
- Understand the effectiveness of quality approaches and tools
- Enhance regression testing based on common sources of rework

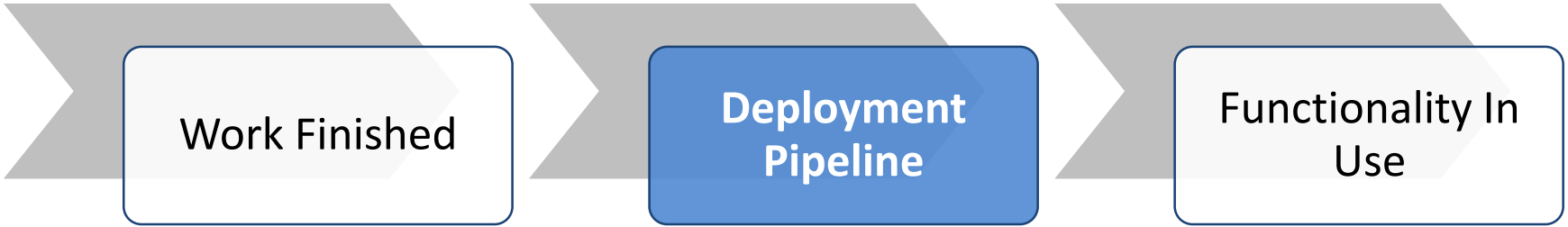


Flow

Lead Time to User Engagement

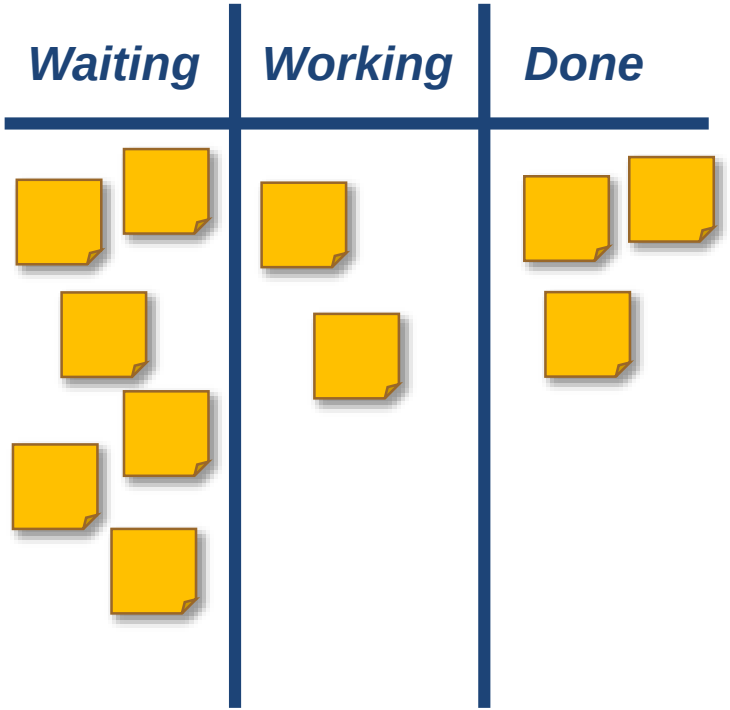
A measure of internal efficiency; time between:

- Developer committing verified code to a repository, and
- User operating the functionality that code enables for the first time

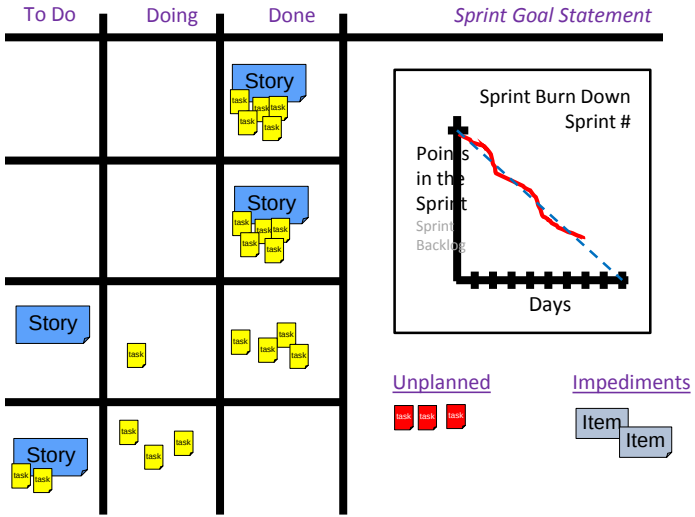


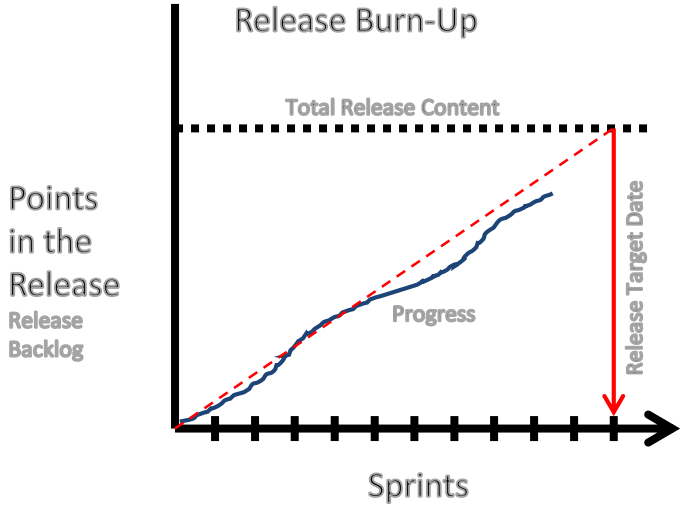
Scaling the Team-Based View to a Pipeline View

Visualizing work in process with a task board is a common building block of agile methodologies and frameworks.



