allan's blog - Agile & Digital Business

I help companies and teams that create software.

Friday, October 30, 2015

Software has diseconomies of scale - not economies of scale

> "Practical men, who believe themselves to be quite exempt from any intellectual influence, are usually the slaves of some defunct economist." John Maynard Keynes

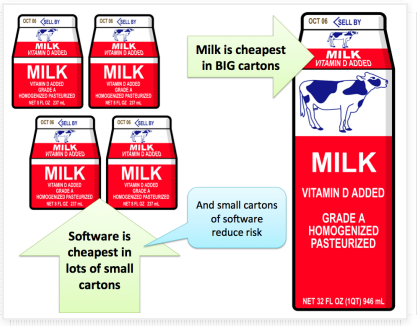
Most of you are not only familiar with the idea of **economies of scale** but you expect economies of scale. Much of our market economy operates on the assumption that when you buy/spend more you get more per unit of spending.

At some stage in our education - even if you never studied economies or operational research - you have assimilated the idea that if Henry Ford builds 1,000,000 identical, black, cars and sells 1 million cars, than each car will cost less than if Henry Ford manufactures one car, sells one car, builds another very similar car, sells that car and thus continues. The net result is that Henry Ford produces cars more cheaply and sells more cars more cheaply so buyers benefit.

(Indeed the idea and history of mass production and economies of scale are intertwined. Today I'm not discussing mass production, I'm talking *Economies of Scale*.)

You expect that if you go to your local supermarket to buy milk then buying one, large - carton of milk - say 4 pints in one go, will be cheaper than buying 4 cartons of milk each holding one pint of milk.

In my #NoProjects talk I use this slide, it always gets a laugh:



Yesterday I put this theory to a test in my local Sainsbury's, here is the proof:

Contributors

- Allan Kelly
- allankelly2015

My books

- Continuous Digital: the alternative to projects
- Xanpan
- Little Book of User Stories
- Business Patterns
- Changing Software Development
- Agile Reader

Newletter sign-up

Sign-up for my newsletter

I am at...

Upcoming events with Allan Kelly

Agile Cambridge
Software Acumen: 27-29 September at Cambridge

Agile Bristol
Agile Bristol: 12 September Evening at TBA

Agile Dice Game
TechCityCoffee: 25 July 2017 2pm-4.30pm at Near Old Street Roundabout

widget @ surfing-waves.com

Speaking events

- Upcoming events
- Events RSS feed

my LinkedIn profile

More

- OnTwitter: @allankelly.net
- My Homepages: allankelly.net

Twitter feed

- 2 pints of milk cost 85p, or 42.5p per pint (marginal cost of one more pint 36p)
- 4 pints of milk cost £1, or 25p per pint (marginal cost of one more pint 7.5p)

(And if you don't know, the UK is a proudly bi-measurement country. Countries like Canada, The Netherlands and Switzerland teach their people to speak two languages. In the UK we teach our people to use two systems of measurement!)

So ingrained is this idea that when it supermarkets don't charge less for buying more, complaints are made (see The Guardian from a few months back.)

Buying milk from Sainsbury's isn't just about the milk: Sainsbury's needs the store there, the store needs staffing, it needs products to sell, and they need to get me into the store. That costs the same for one pint as for four. Thats why the marginal costs fall.

Economies of scale are often cited as the reason for corporate mergers: to extract concessions from suppliers, to manufacture more items for lower overall costs. Purchasing departments expect economies of scale.

But…. and this is a big **BUT**…. get ready….

Software development does not have economies of scale.

In all sorts of ways software development has diseconomies of scale.

If software was sold by the pint then a four pint carton of software would not just cost four times the price of a one pint carton it would cost far far more.

The diseconomies are all around us:

- Small teams frequently outperform large team, five people working as a tight team will be far more productive per person than a team of 50, or even 15. (The Quattro Pro development team in the early 1990s is probably the best documented example of this.)
- The more lines of code a piece of software has the more difficult it is to add an enhancement or fix a bug. Putting a fix into a system with 1 million lines can easily be more than 10 times harder than fixing a system with 100,000 lines.
- Projects which set out to be BIG have far higher costs and lower productivity (per unit of deliverable) than small systems. (Capers Jones' 2008 book contains some tables of productivity per function point which illustrate this. It is worth noting that the biggest systems are usually military and they have an atrocious productivity rate - an F35 or A400 anyone?)
- Waiting longer - and probably writing more code - before you ask for feedback or user validation causes more problems than asking for it sooner when the product is smaller.

The examples could go on.

But the other thing is: working in the large increases risk.

Suppose 100ml of milk is off. If the 100ml is in one small carton then you have lost 1 pint of milk. If the 100ml is in a 4 pint carton you have lost 4 pints.

Suppose your developers write one bug a year which will slip through test and crash the users' machine. Suppose you know this, so in an effort to catch the bug you do more testing. In order to keep costs low on testing you need to test more software, so you do a bigger release with more changes - economies of scale thinking. That actually makes the testing harder but… Suppose you do one release a year. That release blue screens the machine. The user now sees every release you do crashes his machine. 100% of your releases screw up.

If instead you release weekly, one release a year still crashes the machine but the user sees 51 releases a year which don't. Less than 2% of your releases screw up.

