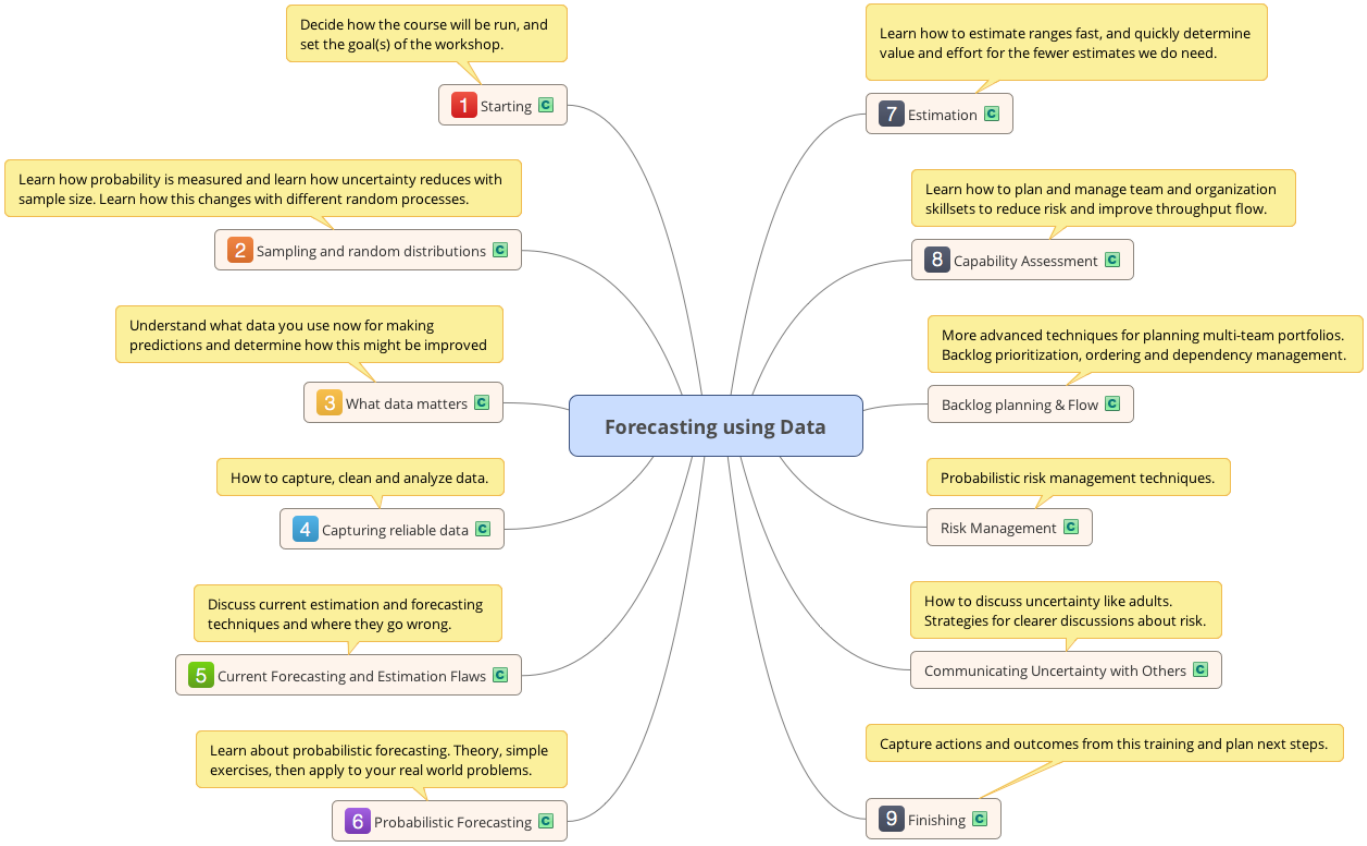
Focused Objective
forecasting - risk - staff - cost of delay

Forecasting using Data

Capturing and using data for forecasting



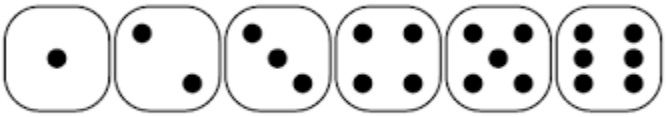
Workshop Manual

Digital materials: http://Bit.Ly/ForecastingUsingData
Email me: troy.magennis@focusedobjective.com

Version 3.0

Understanding probability - Exercises

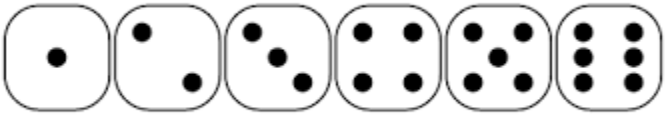
Q1. How many different possible values are there for a standard six-sided dice?



A:

Q2. How many values of a six sided dice are less than 4?

Tip: Circle the values that are less than 4.



A:

Q3. What is the probability of rolling a value less than 4 on a standard six side dice?

Tip: Count the number of “right” values and divide by the total number.

$$p = \frac{Number\ of\ "right"\ values}{Total\ possible\ values}$$ A:

Q4. What is the probability of rolling at LEAST a 2 on a standard six side dice?

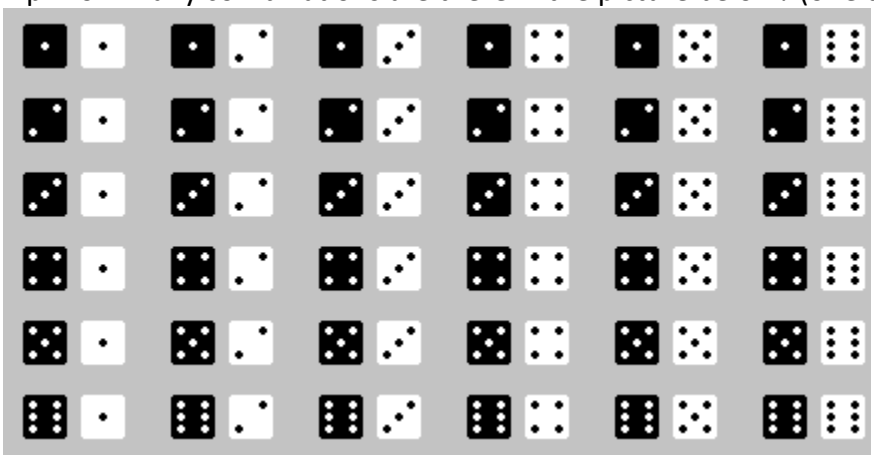
A:

Q5. What is the probability of rolling a value less than 5 on a standard six side dice?

A:

Q6. How many possible outcomes are there for rolling two fair six sided dice?

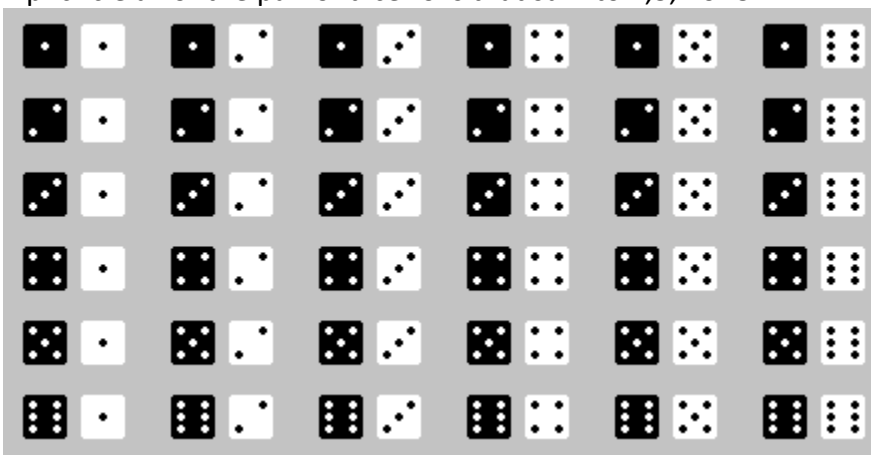
Tip: How many combinations are there in the picture below? (one dice is black, the other white)



A:

Q7. How many values (sum of the two dice) are less than 6?

Tip: circle all of the pair of dice rolls that sum to 2,3,4 or 5



A:

Q8. What is the probability of rolling a combination of less than 6?

Tip: Count the number of "right" values and divide by the total number.

$$p = \frac{Number\ of \text{correct}\ values}{Total\ possible\ values}$$ A:

Answers: Q1: 6 Q2: 3 Q3: 3/6 = 0.5 Q4: 5/6 = 0.83 Q5: 4/6 = 0.67 Q6: 36 Q7: 10 Q8: 10/36 = 0.278

Prediction Intervals – Estimating the chance of where future samples will fall

What is a prediction interval?

Source: https://en.wikipedia.org/wiki/Prediction_interval

In statistical inference, specifically predictive inference, a prediction interval is an estimate of an interval in which future observations will fall, with a certain probability, given what has already been observed.

Estimating the range of actual data by random sampling

When actual data samples can be observed, its handy to know how likely it is that you have discovered the range of likely values. This is useful in understanding how likely there is a lower or higher sample yet to be discovered. Like all random sampling, there is absolutely no guarantee that you have discovered any amount of the range, but prediction intervals give you the probability on average.

Probability the next sample is within the previously seen range after "n" samples = $\frac{(n-1)}{(n+1)}$ X 100

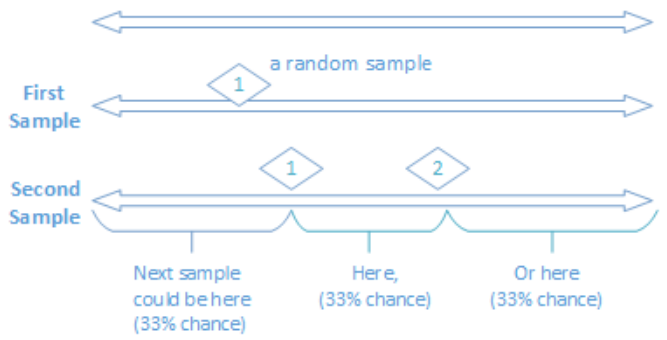
Probability the next sample is lower than the lowest sample so far after "n" = $\frac{1}{(n+1)}$ X 100

Probability the next sample is higher than the highest sample so far after "n" = $\frac{1}{(n+1)}$ X 100

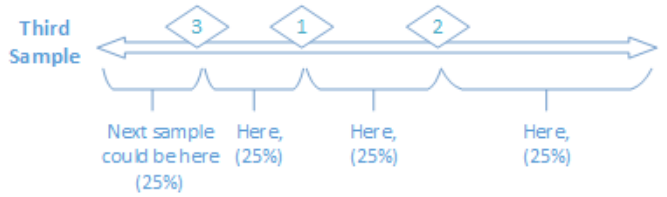
Samples so far (n)	Probability for each interval	Probability next sample in range
1	50.00%	0.00%
2	33.33%	33.33%
3	25.00%	50.00%
4	20.00%	60.00%
5	16.67%	66.67%
6	14.29%	71.43%
7	12.50%	75.00%
8	11.11%	77.78%
9	10.00%	80.00%
10	9.09%	81.82%
11	8.33%	83.33%
12	7.69%	84.62%
13	7.14%	85.71%
14	6.67%	86.67%
15	6.25%	87.50%

Samples so far (n)	Probability for each interval	Probability next sample in range
16	5.88%	88.24%
17	5.56%	88.89%
18	5.26%	89.47%
19	5.00%	90.00%
20	4.76%	90.48%
21	4.55%	90.91%
22	4.35%	91.30%
23	4.17%	91.67%
24	4.00%	92.00%
25	3.85%	92.31%
26	3.70%	92.59%
27	3.57%	92.86%
28	3.45%	93.10%
29	3.33%	93.33%
30	3.23%	93.55%

Q. How can we estimate the chance the next random sample is within the previous found range?



After two samples, there are three spots the next sample could be. Equally splitting the chances, there is a 33.33% chance the next sample is between the previous samples (1) and (2).



After three samples, there are four spots the next sample could be. Equally splitting the chances, there is a 50% chance the next sample is between the lowest (3) and highest (2) so far.

Important assumptions (that are rarely perfectly true)

- The samples are taken at random. Convenient isn't random!
- The distribution is uniform – all values have equal chance.
- The probability is on average. It's when it is more likely than not (~50%)

These are rarely always true in the real world. Milage will vary depending mainly on the underlying distribution. If the distribution is skewed, it can take hundreds of samples to get the lower probability end of the range.

For more probabilistic forecasting resources: FocusedObjective.com Email: troy.magennis@focusedobjective.com

Prediction Intervals Exercise

To find how many samples it takes to find the lower and upper bounds of a sample set on average? This exercise simulates finding the upper and lower boundary of a sequential range by sampling the result of dice rolls.

3 x 6 Sided Dice

2 x 10 Sided

42

7

99

Note: Rolling a 00 and 0 = 100

The process

1. **Roll Dice**: Create a random number with a range of 1 to 100. Options:
    a. A random number generator app on your phone (Randomizers)
    b. Use three rolls of a six-sided dice (see next page for chart)
    c. Sum two 10 sided dice (00 – 90 by 10's) and a traditional (0-9)
2. **Repeat**: Repeat 20 times and record the results in the table below.
3. **Examine Results**: Look at the range between the lowest rolled and highest rolled. Compare against expected.

Questions and discussion topics

1. **What probability distribution is a single roll?**
2. **What guarantee do I have that I have found the range expected?**
3. **What happens if the data is a Normal (bell curve) distribution?**
4. **What happens if the data is left or right skewed?**

Results table

Record each roll & calculate the ranges seen so far after each roll. Compare to expected.

Roll (n)	Value of This Roll	Lowest value seen So Far	Highest Value seen So Far	Range So Far = Highest-Lowest	Expected Range (after roll) $\frac{(n-1)}{(n+1)} \times 100$
1					0
2					33.3
3					50
4					60
5					66.6
6					71.4
7					75
8					77.8
9					80
10					81.2
11					83.3
12					84.6
13					85.7
14					86.7
15					87.5
16					88.2
17					88.9
18					89.5
19					90

Basic Monte Carlo Forecasting – Manually plot work completion from dice rolls of throughput/velocity (1 to 6)

Discover what Monte Carlo forecasting is by performing it by hand. This exercise simulates completing a project many times and plots the outcomes. Perform 7 more trials. Each trial involves filling all rows in a column until the remaining work count reaches zero.

1. Throw a six-sided dice and subtract the number in the row above by this dice roll.
2. When a column reaches zero (or less, just enter 0), move onto the next trial column.
3. Plot each trial as a line graph on the following page. Trial 1, has already been plotted for you.