Things You Can Expect to See Contractor Teams Measuring



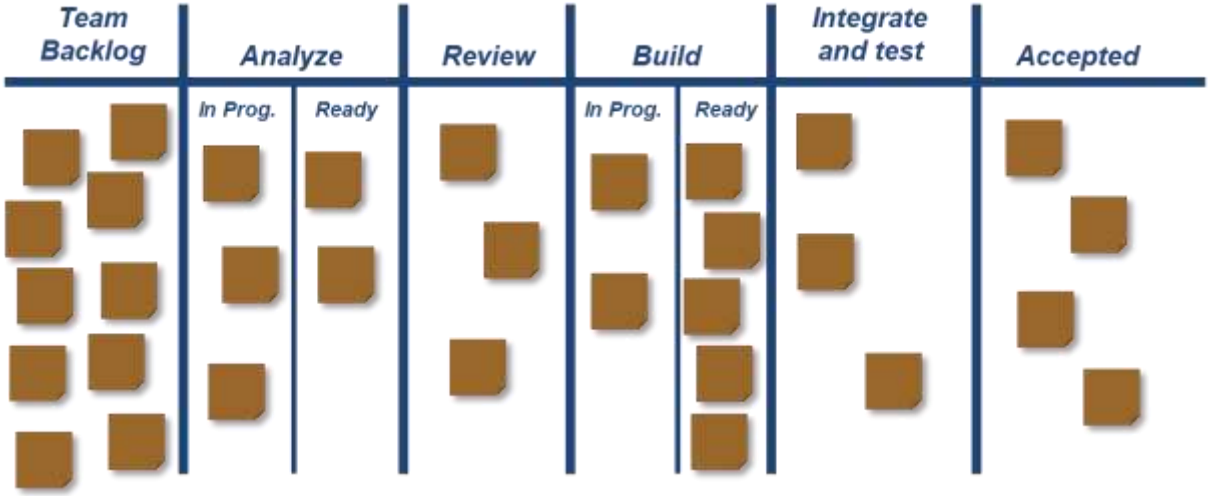


This is the Team's Data

Flow Efficiency

Calculating relative amount of time spent in progress versus waiting for the next state.

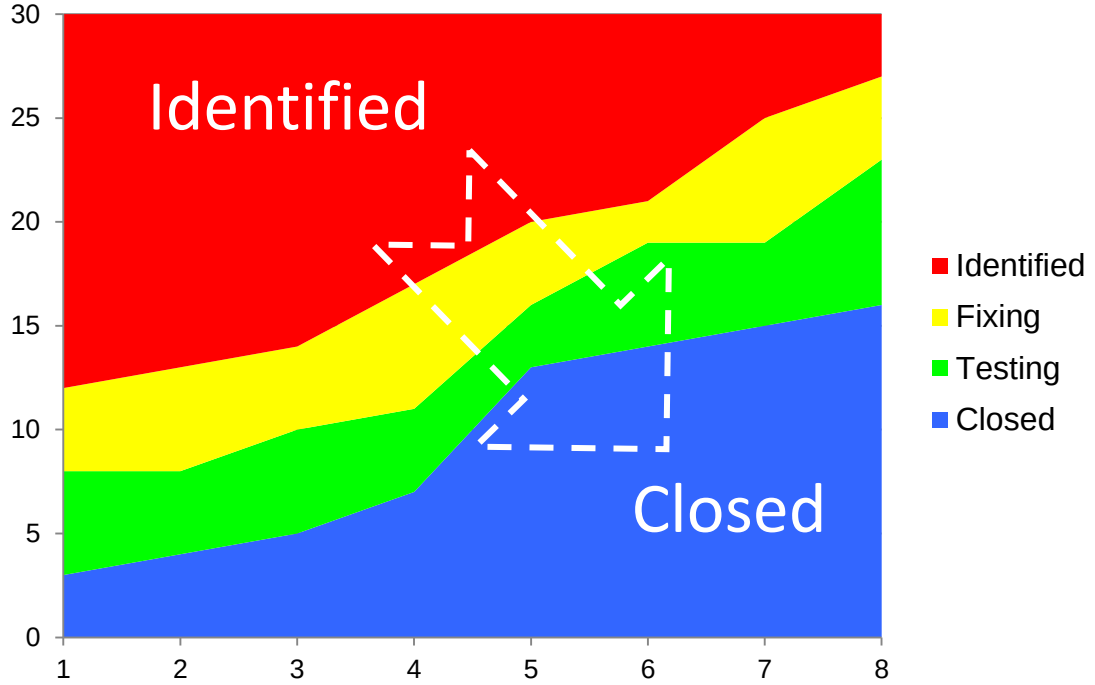
- Benchmark to understand typical range and extreme cases encountered



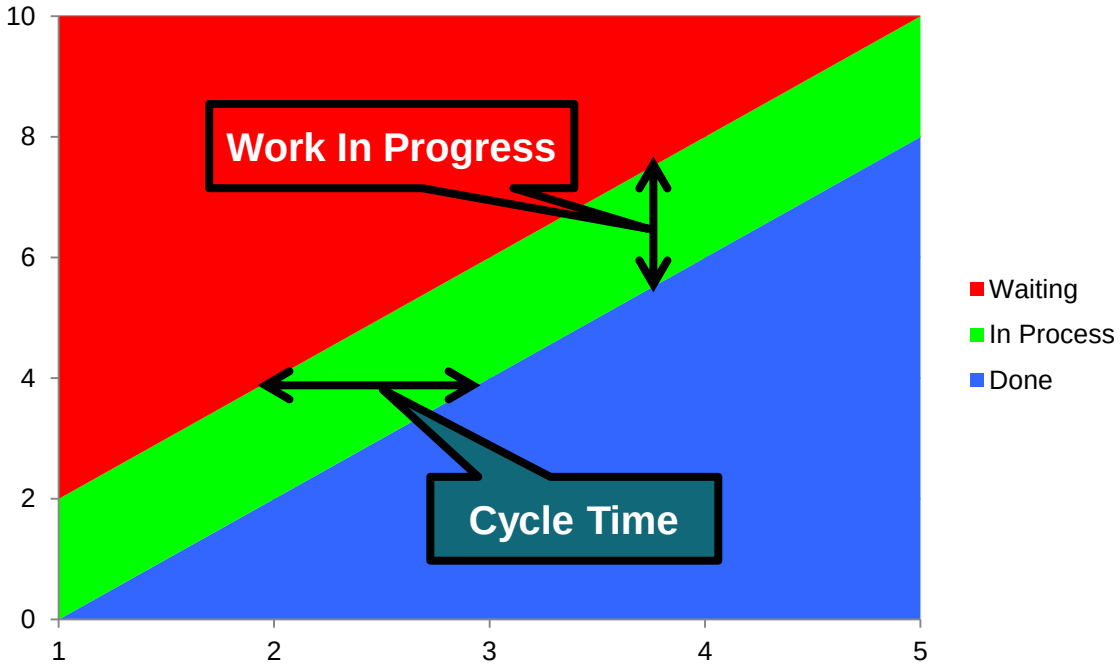
Constructing a Cumulative Flow Diagram$_4$

