Cheatsheets and Worksheets

Project by Focused Objective to capture the essence of Agile math, statistics and forecasting

Download these plus many more spreadsheets and presentations from:

Bit.Ly\SimResources

Please use and share anything you see for work or pleasure.
Just DON'T charge money for them!

Contact troy.magennis@focusedobjective.com with ideas and feedback

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

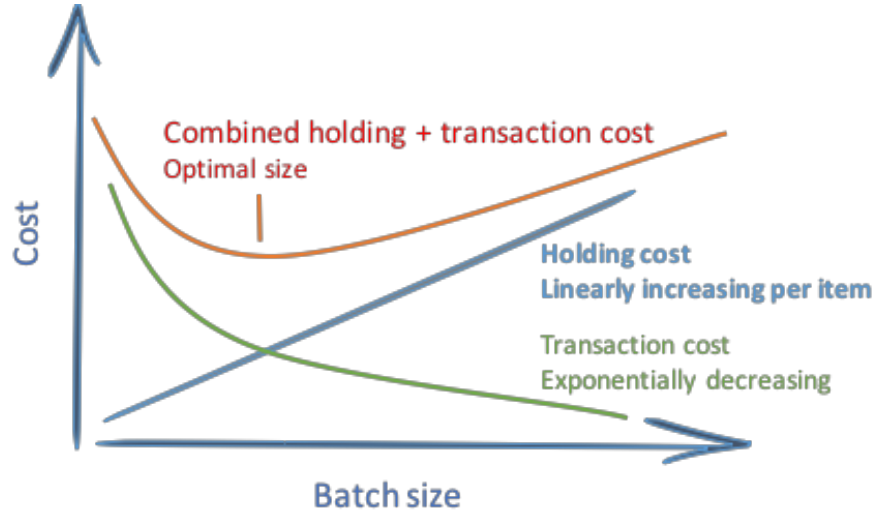
There are two types of cost to balance -

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?

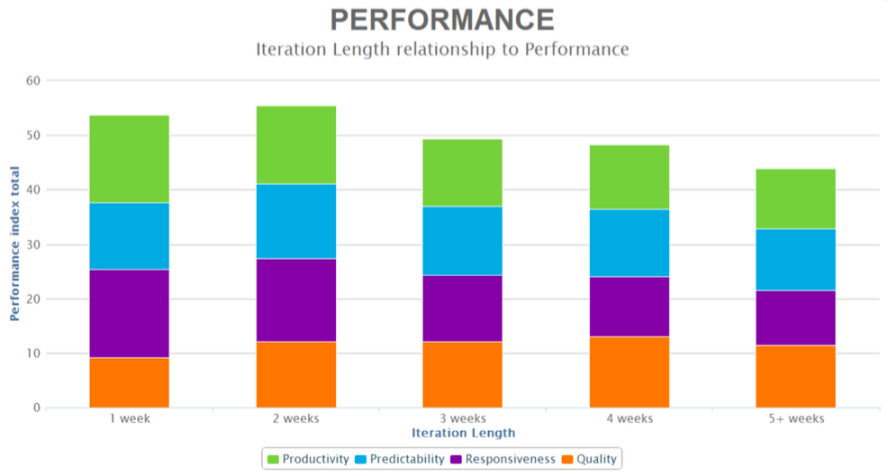
Some projects require performance testing. This type of testing can be labor intensive to setup and time consuming to execute and analyze. Some teams only perform this late in a project causing late feedback on poor performance architecture and code. The right batch size in time and amount of features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

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

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

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

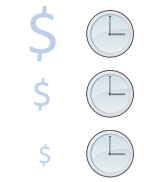
Every item has a different economic impact by being delayed. The impact will be a balance of lost value, and how long they are delayed.