Week	Trial 1 5,1,6,4,2,2	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7
1 (start)	20	20	20	20	20	20	20
2	-(5) = 15						
3	-(1) = 14						
4	-(6) = 8						
5 (shortest)	-(4) = 4						
6	-(2) = 2						
7	-(2) = 0						
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21 (longest)							

4. Compute the probability. Put best (shortest) # weeks in 1/7th, next shortest in 2/7th, etc.

1/7th chance (14%) =

2/7th chance (29%) =

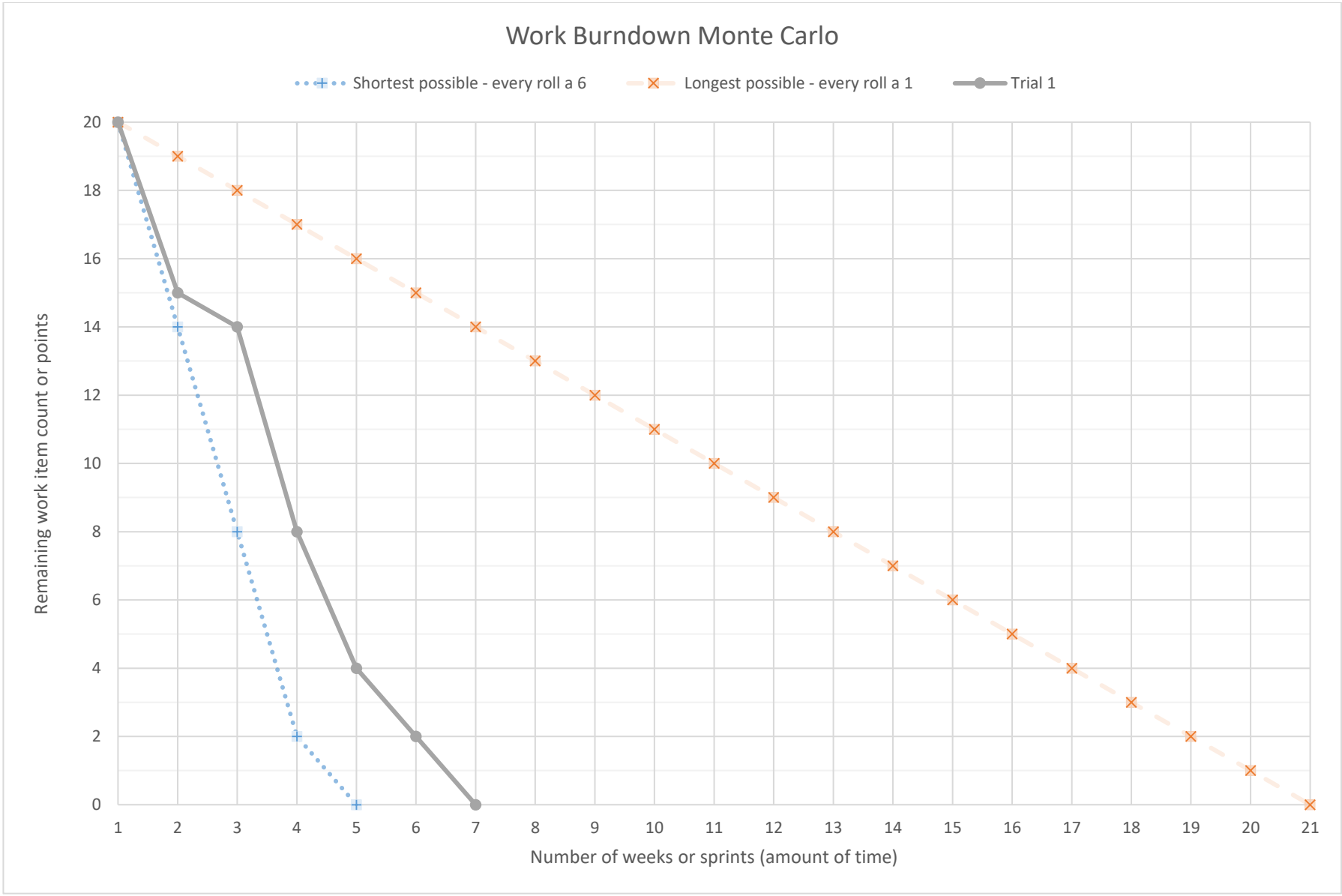
3/7th chance (43%) =

4/7th chance (57%) =

5/7th change (71%) =

6/7th chance (86%) =

7/7th chance (100%) =



Circle the 4th line counting from the left (the lowest) – this has a 4/7th chance (> 50%)

Total Story Count Monte Carlo Forecast Exercise – Read First

Aim

To practice estimating the total number of stories (or story points) in a larger project by sampling just a few feature or epic story count breakdown examples. The goal is to get an estimate of total story count without having to analyze very feature or epic in a proposed project.

Facilitation

4. Discuss with the group the goal "We are going to estimate how many stories (or points) in 10 features."
5. Discuss the basic method "We are going to randomly build sets of 10 story counts and total how many stories are in each group of 10. We will do this for 11 trials, and these trials will allow us to understand the probability of each result. To save time, you just have to build the first two trials, the others are already done for you."
6. Discuss what the samples are "We will be sampling actual story count examples performed by a team. The samples are counts of 36 prior features, but it could be far less, even as few as 7. You can prove this by crossing out some values in random samples and rolling again if you get one of those scrubbed out samples."
7. Discuss how we learn probability "After we have built and totaled the 11 trials, we will count how many trials rounded down to the nearest ten value for simplicity. These counts will tell us how "probable" that many stories are likely. Likelihood is simply the ratio of how many trials are in a group divided by 11 (the number of trials). When done by spreadsheet we might do 1000 trials, but to save time we are just doing 11"

Questions and discussion topics

5. **What could pollute the story count samples (make them a poor predictor of the future)?**
6. **Why can't we just use the average or median values to forecast the story count in 10 features?**
7. **If life depended on this forecast, how many stories would you sign-up for?**
8. **How might you choose a likelihood to target in your company?**
9. **How would you get more definition in the likelihood percentages?**
10. **What does 100% likelihood mean in this case?**

Why it works

If we sample at random the number of stories in features (or epics) analyzed by teams, then we can forecast the number of stories in any given number of features. The assumption is that the pattern in the samples is representative across the rest of the features not analyzed by the team. This will only be true if the features analyzed by the team are actually chosen truly at random. We help you do this here by rolling a two six-sided dice (or one six-sided dice twice). This technique build trials, actual possible sets of 10 features. Each one will be different, but the pattern in the results helps understand how likely each value is by its ratio across the whole set of trials. This is best done by spreadsheet, but this exercise is for learning.

Resources

This exercise is for learning purposes, don't do it by hand!

Exercise – How Many Stories in 10 Features (or Epics)

Aim: To estimate how many total stories there would be for 10 features (or epics). To understand the probability of achieving those story count estimated based on prior sampled history.

1. **Throw dice and record sample in the empty trial cells below using the sample sheet on the next page**

Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11
		2	12	3	12	3	1	3	1	6
		5	15	3	5	10	10	2	6	3
		4	4	3	20	3	3	2	3	2
		3	2	2	2	3	3	3	4	6
		8	6	1	6	6	3	1	2	4
		8	10	8	3	1	2	1	3	12
		1	2	1	3	1	2	5	1	3
		1	1	4	5	3	10	1	1	10
		15	12	3	3	6	6	4	15	12
		6	3	1	8	2	1	6	20	3

2. **Sum each column above and enter the result in the cell below (use your phone calculator!)**

		53	67	29	67	38	41	28	56	61

3. **Round each sum above down to the nearest "ten" e.g. 10+, 20+, 30+, 40+, etc.**

		50+	60+	20+	60+	30+	40+	20+	50+	60+

4. **Compute the probabilities of achieving each result by counting the trials in each group**

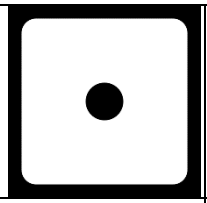
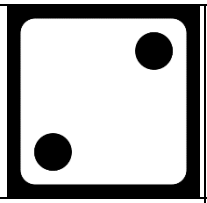
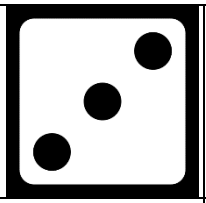
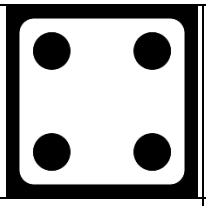
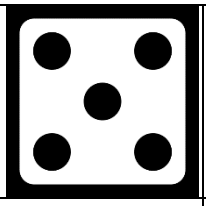
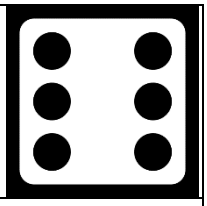
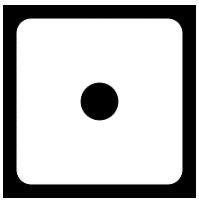
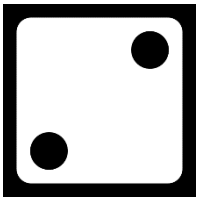
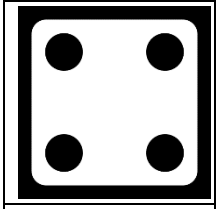
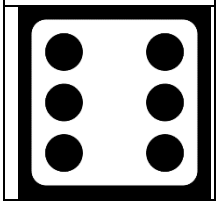
Total Stories in 10 Features (or epics)	Count trial sum groups at least 30,40, 50, etc. stories	(Count / 11) Likelihood
At least **20** stories	11	1
At least **30** stories		
At least **40** stories		
At least **50** stories		
At least **60** stories		
At least **70** stories		
At least **80** stories		
At least **90** stories	0	0

This value is 0 to 1 Multiply it by 100 to get a percentage. 0% = no chance, 100% means every trial achieved at least

Samples: Random Samples of Epic to Story Count

To generate random samples from the story count history, throw two six-sided dice (or throw one six-sided dice twice) and use the sample value at the intersection of the two dice results. It's important to make certain samples are taken at random, and using a dice is often the fairest way to ensure you don't introduce bias!

First dice throw

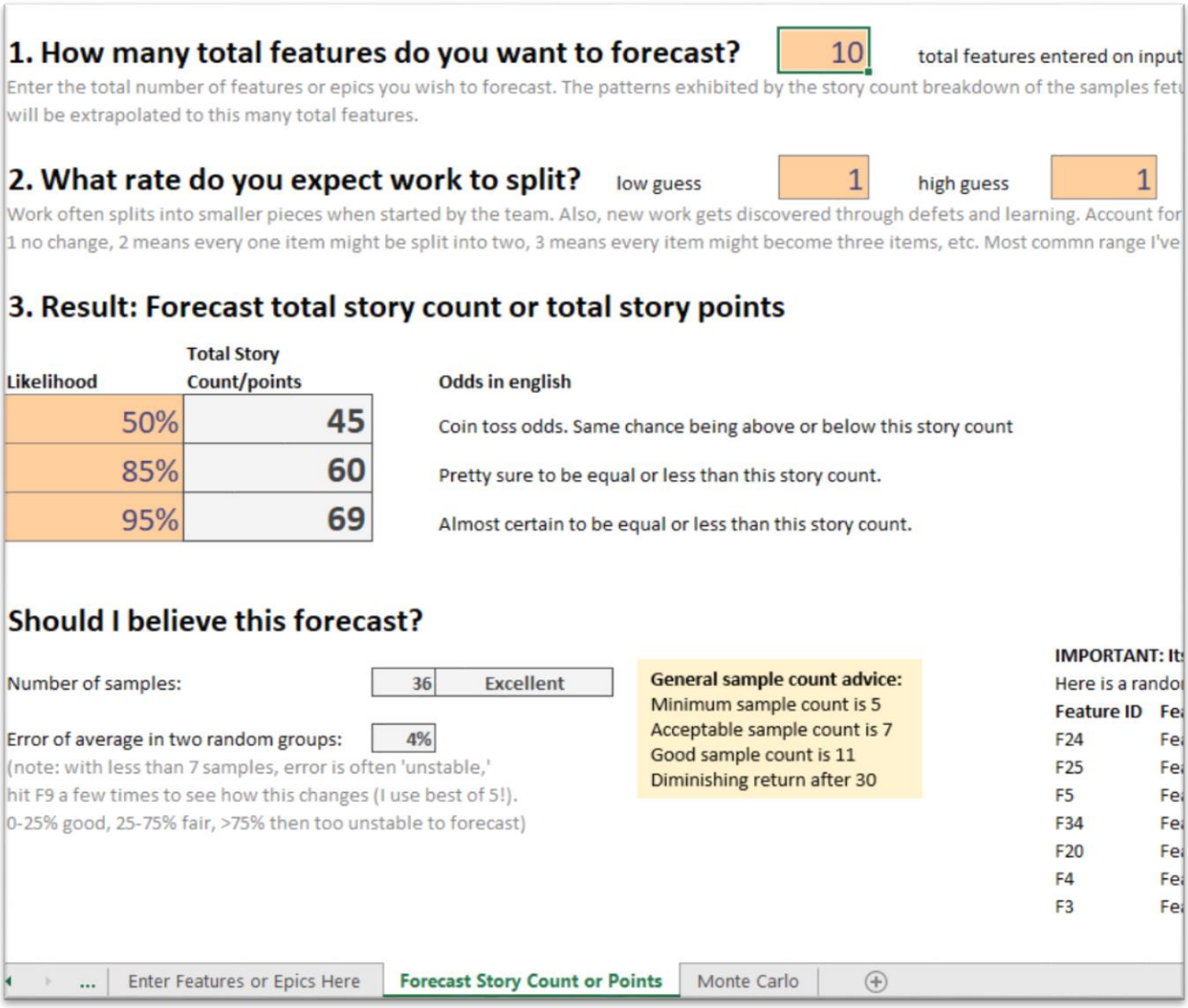
						
	2	5	1	3	8	4
	10	2	20	3	4	1
	2	12	15	5	2	1
	3	1	10	2	3	1
	3	3	8	4	3	1
	6	3	3	6	4	2

Second dice throw

These samples were from an actual project. The team selected 46 epics (features) at random out of 328 and broke them down into story level. They could have stopped at 10 an got the same

Total Story Count Forecasting using Spreadsheet

Download the "**Story Count Forecaster – exercise.xlsx**" spreadsheet from the spreadsheets folder at http://Bit.Ly/SimResources This spreadsheet has the feature story count sample data pre-loaded into the "Enter Features or Epics Here" worksheet.
Click on the "Forecast Story Count or Points" worksheet and enter the number of features you want to forecast, for example 10 as shown here –



Exercises –.

1. The average of all story count samples is 4.6. That would be 46 total story count for 10 features. What probability is that equal to based on your analysis by-hand and spreadsheet?
2. Forecast total story count for 100 features.
3. Delete the estimated story count for all but 7 features. Did the forecast change much?

.

Throughput Forecast Monte Carlo – Read First

To estimate the number of stories that will be completed by a team for a six (6) week timespan using historical weekly throughput samples from the same team. To understand the probability of achieving those estimates.

The process

1. **Simulate one possible result**: A single six week throughput result is simulated (called a trial) by summing together six historical one-week throughput samples picked at random.
2. **Repeat**: This simulation process is repeated many times (eleven here, but it can be thousands of repetitions). Each trial represents a "possible" six-week throughput result given the team's historical rate of delivery.
3. **Calculate likelihoods**: The proportion of trials that meet or exceed a given throughput value versus the total number of trials is the likelihood that value is achievable in the future.

