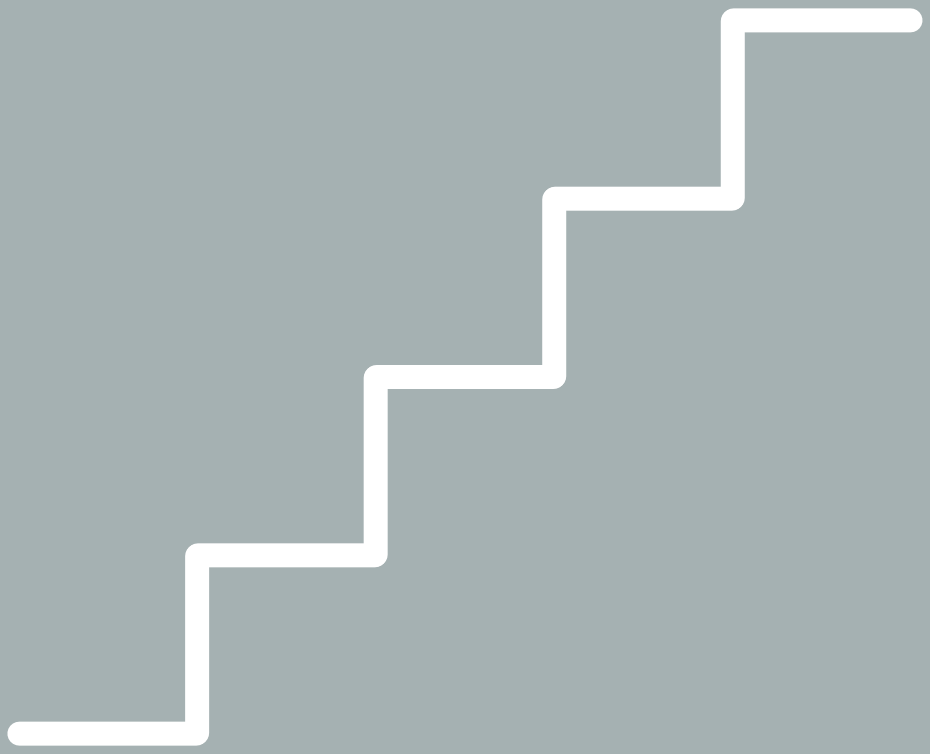
emergn

VFQ



DELIVERING EARLY AND OFTEN

FREE SAMPLE

This publication forms part of Value, Flow, Quality® Education. Details of this and other Emergn courses can be obtained from Emergn, 20 Harcourt Street, Dublin, D02 H364, Ireland or Emergn, One International Place, Suite 1400, Boston, MA 02110, USA.

Alternatively, you may visit the Emergn website at http://www.emergn.com/education where you can learn more about the range of courses on offer.

To purchase Emergn's Value, Flow, Quality® courseware visit http://www.valueflowquality.com, or contact us for a brochure - tel. +44 (0)808 189 2043; email valueflowquality@emergn.com

Emergn Ltd.
20 Harcourt Street
Dublin, D02 H364
Ireland

Emergn Inc.
One International Place, Suite 1400
Boston, MA 02110
USA

First published 2011 - printed 10 October 2018 (version 1.8) - SAMPLE

Printed and bound in the United Kingdom by Apple Capital Print.

Contents

INTRODUCTION

In the Prioritisation session we talked about the importance to an organisation of delivering value early: not because it's a nice thing to do, but because it enables you to make more money sooner. In particular, we showed how the financials of a business case can radically change when you begin delivering a piece of value early. In this session we will focus on the method of delivering that value, primarily achieved through slicing an idea into increments.

This can be hard for people to get their heads around. Oh everyone understands why it would be useful – who wouldn't want to start selling something early and improve cash flow by gaining revenue sooner? But we are very used to the idea of products being delivered whole. It takes an imaginative leap to see how these might be broken down and delivered differently.

Many managers have signed on for the idea of incremental delivery without really understanding what they might have to do. After being asked to think about which element they could deliver early, they quickly start explaining why incremental delivery won't work for their particular project. This project is just so complicated and filled with dependencies that it can't possibly be split into increments. They look around for examples to prove the point. How about a car? they say triumphantly. You can't deliver that in increments, can you? Wheels without an engine are useless; a car chassis without a body is pointless!