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

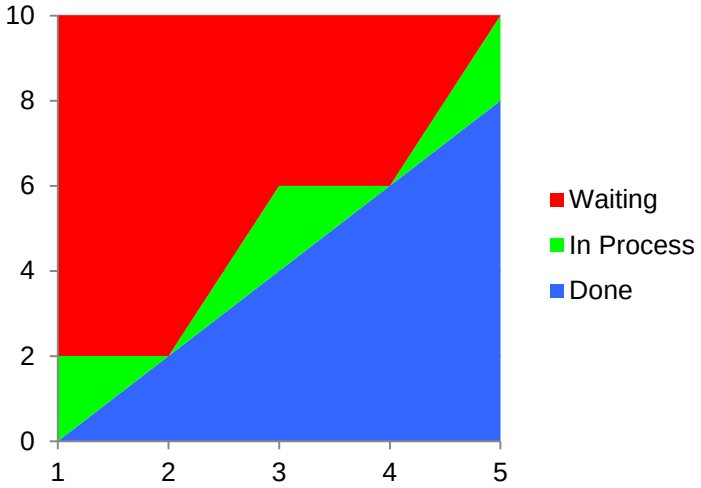
See backup slides for important considerations in constructing meaningful CFDs

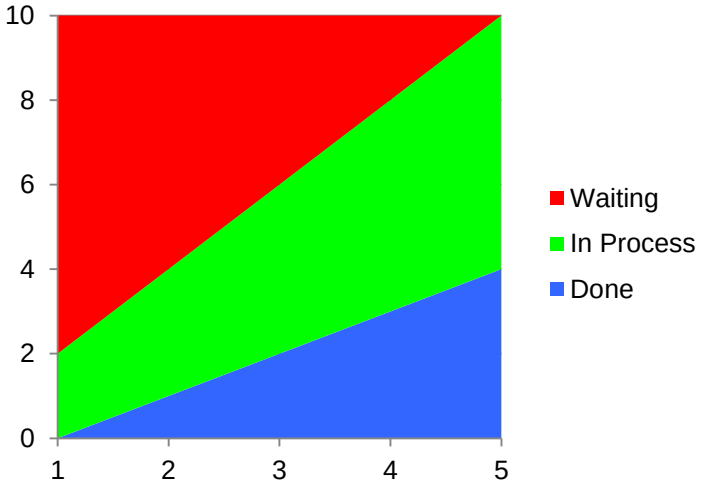


Tell-Tale Signals

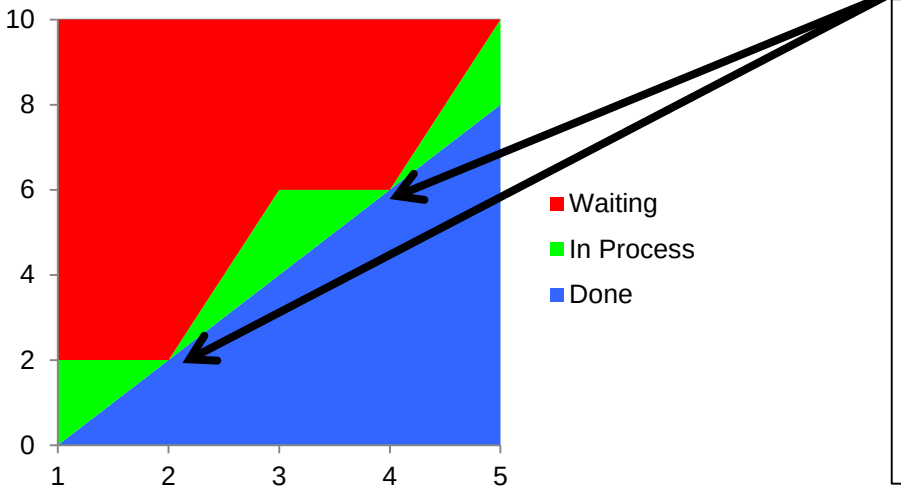


Exercise: What is Going on Here?





Exercise: What *MIGHT BE* Happening$_1$



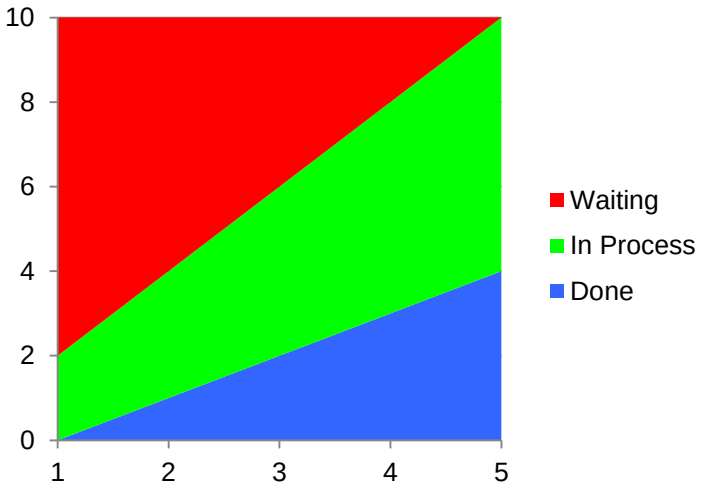
At time 2, and then again at time 4, the number of items “In Process” goes to zero.

- Have we lost the resource(s) that were preparing the items in the “Waiting” state?
- Is this intentional, due to limited resource(s) who can work on items in the “In Process” state?

Exercise: What *MIGHT BE* Happening$_2$

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

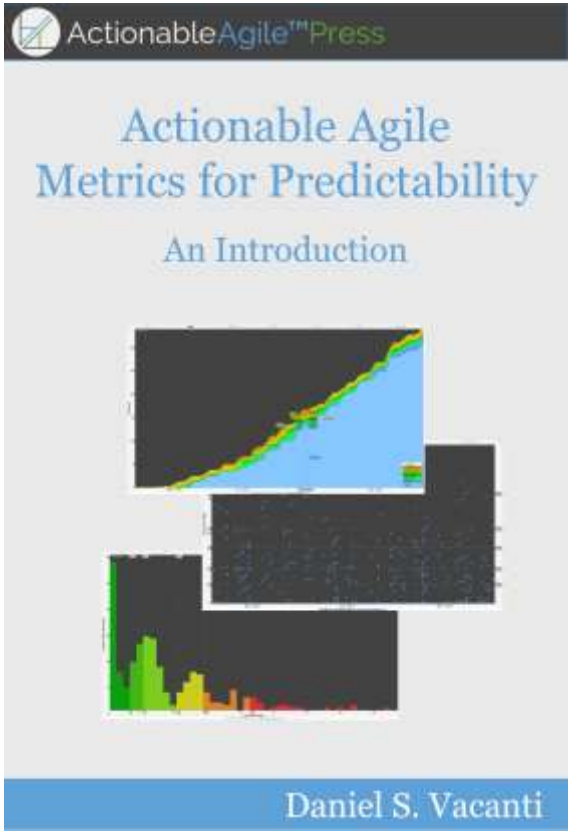
- The rate at which things enter “In Process” is greater than the rate at which things leave “In Process.”
- Are people moving onto new items without completing their work?
- Are new resources being added, who start new work at each time period?
- Are things moving into the “Done” state quickly enough?