Questions and discussion topics

11. **What could pollute the throughput samples (make them a poor predictor of the future)?**
12. **How might you correct for these sample pollution events?**
13. **Why can't we just use the average or median values to forecast the next six weeks?**
14. **If life depended on this forecast, how many stories would you sign-up for?**
15. **How might you choose a likelihood to target in your company?**
16. **How many trials were needed before the actual average (57.75) was included in the range you saw?**
17. **How would you get more definition in the likelihood percentages?**
18. **What does 100% likelihood mean in this case?**
19. **How would you track progress against this forecast?**
20. **What is the impact of not returning the sample each time?**

Why it works

Historical throughput data for teams measures delivery rate for a wide portion of the development system (the wider the better). Team throughput per week accounts for delays; for example waiting time, impediments, staff availability, interruptions and un-recorded work. The impact of these delays is more significant to a forecast than the hands-on time alone. This is a reason developer estimates are unreliable when forecasting projects, they don't account for delays and system dynamics. In a stable system (e.g. the team isn't blown-up), throughput will be a good predictor of future delivery rate even with large item size variability.

Team Throughput Sample Data

Samples represent the number of stories completed per week by the same team taken from an actual project.

Samples: 16,3,10,6,19,11,17,17,15,9,11,8,5,13,5,7,8,6,10,10,8,5,5,7

Count: 24 Sum: 231 Minimum: 3 Median: 8.5 Average: 9.625 Maximum: 19

Resources

Forecasting spreadsheets: https://github.com/FocusedObjective/FocusedObjective.Resources
(these spreadsheets do the process described here thousands of times instantly. This exercise is for learning purposes, don't do it by hand!)

Exercise – Throughput Forecast Monte Carlo Worksheet

Aim: To estimate the number of stories that will be completed by a team for a six (6) week timespan using historical weekly throughput samples for that team. To understand the probability of achieving those estimates.

Process:

5. Shuffle the 24 throughput cards or dice (whichever method you choose)
6. Pick a card at random or throw dice and record sample in the table below
7. Return the card to the deck and reshuffle ("sample with replacement")
8. Repeat until all squares are filled

We randomly sampled trials 4 to 11 for you to save

Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11
			7	11	7	5	17	5	10	16
			19	7	10	5	13	13	5	7
			6	5	5	3	5	16	6	5
			6	19	5	3	5	3	6	3
			5	7	10	5	6	8	8	6
			5	7	19	10	16	8	10	16

9. **Sum of all samples for each trial by column (upper) / Nearest "tens" grouping rounded down (lower)**

			48	**56**	**56**	**31**	**62**	**53**	**45**	**53**
			40+	50+	50+	30+	60+	50+	40+	50+

10. **Probabilities of achieving at least n stories for a six-week timespan**

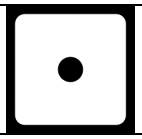
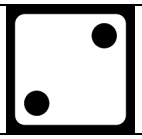
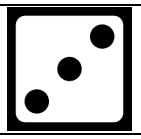
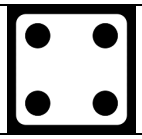
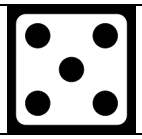
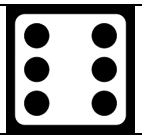
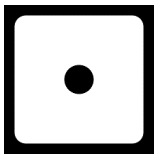
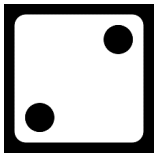
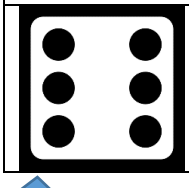
Six Week Throughput	Count trial sum groups at least 30,40, 50, etc. stories	(Count / 11) Likelihood
At least **30** stories		
At least **40** stories		
At least **50** stories		
At least **60** stories		
At least **70** stories		
At least **80** stories		
At least **90** stories		

This value is 0 to 1 Multiply it by 100 to get a percentage. 0% = no chance, 100% means every trial achieved at least this level.

Samples: Random Samples of Throughput by Six-Sided Dice

To generate random samples from the throughput history, throw two six-sided dice (or throw one six-sided dice twice) and use the sample value at the intersection of the two dice results. It's important to make certain samples are taken at random, and using a dice is often the fairest way to ensure you don't introduce bias!

First dice throw

						
	16	3	10	6	19	11
	17	17	15	9	11	8
	5	13	5	7	8	6
	10	10	8	5	5	7
	Roll again	Roll again	Roll again	Roll again	Roll again	Roll again
	Roll again	Roll again	Roll again	Roll again	Roll again	Roll again

Second dice throw

Capture Recapture Exercise 1: **DO NOT START UNTIL ASKED**

Circle the spelling mistakes in the following paragraph.

George new that he shouldn't drink alchohol on a Wedsday night, especially since his govermnet proffesor had schedualed an important exam on Thrusday. However, he beleived he would loose his friends if he didn't go out with them. The pressure to fit in with his peers was worst then the fear of bad grades. To be popular among his friends, one had to be either a musclar athelete or a wild and crazy drinker. George realy could not concieve how it was posible for a student to consume huge quanities of liquor and still suceed in school. Maybe the drinkers were just more briliant than he was. He didn't even enjoy the passtime of spending ours in a bar trying to persue a temperary feeling of excitement and "fun." Somehow he expected the cheif of campus security to catch him and the university administration to expell him. But George didn't posses enough courage to express his opion to his friends. He was certian they would tell him to mind his own buisness. Also, he did't want to be seperated from his friends. So he planed to meet them at a local restaraunt, have a few drinks, leave early, take some asprin, and spend a few ours studing for the exam.

Total mistakes found:

Total mistakes found by both groups:

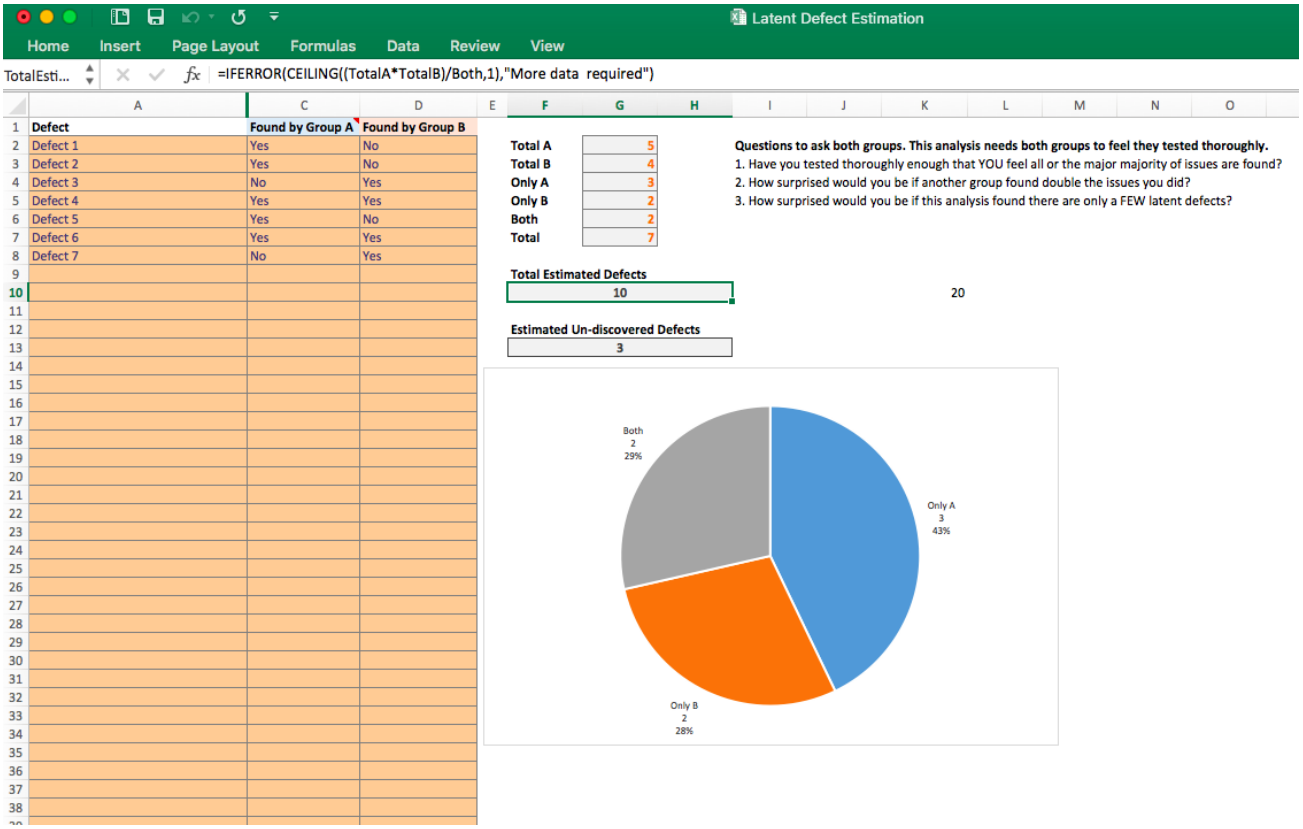
Latent Defect Estimation

Get from: Bit.ly/SimResources – Spreadsheets/Latent Defect Estimation

Use it to estimate the number of defects remaining after two groups independently check,

Discuss –

- How to perform this analysis when beta testing
- How to perform this analysis with bug bash days



Technical Risk Examples : Things that cause rework or additional work to be completed before delivery of an item

- Solution performs too slowly and needs to rework to hit releasable performance goals
- Solution uses too much server runtime memory and risks server failure. Rework required to reduce memory needs
- Solution has security vulnerabilities that causes rework to be releasable
- Solution uses too much data storage space and needs rework
- Solution fails to scale across multiple dynamic servers as expected and needs rework to support scaling
- Solution has hard coded configuration and cant be deployed to testing or production environments without rework
- Solution fails to work on different browsers, used to be IE, now think Safari
- Solution breaks previously working features that were thought to be unrelated and those need to be fixed/reworked
- Solution doesn't have required level of production monitoring features and needs rework to move into production environments
- Solution works on test data, but becomes unusable when exposed real customer data, requiring rework

Process Risk Examples: Things that delay work irrespective of the item itself

- Work sits idle, queued before a constraint for some resource that we didn't anticipate
- Images or other assets aren't available to develop a solution
- Test data isn't available to develop a solution
- Un-planned work increases beyond what was anticipated and slows progress
- Team isn't in a position to begin building a solution when planned (physically present, with everything they need to code, not shared)
- Test environments not available when needed

External Risk Examples: Dependencies, things we need from "others" before delivery of an item

- Blocked waiting for external hosting vendor to configure and install servers
- External stakeholder is slow in giving sign-off approval
- Legal / Geopolitical or other regulatory sign-off is slow

Likelihood 1 - Rare
Rare = 1 in ____ times
Likelihood 2 - Sometimes
Sometimes = 1 in ____ times
Likelihood 3 - Often
Often = 1 in ____ times
Likelihood 4 – "Certain"
Certain = Expected Every time
Impact - High
High = _____ days/weeks
Impact – Medium
Medium = _____ days/weeks
Impact - Low
Low = _____ days/weeks

Preferred Order of Action

What risks are most important to deal with first? Deciding which risks are cost effective and beneficial to go after is important, and it's not as easy as you might think. How much "sometimes" turns a "medium" impact into more important than a "rare/high"? It will depend.

Group Exercise

In a group, discuss what order you would address the un-numbered zones.

1. First, make clear the definitions for Likelihood and Impact. Agree on some numerical way to measure these in your context (Document these on the risk canvas).
2. Then, agree on how much impact would equal an increase in likelihood to make it more sense to do one medium impact before a high impact.
3. Fill all of the un-number zones.

Tips and Ideas

1. It's often hard in the un-clear zones to have an exact optimal order. This means that there may not be an exact optimal order! It's OK to best guess.
2. There will be process risks, and item specific risks. Look to understand that some risks will be "certain" for some kinds of work and "Rare" for others
3. A "medium" "sometimes" might be more important than a "high" "often" if it applies to more items in the backlog. Think frequency and exposure
4. For each risk, think what type of backlog item it applies to and how many of them there are. Drop risks that are no longer applicable.
5. Don't solve risks where the impact is less disruptive than the fix. The goal is to not eliminate every risk, just the impactful ones with easy fixes!
6. For each risk, decide on a "Solvability" score – this will help find the easiest solved risks that have big benefit; especially in the unclear zone.

Balanced Metric Dashboard – Description and Examples

Quality – How Well (Done things stay done)

What is the intended behavior?
Help teams continuously see if actions they are taking are causing a delay in delivery or any decline in product quality that would lead to customer dissatisfaction.

Examples
- Escaped defects – defects detected outside of the team
- Forecast days to complete all defects if team did nothing else
- Measure of release readiness - crowdsourced view on releasability
- Passing test percentage (sometimes of the last 5 runs)

Detectable impacts when overdriven?
- Productivity measure declines – team declares "Done" slower
- Responsiveness measure declines – team starts new work slower

Responsiveness – How Fast (The right things get done fast)

What is the intended behavior?
Help teams continuously see if they are responsive to new requests, especially those of the highest priority and criticality. Avoid measuring responsiveness for non-critical items which causes poor prioritization.

Examples
- Lead Time for high(er) severity defects
- Cycle time for committed items (eg. items chosen for a sprint)
- Lead time for items that have ever been Top 5 in the backlog

Detectable impacts when overdriven?
- Quality measure declines – Doing things faster causes defects
- Predictability measure oscillates – Inconsistent rate of delivery

Productivity – How Much (Things are getting done)

What is the intended behavior?
Help teams continuously see the delivery rate of completed work and see if actions they are taking are causing any increase or decrease of that delivery rate.

Examples
- Throughput. Completed items per week (divided by team size?)
- Velocity. Sum of completed points per sprint
- Releases per day/week

Detectable impacts when overdriven?
- Quality measure declines – "Done" things prematurely accepted
- Predictability measure oscillates – Doing too much causes chaos

Predictability – How Consistently (Things are getting done consistently)

What is the intended behavior?
Help teams continuously see if their delivery rate (productivity) is consistent and see if actions they are taking are causing uncertainty in that rate. Low predictable measure means less ability to forecast.

Examples
- Variation of the productivity measure (Standard Deviation)
- Coefficient of Variation of productivity measure (S.D./Average)
- Committed work / Delivered Work ratio

Detectable impacts when overdriven?
- Productivity measure declines – doing less means more consistency

For more probabilistic forecasting resources: FocusedObjective.com Email: troy.magennis@focusedobjective.com

Quality – How Well (Done things stay done)

What is the intended behavior?
Help teams continuously see if actions they are taking are causing a delay in delivery or any decline in product quality that would lead to customer dissatisfaction.

Examples
- Escaped defects – defects detected outside of the team
- Forecast days to complete all defects if team did nothing else
- Measure of release readiness - crowdsourced view on releasability
- Passing test percentage (sometimes of the last 5 runs)

Group Exercise (form groups of 3 to 5 people)
1. Brainstorm and discuss any measures of Quality you currently have and write one post-it note per measure
2. Brainstorm and discuss what data you have that may be used as a measure of this metric and add a post-it one note per measure
3. Discuss and dot vote what measure you feel as a group offers the best way to detect improvement or decline for this metric
4. For the top choice, Brainstorm measures that would detect if this metric was improved at the expense of everything else
5. Complete the paragraph at the bottom of this sheet.

1. List Current Quality Measures **2. List New Potential Quality Measures** **4. How would we detect when overdriven**

We will measure _____ trended every _____ (day/week/sprint) as our measure of Quality. We will also measure _____ to detect if we over-drive improving Quality and suffer elsewhere.