Figure 1. Cars in production, not ready for delivery!

This is how we're used to thinking about products. They seem indivisible to us. First of all, of course, software is not a car. Bits on a hard-drive can be configured more easily than car parts. Yet, even a car can be imagined differently if we try hard enough. Design enhancements to an original model can be delivered over time – brakes that last longer, a higher performance engine, an optional sunroof – together these changes can add up to a bigger change than launching a whole new model. Or we could look at it another way – the car parts do have independent value and could be sold to other manufacturers or as 'kit cars'. Or we could launch our car to the UK market only, and not worry about fitting left hand drive for another year. In short, there are many ways of considering incremental delivery – even with something as physical and difficult to divide as a car.

The exercise is far easier with software. Most elements of software can be delivered separately and still have value. A complex software application has value before it is complete: for example, a piece of search functionality might begin with a simple search for a name, but it might not yet allow us to do anything with the resulting list, or even order it. Yet the list of people called 'John Smith' in our total database

might well be of use to us. A second characteristic that makes software perfect for both incremental and iterative development, is that it is inherently malleable. Unlike the metal and plastic of car parts, which once moulded into a shape are very hard to reconstruct into a different design or size, software can expand and contract, be reconfigured or changed or updated.

By the end of this session you will:

1. Understand how increments and iterations are defined.
2. Appreciate the benefits of delivering early and often:
    - financial, marketing, engineering and other.
3. Identify examples of where incremental delivery is used as a successful business model.
4. Begin to break projects into increments through:
    - developing a mindset to split dependencies and ideas
    - using different prisms to split ideas: value, risk, stakeholder, urgency, geography, necessity.
5. Know how small to go with your increment.
6. Be able to map a customer journey or story strand to create slices or bites of functionality.
7. Put in place practices that enable delivering early and often.
8. Balance potential drawbacks or limits to delivering early and often including transaction cost and technological break-through.
9. Demonstrate the value of incremental delivery.

1 SO WHAT IS DELIVERING EARLY AND OFTEN ANYWAY?

Delivering early and often describes the development style known as Incremental and Iterative Delivery. The Agile Alliance's *Guide to Agile Practices* defines an incremental development strategy as 'each successive version of the product is usable, and each builds upon the previous version by adding user-visible functionality.' The 'version' is the increment: usable software that has an innate value. The Agile Manifesto states, 'Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.' And all Agile methodologies stress the importance of an incremental approach to software delivery – that is, to delivering value and delivering it early.

Incremental delivery is such an important element of Agile methodologies that there are several different terms which you may hear – all of which essentially mean 'increment'. In Scrum, the increment is called 'potentially shippable product'. Other methodologies use alternative terms. You may also hear people talking about 'Minimum Marketable Feature' (MMF) or 'Minimum Viable Product' (MVP). The terms draw attention both to size (small) and to its independent value (marketable or shippable – that is, something a customer can use).

It's importance to Agile doesn't mean that more traditional waterfall methods can't also do incremental delivery of value. Many have a phased launch delivery plan, which is essentially a form of incremental delivery. Because the phases tend to be made up of large batches, the value is not being delivered as early as it could be – the increments could be smaller. We will go on to discuss the problems this can cause in a further session – Batch Size Matters.

Delivering an increment requires you to look at the structure of a project in a different way and this can often require an imaginative leap. Instead of the traditional horizontal process (based on supposed efficiencies of large batches), incremental delivery asks us to slice our product vertically and deliver sections of value.

Activity 1 – Increments in your organisation

This activity will take 10 minutes. You can do this activity on your own or as part of a conversation with colleagues.

Consider a current project, was it delivered in increments? If so, consider of what the first increment consisted (think about the features that were released to customers, for example).

Reflect on how the project could have been split differently and what the effects would have been. Could you have launched faster? What might you have learnt sooner?