If You Only Read One Book About Agile Measurement…This is It!!!!

Simply stated, flow is the movement and delivery of ***customer value*** *through a process.*

** Excerpted from page 11 of Daniel Vacanti's book*



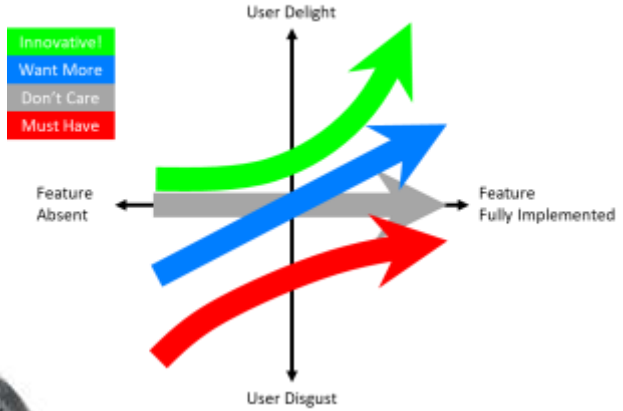
Are we delivering enough important capabilities to match the level of investment?

Warfighter-in-the-loop development

- Analysis of alternatives including Cost Of Delay
- Iteration- and Increment-level demonstrations

Trading cost and schedule for value

- Focus on the outcome, not just the inputs
- Relative estimation in context
- Precision in service of consistency



Are we delivering capabilities that satisfy the demands of the operational environment?

Quality measured in terms of the systems' operating environment

- Measured changes in system user workflows
- Change in system capacity, speed, interoperability, yield or volume of output
- Focus on 'fitness for purpose' in the field

Quality performance of engineering work

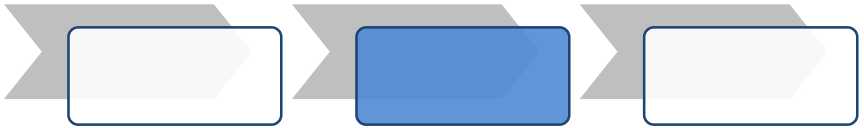
- Measuring speed to quality

Quality

Are we delivering capabilities fast enough to keep pace with the user's operational needs?

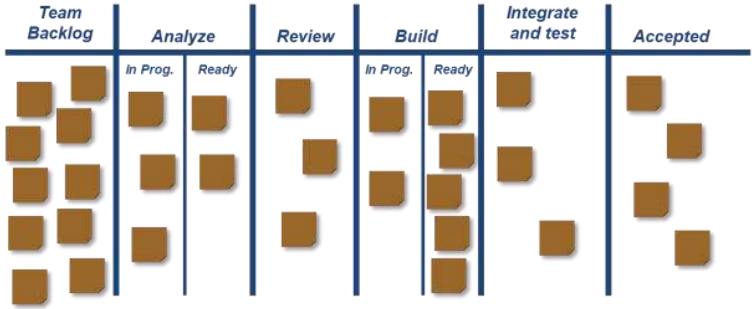
Time to field increments of functionality

- Cycle time benchmarks for value stream elements to measure and manage flow
- Active management of Work In Progress for Flow Efficiency



Rapid verification through automation

- Measuring shift of testing focus
- Coverage and speed of testing



Flow

A Few Agile and Measurement Resources

(These are also in your Learning Resources Package)

Long (1hr 24 min) Video from Will Hayes on Successful Agile Metrics:

https://www.youtube.com/watch?v=zYmmOpbtx_Q

SEI Agile Measurement Technical Note:

https://resources.sei.cmu.edu/asset_files/TechnicalNote/2014_004_001_77799.pdf

For specialists in EVM, there is an annual EVM Forum that usually includes Agile topics:

https://www.evmpforum.com/

Contact Information

Will Hayes

Principal Engineer

Continuous Deployment of Capability Directorate

Software Engineering Institute

Telephone: +1 412.268.6398

Email: wh@sei.cmu.edu

Web: www.sei.cmu.edu/go/agile

SuZ Miller

Principal Researcher
Continuous Deployment of Capability Directorate

Software Engineering Institute

Telephone: +1 412.268.9143

Email: smg@sei.cmu.edu

Web: www.sei.cmu.edu/go/agile

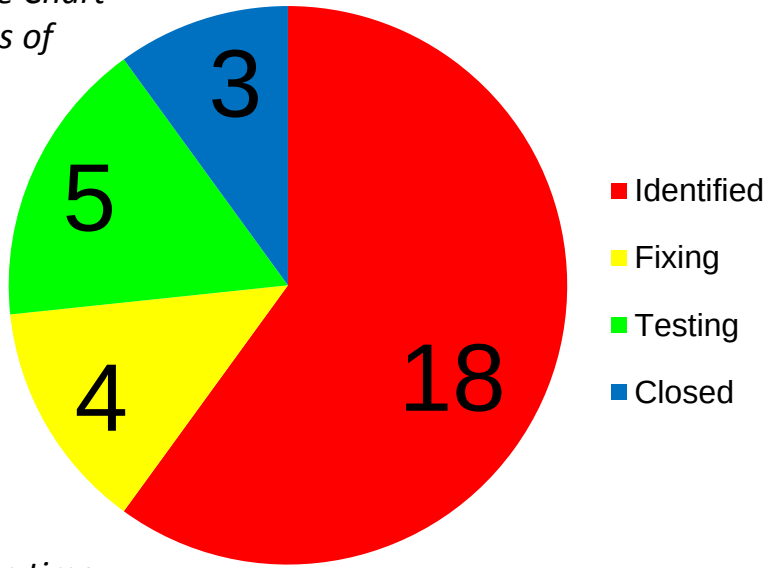
BACKUP

Useful Graphical Tool

How a Cumulative Flow Diagram Relates to Other Common Visualizations

Constructing a Cumulative Flow Diagram$_1$

Here we have a Pie Chart showing the status of 30 defects across the four stages of the defect handling life-cycle.