Yes I'm talking about batch size. Software development works best in small batch sizes. (Don Reinertsen has some figures on batch size The Principles of Product Development Flow which also support the diseconomies of scale argument.)

Ok, there are a few places where software development does exhibit economies of scale but on most occasions diseconomies of scale are the norm.

This happens because each time you add to software software work the marginal cost per unit increases:

- Add a fourth team member to a team of three and the communication paths increase from 3 to 6.
- Add one feature to a release and you have one feature to test, add two features and you have 3 tests to run: two features to test plus the interaction between the two.

In part this is because human minds can only hold so much complexity. As the complexity increases (more changes, more code) our cognitive load increases, we slow down, we make mistakes, we take longer.

(Economies of scope and specialisation are also closely related to economies of scale and again on the whole, software development has diseconomies of scope (be more specific) and diseconomies of specialisation (generalists are usually preferable to specialists).)

However be careful: once the software is developed then economies of scale are rampant. The world switches. Software which has been built probably exhibits more economies of scale than any other product known to man. (In economic terms the marginal cost of producing the first instance are extremely high but the marginal costs of producing an identical copy (production) is so close to zero as to be zero, Ctrl-C Ctrl-V.)

economies of scale since school. You need to start thinking diseconomies of scale.

Second, whenever faced with a problem where you feel the urge to go bigger run in the opposite direction, go smaller.

Third, take each and every opportunity to go small.

Four, get good at working in the small, optimise your processes, tools, approaches to do lots of small things rather than a few big things.

Fifth, and this is the killer: know that most people don't get this at all. In fact it's worse…

In any existing organization, particularly a large corporation, the majority of people who make decisions are out and out economies of scale people. They expect that going big is cheaper than going small and they force this view on others - especially software technology people. (Hence Large companies trying to be Agile remind me of middle aged men buying sports cars.)

Many of these people got to where they are today because of economies of scale, many of these companies exist because of economies of scale; if they are good at economies of scale they are good at doing what they do.

But in the world of software development this mindset is a recipe for failure and under performance. The conflict between economies of scale thinking and diseconomies of scale working will create tension and conflict.

Have I convinced you?

Get small.

Finally, I increasingly wonder where else diseconomies of scale rule? They can't be unique to software development. In my more fanciful moments I wonder if diseconomies of scale are the norm in all knowledge work.

Even if they aren't, as more and more work comes to resemble software development - because of the central role of individual knowledge and the use of software tools - then I would expect to see more and more example of diseconomies of scale.

(PS, if you didn't see my last post I've started a newsletter list, please subscribe.)

Posted by Allan Kelly at 9:31 am

► August (4)
► July (2)
► June (5)
► May (1)
► April (4)
► March (5)
► February (4)
► January (3)
► 2014 (41)
► 2013 (43)
► 2012 (45)
► 2011 (53)
► 2010 (70)
► 2009 (90)
► 2008 (85)
► 2007 (79)
► 2006 (77)
► 2005 (62)

JavaCodeGeeks

Syndicated to Java Code Geeks

Add a comment as Allan Kelly

Top comments

julien tayon 1 year ago - Shared publicly

bigger efforts need coordination.
Coordination introduces politics in software. Politic has unstable point of views based on power struggle. Computer deals poorly with non determined values.

Computer does or not. It does not "maybe/might" yet. The uncertainty in the specs

+5 · Reply

Marcos Brigante 1 year ago (edited)
Your words are so true, nicely said.

Piotr Kalinowski via Google+ 1 year ago - Shared publicly

I haven't thought about it that way.

+3 · Reply

Mario Lucero via Google+ 1 year ago - Shared publicly

Great article!!!

+1 · Reply

Luke Micono 9 months ago - EMP5031-Fall2016 (Discussion)

I believe there are a few of us in both EMEN 5010 and EMEN 5031 in this class, so I thought I'd share the below, as it addresses lean development as well as economies of scale (which was recently addressed in 5010).

I'm curious to hear what others in the class think. Do the principles of economies of

Nazly El Shazly 9 months ago

That's an interesting perspective. But I can also see software as the perfect economy of scale. Because essentially, you build one version of software using one process cycle (or maybe a few if you consider different operating systems or different architectures) and then you can instantly deploy it on an infinite number

Jim Lampariello 9 months ago

I think the author is confusing scaling software with making a product more complex. Adding more lines of code or more features to a project isn't scaling it – it's making it more complicated. It would be like if Henry Ford decided to add four wheel drive, an autopilot, and a coffee maker to his cars. In that case, of course

Overture Project 1 year ago - Shared publicly

"In part this is because human minds can only hold so much complexity. As the complexity increases (more changes, more code) our cognitive load increases, we slow down, we make mistakes, we take longer." http://buff.ly/29qmOrv

· Reply

BLUE SCRUM 1 year ago - Shared publicly

Small is beautiful in software development **#bluescrum #team**

· Reply

Dexter Miguel 1 year ago

What's Blue Scrum...? Small does not equal Scrum.

Mike Pearce via Google+ 1 year ago - Shared publicly

This is brilliant. A really useful way of explaining why smaller iterations and fewer deliverables as actually healthy and stand more chance of success. Economies of scale is something most stakeholders will understand.

· Reply