Case Study – World Bank

The World Bank was considering a project to improve the productivity of 120,000 small-scale farmers in Nicaragua by 30% in 16 years. A team of World Bank experts and Nicaraguan Ministry of Agriculture officials began conducting surveys and analysing data. They were creating a long-term plan that would help huge numbers of farmers meet the goal. There were various long-term initiatives the planners believed would stimulate productivity. Implementation could then begin. This is what we would describe as horizontal slicing of the project – analyse, plan, implement, observe results.

It was taking a very long time. So long, that the World Bank also sliced the project vertically. They created little mini-projects to deliver small increments of value as an immediate pay-off and also as test-case pilots from which they could learn for the longer-term plan. After all, they realised, success in raising production for just one farmer in year 1 of the project is as valuable as raising productivity for 16 farmers by year 16. There would also be a positive effect on the project's reputation and morale, increasing the likelihood that other farmers would sign up to the plan.

One such increment was to increase Grade A milk production in the Leon municipality from 600 to 1,600 gallons per day in 120 days with 60 small and medium-size producers. The team with this goal discovered early on that the issue was not production, but quality. The farmers could produce the quantity of milk, but distributors had to throw away nearly half due to contamination, spoilage and other problems. The team brought in a representative of Nicaragua's largest private company in the dairy sector – a potential customer, in other words, who could state what quality standards were necessary and thus what hygiene standards and testing equipment needed to be introduced for the farmers. Improvements were made rapidly, but a new 'quality' issue raised its head. Small farmers didn't have the storage facilities to keep the extra 'Grade A' milk now being produced, while investing in refrigeration facilities would be an excessive capital investment. The customer representative on the team took the opportunity to change from being an advisory potential customer to a real customer. Now his company could be assured of a quality supply, he was prepared to have the milk collected daily rather than twice weekly.

The overall goal may well benefit from more farmers adopting the same practices and the project can clearly learn from these results, but the increment's value does not depend on these further benefits. The increase of 1,000 gallons of milk a day has independent value in and of itself.