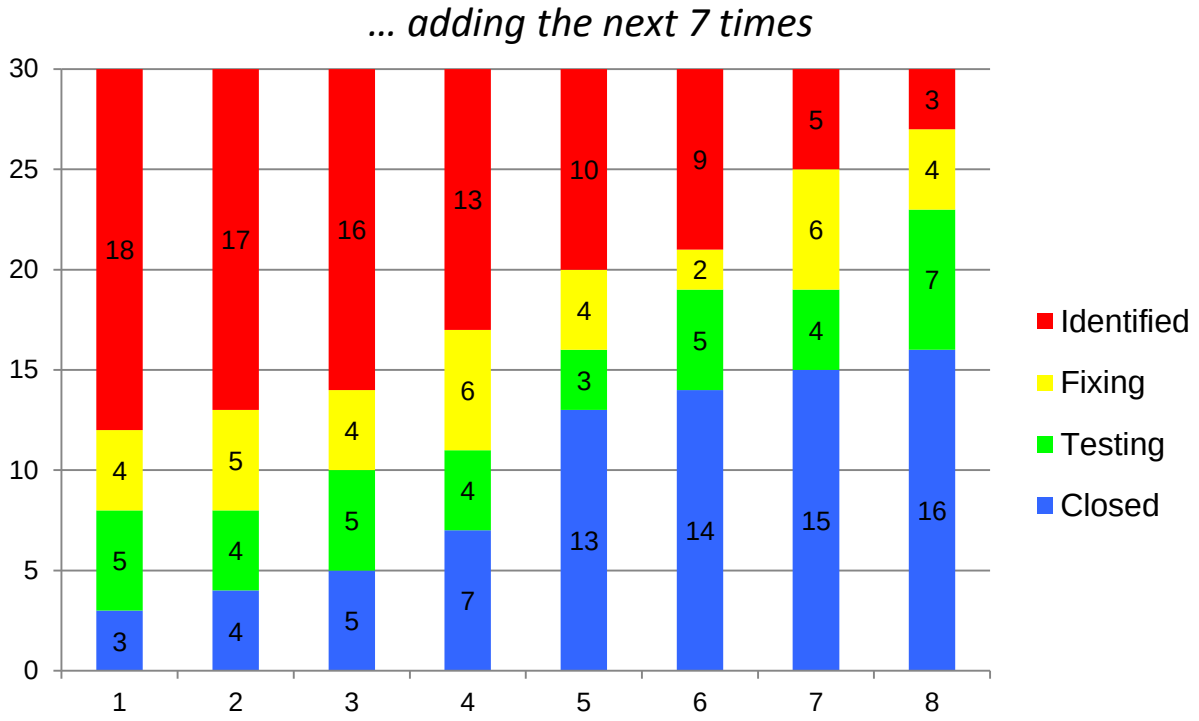
This is a snapshot for a single point in time.

Constructing a Cumulative Flow Diagram$_2$

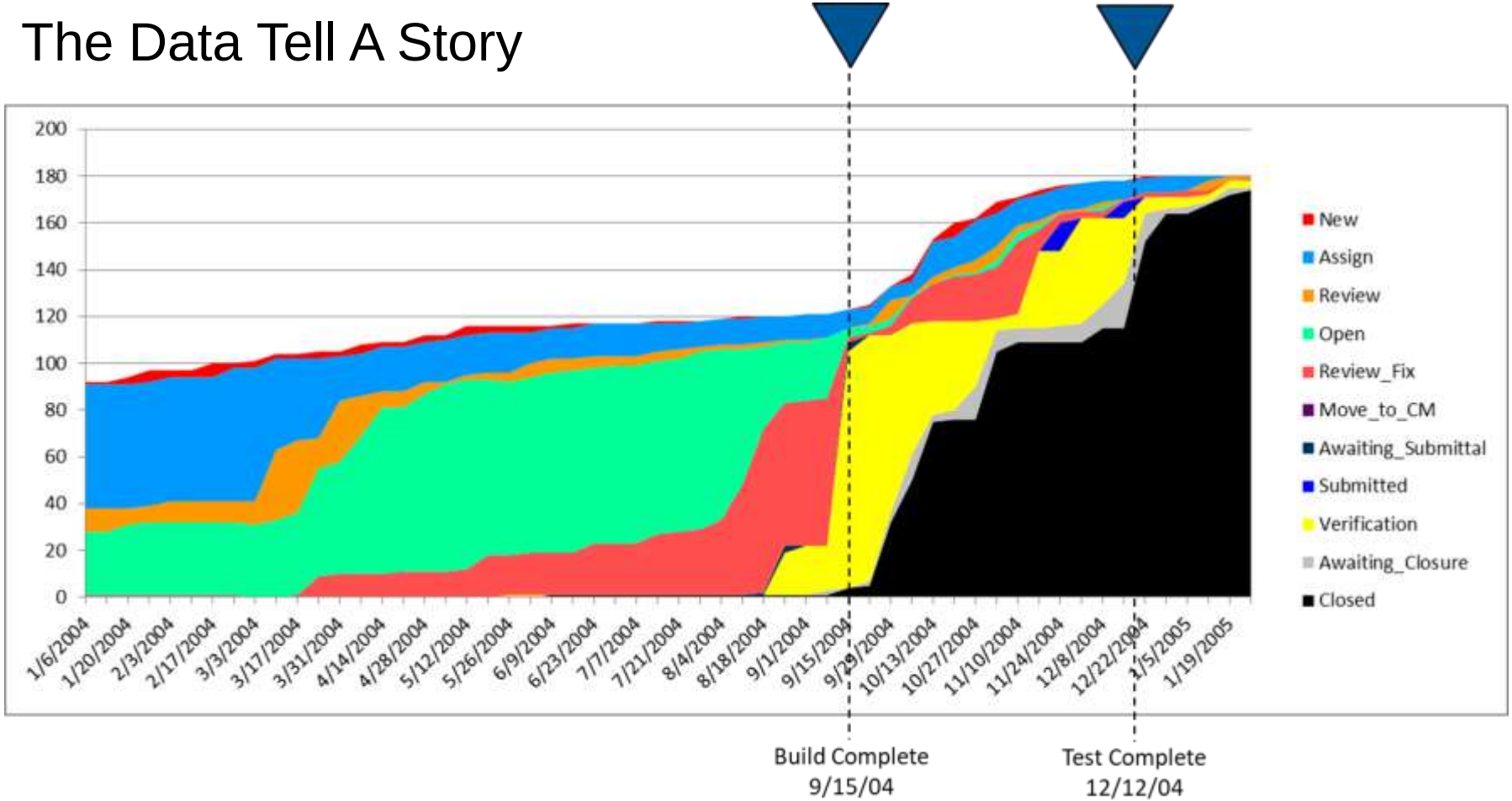
Same data, but presented in a stacked column chart

For a single point in time.