"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



Harder to decide when both value of what NOT done and duration change

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

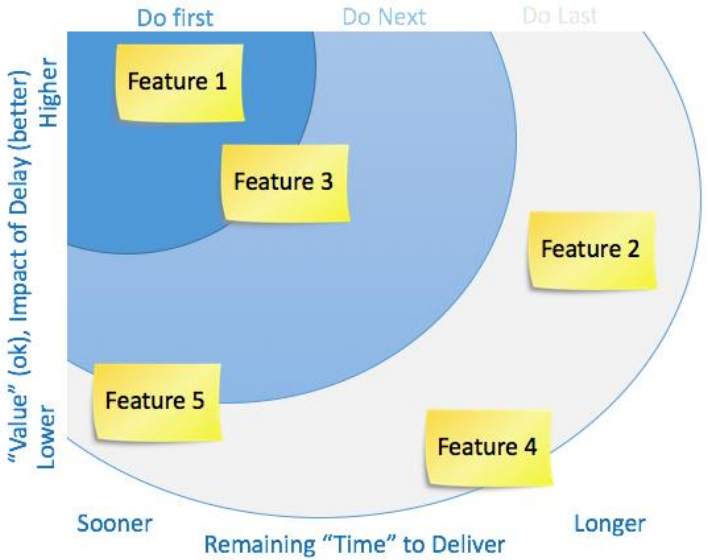
SAFe's Weighted Shortest Job First formula (upper), and a typical data capture table (lower)
More info: http://www.scaledagileframework.com/wsjf/

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

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).



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!

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.

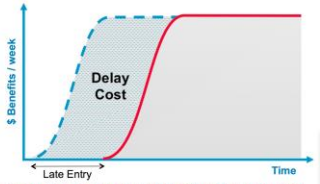
1. "Arnold Mod"
In an email thread conversation between Martin Burns and Joshua Arnold, the suggestion of making Time Criticality more dominant was suggested. This solves the theoretical problem that something "Critical" might score a lower WSJF due to a high business value or risk reduction or opportunity 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

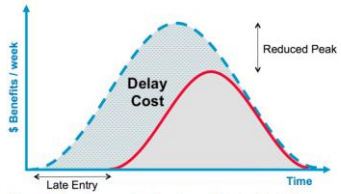
Value erodes differently for different products and markets. The most commonly calculated is just the loss of revenue on the front end because of being late. But, the lost value can be much more than that if the delay causes a permanent erosion of market share, or if the market window is short. It can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



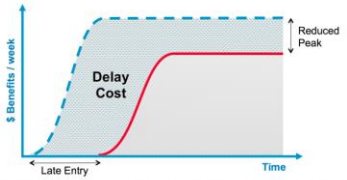
For ideas with a very long-life, with peak unaffected by delay

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

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/

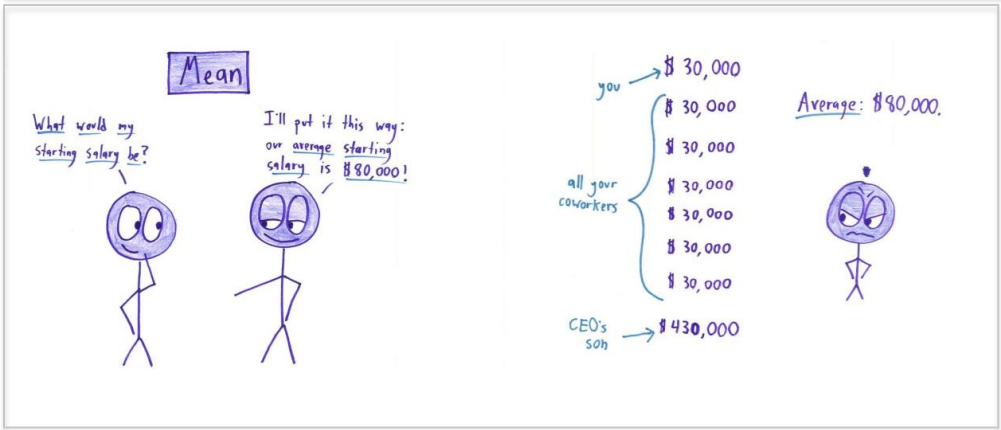
Joshua Arnold's blog:
http://blackswanfarming.com/category/cost-of-delay/