Responsiveness – How Fast (The right things get done fast)

What is the intended behavior?
Help teams continuously see if they are responsive to new requests, especially those of the highest priority and criticality. Avoid measuring responsiveness for non-critical items which causes poor prioritization.

Examples
- Lead Time for high(er) severity defects
- Cycle time for committed items (eg. items chosen for a sprint)
- Lead time for items that have ever been Top 5 in the backlog

Group Exercise (form groups of 3 to 5 people)
1. Brainstorm and discuss any measures of Quality you currently have and write one post-it note per measure
2. Brainstorm and discuss what data you have that may be used as a measure of this metric and add a post-it one note per measure
3. Discuss and dot vote what measure you feel as a group offers the best way to detect improvement or decline for this metric
4. For the top choice, Brainstorm measures that would detect if this metric was improved at the expense of everything else
5. Complete the paragraph at the bottom of this sheet.

1. List Current Responsiveness Measures **2. List New Potential Responsiveness Measures** **4. How would we detect when overdriven**

We will measure _____ trended every _____ (days/week/sprint) as our measure of Responsiveness. We will also measure _____ to detect if we over-drive improving Responsiveness and suffer elsewhere.

Productivity – How Much (Things are getting done)

What is the intended behavior?
Help teams continuously see the delivery rate of completed work and see if actions they are taking are causing any increase or decrease of that delivery rate.

Examples
- Throughput. Completed items per week (divided by team size?)
- Velocity. Sum of completed points per sprint
- Releases per day/week

Group Exercise (form groups of 3 to 5 people)
1. Brainstorm and discuss any measures of Quality you currently have and write one post-it note per measure
2. Brainstorm and discuss what data you have that may be used as a measure of this metric and add a post-it one note per measure
3. Discuss and dot vote what measure you feel as a group offers the best way to detect improvement or decline for this metric
4. For the top choice, Brainstorm measures that would detect if this metric was improved at the expense of everything else
5. Complete the paragraph at the bottom of this sheet.

1. List Current Productivity Measures **2. List New Potential Productivity Measures** **4. How would we detect when overdriven**

We will measure _____ trended every _____ (day/week/sprint) as our measure of Productivity. We will also measure _____ to detect if we over-drive improving Productivity and suffer elsewhere.

Predictability – How Consistently (Things are getting done consistently)

What is the intended behavior?
Help teams continuously see if their delivery rate (productivity) is consistent and see if actions they are taking are causing uncertainty in that rate. Low predictable measure means less ability to forecast.

Examples
- Variation of the productivity measure (Standard Deviation)
- Coefficient of Variation of productivity measure (S.D./Average)
- Committed work / Delivered Work ratio

Group Exercise (form groups of 3 to 5 people)
1. Brainstorm and discuss any measures of Quality you currently have and write one post-it note per measure
2. Brainstorm and discuss what data you have that may be used as a measure of this metric and add a post-it one note per measure
3. Discuss and dot vote what measure you feel as a group offers the best way to detect improvement or decline for this metric
4. For the top choice, Brainstorm measures that would detect if this metric was improved at the expense of everything else
5. Complete the paragraph at the bottom of this sheet.

1. List Current Predictability Measures **2. List New Potential Predictability Measures** **4. How would we detect when overdriven**

We will measure _____ trended every _____ (day/week/sprint) as our measure of Predictability. We will also measure _____ to detect if we over-drive improving Predictability and suffer elsewhere.

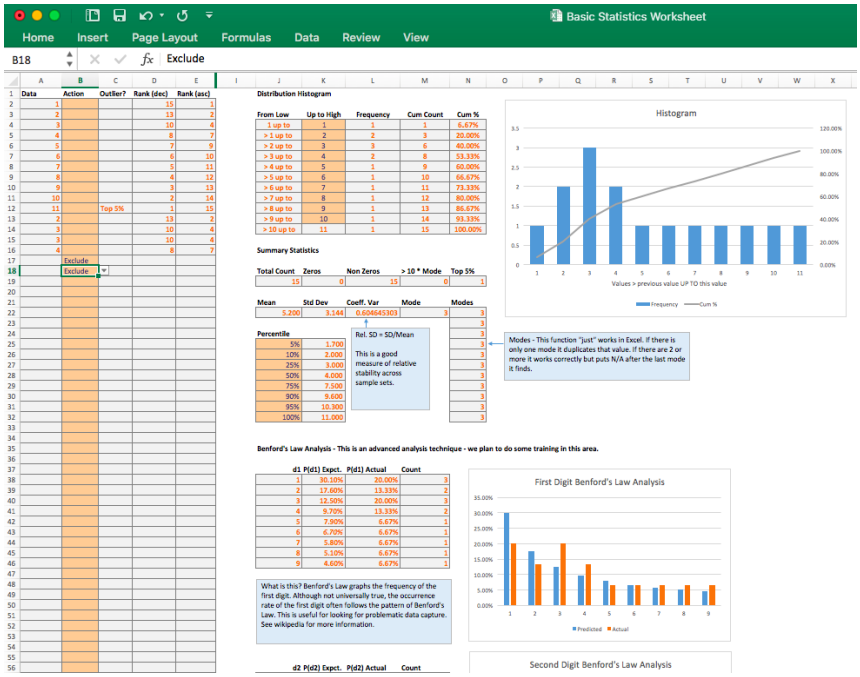
Tool: Basic Statistics Worksheet

Get from: Bit.ly/SimResources – Spreadsheets/Basic Statistics Worksheet

Use it to analyze numerical data for consistency, outliers and special cases that might reflect good or bad data quality issues.

Discuss –

- Statistical Terms: mean vs median
- Histograms: Why they are useful
- Outlier Management: What is an outlier, what do you do
- Zero and empty management: How to avoid these causing errors
- Benfords Law: Detecting non-random….



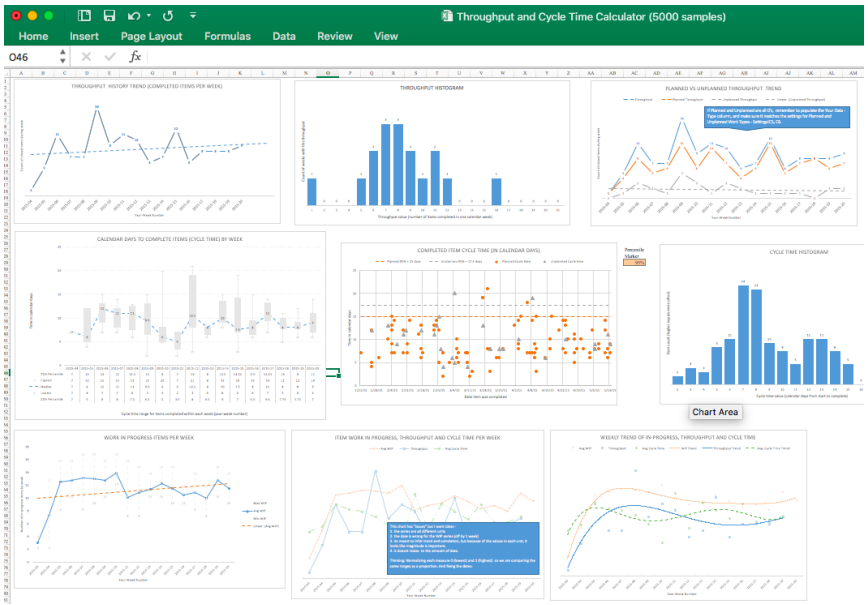
Tool: Throughput and Cycle Time Calculator / Team Dashboard

Get from: Bit.ly/SimResources – Spreadsheets/ Throughput and Cycle Time Calculator (5000 samples).xlsx and Spreadsheets/Team Dashbord.xlsx

Use it to analyze story completion and start dates and generate throughput and cycle time. Also produces 17 other charts.

Discuss –

- How to get the date data
- The different charts it produces
- Planned versus un-planned work settings
- Where to copy cycle time and throughput values



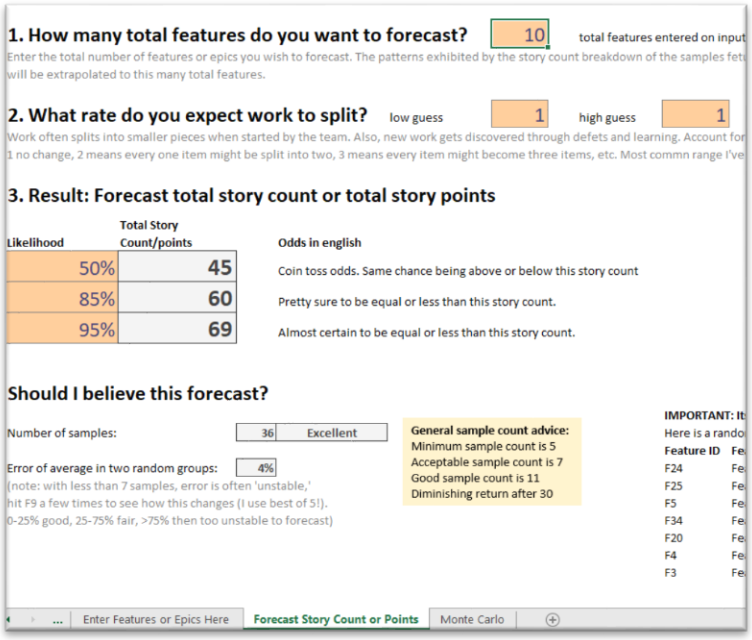
Tool: Story Count Forecaster

Get from: Bit.ly/SimResources – Spreadsheets/Story Count Forecaster.xlsx

Use it to forecast a total number of stories in a project/feature by breaking down just a sample set of estimates

Discuss –

- The goals of story count/points forecasting
- How to estimate and measure split rate by comparing actuals
- How the error average helps understand number of samples
- The charts and how it works



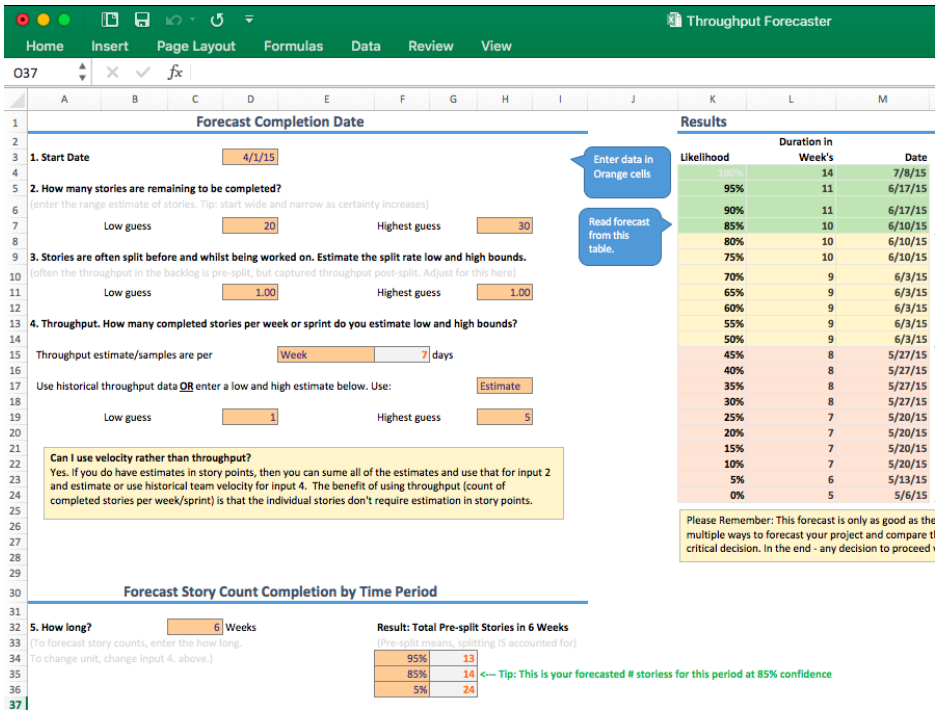
Tool: Throughput Forecaster

Get from: Bit.ly/SimResources – Spreadsheets/Throughput Forecaster.xlsx

Use it to forecast a single feature or how much work fits within a fixed time-frame. Can use historical data or range guesses

Discuss –

- The charts and how it works
- How to get the throughput data
- How to perform the range estimates
- How to track actual versus estimates
- Modeling and Forecasting Risks



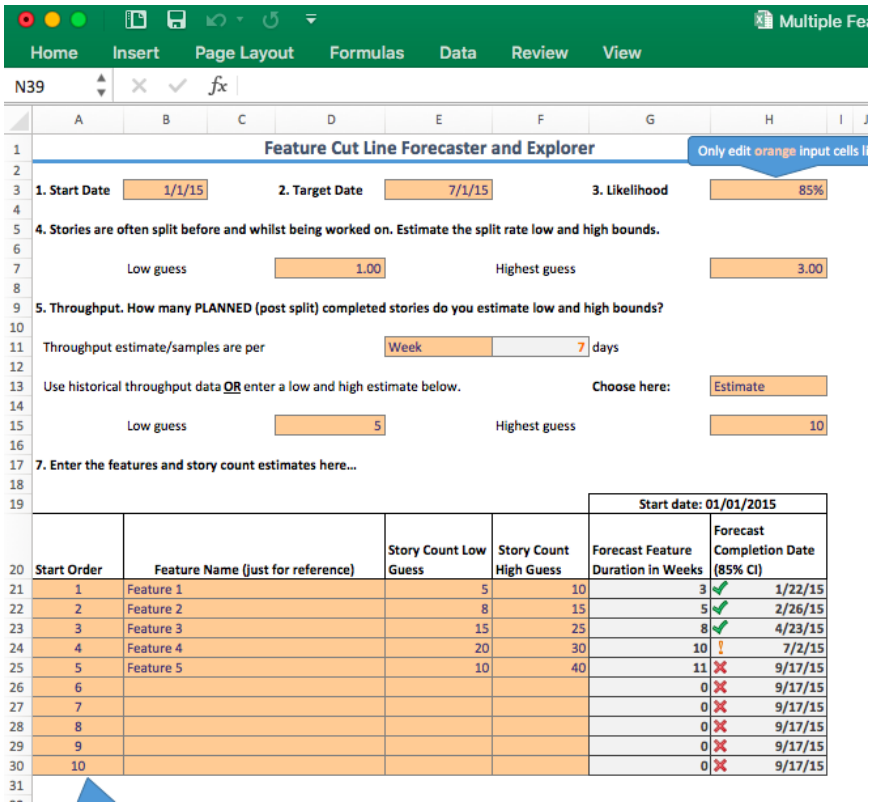
Tool: Multiple Cut Line Forecaster

Get from: Bit.ly/SimResources – Spreadsheets/Multiple Feature Cut Line Forecaster.xlsx

Use it to forecast a what features will likely deliver by a target date, and to simulate different start orders of those features to maximize value delivery.