Constructing a Cumulative Flow Diagram$_3$



The Data Tell A Story



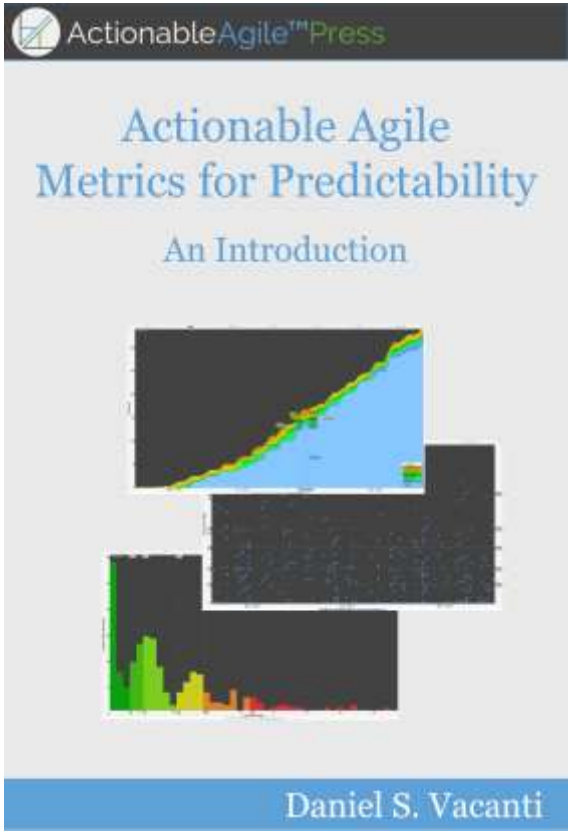
Work In Progress, Cycle Time and Throughput

Concepts that Underpin Measuring and Analyzing Flow

Flow and the Basic Metrics of Flow

Simply stated, flow is the movement and delivery of customer value through a process.

** Excerpted from page 11 of Daniel Vacanti's book*



The Three Primary Flow Metrics and Little's Law

Work In Progress - the number of items that we are working on at any given time

Cycle Time - how long it takes each of those items to get through our process

Throughput - how many of those items complete per unit of time

$$\textbf{Average}\ \text{Cycle Time} = \frac{\textbf{Average}\ \text{Work In Progress}}{\textbf{Average}\ \text{Throughput}}$$

Anti-Patterns Threatening Flow

Work items tend to pile-up at the bottlenecks in your process due to:

- Multi-tasking staff forced to spread time across a growing list of parallel work
- Introduction of new (hi priority) work which displaces work in progress
- Insufficient resources to perform a specialized task or work step
- Challenges in coordinating external dependencies

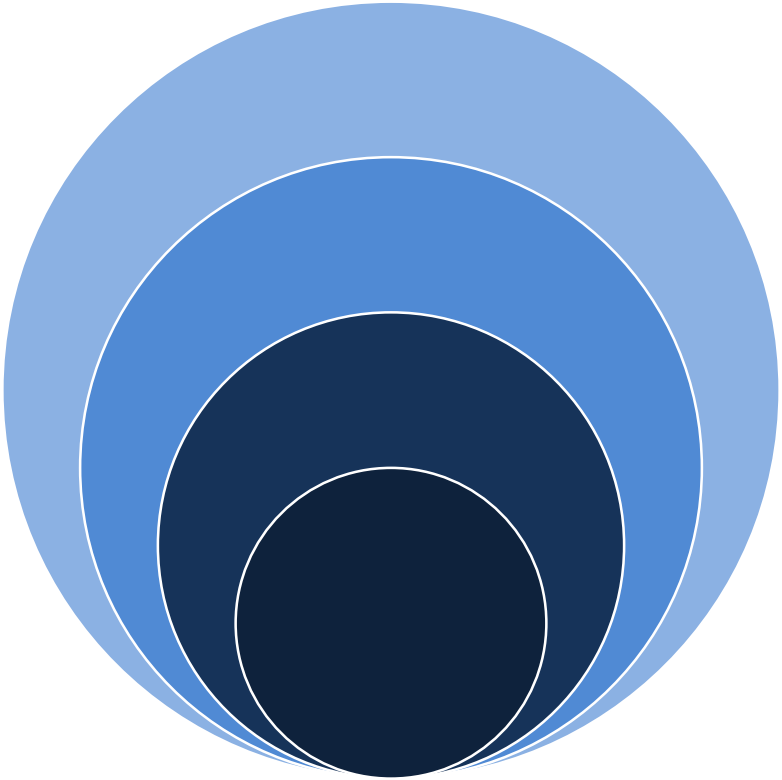
Unit of Analysis and Outcome of Interest

These methods can be productively applied at various levels of analysis

- Projects
- Releases
- Epics
- Stories
- Defects

Different outcomes are implied, as well as different data sufficiency considerations.

You must define “work item” for yourself.



Understanding Lead Time

Cycle Time – the time it takes to cycle through a process step is well understood.

Lead Time – a closely related concept, is focused on the time it takes to achieve a target state.

Both require a precise understanding of entry/exit criteria for your workflow.

Representing the Magnitude of the Work

Teams use story points to build a shared economic framework

- Standardizing this framework across an enterprise is expensive
- Re-calibration as teams learn becomes disruptive to enterprise
- The dis-incentive to act on feedback erodes utility for the team

Velocity is an amalgamation of different sources of variation

- True differences in the magnitude of work to be performed
- True differences in the complexity of problems to be solved
- Errors in people's ability to forecast each of the above and more
- Aggregation across teams does not make the data more accurate

Time Intervals Allow Precise Measurement

Variation in recorded cycle times reflects variation in how long it took to finish the work

- Limitations in our ability to foresee future events do not limit accuracy of data

The unit of measurement reflects what stakeholders directly care about

- When will the work be done?