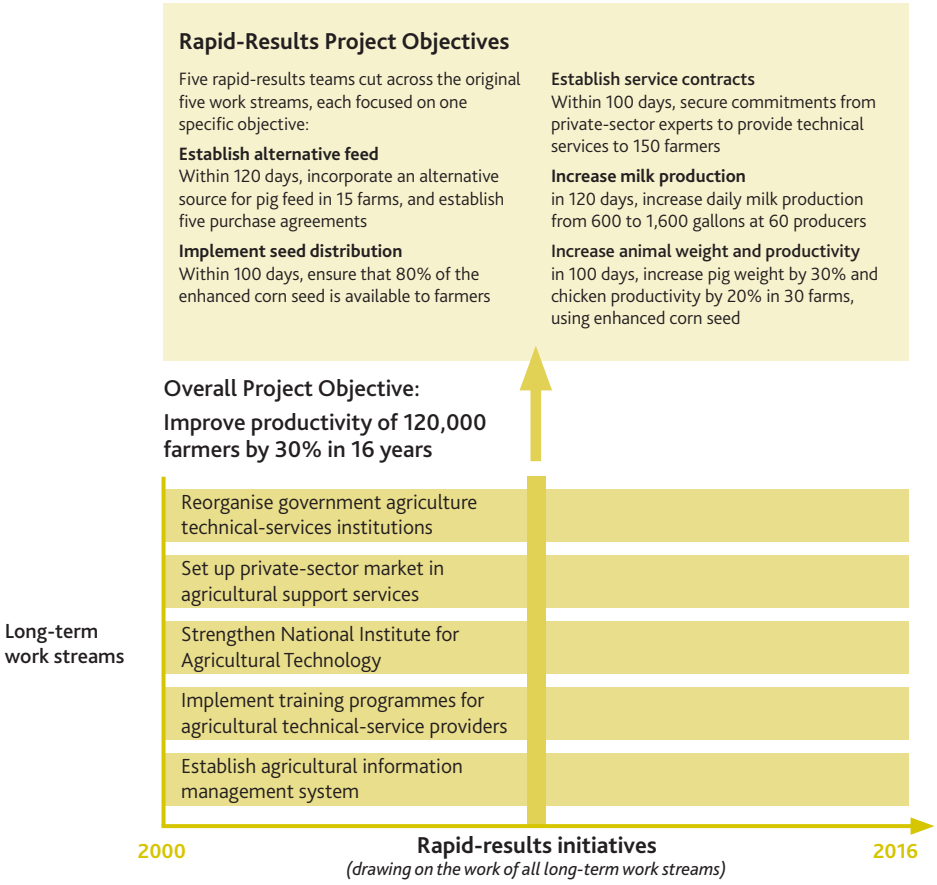


Figure 2. The World Bank's project plan

1.1. The difference between increment and iterate

You say increment, I say iterate (let's call the whole thing off). These two terms are frequently bandied about as if they were identical, but in fact the two concepts are quite different. In essence, an iteration is a process designed to get you towards the unit of value – the increment.

Craig Larman in *Agile & Iterative Development: A Manager's Guide* writes, 'Incremental delivery is often confused with iterative development. A six-month delivery cycle could be composed of 10 short iterations. The results of each iteration are not delivered to the marketplace, but the results of an incremental delivery are.'

Each iteration is a complete mini run through the development process, incorporating analysis, design, programming and test. An iteration aims to produce an increment of software, but it may not necessarily do so. In fact, you might take a step back and say that an iteration really exists in order to elicit feedback. At the end of the iteration, we should have a piece of working software. Now we need to test/demonstrate this software with a customer. It is quite likely that we will need to modify the software.

'Oh, I see!' the customer may exclaim. 'I love how it looks but that's absolutely useless to me.' The high-quality piece of working software is not, in other words, a shippable increment. A new iteration is required to work further on the software, incorporating the customer feedback. (And on a quick pedantic note, iteration comes from the Latin, meaning to do again, to repeat. If you hear anyone saying that they need to 'reiterate' or even worse 'to reiterate over again' they are guilty of tautology and should probably be smeared in honey and buried in an ant nest.)

Jeff Patton uses the Mona Lisa to illustrate his point:

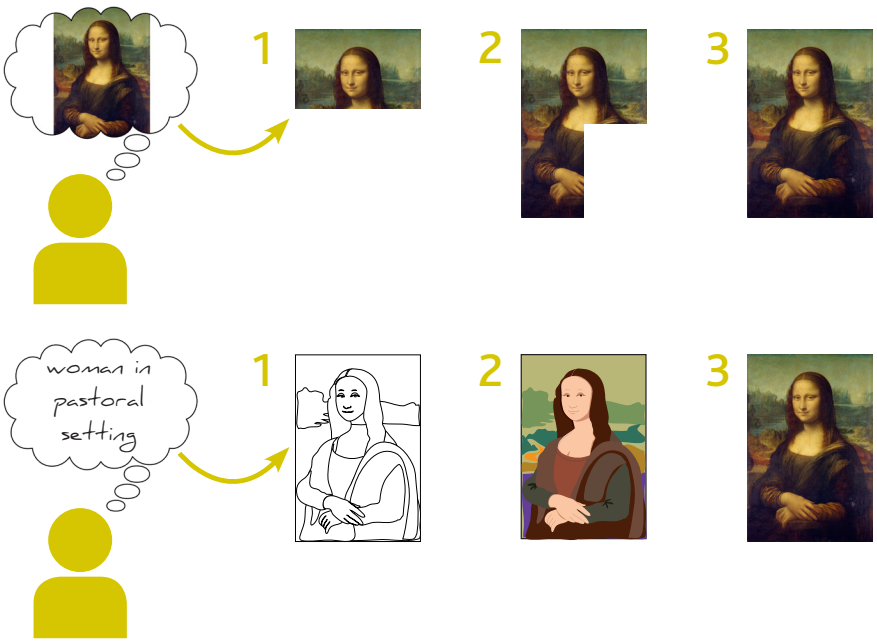


Figure 3. An incremental versus an iterative and incremental delivery approach

Artists work their way towards the final picture by a series of iterations including sketches, false starts and rough blocking. They don't start from a corner and work inch-by-inch without mistakes until they've covered the whole canvas. Software, like art or writing, is a creative process, and is especially suited to iterative development – a journey towards the finished product. Software development is also rather more costly than starving in a garret while you work on your masterpiece, and for this reason, software tries to deliver parts of value early. This is rather like the artist who managed to sell the sketches before the finished picture was ready, or the writer who sold the story in serialised chapters as he wrote it (as Dickens did for most of his novels).