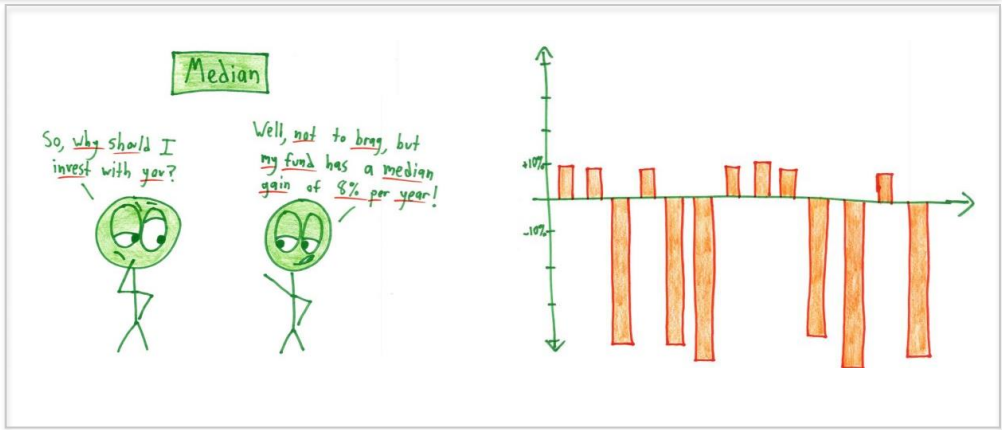
Chris Matts Blog:
https://theitriskmanager.wordpress.com

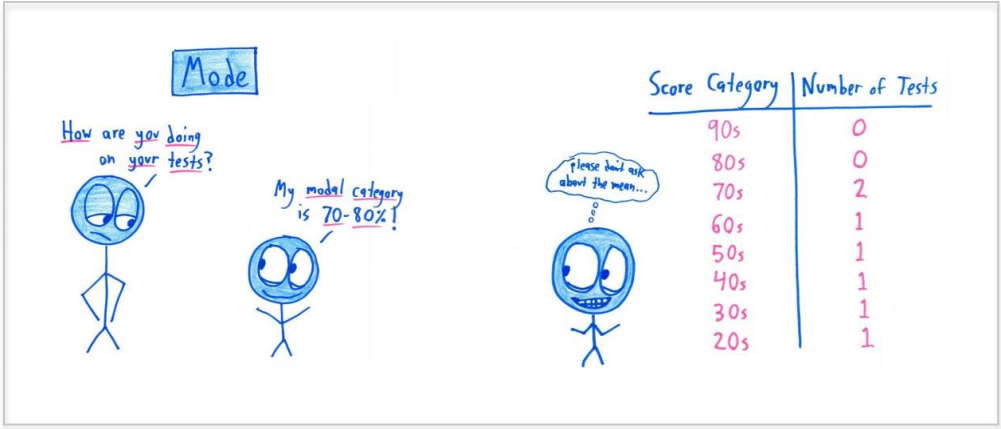
Troy Magennis:
Spreadsheets for Cost of Delay http://Bit.Ly/SimResources
Blog: http://focusedobjective.com/blog/