Discuss –

- How to split features
- How to set split rate
- Using estimates before sample data is available
- How to simulate order change
- Why to use month adjustments (vacation, staff increases, conferences, etc)



Feature Cut Line Forecaster and Explorer

Only edit orange input cells

1. Start Date: 1/1/15 — 2. Target Date: 7/1/15 — 3. Likelihood: 85%

4. Stories are often split before and whilst being worked on. Estimate the split rate low and high bounds.

Low guess: 1.00 — Highest guess: 3.00

5. Throughput. How many PLANNED (post split) completed stories do you estimate low and high bounds?

Throughput estimate/samples are per: Week: 7 days

Use historical throughput data OR enter a low and high estimate below. — Choose here: Estimate

Low guess: 5 — Highest guess: 10

7. Enter the features and story count estimates here...

Start date: 01/01/2015

Start Order	Feature Name (just for reference)	Story Count Low Guess	Story Count High Guess	Forecast Feature Duration in Weeks	Forecast Completion Date (85% CI)
1	Feature 1	5	10	3	1/22/15
2	Feature 2	8	15	5	2/26/15
3	Feature 3	15	25	8	4/23/15
4	Feature 4	20	30	10	7/2/15
5	Feature 5	10	40	11	9/17/15
6				0	9/17/15
7				0	9/17/15
8				0	9/17/15
9				0	9/17/15
10				0	9/17/15

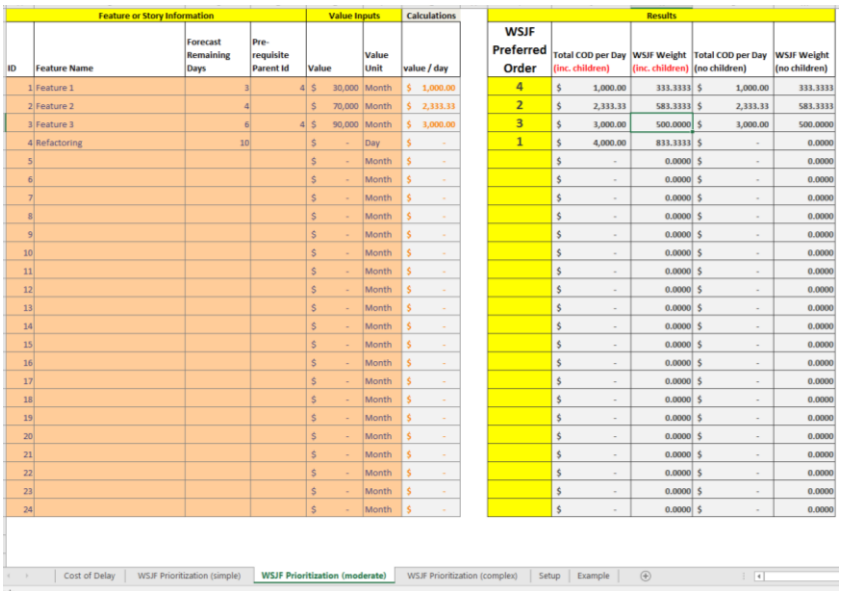
Tool: Cost of Delay Prioritization Calculator

Get from: Bit.ly/SimResources – Cost of Delay Prioritization Calculator.xlsx

Use it to compute total cost of delay and to find an "optimal" start order for proposed work.

Discuss –

- The different cost of delay ordering techniques (see cheat sheet)
- How to introduce cost of delay
- How complex is too complex
- How to account for dependencies and enablers



Feature or Story Information				Value Inputs		Calculations	Results				
ID	Feature Name	Forecast Remaining Days	Pre-requisite Parent Id	Value	Value Unit	value / day	WSJF Preferred Order	Total COD per Day (inc. children)	WSJF Weight (inc. children)	Total COD per Day (no children)	WSJF Weight (no children)
1	Feature 1	3	4	$ 30,000	Month	$ 1,000.00	4	$ 1,000.00	333.3333	$ 1,000.00	333.3333
2	Feature 2	4		$ 70,000	Month	$ 2,333.33	2	$ 2,333.33	583.3333	$ 2,333.33	583.3333
3	Feature 3	6	4	$ 90,000	Month	$ 3,000.00	3	$ 3,000.00	500.0000	$ 3,000.00	500.0000
4	Refactoring	10		$ -	Day	$ -	1	$ 4,000.00	833.3333	$ -	0.0000
5				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
6				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
7				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
8				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
9				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
10				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
11				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
12				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
13				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
14				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
15				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
16				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
17				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
18				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
19				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
20				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
21				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
22				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
23				$ -	Month	$ -		$ -	0.0000	$ -	0.0000
24				$ -	Month	$ -		$ -	0.0000	$ -	0.0000

Quarterly Portfolio Planning - one item at a time until you exceed capacity

The scene:

A group of product owners and stakeholders meet at regular intervals to plan an upcoming period of time. Everyone brings their laundry list of ideas to a full day meeting to plan what will get done. Normally, there are thousands of times more ideas than capacity, and even the MUST HAVES would take the next 12 months to complete.

The goal:

To ensure that the most important items (highest cost of being delayed) make it into the next quarter. To make sure that no one team or skillset is burdened with more than it has achieved historically.

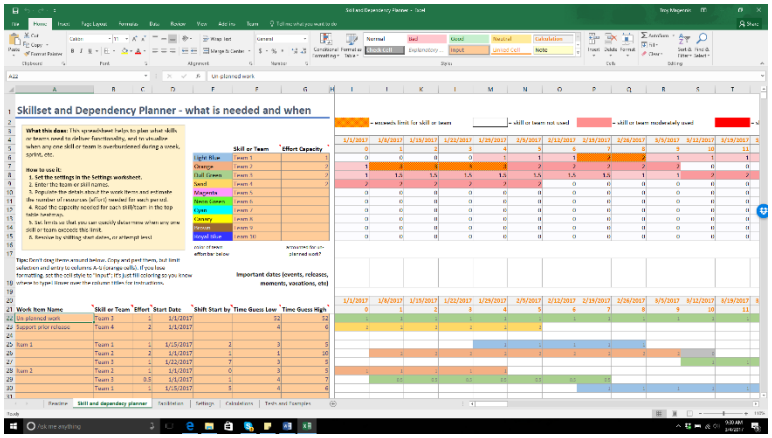
Suggested process:

1. Debrief last quarters plans. Identify carry over work. Identify how well the last plan played out with respect to keeping teams within their effort budget. No blame, just looking to learn how to set correct capacity limits and what type of work we plan poorly.
2. Reserve capacity for teams and skills.
    - To account for warranty of prior released features
    - To account for any team un-planned work in addition to this plan
3. Identify key dates. Are any dates immovable, for example a consumer promotional campaign, sporting event, or conference. List these dates and decide when work will need to be "finished" to allow other activities to take place (for example, training guides or website updates based on the new features or products.
4. Identify the total capacity limit for each team or skill. Decide what units make sense here. It can be people, it can be throughput, it can be story points. Just decide upfront and be consistent with this limit and the effort estimate for each line item.
5. Have the group identify THE ABSOLUTE ONE FEATURE that must deliver. It MUST be one single highest priority.
6. Brainstorm what teams or skills are needed to deliver this ONE feature. Enter this into this spreadsheet and break out one row per team or skill involved, have someone from each team perform a quick estimate of lowest and highest delivery time and effort estimates (these are broad 1 week or 2 week estimates, NOT detailed estimates. Allow the team the final say on these estimates, don't second guess them.
7. Once all of the dates and effort estimates are entered, look at the team and skill heatmap. Have any of the skills hit their limit yet?
    - No: Go back to step 5 and repeat for the net feature
    - Yes: Can you move the start date of this feature to reduce peak demand, for example can you move the start date later to avoid overwhelming the exceeded skill or team? If NOT then that’s it. You have your plan.
8. Still have some teams lightly loaded? Can they help teams that are overwhelmed and get the next highest priority feature? Rule this out before loading them with "Nice to have work."

Tool: Skill and Dependency Portfolio Planner – make dependencies visible, determine constraints and finding optimal start dates

Get from: Bit.ly/SimResources – Spreadsheets/Skill and Dependency Planner.xlsx

Use it to understand team or skill utilization in portfolio and dependency planning. Find ways to balance capacity by shifting start dates. Visually represent the load on teams for discussion and constraint analysis.



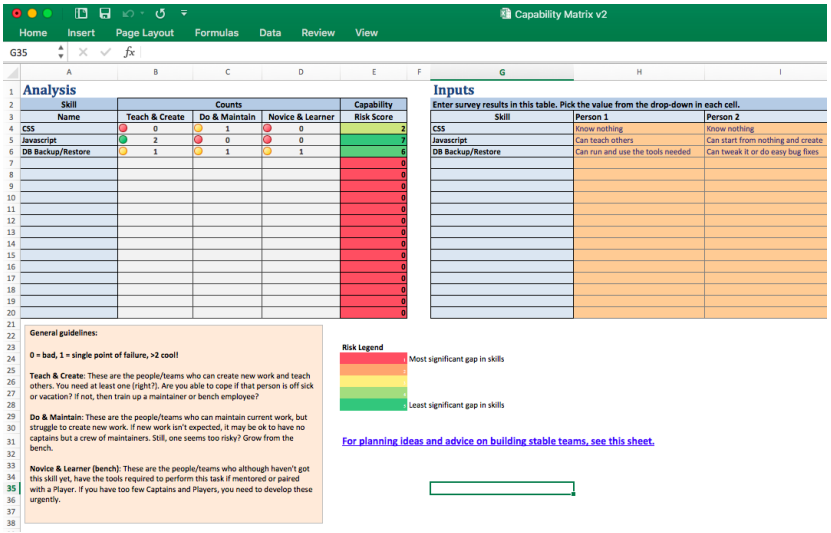
Discuss –

- How to find the constraint during planning
- What to do when a constraint is hit
- How to enter data in a group facilitation environment
- How to reserve capacity and set realistic limits

Tool: Capability Matrix v2

Get from: Bit.ly/SimResources – Spreadsheets/Capability Matrix v2.xlsx

Use it to quickly survey for available skills and understand who can teach those skills.



Discuss –

- How to decide what skills to survey
- How to maximize honest answers
- How to plan team growth and splitting points

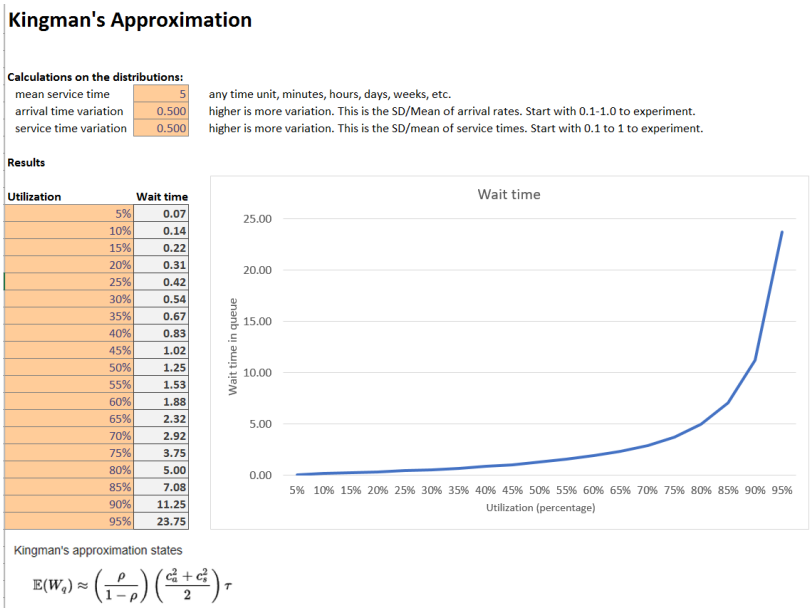
Simulation: Kingman's Formula – utilization and variation on cycle time

Get from: Bit.ly/SimResources – Spreadsheets/Kingmans Formula.xlsx

Use it to teach and understand how utilization and variation can impact cycle-time.

Discuss –

- Double the mean service time. What happens to wait time?
- Set the service time variation to 0, play with service time variation. Double and halve
- Which variability matters more, arrival rate or service time?



Kingman's Approximation

Calculations on the distributions:

mean service time	5	any time unit, minutes, hours, days, weeks, etc.
arrival time variation	0.500	higher is more variation. This is the SD/Mean of arrival rates. Start with 0.1-1.0 to experiment.
service time variation	0.500	higher is more variation. This is the SD/mean of service times. Start with 0.1 to 1 to experiment.

Results

Utilization	Wait time
5%	0.07
10%	0.14
15%	0.22
20%	0.31
25%	0.42
30%	0.54
35%	0.67
40%	0.83
45%	1.02
50%	1.25
55%	1.53
60%	1.88
65%	2.32
70%	2.92
75%	3.75
80%	5.00
85%	7.08
90%	11.25
95%	23.75

Kingman's approximation states

$$\mathbb{E}(W_q) \approx \left(\frac{\rho}{1-\rho}\right)\left(\frac{c_a^2 + c_s^2}{2}\right)\tau$$

Simulation: Traffic Light Simulator – Showing how do delays cascade

Get from: Bit.ly/SimResources – Traffic Light Simulator.xlsx

Use it teach how delays cause different impacts on cycle time. Learn how to determine what factors influence extremely long cycle times and how this changes what coaching advice will be successful.

Discuss –

- The two basic cycle time distributions
- Find ways to create the different cycle time distributions

Capture-Recapture Exercise Answers:

George new that he shouldn't drink alchohol on a Wedsday night, especially since his govermnet proffesor had schedualed an important exam on Thrusday. However, he beleived he would loose his friends if he didn't go out with them. The pressure to fit in with his peers was worst then the fear of bad grades. To be popular among his friends, one had to be either a musclar athelete or a wild and crazy drinker. George realy could not concieve how it was posible for a student to consume huge quanities of liquor and still suceed in school. Maybe the drinkers were just more briliant than he was. He didn't even enjoy the passtime of spending ours in a bar trying to persue a temperary feeling of excitement and "fun." Somehow he expected the cheif of campus security to catch him and the university administration to expell him. But George didn't posses enough courage to express his opion to his friends. He was certian they would tell him to mind his own buisness. Also, he did't want to be seperated from his friends. So he planed to meet them at a local restaraunt, have a few drinks, leave early, take some asprin, and spend a few ours studing for the exam.

Corrected -

George knew that he shouldn't drink alcohol on a Wednesday night, especially since his government professor had scheduled an important exam on Thursday. However, he believed he would lose his friends if he didn't go out with them. The pressure to fit in with his peers was worse than the fear of bad grades. To be popular among his friends, one had to be either a muscular athlete or a wild and crazy drinker. George really could not conceive how it was possible for a student to consume huge quantities of liquor and still succeed in school. Maybe the drinkers were just more brilliant than he was. He didn't even enjoy the pastime of spending hours in a bar trying to pursue a temporary feeling of excitement and "fun." Somehow he expected the chief of campus security to catch him and the university administration to expel him. But George didn't possess enough courage to express his opinion to his friends. He was certain they would tell him to mind his own business. Also, he didn't want to be separated from his friends. So he planned to meet them at a local restaurant, have a few drinks, leave early, take some aspirin, and spend a few hours studying for the exam.

Total errors: 36

Source: **STUDENT LEARNING ASSISTANCE CENTER (SLAC).** Texas State University-San Marcos

20 COGNITIVE BIASES THAT SCREW UP YOUR DECISIONS

1. Anchoring bias.

People are **over-reliant** on the first piece of information they hear. In a salary negotiation, whoever makes the first offer establishes a range of reasonable possibilities in each person's mind.

2. Availability heuristic.

People **overestimate the importance** of information that is available to them. A person might argue that smoking is not unhealthy because they know someone who lived to 100 and smoked three packs a day.