Since iteration is all about gaining feedback in order to improve the solution, it follows that we want to iterate quickly. The focus is not on creating a complete and perfect design, but rather on an end-to-end piece of working software which we can then validate. On greenfield projects, this could be a walking skeleton – something which you have no intention of ever releasing externally, but which can provide significant internal value by validating the architecture. On brownfield projects, the iteration might be more about improving design than changing behaviour. It can be as simple as a clean-up or performance tuning – making the architecture more robust or scaling the ability of the software to cope with more users, for example.

Alaistair Cockburn sums up by pointing out that since incremental development works on a cycle, it helps you improve your process. But the choice of process depends on you. 'It neither implies, requires nor precludes iterative development or waterfall development – both of those are rework strategies. The alternative to incremental development is to develop the entire system with a 'big bang' integration.'

Case Study *– The appeal and risk of the big bang*

boo.com

Boo.com is famous. Or perhaps infamous is a better term. During its glory days, this sports/fashion online retailer was named as one of 'Twelve Start Up Superstars' and was lauded in The Wall Street Journal, the Financial Times, Fortune magazine and Vogue, amongst many others. Now the company tends to be named on other lists, 'Boo and the hundred other dumbest moments in e-Business History', 'The top 10 dotcom flops' and even the founder's own book ***Boohoo 'from concept to catastrophe'***.

For anyone who was asleep or at school during the dotcom boom and crash, boo.com was seen as embodying the excessive consumption and management errors of an era. They burned through $135 million of investment in 18 months, (to be fair, Ernst Malmsten, the CEO, denied charges that they wasted money on champagne, by pointing out he only drank vodka). There were usability troubles with the website and glitches with the technology platform – the 3D imaging

and fancy graphics were unbearably slow at a time when most people had dial-up internet connections. Flaws in the business model, like the number of returns, undercut expectations. Boo.com had a vast staffing cost, employing 400 people in multiple countries around the world. Because it was launching in numerous countries with their own tax laws, languages, and fulfilment methods, the system set-up complexities were enormous. Dealing with these took time (delaying launch), and required vast technical support. A blog from former employee Tristan Louis, drew several lessons, including: 'Plan early, think of all that can go wrong, and then plan it again. Usually, spending more time on specs saves you from many headaches down the road.'

It's a tempting conclusion – perhaps the one that most employees would have reached. But actually, such a lesson is symptomatic of what was wrong at the heart of the business. The planning may not have been perfect, there were management errors and technical messes, but the business's original sin was to turn its back on incremental delivery. As Ernst Malmsten admitted, 'instead of focusing single-mindedly on just getting the website up and running, I had tried to implement an immensely complex and ambitious vision in its entirety. Our online magazine, the rollout of overseas offices, the development of new product lines to sell on our site – these were all things that could have waited until the site was in operation.'

These things could certainly have waited, but the website that Malmsten envisaged was still fronting an overambitious model. Incremental delivery would have highlighted problems earlier, from the user experience to returns. By the time boo.com collapsed, it was turning over $500,000 in orders every 2 weeks with good conversion rates and a healthy repeat purchase rate. These were great figures – but too late. A revenue stream that had come in earlier, even a smaller stream, would have offset the huge quantities of investment being poured in.

So why hadn't Malmsten considered incremental delivery? Did he just not think of it? Possibly. He certainly made plenty of mistakes. But we should also point out that the decision not to use incremental delivery was tempting. There were several reasons why Malmsten and his advisors were seduced by the big bang.

Suppliers – the people that boo.com needed to win over in order to have something to sell – were attracted by the idea of a global company. They weren't interested in dealing with an internet company who simply seemed like an alternative distribution channel for a single country. The investors also wanted to deal with a global company – time and again the feedback Malmsten received was that the global reach was attractive and exciting to large investors. The team were worried about competition. They wanted to raise the barriers to entry so that no-one else could copy their ideas. Instead, they should have been worrying about how easily they could be asset-stripped when they went bust. These factors led the team to reject geographical incremental delivery, launching country by country, and instead to launch in many at once. This decision led to enormously increased complexity in the back-end of the business in order to sort out taxes, delivery and pricing.

