Table of Contents

"Second Generation Lean Product Development" -- Don Reinertsen

AWA Meetup London England September 2015

Notes and Review

As usual a very “dense” pitch with a lot of information. If you want to understand impacts of things like queuing theory, lean, and why variability should be preserved for new product development (hint: Black Scholes option pricing model) then this is the video for you.

Preamble

I.e. Not traditional upfront planning, all requirements which is 1st generation

Have you ever had a product development where there wasn't a legitimate business need to change direction? Never so old models don't work.

7 big ideas in 2nd generation lean product development

Understand your economics Manage your queues Exploit variability Enable smaller batches Control WIP and start rates Prioritize based on economics Accelerate feedback

Understand your economics

All product development decisions have more than one moving part. Therefore have to quantify to make decision

Eg cost of delay. Turns out to be very easy to calculate

Should we operating our testing process at 80% utilization with a 2 week queue or 90% utilization with a 4 week queue. Two moving parts therefore basic lean (i.e. “Remove waste” etc) won't help. Comparing two weeks of cycle time to 10% points of utilization. Which is better? But two different units of measure

So what to do Bring to same unit of measure Therefore what should be (Who are you trying influence) - people who control the money Metric might be something to do with $ Life cycle profits

Life cycle profits made up of Waste Cycle time Variability Efficiency Revenue Unit cost Value added

But all need to converted to see impact of one dial on life cycle cost

Cost of delay is just the partial derivative of life cycle cost wrt cycle time.

How do I figure out the cost of being late. Make a model that helps me determine

Create baseline model (everything goes to plan) Add in factors - model expense overrun, model cost overrun, model value shortfall, model schedule delay, model risk change Use this to determine total profit impact of missing a MOP Use this to calculate sensitivity factors

Can see things like “a one percent expense overrun will result in loss of $40K” and compare to things like “a one month delay in release will result in loss of $500K” so have the basis for decision making.

Plus can process them almost instantly - not through accounts of department, so can reach and touch lots of little decisions not just the big go / no go decisions

How accurate does it need to be - two three significant digits We have to outrun “intuition” 10 people in the room who all have the ability to delay this project “What it will cost the shareholders if I introduce 60 days late” Rank high to low Do this to do to people as this is your accuracy benchmark Average range is 50:1 Range of cost of delay - 10:1 range with calculation If improve decision process to so that it is better

But what if cannot calculate cost of delay “Garbage in means garbage out” because we cannot estimate sales If you go through the same process as above and rank sales estimates, the range sales is say 4:1 which is way better than the estimates on cost of delay at 50:1

We are better at sales because we've been estimating sales for a long time. We have not done this with cost of delay.

Get better with cost of delay calculations

Quote Message - “any analysis beats intuition”

Should you worry that you have a 2 order magnitude delta in cost of delay.? If you have smart people making decisions based on their beliefs and you have a 2 orders of magnitude variation in the belief then there is a problem in the decision making - i.e. Generating chaos, and not creating alignment (marketing says this is important, engineering says that, purchasing another)

Boeing 777 Story Pass decision process - you can make decision for $300 / pound yourself mr engineer, supervisor gets 600 while program manager gets $2500 $300 is system level optimal point for 5000 engineers - each making decisions every day Not that it's not the most important person making a decision Program manager has overall picture - decision rule so he does have to participate in these decisions

This is another use of economic model

Manage your queues

What happens when you look at product development through the lens of queuing theory.

Too many people think that higher utilization means you get more and so don't bother measuring the queues. Miss true cost of overloading process and this hurts all aspects of development performance.

Problem is that queues are invisible (queues are information)

Traffic at rush hour illustrates classic characteristics of queuing system 4 lane traffic at rush hour - one direction Now take out one lane so reduce capacity by 25% Does this lead to 25% reduction in average cycle time - no it doubles and trebles it If I do this at 3am there is no impact on average cycle time

Quote “Changes in loading have a non-linear impact and the non-linearity depends on the capacity utilization.”

Effect of capacity utilization on queue size See chart for M/M/1 /infinity queue As increase utilization percentage Every time we halve the distance to the right hand access we double the queue

When capacity utilization (rho) approaches 100% queue size approached infinity Rho / (1 - rho)

If you ask manufacturing guy about max utilization of factory they would say “max 85%” for example. Very consistent

What has more variability engineering (done one time in a row) or manufacturing (done n times in a row) Expect lower utilization to deal with variability But we often hear about loading our work to 95% and then wonder why things don't flow through the system. We are proud we are keeping engineers busy but work product is sitting around waiting for the next step in the process.