3. Bandwagon effect.

The probability of one person adopting a belief increases based on the number of people who hold that belief. This is a powerful form of **groupthink** and is reason why meetings are often unproductive.

4. Blind-spot bias.

Failing to recognize your own cognitive biases is a bias in itself. People notice cognitive and motivational biases much more in others than in themselves.

5. Choice-supportive bias.

When you choose something, you tend to feel positive about it, even if that **choice has flaws**. Like how you think your dog is awesome — even if it bites people every once in a while.

6. Clustering illusion.

This is the tendency to **see patterns in random events**. It is key to various gambling fallacies, like the idea that red is more or less likely to turn up on a roulette table after a string of reds.

7. Confirmation bias.

We tend to listen only to information that confirms our **preconceptions** — one of the many reasons it's so hard to have an intelligent conversation about climate change.

8. Conservatism bias.

Where people favor prior evidence over new evidence or information that has emerged. People were **slow to accept** that the Earth was round because they maintained their earlier understanding that the planet was flat.

9. Information bias.

The tendency to **seek information when it does not affect action**. More information is not always better. With less information, people can often make more accurate predictions.

10. Ostrich effect.

The decision to **ignore dangerous or negative information** by "burying" one's head in the sand, like an ostrich. Research suggests that investors check the value of their holdings significantly less often during bad markets.

11. Outcome bias.

Judging a decision based on the **outcome** — rather than how exactly the decision was made in the moment. Just because you won a lot in Vegas doesn't mean gambling your money was a smart decision.

12. Overconfidence.

Some of us are **too confident about our abilities**, and this causes us to take greater risks in our daily lives. Experts are more prone to this bias than laypeople, since they are more convinced that they are right.

13. Placebo effect.

When **simply believing** that something will have a certain effect on you causes it to have that effect. In medicine, people given fake pills often experience the same physiological effects as people given the real thing.

14. Pro-innovation bias.

When a proponent of an innovation tends to **overvalue its usefulness** and undervalue its limitations. Sound familiar, Silicon Valley?

15. Recency.

The tendency to weigh the **latest information** more heavily than older data. Investors often think the market will always look the way it looks today and make unwise decisions.

16. Salience.

Our tendency to focus on the **most easily recognizable features** of a person or concept. When you think about dying, you might worry about being mauled by a lion, as opposed to what is statistically more likely, like dying in a car accident.

17. Selective perception.

Allowing our expectations to **influence how we perceive** the world. An experiment involving a football game between students from two universities showed that one team saw the opposing team commit more infractions.

18. Stereotyping.

Expecting a group or person to have certain qualities without having real information about the person. It allows us to quickly identify strangers as friends or enemies, but people tend to **overuse and abuse** it.

19. Survivorship bias.

An error that comes from focusing only on surviving examples, causing us to **misjudge a situation**. For instance, we might think that being an entrepreneur is easy because we haven't heard of all those who failed.

20. Zero-risk bias.

Sociologists have found that **we love certainty** — even if it's counterproductive. Eliminating risk entirely means there is no chance of harm being caused.

SOURCES: Brain Biases; Ethics Unwrapped; Explorable; Harvard Magazine; HowStuffWorks; LearnVest; Outcome bias in decision evaluation, Journal of Personality and Social Psychology; Psychology Today; The Bias Blind Spot: Perceptions of Bias in Self Versus Others, Personality and Social Psychology Bulletin; The Cognitive Effects of Mass Communication, Theory and Research in Mass Communications; The less-is-more effect: Predictions and tests, Judgment and Decision Making; The New York Times; The Wall Street Journal; Wikipedia; You Are Not So Smart; ZhurnalyWiki

BUSINESS INSIDER

Latent Defect Estimation Worksheet

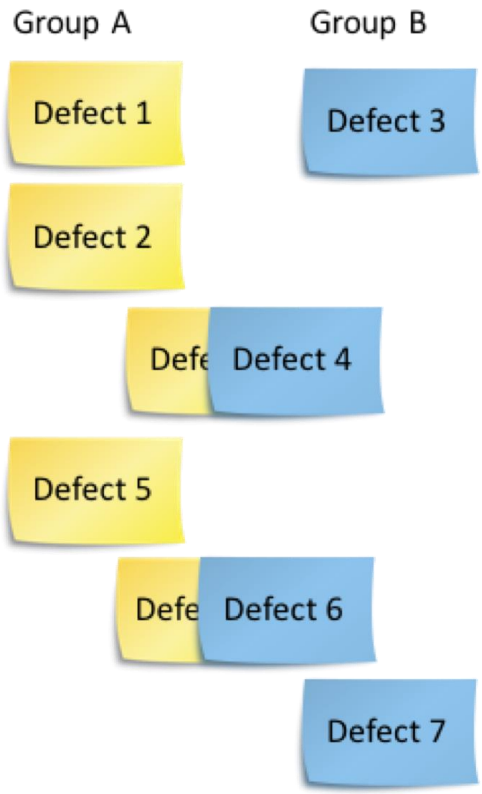
Capture-Recapture Method for latent defect estimation

Capture-recapture is a way to estimate how well the current investigation for defects is working. The basic principle is to have multiple individuals or groups analyze the same feature or code and record their findings. The ratio of overlap (found by both groups) and unique discovery (found by just one of the groups) gives an indication of how much more there might be to find.

References: Walt Humphries in the Team Software Process (SEI/CMU), Joseph Schofield in Estimating Latent Defects using Capture-Recapture: Lessons from Biology.

$$\text{Estimated total defects} = \frac{\text{Found by group A} \times \text{Found by group B}}{\text{Found by BOTH group A and B}}$$

$$\text{Estimated defects un-discovered} = \text{Estimated total defects} - \text{Unique defects found so far}$$



Total found by A = 5

Total found by B = 4

Found by BOTH = 2

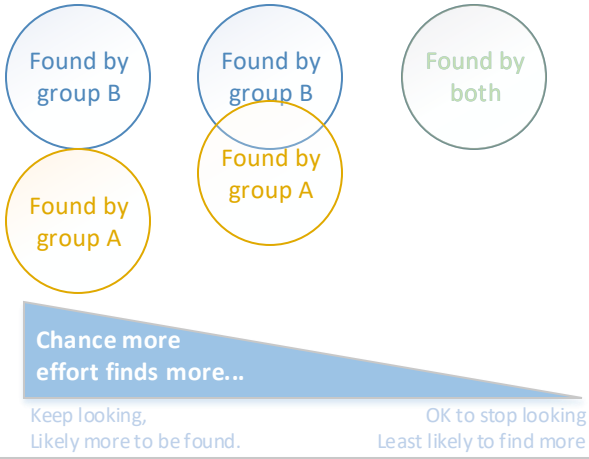
Total found = 7

$$\text{Estimated total} = \frac{5 \times 4}{2} = 10$$

$$\text{Estimated un-discovered} = 10 - 7 = 3$$

General idea...

The more overlap in the defects two independent groups find, the fewer defect remain un-discovered



How to -

1. Set two independent groups the task of testing or reviewing the same code or document
2. Have each group record the defects or errors they find
3. After a period of time, match the defects found by each group to count the duplicates
4. Estimate the number of remaining defects versus how many unique defects have been found using the formula

What to avoid -

1.Team not reporting duplicates. You need to encourage the groups to report EVERYTHING they find
2. Teams testing too quickly. If testing isn't thorough enough, latent defects will appear less than actual. Make sure each team at least "feels" they have looked thoroughly
3. Assuming this estimate is perfect. This technique is an estimate to decide whether more time is worth it.

Some ideas when to use it -

Bug bash days: set two groups the task of looking for defects in the same feature area. Get a feeling for stability.
Customer beta programs: create two groups of beta testers (A-K and L-Z first letter of first name) and analyze the defects they report as from group A and B. Helps release earlier with confidence.

Optimal Batch Size

What is the optimal batch size?

Some jobs are too expensive to perform for every individual item. There are costs incurred by delaying delivery of items. What is the optimal number of items to perform as a batch, or what is the right amount of time for regular delivery?

The basic concept – balance per item cost versus delayed delivery cost

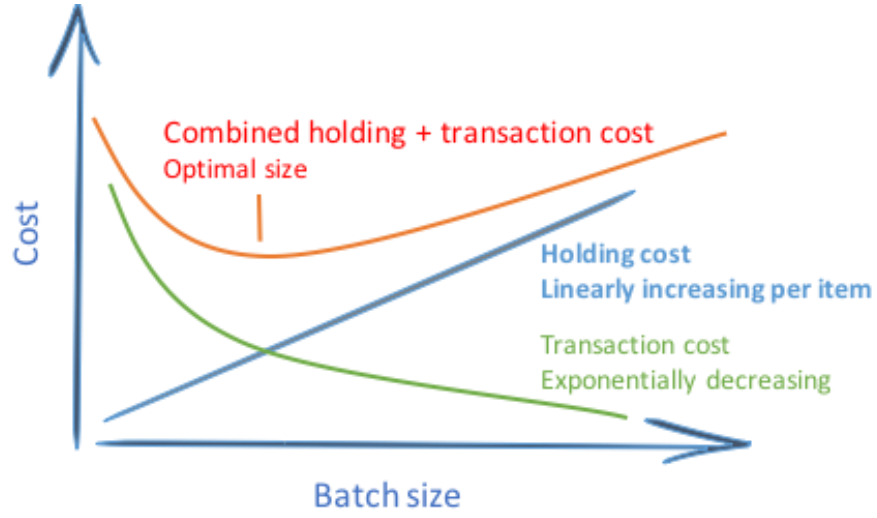
There are two types of cost to balance -

1. The costs of setup and execution of the process (per unit) = Per Item Transaction costs
2. The costs of holding back items from delivery e.g. delayed feedback = Per Item Holding costs

Optimal batch size is the number of items that minimizes the total combined costs. There are various types of costs that need to be considered. Delayed feedback, storage cost, decay in value.

If setup and execution costs of a task is high, it gets exponentially more efficient (as a per item cost) for higher batch sizes. Holding costs are often more linear (it is assumed all items deliver equal value, and that any item could give valuable feedback). Brainstorm all the transaction costs and holding costs. Think about how they respond to different batch sizes.

Most importantly: Just DON'T be near either end. That is where total cost escalates. It is almost always safe to halve the batch size putting you in the flatter section of the total cost U-curve. You will know when batch size gets too small, costs and effort escalate enormously.



References and more information

Google for "Economic Batch Size" or "Economic Order Size" in production/inventory management.
Read "Principles of Product Development Flow" by Donald Reinertson, Chapter 5 – Batch Size
Read "An optimal batch size for a JIT manufacturing system" Lutfar R. Khana, Ruhul A. Sark

Example: Performance Throughput Testing – How often?

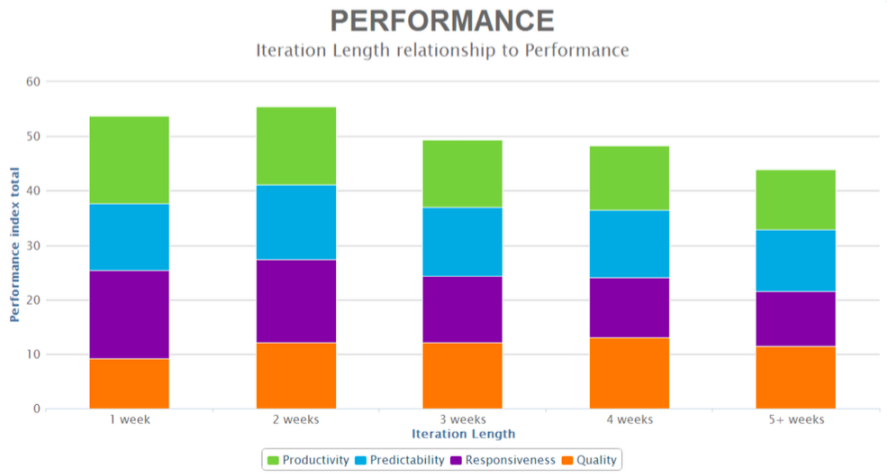
Some projects require performance testing. This type of testing can be labor intensive to setup and time consuming to execute and analyze. Some teams only perform this late in a project causing late feedback on poor performance architecture and code. The right batch size in time and amount of features to test in one batch is important to consider. Balance the costs -

Transaction Costs	Holding Costs
Setup time	Delayed feedback on architecture decisions
Blocked testing environment	Harder to find the culprit of any regression
	Late knowledge release will be delayed

Tip: Do twice as often as you do now. Once you achieve that rate, halve that again. Stop when the disruption of performing these tests becomes intolerable

Study: Sprint Length – How many weeks are optimal?

Two weeks is the most commonly used sprint cadence, but is it optimal? Turns out, "yes, mostly." In a study, Larry Maccherone then with Rally Software performed looking at approximately 10,000 projects. Measured with a balanced set of metrics across four different categories (quality, predictability, performance and responsiveness), two week sprints performed best overall. However, teams with one week sprints had better responsiveness scores, and teams using three-four weeks sprints had higher quality scores. The optimal length might depend on the trade-off between quality and responsiveness in your context. Tip: Start at two weeks and experiment over time. Go lower if responsiveness is important, but keep an eye on quality due to moving too fast.



http://maccherone.com/publications/KEYNOTE-MetricsAndInsights-Maccherone-0604-1245PM.pdf

Dependencies

Definition: Dependency

Progress of one action relies upon the timely output of a previous action, or the presence of some specific thing. Source: Strode & Huff [1]

Impacts of Dependencies

1. Reduced freedom of ordering work item starting priority [2]
2. Increased lead-time of work items waiting for dependencies to resolve
3. Friction between teams who have different priorities and rewards

Reduced Ordering Options – One dependency halves options

Feature A	Feature B	A before B
1	2	Yes
2	1	No

One dependency cuts allowed options in half

Feature start order

Two dependencies cuts allowed start options to 1/6th

Feature A	Feature B	Feature C	A before B	A bf B, B bf C
1	2	3	Yes	Yes
1	3	2	Yes	No
2	1	3	No	No
2	3	1	No	No
3	1	2	Yes	No
3	2	1	No	No

Number of Dependencies	Valid Ordering Options
0	100%
1	50%
2	16.667%
3	4.167%
4	0.833%
5	0.278%

Dependencies narrow the valid options for starting work. This is especially debilitating when planning multiteam delivery of features. Even a single dependency between backlog items halves the allowable start order. By two dependencies only 16% valid options remain, by three only 4% start order options are valid. [2]

Work should be prioritized to minimize cost of delay and maximize value. Dependencies between backlog items or other teams erodes the ability to optimize start order.

Taxonomy of Agile Software Project Dependencies

Source: Strode & Huff [1]

A dependency is created when the progress of one action relies upon the timely output of a previous action, or the presence of some specific thing. Dependencies lead to potential or actual constraints on projects. Potential constraints are those that are currently organised or managed well, causing no problems in the progression of a project. Actual constraints are bottlenecks or points in a project that stakeholders are aware of, but have no immediate means to circumvent.