Their conviction that the online magazine, fancy 3D graphics and virtual sales assistant Miss Boo, were the reason customers would shop at boo.com made it hard for them to split their product offering. When the first round of cuts axed these elements, the co-founder Kajsa Leander threatened to resign, convinced the company was cutting priorities, not 'extras'. With hindsight it may be easy to scoff at her conviction, but the fact remains that not all people see priorities (and thus what enters the first increment) the same way.

Boo.com envisaged a near-instant global dominance that would discourage competition, create a ready-made image of cool and could be extended swiftly to other market sectors. In giving way to this 'big win' temptation and rejecting incremental delivery, they lost everything. It's a reminder that business as a whole, as well as IT projects or software products, need to stay focused on delivering small, valuable increments early.

1.2. Benefits of delivering early and often

We've mentioned many of the benefits above, but here they are, presented in a nice tidy table:

Financial	Earlier profits Earlier cashflow Less investment
Marketing	Shorter planning horizon Earlier feedback from customers More reliable feedback Flexibility Image of consistent product improvement Customer lock-in
Engineering	Lower technical complexity Early field experience with technology Spreads technological commitments
Other	More motivating to teams Accelerated learning (Feedback)

Financial

This is the biggie. The most obvious benefit, and most intuitive, is that the sooner you start using the software (either intended to generate revenue directly or indirectly or make savings), then the sooner you will see a return on your investment. Quarter by quarter, this extra time in market can radically improve a business case, bringing in cashflow sooner (at a time when the cash is worth slightly more) and significantly reducing investment risk. Your revenue will be higher even over the same product lifecycle because it starts earlier. Your break-even point will come sooner and thus your ROI and other measures will look more attractive. Delivering any increment will help, but by delivering your most valuable increments first, you can really transform the overall value to the organisation (see the Prioritisation session for further discussion on this point).

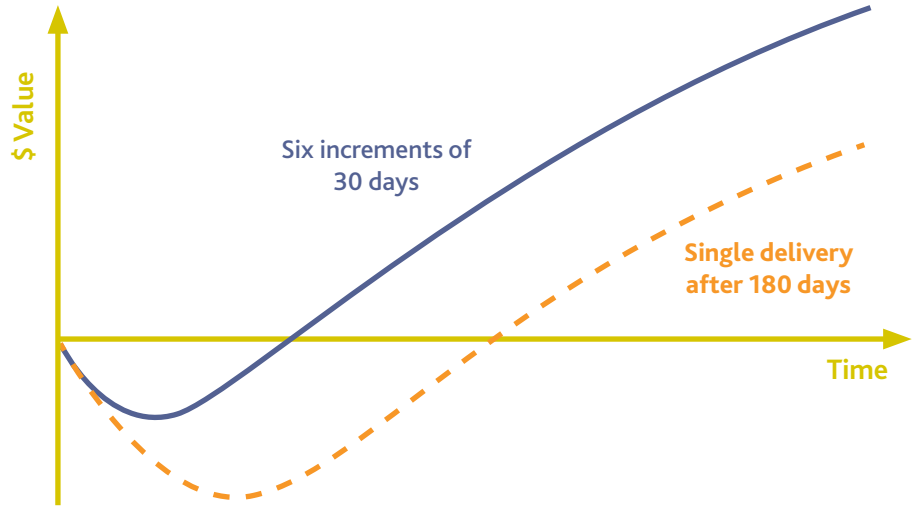


Figure 4. Earlier delivery of value through smaller increments

Perhaps you want to encourage your customers to receive their statements online, rather than printing and posting them out. You know that this will eventually save you several hundred thousand pounds per year. Migrating your total, highly complicated customer base online is going to be difficult. So perhaps you start with your most internet-engaged customers – just the 400 who signed up online to a direct debit payment scheme. This will still save you a bit of money.