Manage queue Means don't just worry about cost of excess capacity Need to also think about cost of delay Queues cost more money than the cost of engineers Results in U curve to determine optimal How many jobs are waiting are waiting in the cad line, vs cost of new machines

Use queue size as the control variable, don't do capacity utilization Sell by “We have too many projects in process, we can cut the cycle time for each project”

Why queues matter? Queues create Longer cycle time Lower quality - delay feedback loops More variability Increase risk - force you to plan with longer planning horizons. More overhead - too much stuff to track in the process Less motivation - “I'd really like this to be finished by Friday so it's ready for testing” “but testing has 3 week delay so nothing helps” Enables faster feedback.

Quote: “If you give a coder feedback after 90 days after he wrote it the most common response is 'that's not my code” or at least don't remember it.

Exploit variability

Doesn't make sense to eliminate variability in software world

Problem with reducing variability in product development is that a byproduct is that you remove innovation.

6 sigma - makes everyone risk adverse Track “revenue from new products” to understand the effect over time

Quote - Product development produces the recipe for a product. Use it over and over again once you have it. But for us if we produce the same recipe twice you've added no value. You have to change the recipe to add value.

Financial theory Help explain this to people

Black Sholes pricing model

Calculate by multiplying two functions together Expected price - almost normal distribution Payoff price - as there is a value on option that you won't exercise the option Result multiplication of two functions is curve With Asymmetric payoff and option pricing Area above the line on right is bigger than stuff on left All the variation comes not from the expected price, but from the payoff price

Increase variability on a stock price Increases return I.e. It's wise to have increased variability Payoff function for manufacturing is different In product development we have asymmetric payoffs - not true of manufacturing.

Asymmetric payoff in product development? Huge difference between payoff and the cost Frequently see this in product development

Answer is not “celebrate variability” Only works when you encounter an asymmetric payoff function

I.e. Can add variability to product development and not add value. Problem is cannot add value without introducing variability

Example If you had 10 runners in a race all with the same mean race time, would you invest in the runner with the highest standard deviation in the race, or the lowest. Answer is highest because they have long tail, i.e. Higher chance to win overall (this is example of asymmetric payoff. Only one person wins, rest lose)

Enable smaller batches

Software industry is example of best at doing this especially in test. Better at lean than people who have 40 years experience. Because we assume we have change happening

We've tended to under manage batch sizes in product development and sometimes we've institutionalized large batches (eg phase / stage gate → big batch to big batch - all requirements into design - then 100% of the work product going from requirements to design. This means we are doing

maximum theoretical batch size which from queuing theory means maximum theoretical cycle time

Opposite of “lean”

Linear relationship between batch size and cycle time Cut batch size in half and you cut cycle time in half.

Cheap, easy to do Can undo if it doesn't work out

Huge advantage in speed of feedback with small batches

Deliver work in multiple items of work.

Metaphor - who arrives first if have 40 people traveling in one bus. Answer everyone. Alternative is to have 4 people travel in each of 10 cars. Now have opportunities to change things around.

Batch size reduction allows you to change order of execution Take advantage by putting things with high risk in early batches if small batch size

Benefits of small batches with testing Smaller changes → fewer open bugs → faster cycle time → early feedback → better economics

Smaller changes → less debug complexity and more efficient debug → cheaper debug Fewer open bugs → more uptime and higher validity → cheaper testing Fewer open bugs and faster cycle time → fewer status reports → less non value add Faster cycle time → early feedback → faster learning and lower cost change → better code and cheaper correction

Quote “The longer something is in transit in a process, the more likely it is the requirement will change”

Separate the triage (decision) from the dispatch decision. You have to finish one thing before taking on the next into the system.

If you change one line of code and have a defect it's pretty easy to figure out where the problem is. If two lines then defect could be from either of those lines, or an interaction between those lines. Interactions go up 2**n (where n is the number of changes) - massive rise in defect complexity

Setting batch size Trade off between transaction cost and holding cost - u shaped curve Eg go shopping for eggs once a year instead of every week Transaction cost low (once a year). Holding costs high (I'd probably have to buy extra refrigerators, money tied up in egg inventory etc). Alternative is one piece flow - we eat and egg, we buy and egg. Obviously bad.