Meaningful efficiency gains can be communicated well

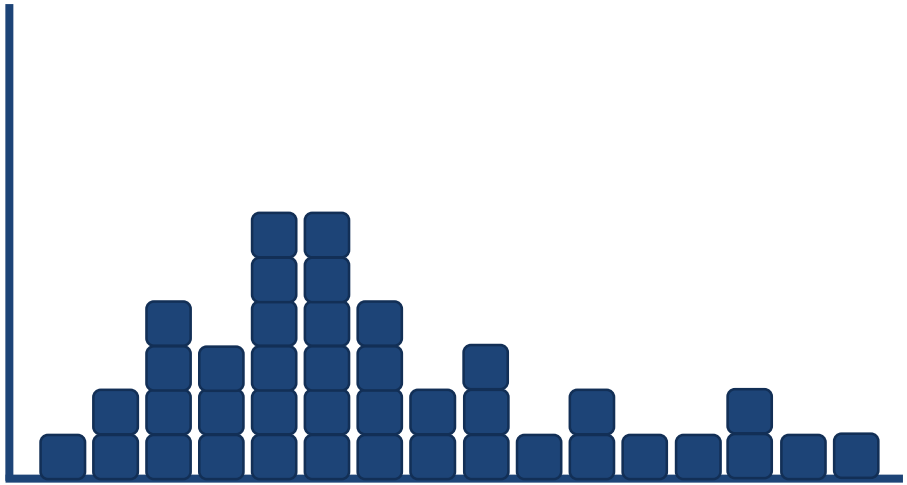
- We now consistently finish this activity 2 days earlier than past projects could