You implement the increment and 100 customers ring up to complain that they hate the new system. You have learned something that has saved you a very costly mistake – if one in 4 of your total customers had got annoyed with you, the effects on your business might have been catastrophic. Now you need to rethink your strategy and customer motivation. How much should you invest in persuading your customers to change – perhaps you could offer them a 5% decrease in their bills. Does the business case still work in your favour? Once you fine tune your offer and get a success with your best customers, the business can observe the savings and decide whether it's worth rolling the project out to the next group. This is known as incremental funding – more money is allocated to a project as its success is proved.

The opposite would be to deliver all the value at the end of the project. Set up a completely online system and insist that customers either migrate to it or lose their account. This would be such a risky idea that few companies would do it. Instead, banks and utility companies try to entice customers to move to paper-free accounts with prize-draws, discounted fees or reminding customers of the environmental benefits – not by trumpeting how much it will save the company. Yet in software, we often see precisely this 'big-bang' implementation approach, with all the associated risk, potential cost and reduced value flow. Managers may be so focused on the economy of production cost that they cannot see beyond it, or they may be convinced that the dependencies within the project make it too difficult to break it down into increments - more on this later!

Increments also mean you know when to stop – the point at which you have gained all the value there is and everything on top just weighs the project down. As we will say many times in this course, many software features are unused. Sometimes known as 'gold-plating', sometimes referred to as YAGNI (you ain't gonna need it), these excess features represent waste. Put simply, they cost you money and time to add to a project and maintain. In return they don't offer sufficient value to justify their inclusion. But how do you know in advance what will turn out to be a killer feature and what will be a frill? Increments give you feedback early so you can kill the loser and invest in the winner.

Risk is an essential characteristic of any innovation, but it does not boil down to a project's ultimate success or failure. Other indirect financial benefits of incremental delivery include lower costs and lower risk associated with shorter lead times and lower inventory (such costs are lower in spite of a higher transaction cost). This is sometimes hard for people to accept because it seems so counter-intuitive. Consider the number of companies that centralise purchasing, for example, in order to drive down supplier costs and decide to order everything in bulk at the beginning of a project. This can end up causing higher costs later, when the 50 licenses for development software purchased at the start turn out to be a waste of money, because the developers now realise they need a different piece of software.

Marketing

The examples above remind you how financial benefits are tied to others. Early sales tend to equate to market share, which has a series of further benefits: establishing you as market leader; forcing competitors to play catch-up, and through frequent releases, giving a perception of your product as being continuously improved. By allowing you to fine-tune your offering to provide customers with the best value early, early delivery opens the possibility of locking customers into a relationship or charging more for your products.

Customer satisfaction may not show in immediate financial benefit to the company, but it is essential to future success. In the example of moving to paperless statements, incremental delivery permitted us to adapt to customer feedback – this controls the risk of annoying customers and allows you to please them sooner.

Engineering

Incremental delivery often enforces a degree of technical and architectural simplicity because it demands breaking systems into component parts. Even the most complex of databases must be built one section at a time and systems need to be decoupled to enable this. This can sometimes feel difficult to those dealing with legacy systems, but monolithic technical systems have their own problems – problems often ameliorated by a simpler 'lego-block' style of build. However daunting the task of decoupling the system or splitting it along the seams may feel, the benefits are likely to far outweigh the time required for doing it.

Incremental delivery allows teams to try out new technologies in a risk-limited manner. That might mean switching to a new programming language on a specific component or a new version of a database server on just one particular aspect of the system. While an additional piece of work might be required to link the new system to the old, by incremental delivery you can test the proposed functionality in a low-risk and low-cost manner while maintaining service on the old system.

Other

Knowing that you are adding real and measurable value is one of the most motivating experiences for a team. Seeing a customer using the feature, watching sales figures rise or complaints fall – such measures are a tangible expression of success – and success motivates more than a stirring speech from the boss or even a pay-rise. Knowing that the feature you are working on will go live next month (as opposed to in three years time when you may have left the company), is not only more exciting, but it is likely to force you to pay close attention to the quality of what you are writing. Together, these forces can significantly boost team performance.

Incremental delivery is driven through a feedback loop, normally because we use iterations to get us there. Feedback means learning, and since each piece of learning can be applied to the next iteration, the smaller the feedback loop the faster the learning process and the greater the chance of the painfully acquired knowledge being of benefit to the project and the wider organisation.

emergn

VFQ