Automated testing drives down transaction cost, enabling small batches. The ways you reduce batch size by focusing on transaction costs. Doesn't work to just mandate small batches as that will just raise the costs. DevOps at Facebook have done this - drove down transaction costs (if code doesn't work annoys a lot of people, so reduce transaction cost by only have a few people see it)

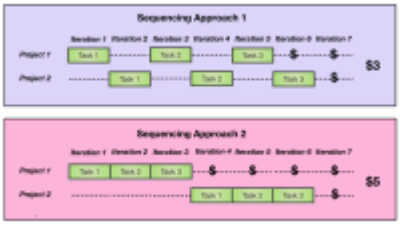
Control WIP and start rates

Mistaken assumption is the sooner you start work the sooner you finish. Dilutes resources.

Littles law - Wait time in the queue is the length of the queue divided by the departure rate. Therefore reduce amount of WIP, reduces the amount of transit time in the system. Focus on WIP instead of focusing on cycle time

Therefore Control the number of active projects.

Same number of projects, but if you focus on one project you will finish early one first one. Overall will take the same amount of time Discussion for management - “help me understand why you are working in such a way as to produce the least amount of profit for the company?”



Also helps 2nd projects as well as you make decision late - improve requirements, learn from first project, etc

How many degrees of freedom do you have in your requirements Avoid long planning horizons “We developed the product so fast the marketing didn't have time to change their mind” Actually this generally - short planning horizons are more stable Improve quality of requirement is to shorten the time horizon

Also reduce the queue time

Control WIP with visual control board

WIP constraints can be local, regional, or global

On visual control chart horizontal axis is state Vs on Gantt chart horizontal axis is time This is fundamental difference in thinking approach for how we manage work. And that changes everything. Because if we know the state, we know how big the queue is and since the time in queue is the majority source of cycle time we are aware of where that time is spent (not true of a Gantt chart) Simple look, but massive change

Prioritize based on economics

Use FIFO for homogenous flow - cost of delay against duration are the same for 3 projects Manufacturing use this

Want to prioritize based on weighted shortest job first Maximum benefit per unit cost consumed Changing the queuing discipline to WSJF produces dramatic differences in delay cost (eg 96% reduction) without changing demand etc in the system. All I am doing is working on high cost of delay items first.

Need to have cost of delay - must have this

Weighted shortest job first - operating system term

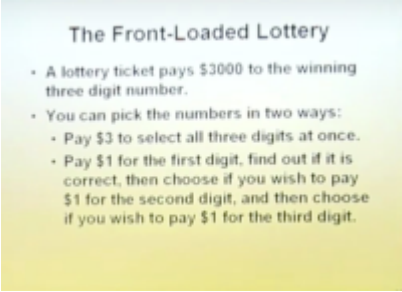
Economic advantages are compelling - safe is about the way to apply it. Heading in the right direction and way better than alternatives in prioritization.

Accelerate feedback

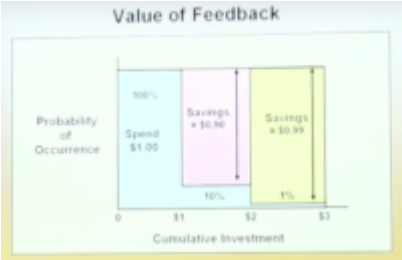
Queue / batches are large means feedback loops

Why feedback.

Use front end loaded lottery to help explain the benefit of “early feedback”



Radically changes the economics Average pay off is $1.89 not break even Difference between two games - option of shutting down the after first number and again on the second. This allows us to improve the asymmetry of the payoff This is the value of feedback:



Traditional approach to product / project development is like watching a horse race. Place bets at beginning and hope that your horse comes in. Product development should be see as a horse race where you can move your bets after the horses have started running.

Lean startup - option to “stop persevering”

You start working on the feature that you think is useful to 50% of your customers and cost about 1 month to develop. You then find out that it is only useful to 5% of your customers and the cost is now 6

months. The economics of the feature have massively changed so now you make adjustments. Time to throw that idea the bus.

Everything we are delivery today is more valuable than everything we threw under the bus. Allow to add things that are more valuable. Original plan is a based on noisy economic value

Want to Know More?

- Second Generation Lean Product Development Flow by Don Reinertsen

Webinar, Video, System, Queuing, Utilization, Review