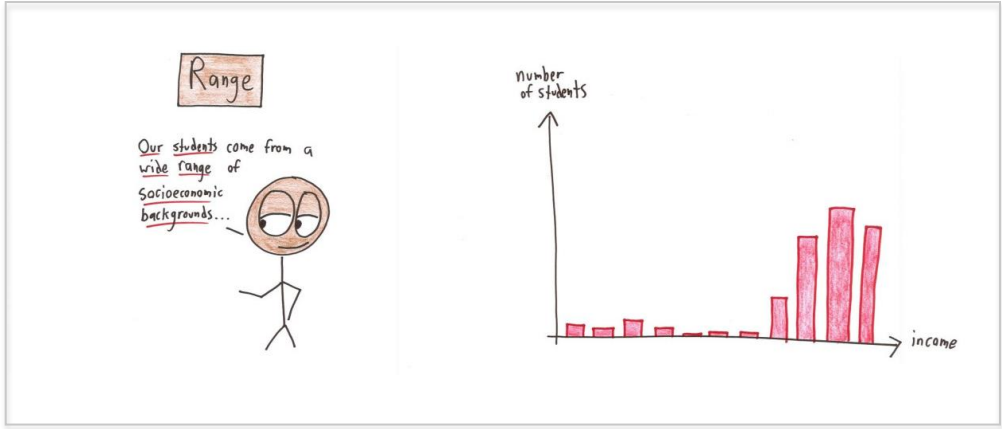
Jointly designed in an online conversation by Martin Burns, Don Reinertsen, Chris Matts, Joshua Arnold, Tony Grout and Troy Magennis sometime during 2016. It is, and will remain a work in progress. Trademarks used NOT ours.

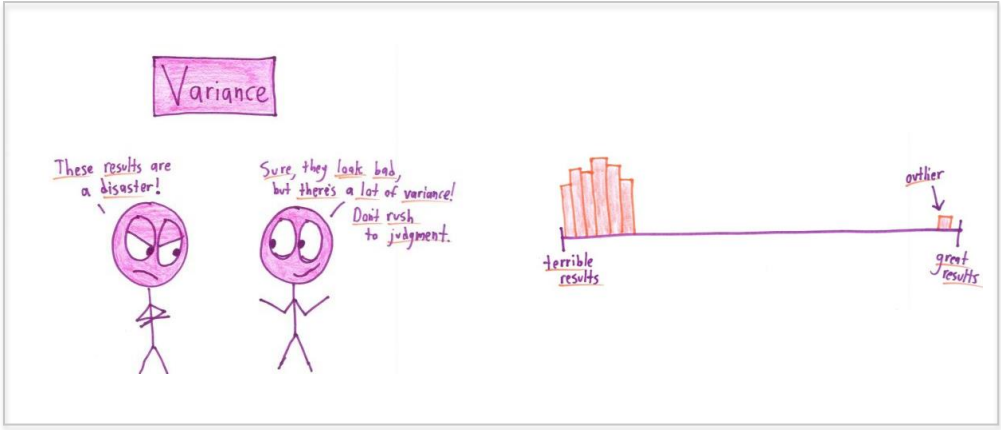
Statistical Measures – mathwithbaddrawings.com by Ben Orlin

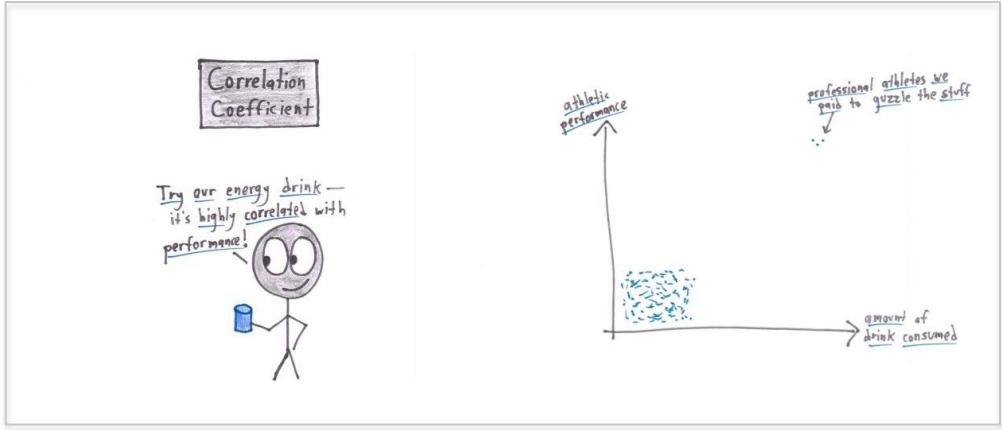












Why not to trust statistics – source and credit for these images https://mathwithbaddrawings.com/2016/07/13/why-not-to-trust-statistics/