Useful Graphical Tool

Cycle Time Histogram

Cycle Time Histogram

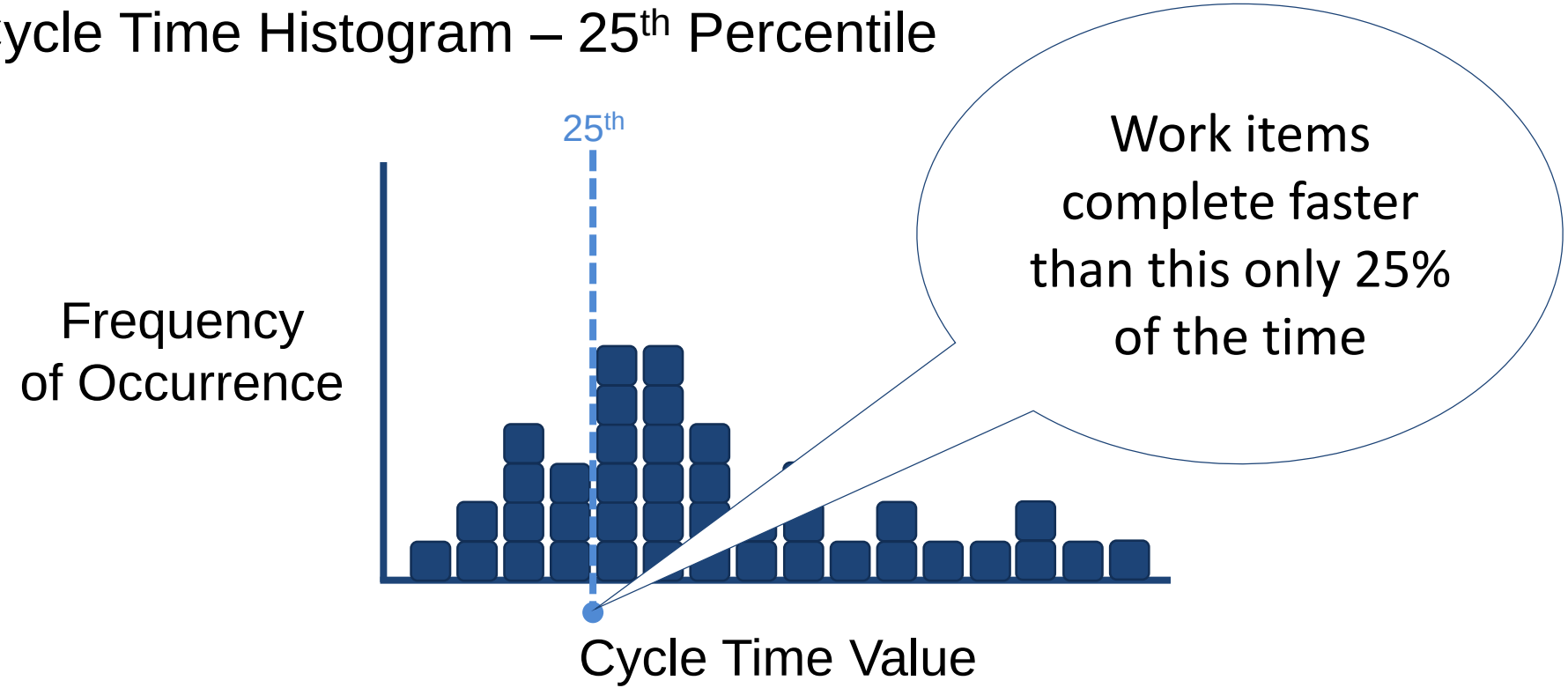
Frequency of Occurrence



Cycle Time Value

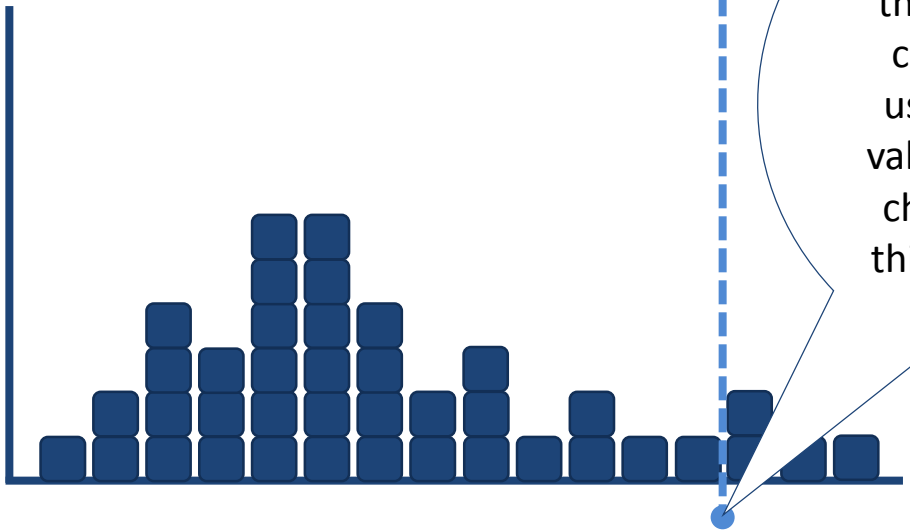
Sample Size = 40

Cycle Time Histogram – 25th Percentile



Sample Size = 40

Cycle Time Histogram – 90th Percentile



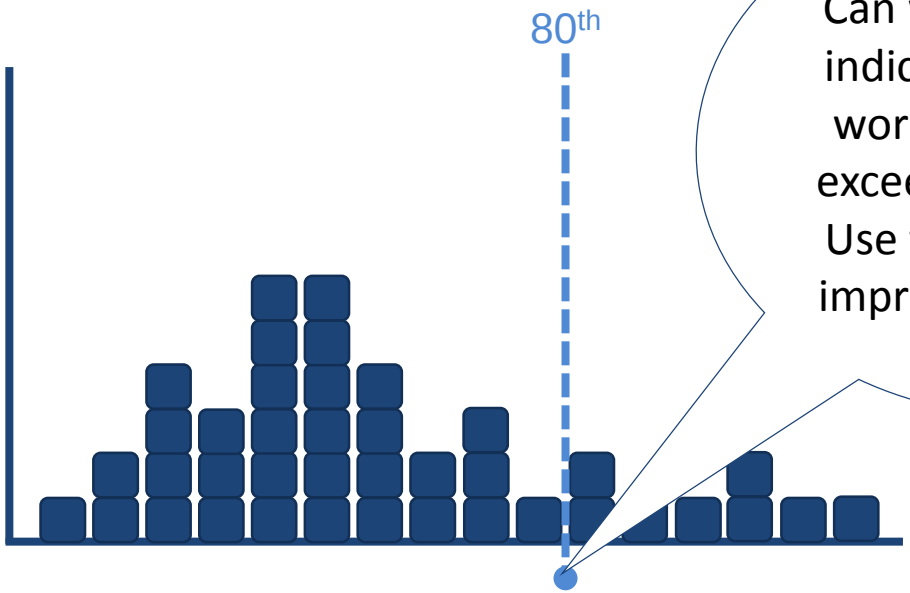
Frequency of Occurrence

Cycle Time Value

If work items complete in less than this 90% of the time, can we build a plan using this cycle time value – accepting 10% chance of exceeding this cycle time on any given item?

Sample Size = 40

Cycle Time Histogram – 80th Percentile



Can we build leading indicators to identify work items that will exceed this duration? Use this as a process improvement target?

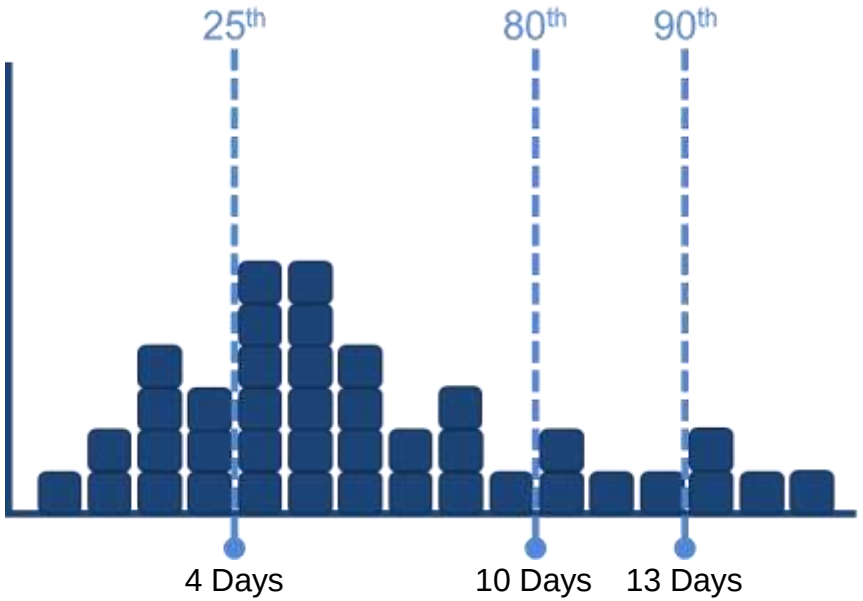
Sample Size = 40

Forecasting with Cycle Times

Implications

- Work items that start more than 13 days before planned release have a 90% chance of finishing in time
- Increasing the percentage of work that complete in less than 4 days is a goal for process improvement
- Decreasing the percentage of work that completes in more than 10 days removes unwanted variance

Big changes in the workload or work processes require new benchmarks



Cycle Time Benchmarking

Benchmark using percentiles (e.g., 25th and 85th) to avoid distraction from outliers

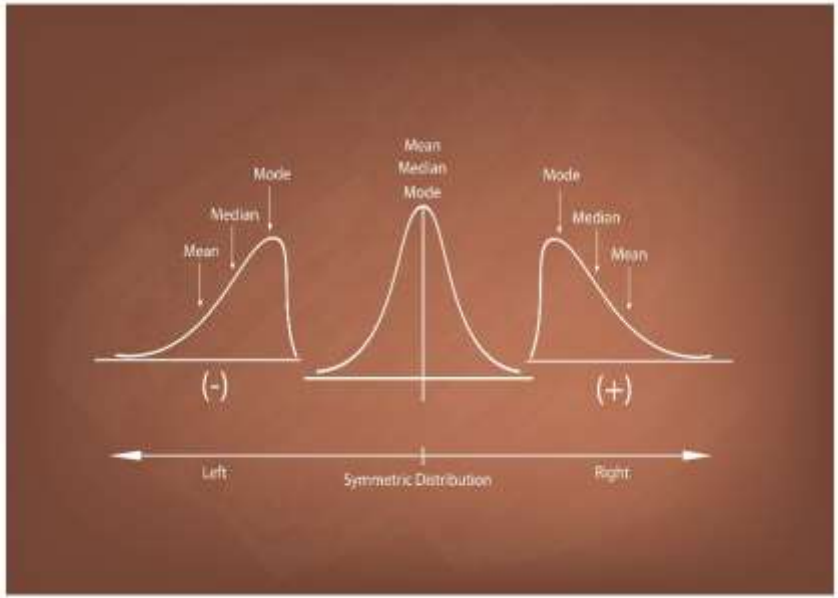
- The “average” may shift dramatically due to extreme data points that occur only rarely

Use the benchmarks to understand the feasibility of targets

- Yesterday’s weather helps anticipate today, but is not a perfect predictor

Targets for improvement need to be informed by knowledge of the data

- Forecast with range of likely outcome and/or level of confidence



Process Flow by the Numbers

Understanding the probable range of cycle times for components allows us to:

- Assess the probable performance of the value stream overall (e.g., using Monte Carlo Simulations)
- Screen for potential bottlenecks in flow
- Identify patterns of flow debt to remedy
- Simulate the effect of hypothetical changes in performance