Knowledge dependency

A knowledge dependency occurs when a form of information is required in order for a project to progress. There are four forms of knowledge dependency:

Requirement - a situation where domain knowledge or a requirement is not known and must be located or identified and this affects project progress

Expertise - a situation where technical or task information is known only by a particular person or group and this affects project progress

Task allocation - a situation where who is doing what, and when, is not known and this affects project progress

Historical - a situation where knowledge about past decisions is needed and this affects project

Task dependency

A task dependency occurs when a task must be completed before another task can proceed and this affects project progress. There are two forms of task dependency:

Activity - a situation where an activity cannot proceed until another activity is complete and this affects project progress

Business process - a situation where an existing business process causes activities to be carried out in a certain order and this affects project progress

Resource dependency

A resource dependency occurs when an object is required for a project to progress. There are two forms of resource dependency:

Entity - a situation where a resource (person, place or thing) is not available and this affects project progress

Technical - a situation where a technical aspect of development affects progress, such as when one software component must interact with another software component, and its presence or absence affects project progress

Increased Lead Time and Risk of Delay: Chance of on-time = 1 in 2^n (where n = dependencies)

Team 1	Team 2	Team 3	Team 4

Team 1	Team 2	Team 3

If multiple teams all need to synchronize there work (have serial dependencies on each other), every dependency doubles the chance of being late due to AT LEAST one of the teams delivering late. In fact, the chance of on-time delivery is 1 chance in 2^n (n = number of dependencies).

For example, if 4 teams had dependencies (left table), there is 1 chance in 16 of on-time delivery (15 chances in 16 of being late due to at least one delay). If one dependency can be removed leaving just 3 dependent teams (right table), the risk of delay is now 1 chance in 8.

Note: a red cell means team delivered late, green means as expected.

References

[1] Strode D, Huff S. A taxonomy of dependencies in agile software development. 23rd Australasian Conference on Information Systems, https://dro.deakin.edu.au/eserv/DU:30049080/strode-taxonomyofdependencies-2012.pdf

[2] Sheerer A., Bick S., Hildenbrand T. and Heinzl A. The Effects of Team Backlog Dependencies on Agile Multiteam Systems: A Graph Theoretical Approach. http://conferences.computer.org/hicss/2015/papers/7367f124.pdf

Diane E. Strode*, Sid L. Huff, Beverley Hope, Sebastian Link. Coordination in co-located agile software development projects. https://pdfs.semanticscholar.org/7cc7/2fdff2c46c94fbfaccedbfb01843c529df84.pdf

For more probabilistic forecasting resources: FocusedObjective.com Email: troy.magennis@focusedobjective.com

Better Backlog Prioritization (from random to lifetime cost of delay)

The Goal – what to start next

Given a set of things we could do next, is one more economically advantageous to start first.

The Challenge – Doing one thing delays others

Every item has a different economic impact by being delayed. The impact will be a balance of lost value, and how long they are delayed.

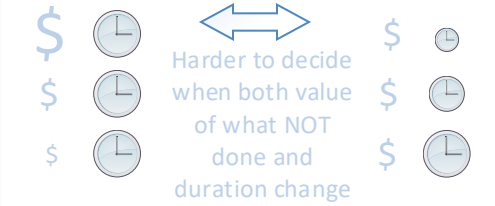
"The problem with any prioritization decision is [it is a] decision to service one job and delay another." Don Reinertsen

The basic concept – balance $ & time

$ = Value lost due to delay 🕒 delay duration

Same duration, do > lost $ first

Same lost $, do < duration first



There are many ways that a more economically optimal work order can be derived. These rank from no attempt to find a more optimal order to computing the impact of delay on profitability over the products useful lifespan. The goal is to use the technique that is appropriate for the level of impact if wrong. For major decision with huge financial impact, use the techniques more to the right.

The leftmost techniques focus on quickly determining an items value and prefer to do those first. As we move to the right, the "value" estimate starts to consider more factors. Techniques on the right start to estimate the value based on total market profitability impact due to doing something else first. All techniques are a balance between analysis time and how impactful it might be to be wrong.

Most common
Subjective

Small to medium impact decisions
Gets close when the impact of being wrong is moderate

Medium to large impact decisions
Start understanding the total cost of delay (products and major features)

HiPPO
Just make a choice

Relative Value & Effort (delay) Matrix
Balance duration and value

SAFe "Arnold"
Make urgency dominate

SAFe (real estimates)
Normalize & weight values against $ and time range scales

Economic Impact of delay over a product's entire lifespan is modeled (long term value)

Random choice ← → **Crystal Ball**

Highest Value First
Common in Scrum. Some attempt as estimating "value"

SAFe
Consider more than one value type (estimated using Fibonacci point scale)

Cost of Delay Curve Categorization
(e.g. Kanban's class of service)

Quantitative $ per Fixed Period
(e.g. CD3)

Note: a naïve assumption is that good analysis is performed in every technique shown here. Its possible to do all of these techniques poorly and make poor decisions. Use care!

Trade offs

Chance of making a sub-optimal decision

Effort required to get optimal decision

Highest Value First: Common in Scrum

Scrum proposes starting the highest customer value work first. Some teams use a qualitative low, medium and high. Some attempt to estimate it in dollars. This is better than random ordering, but often leads to "Biggest liar wins." It also doesn't consider how long each item will take, meaning more value might be delivered in a number of smaller items that sum to greater value.

SAFe and SAFe "Arnold Mod"

SAFe (Scaled Agile Franework) uses a weighted shortest job first balancing technique. It uses subjective measures and approximates optimal starting order using the following formula (highest first):

$$\frac{\text{Value + Criticality + Risk Reduction or Opportunity Enablement}}{\text{Job Size}}$$

Joshua Arnold offers the following modification to make time criticality more dominant:

$$\frac{\text{Criticality x (Value + Risk Reduction or Opportunity Enablement)}}{\text{Job Size}}$$

Economic Models and WSJF

Donald Reinertsen in his book "Principles of Product Development Flow" offers a variety of scheduling techniques. The most popular is Weighted Shortest Job First where optimal starting order is calculated using delay impact in dollars and size. Optimal order (highest to lowest) is calculated using the formula:

$$\frac{\text{Cost of delay}}{\text{Duration of delay}}$$

Reinertsen suggests it's prudent to consider the total market impact of a delay, not just the immediate lost value.

Jointly designed in an online conversation by Martin Burns, Don Reinertsen, Chris Matts, Joshua Arnold, Tony Grout and Troy Magennis sometime during 2016. It is, and will remain a work in progress. Trademarks used NOT ours.

Better Backlog Prioritization (from random to lifetime cost of delay) - Detail

SAFe Weighted Shortest Job First

The Scaled Agile Framework proposes an ordering system based on Don Reinertsen's Weighted Shortest Job First (WSJF) principles. Proposed features are assessed on multiple value and size axis using relative Fibonacci story point estimates. The process is described as -

1. Rate each parameter against the other features using the scale: 1,2,3,5,8,13,20. Do one column at a time, and calibrate the lowest value to be a "1" - each column MUST have one "1"
2. Calculate the WSJF value for each column using the formula shown below
3. Do the feature that has the HIGHEST WSJF value first if possible

Pros: helps prioritize more than one type of value, and balances time based on the proxy job size
Cons: story point estimates don't handle extreme variation in value, job size not always duration

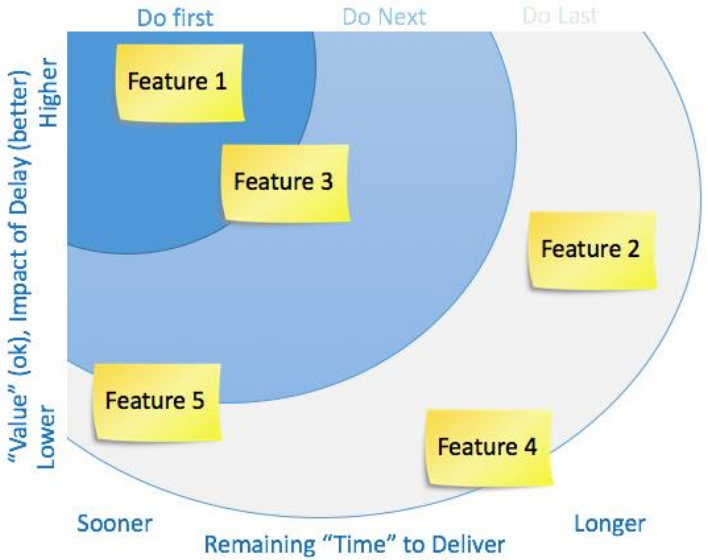
SAFe's Weighted Shortest Job First formula (upper), and a typical data capture table (lower)
More info: http://www.scaledagileframework.com/wsjf/

$$WSJF = \frac{\text{User-Business Value + Time Criticality + Risk Reduction | Opportunity Enablement Value}}{\text{Job Size}}$$

Feature	User-Business Value	Time Criticality	RR | OE Value	Job Size	WSJF

Matrix Techniques – quick filtering

Have the teams place a feature on a matrix of delay time and impact of delay (start with the value you use today, strive for more complete cost of delay). Do higher impact, shortest delivery time items first. Erik Willeke uses a variation where ONLY product owners can move items up or down (indicating higher or lower impact) and the development team left and right (shorter or longer to deliver).



SAFe Weighted Shortest Job First Variations (un-sanctioned)

Some variations of the basic SAFe formula and technique have evolved to make the computation more likely to match ideal.

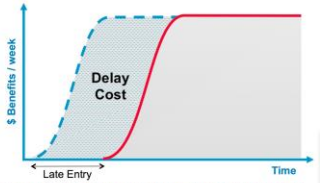
1. "Arnold Mod"
In an email thread conversation between Martin Burns and Joshua Arnold, the suggestion of making Time Criticality more dominant was suggested. This solves the theoretical problem that something "Critical" might score a lower WSJF due to a high business value or risk reduction or opportunity enablement or a low size. Martin noted that these rarely occur due to earlier decision processes, but this suggestion would solve these even if they slipped through.

$$WSJF = \frac{\text{Time Criticality x (Value + Risk Reduction or Opportunity Enablement)}}{\text{Job Size}}$$

2. Scale and Weight the Arguments (solve the mathematical issues of different argument units)
The use of Fibonacci numbers for the input arguments is an attempt to make the estimates relative to each other for the same input argument, but there is a chance that the magnitude is different for each value. For example, a "5" in value might be $100,000, but a "5" in risk reduction might be $500,000. If we just added them as the original SAFe formula says, the result makes little intuitive sense. To correct, either scale the values to normalize across arguments, or multiple each Fibonacci value by a weighting multiplier to correct the magnitude mismatches.

How value is lost due to a delay – Urgency Profiles

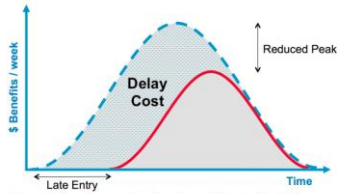
Value erodes differently for different products and markets. The most commonly calculated is just the loss of revenue on the front end because of being late. But, the lost value can be much more than that if the delay causes a permanent erosion of market share, or if the market window is short. It can be difficult to calculate the longer term value erosion, but it may be significant.



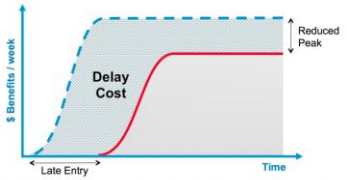
For ideas with a very long-life, with peak unaffected by delay

Tips:
1. Estimate the lifespan of the feature or product. If it has a shorter market window, do it sooner.
2. Estimate if a delay will permanently impair market share. If it does, do it sooner.

Images credit: blackswanfarming.com/urgency-profiles/



Short benefits horizon, and reduced peak due to late delivery



For ideas with a very long-life, with reduced peak due to later delivery

DOs
- Encourage better economic decisions
- Use the lightest analysis method to get a decision
- Use these methods to help have conversation about what value means to each feature or product
- Find better ways to measure and estimate
- involve a diversity of viewpoints on both value and delay.
- Consider reducing risk in a project earlier as adding value

DONTs
- Use complex analysis on small items. Ideally only for features and larger
- Ignore delivery time or its proxy job size; this leads to sub-optimal ordering
- Create an arms race for "value" by prioritizing on it alone (biggest liar wins syndrome)
- Use the highest paid persons opinion if at all possible, offer alternatives!

REFERENCES

Donald Reinertsen:
Books: Principles of Product Development Flow has great Cost of Delay ideas and concepts.
Video: Cost of Delay: Theory & Practice with Donald Reinertsen https://www.youtube.com/watch?v=OmU5yIu7vRw

SAFe:
http://scaledagileframework.com/wsjf/

Joshua Arnold's blog:
http://blackswanfarming.com/category/cost-of-delay/

Chris Matts Blog:
https://theitriskmanager.wordpress.com

Troy Magennis:
Spreadsheets for Cost of Delay http://Bit.Ly/SimResources
Blog: http://focusedobjective.com/blog/

Size – Growth – Pace Forecasting Model

How Long - Duration

Computes a duration of time in the same period as the pace estimate (e.g. weeks, sprints) To turn this into a calendar date, you will need to add this to a start date (when known), and account for weekends, holidays, etc.

$$How\ long = \frac{Size\ x\ Growth\ factor}{Pace}$$

Size – How Big

How much work required to deliver a feature or project

What is size?
The size argument is a range estimate of how much work is needed to fully deliver the value to customers of a feature or project.

What units can it be measured in?
The same units that actual delivery pace can be measured or estimated in.
Examples from "best" to "worst" -

- A count of work items (stories or epics)
- A sum of size buckets (low's, med's, high's)
- A sum of story points (points)
- A sum of times (hours or days or weeks)

How should it be estimated?

- As a range, not as an individual number
- With a goal the eventual actual falls within the estimate range.
- See the Size Assumption worksheet

Suggested actions -

- Use relative estimation. Lay out stories on a table and sort from left (smallest) to the right (hardest)
- Keep a record of prior completed feature story counts to calibrate new estimates
- Reflect during retrospectives why some estimates worked better than others

Growth (factors) – How Known

How much work gets added after starting the feature or project that must be completed

What is growth?
The growth factor argument is a range estimate of how much additionally discovered work is needed to deliver a feature or project.

What units can it be measured in?
The same units that size has been estimated in. Some growth applies to every item (multiplicative) and some is additional.
There are four basic types of growth -

- Time based (new ideas after we start)
- Rate Based (every item add 1 to 3 items)
- Scale Based (unit correction size & pace)
- Event based (risk of extra work)

How should it be estimated?
See the Growth Assumption Worksheet.

Suggested actions -

- Release more frequently to limit time based growth
- Keep a record of prior completed item rate based growth factors
- Be alert to how items have split from the backlog into delivered items. Its normal items split from 1 to 3 times as teams look at stories in detail.

Pace of Delivery – How Fast

How fast work is completed in a deliverable state

What is pace?
The pace argument is a range estimate of how much work is completed over a fixed period of time. E.g. stories per week, points per sprint.

What units can it be measured in?
The same units that size is estimated. Decide the time period these units are counted over. The smaller the period (day, week, month) the more granular forecast period can be computed.
There are normally three paces to consider -

- Ramp-up time – as the team form/learn
- Stride pace – the fastest consistent pace
- Ramp-down pace – as the team delivers

How should it be estimated?
See the Pace Assumption worksheet.

