BATCH SIZE MATTERS

Idea in brief

In software development a batch is either the quantity of work required for or produced by a single process. Unfortunately, in IT we tend to work with large batches because an internal rule of thumb insists this is the most efficient way of doing work. When working with smaller batches, it soon becomes apparent that the benefits far outweigh the costs of creating them.

Stage gates and the need for information drive behaviour that tends toward the creation of large batches. The batch then becomes dependent on the slowest element when it comes to passing a stage gate – flow is impeded and queues build. This means that, within a large batch, valuable items that could travel faster are held up by slower items, and value is not realised as soon as it could be.

Smaller batches produce fewer queues, give faster cycle time, increase urgency and decrease risk through faster feedback. Because reducing your batch size is cheap and reversible, you can experiment by changing batch size without suffering drastic economic penalties. And because feedback arrives sooner, this information allows us to respond to opportunities more effectively.

Enabling effective use of batch sizes in not always straight forward but the cost of not doing so is high. For companies with a need for flexibility/agility, reduction of batch size is a key step to implement. Remember the new rule of thumb: your batch size should probably be smaller.

Ideas in practice

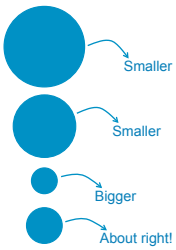
a **BATCH** is the
QUANTITY OF WORK
or ideas that we
PROCESS TOGETHER

"Big batches (waterfall), small batches (almost all forms of agile)... the number of ideas processed is bounded by time or size."
David Anderson

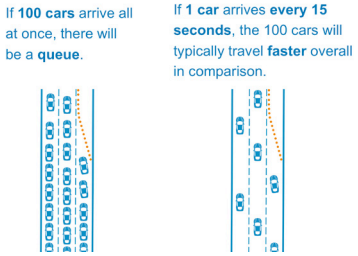
- The batch may be a number of features, requirements, bugs to fix, etc. processed at any one time.

We can experiment with changing batch sizes.

It is cheap and reversible.

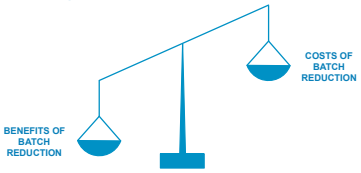


- Software is not neat and easy to divide up into batches... but choosing a new batch size shouldn't be arduous.
- Experiment – pick a batch size, and then change it depending on what you observe.



- Changing batch size can optimise flow.
- Reducing batch sizes helps to reduce the variability of flow and induce a steady state that minimises the chance of a queue forming.
- Small batches drive shorter cycles, which can encourage responsibility and motivation.

But the benefits typically outweigh the costs



- As we reduce batch size, it leads to an increase in repetition which means an increase in overall transaction cost.
- Investing in automation and infrastructure becomes key in keeping transaction costs down.
- The benefits of batch size reduction typically outweigh the costs.

Well, I'm glad you asked...

C_T = Total cost per year
C_t = Transaction cost per year
C_h = Holding cost per year
F = Fixed cost per batch
N = Items per year
Q = Items per batch
H = Holding cost per item per year

$$C_T = C_t + C_h$$
$$C_T = \frac{FN}{Q} + \frac{HQ}{2}$$
$$\frac{dC_T}{dQ} = 0 = -\frac{FN}{Q^2} + \frac{H}{2}$$
$$\frac{FN}{Q^2} = \frac{H}{2}$$
$$Q^2 = \frac{2FN}{H}$$
$$Q_0 = \sqrt{\frac{2FN}{H}}$$

Taken from *The Principles of Product Development Flow* by Don Reinertsen

- You can ignore the formulae! Instead use this new rule of thumb:

Your batch size can probably be smaller

- Note that the optimum batch size graph is a U curve – that means it is what Reinertsen calls 'very forgiving of errors'.

So the morals of this story are...

reduce input to limit batch size

small batches give early feedback on what the customer really wants

lots of analysis up front leads to larger batches

detailed project plans up front are likely to lead to larger batches

- Today, IT tends towards lots of analysis and detailed project plans up front – both of which are likely to lead to larger batches.
- Reducing input is the simplest and cheapest method of limiting batch size.
- Small batches drive early, regular and rapid feedback.

emergn