These drawings come from Ben Orlin's fabulous Math With Bad Drawings blog. This set of drawings shows a different statistical measure and how it can be misused in some way. The intention is to be aware when you are given a statistical measure how to interpret how it may not be giving you the larger picture. Take care! Follow Ben on Twitter @benorlin

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

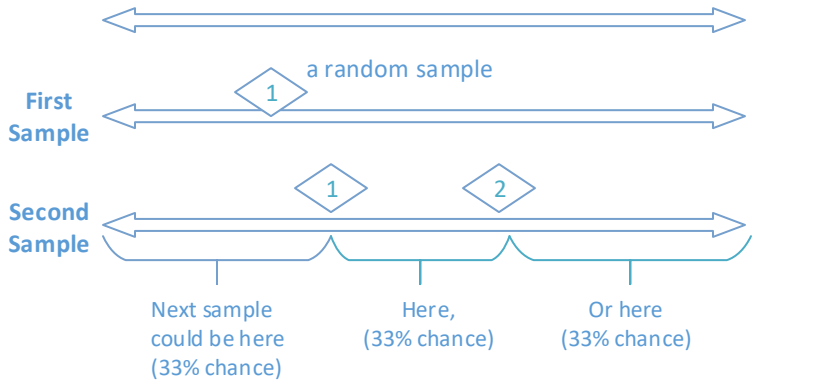
Probability the next sample is lower than the lowest sample so far after "n" = $\frac{1}{(n+1)}$ X 100

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

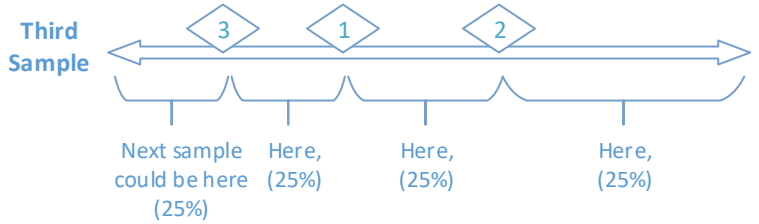
Samples so far (n)	Probability for each interval	Probability next sample in range
1	50.00%	0.00%
2	33.33%	33.33%
3	25.00%	50.00%
4	20.00%	60.00%
5	16.67%	66.67%
6	14.29%	71.43%
7	12.50%	75.00%
8	11.11%	77.78%
9	10.00%	80.00%
10	9.09%	81.82%
11	8.33%	83.33%
12	7.69%	84.62%
13	7.14%	85.71%
14	6.67%	86.67%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

Latent Defect Estimation Worksheet

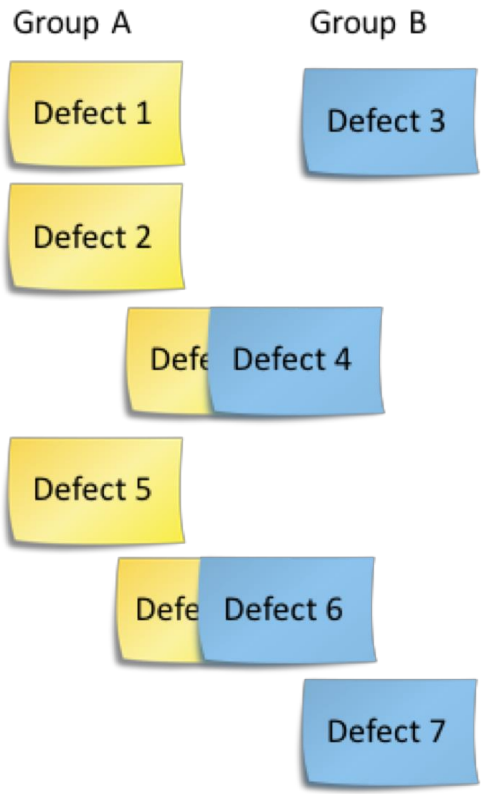
Capture-Recapture Method for latent defect estimation