Suggested actions -

- Start with a range estimate, move to actual data after 7 to 11 samples
- Consider ramp-up time and ramp-down time. Pace often is half of the assumed stride pace for the first and last 20% project time
- Often it's easier to estimate the team's stride delivery pace and adjust down based on ramp-up and down impact

Feature or Project Size Assumption Worksheet

Step 1 – Split the project or feature if possible.

Splitting features and projects into smaller batches helps keep uncertainty to manageable levels. People are better at judging small to medium size features and projects versus large and huge sized projects. Always look for ways to split work into smaller batches before estimating and forecasting.

Step 2 – Choose the units of measure for size.

Best (lowest effort) → Worst (highest effort)

Count of stories

A low and high estimate of how many features or stories it will take to deliver a bigger feature

Pros:
- Lowest effort
- Supports ranges

Cons:
- Diverges if stories are wildly different sizes for one feature versus another

Counts of size buckets

Counts of how many "large," "medium," and "small" stories to deliver a bigger feature

Pros:
- Supports ranges
- Handles coarse size variation

Cons:
- Requires stories to be bucketed in each of the groups

Sum of "story points"

Each feature or story is estimated in relative arbitrary units calibrated to time by velocity

Pros:
- Supports ranges (but rarely done)
- Handles story size variation

Cons:
- Requires every story to be analyzed and estimated

Sum of time estimates

Each feature or story is estimated is expected actual delivery time

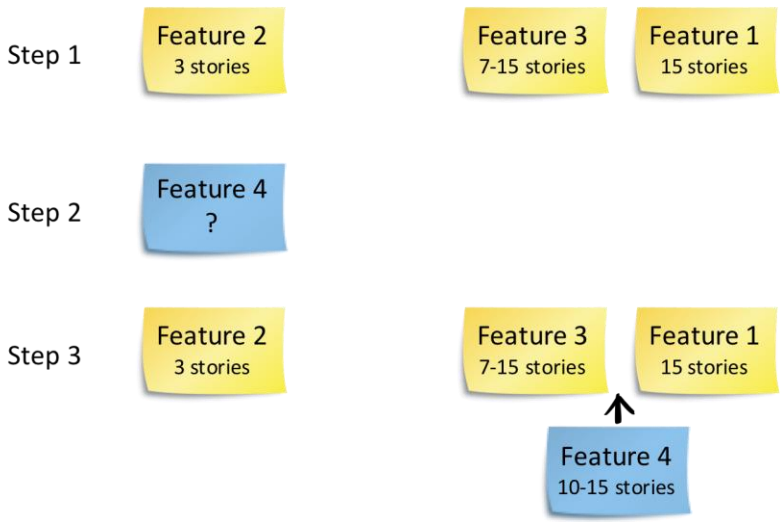
Pros:
- Handles story size variation

Cons:
- Highest effort
- Doesn't account for system impacts on delivery time

Step 3 – Create range estimates (low to high) for each project or deliverable feature in the size unit chosen in step 2.

Method 1: Reference Class Range Forecasting (preferred)

1. Lay out prior features across a table from lowest to highest story count.
2. Introduce the feature being estimated.
3. The group picks a position on the table where this feature fits relative to the others
4. Get agreement on an optimistic (low count) and a pessimistic (high count)



Method 2: Low and High Range Estimate (with 90th Confidence)

1. Start with some calibration exercises. Practice guessing value ranges and researching you are right 9 times out of 10 (90th Confidence Interval). Help people expand the ranges until they are truly confident estimating to 9 in 10 chance.
2. Introduce the feature being estimated.
3. Discuss different approaches to delivering the feature. Decide what approach the group is pursuing. Failure to do this will have half the group estimating one thing and the other half estimating something else.
4. Ask: What would be the likely minimum number of stories needed to do this feature. Discuss and find some agreement.
5. Ask: What would be the likely maximum number of stories needed to do this feature. Discuss and find some agreement.
6. Challenge the group. Make an equivalence bet. Ask the group if they would rather take a chance of winning or losing $1,000 hypothetical dollars if the actual falls in the range, OR the chance of pulling a green ball out of a bag with 9 green balls and one red. Adjust the range until more confidence in the estimates.

References: How to measure anything (Douglas Hubbard). The Failure of Risk Management (Douglas Hubbard). Risk Intelligence (Dylan Evans).

For more probabilistic forecasting resources: FocusedObjective.com Email: troy.magennis@focusedobjective.com

Feature or Project Growth Assumption Worksheet

Time Based Growth

The longer we go the more alterations to original scope get added.

What is time based growth?
The longer the time between committing to delivery and actual delivery the higher risk feature requirements will change. These changes can be additional ideas to current plans, or reactions to a competitive market change. This type of growth isn't bad, it's just inconvenient from a forecasting perspective. For companies to be innovative and react fast to change, adapting plans to accept new ideas and latest information is critical.

1. How much more scope?

Release Frequency	Technically Easy	Hard
Cont. – 2 weeks	1x	1.25x
3 – 6 weeks	1.25x	1.5x
7 – 12 weeks	1.5x	1.75x
13 – 26 weeks	1.75x	2x
26+ weeks	2x	4x

Suggested actions -

- Releasing more frequently limits exposure to time growth
- Don't try and eliminate ALL time based growth. Some is healthy, it means recently learnt lessons become incorporated into plans
- Bent Flyberg is a good resource of material on Mega-projects and recommends avoiding Mega-projects altogether!

Rate Based Growth

The more work we complete the more we learn about what we need to do to deliver.

What is rate based growth?
This type of growth comes from actually completing work. Any growth in story count that has a relationship to completed work falls into this category. Defects and rework discovered are examples of rate based growth.

To capture these, have the team brainstorm items that performing work might cause new discovered work. Keeping an ongoing list of items like this from prior projects helps do this more accurately next time.

2.Things we will also need to do

Growth due to...	Occurr.	# Stories
E.g. Defects	100%	1-3
E.g. Localization	20-30%	3-4

Suggested actions -

- Create a rate based growth table for your feature and projects
- Consolidate the rate based growth knowledge across other teams, limit to top 10
- Use actual data to refine the occurrence rate estimate and the impact for growth item**s.**

Scale Based Growth

The size of completed work is different than the work in our backlog. E.g. work splits

What is scale based growth?
This is the most overlooked growth of work. It is more properly a correction rather than growth. It is caused when the pace of delivered items is assumed to be the pace remaining backlog items will be completed. Why isn't this true? Commonly work is split into multiple items when the team is analyzing the details just prior to adding them to their sprint or pulling that work into the team. Left un-adjusted, it looks like the team will complete faster than they will.

3a. Low guess: Items in the original backlog become x items in complete?
1 2 3 4 5 other:
(default)

3b. High guess: Items in the original backlog become y items in complete?
1 2 3 4 5 other:
(default)

Ask the team what is the lowest and highest likely split rate.

Suggested actions -

- Be alert to anytime the backlog count is combined with historical data; The result will often be optimistic.
- Start with the estimate range of 1 to 3 times.
- Measure the actual rate and adjust.

Event Based Growth

Feedback or things that go wrong in the approval to release process.

What is event based growth?
Sometimes we have a hint that something might go wrong requiring rework to fix. For example, sometimes you build a feature but have little insight to how it will perform with thousands of users. Until it's built and tested under stress, you can't know for certain that it isn't going to need improvement. This is an event based scope increase. Just one risk can make or break a good software forecasts.

4.Things we might need to do

Risk	Prob.	# Stories
E.g. Performance	50-75%	20-30

Suggested actions -

- Pick the top five. If you have more than this, forecasting is pointless, you are guessing. Don't start!
- There will ALWAYS be a few un-avoidable risks. Do these earlier in the project to avoid late surprises..
- Don't spend all your time on total risk avoidance, accept some will occur and forecast accordingly.

Feature or Project Delivery Pace Assumption Worksheet

Ramp-up / Starting Pace
The pace as the team is forming and learning. Often understrength in skills.

Stride Pace = Sustainable completion pace as a team
The sustainable pace delivered as a team. Often limited by flow through a system constraint. Determine where the constraint will be and estimate low and high completion rate through that step.

Ramp-down / Delivery Pace
The pace as the team is in the final delivery phase into production environment.

Start with FIRST 20% of the original work completed at ½ pace and adjust

Start with LAST 20% of the original work completed at ½ pace and adjust

Backlog

Work Completed

Things that DECREASE ramp-up
- Existing team
- Similar recent work types
- Pairing and sharing skills

Things that INCREASE ramp-up
- New team (> 1/3 people new)
- New type of work
- New innovation or technology

1. Where is the likely system constraint limiting flow?

2. How many people do we have who can work at the constraint [1]?

3. How many items per week/sprint can each person (or team) [2] deliver at the constraint?
Lowest guess: Highest guess:

6. How long will ramp-up take?
0% 10% 20% 30% 40%
(default)

4. Low pace = [2] x [3 low]

5. High pace = [2] x [3 high]

8. How long will ramp-down take?
0% 10% 20% 30% 40%
(default)

7. Pace impact during ramp-up?
0.1 0.25 0.5 0.75 none
(default)

9. Pace impact during ramp-down?
0.1 0.25 0.5 0.75 none
(default)

Things that DECREASE ramp-down
- Continuous automated delivery
- Early focus on quality & *done*
- Internal authority to release

Things that INCREASE ramp-down
- Late integration testing
- Batch steps: e.g. Localization
- External authority to release

Delivery pace over time

How do we find the constraint?
The constraint is the limiting factor of a system delivering faster. If you can't observe a system to visibly see where work is queuing (a great indicator of a constraint), look for where specialist skills are needed and you expect to have too little. I often ask what skill would you add more of to increase flow - that's generally the constraint!

Why do we only estimate at the system constraint?
Every system will have a constraint. The speed that work enters the constraint, and the speed that work completes after the constraint doesn't impact total system pace – system pace is the same as the pace at the constraint.

What if there are multiple possible system constraints?
It can be hard to predict where the limiting system constraint will be. If this is the case, brainstorm and estimate the top 3 most likely, then average the low estimates for an average Low Pace for them. Do the same for the High Pace estimate.

What values should I use for average pace?
If you are using a forecasting tool that doesn't support multiple feature/project phases, an overall average low and high pace can be calculated using the following formulas (note: [6] = value from box 6)-

X = Ramp up interval = [6]/100
Y = Ramp down interval = [8]/100
Z = Stride interval = 1 – (X + Y)
Up avg = X x ([7] x [4])
Stride avg = Z x [4]
Down avg = Y x ([9] x [4])

Low Average = Up avg + Stride avg + Down avg
(Repeat for the high, replace [4] with [5])

What about using actual data?
Use actual throughput or velocity data as soon as possible to confirm these assumptions. Remember, the early samples will be the in ramp-up phase, and be slower than the expected stride pace by the factor estimated in box [7] above. Adjust using the formula -
Stride pace = (1 / box [7]) x Measured Pace

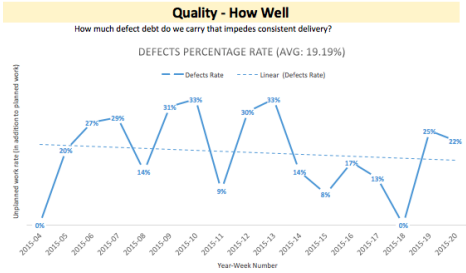
Team Dashboard Spreadsheet Cheat Sheet

What is the intended behavior?
Help teams continuously triage and complete defect backlog items at a consistent rate. Try and drive this percentage lower without deferring defects.

How is it gamed?
- Defects are entered as additional stories (seen by growing throughput, but lowering customer satisfaction and complaints)

When overdriven, what moves?
- Responsiveness metric increases for defect types.
- The number of open defects grows if story work is done in preference.

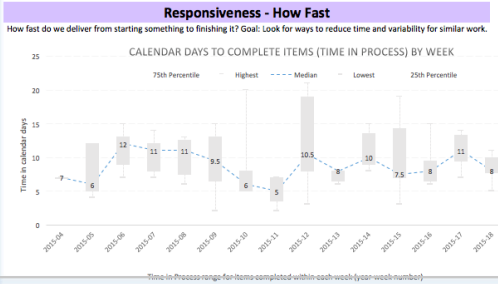
What is this chart?
Quality is measured as the percentage of finished work that is marked as defect type. The number of defects finished each week is divided by the total count of finished items that same week.



What is the intended behavior?
Help teams understand the time in-process (cycle time) for completing items, and to look for ways to reduce the time (and variability) by eliminating unnecessary steps and waste.

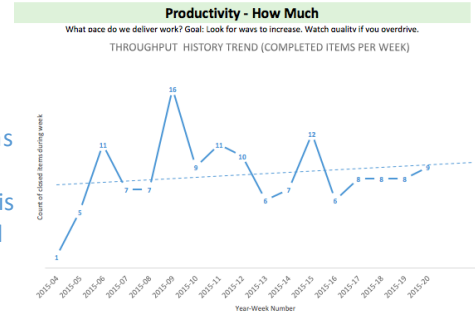
How is it gamed?
- Only fast simple work is started (seen by initial increasing productivity only to regress later)
- Premature sign-off of items as finished (seen by growing defect counts)

When overdriven, what moves?
- Quality metric goes up as more defects are reported due to premature completion.
- Predictability metric oscillates into higher peaks as more work and more defects get opened



What is this chart?
Responsiveness is measured as the median (blue line) calendar time in-progress for all items completed in the same week. The minimum, 25th & 75th percentile and maximum values are shown to see the variability each week.

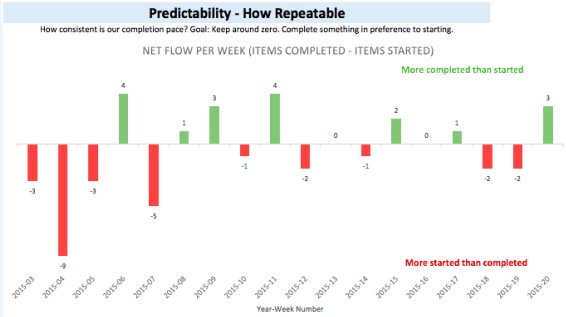
What is this chart?
Productivity is measured as throughput per week. The spreadsheet counts how many items were finished each calendar week helping see the trend over time. This chart includes all items, defects and story work.



What is the intended behavior?
Help teams find what level of throughput per week is consistently achievable and to find ways to increase this over time by improving their processes.

How is it gamed?
- Throughput can be increased by breaking down items into smaller pieces (seen by decrease in responsiveness).
- Teams could also prematurely sign-off work only to have defects reported later (seen by increase in quality defect ratio).

When overdriven, what moves?
- Quality metric goes up as more defects reported.
- Predictability moved down into more “red” bars.



What is this chart?
Predictability is measured as net flow per week. The spreadsheet counts how many items were finished and subtracts how many items were started for the same calendar week creating a positive bar (green) if more is finished than started, or a negative bar (red) if more items were started than finished.

What is the intended behavior?
Help teams focus on finishing something in-progress (or helping someone on the team finish something) before starting something new. Help teams see when impediments inhibit finishing work already in progress.

How is it gamed?
- Teams can prematurely sign-off work only to have defects reported later (seen by increase in quality defect ratio).
- Teams can slow down starting new work (seen by a decrease in throughput productivity)

When overdriven, what moves?
- Quality metric goes up as more defects reported.

For more probabilistic forecasting resources: FocusedObjective.com Email: troy.magennis@focusedobjective.com