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.

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

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



Total found by A = 5

Total found by B = 4

Found by BOTH = 2

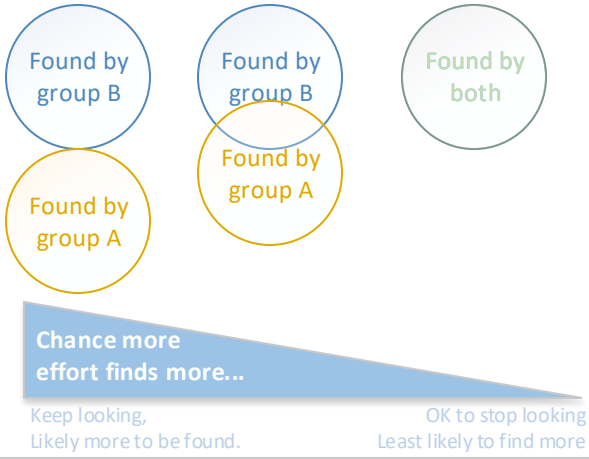
Total found = 7

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

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

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.

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

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

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

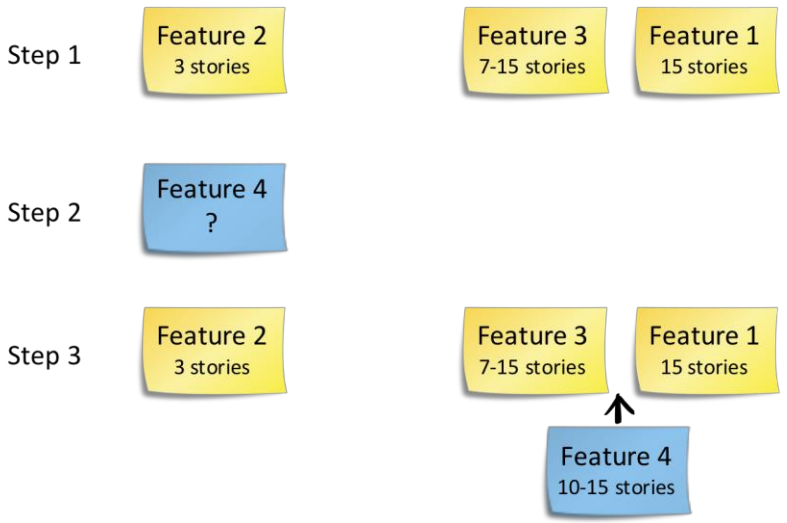
Pros:
- Handles story size variation

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.

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

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

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

Forecast Assumption Worksheet

1. What are we planning to build?

2. We know we have achieve this when?
Who gives the final "go live" decision?

3. Desired delivered to customers date:

4. Cost of delay (per week or month):

5. To START, we need

The following tools and equipment -

The following questions answered -

Minimum dedicated team and skills -

Dependencies (other projects or infra. needs)

6. What things might IMPEDE progress?

Other projects that might inhibit focus -

Events (holidays, conferences, training) -

7. To DELIVER to customers, the following things need to be completed as well

To beta or internally test -

To make live to all customers -

8. What do we need to LEARN to deliver

Learn what | How?

9. What SKILLS do we need to deliver

Skill | # People (T=teachers, D=doers)

10. How will we avoid finding quality or issues too late?

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