PRIORITISATION

VALUE – DELIVERING VALUE

Hi there

Contents

Hi there

INTRODUCTION

If you have lots of children demanding slices of birthday cake, you could do several things. You could serve those who arrive first or yell loudest, and then panic when you run out. Since that will mean lots of tears, another idea might be to count the children first and then try dividing the cake up into the right number of very small slices. Funnily enough, IT departments often try to allocate their time and money in rather similar ways – giving the most to those who shout the loudest or trying to divide it fairly between everyone.

Much of life contains difficult decisions that can basically be summarised as 'who is going to get a big slice, who will get a small slice, and who will get none at all?' This is prioritisation in essence.

When there is more demand than there is supply, people have to prioritise. We handle the concept all the time in our daily lives, from the trivial, 'which email shall I open first?', to the serious, 'shall I save this money for future college fees or shall we go on a family holiday?'.

Prioritisation can be difficult, involving painful decisions and trade-offs. During the First World War, medical staff developed a rough and ready form of prioritisation called triage. In battlefield 'clearing stations', there were vast numbers of wounded men, and only a few doctors attempting to process thousands of casualties. Triage was the brutal but effective method of maximising the value of that resource. The original groups were: those likely to die no matter what treatment they were given; those likely to live no matter what treatment they were given; and those for whom immediate treatment might make a difference. Only the last group would be treated.

For IT, prioritisation is not only about ordering projects by value or importance, nor is it about the fair allocation of resource... Just as with the battlefield application of triage, prioritisation includes choosing whether you will invest any resource at all.

According to Jeanne Ross and Peter Weill's research for their book *IT Governance*, 'The companies that manage their IT investments most successfully generate returns that are as much as 40% higher than those of their competitors.' In a separate article, Jeanne Ross also quoted a leader of a $15 billion retail enterprise as saying, 'IT investments are like any other investment. You must make a decent return or you go bust. It just happens faster with IT!'

By the end of this session you will appreciate:

1. Where there is a mismatch between supply and demand, prioritisation is a necessity:

   - strategies devoted to managing demand

   - strategies devoted to managing supply

2. Problems with focusing on 'right decision' not 'right outcome'.

3. How to unpick 'business value' into value, cost, risk and new knowledge.

4. Differences between ordering, selection and prioritisation.

5. When to use numbers or approximate 'labels'.

6. Common prioritisation techniques, their advantages and disadvantages, when to use them and the type of flow process they suit best:

   - MoSCoW

   - Classes of service

   - Equity

   - HiPPO decisions (HiPPOQ)

   - Value divided by effort

   - Cost benefit analysis

   - Incremental Funding Model

   - Weighted look ahead approach

   - Weighted shortest job first (CD3)

7. The non-rational decisions that affect prioritisation.

Hi there

1   DEMAND AND SUPPLY

1.1.   Prioritisation (demand)

In an ideal world, supply meets demand exactly. In the same ideal world the sun shines whenever you wish to go out and it rains whenever you happen to wish to stay in. This ideal world, does not exist.

As our lives rely more and more on IT, the demand on IT increases. It is not simply a case of deciding which new products to deliver, our organisations also rely upon IT for how we communicate, store information, manage the most basic of our processes. As this reliance is increasing, so do the possibilities of what IT could do further for the organisation as a whole. In short, IT is busier than ever and more essential than ever.

As we've covered elsewhere, this importance and busy-ness does not translate to endless budgets and unlimited resource. Instead it means that IT and the wider organisation need to be very intelligent in how they allocate the resources they do have. Do we do this big project, or these three little ones? Do we ensure our systems are super stable, or do we invest in a new revenue opportunity? All too often when faced with such questions, the response is: but I want you to do all of them!

It might be easier to think of these choices as investments. Just as a company only has a certain amount of cash to invest in different opportunities, so IT can only invest in so many projects. The question is to decide how to maximise this investment. If there is a great deal of demand being placed on IT, prioritisation will be essential. By prioritisation we mean deciding how to invest our IT resource in order to maximise value for the organisation. We need to do this at both the macro level (what project shall we work on) and the micro (which feature or requirement shall we do first, if at all).

In the introduction we discussed battlefield triage. As a prioritisation technique, it continues to be used, but with more sophisticated decision rules. In the Emergency Room of a busy hospital, for example, triage separates those who require immediate assistance from those who can wait. Someone arriving with a gunshot wound will be seen more quickly than someone who has sprained their ankle. As cases become more equal, the decision rules become more subtle, relying upon expert knowledge about the potential consequences of delay. Someone who has suffered a blow to the head but who isn't complaining may be seen more quickly than someone with a broken leg who is screaming the place down. The broken leg may look spectacular (and feel horrible!), but the possibility of internal brain damage is so serious that the head injury receives priority treatment.

Figure 1.   Heart emergencies only

Most IT projects are not dealing with life or death situations, but the principle of expert knowledge which influences our decision rules remains sound. It matters because (to drop into cliché for a moment) we are normally comparing apples and pears – unlike items with varying levels of value, risk and urgency. In order to know which project to do first we need measures that will allow us to make meaningful comparisons. Only when we can do this will we be able to maximise the return on our IT investment for the organisation.

1.2.   Managing demand

The macro: project approval

Prioritisation happens at both the macro and the micro level. The macro level is often very visible because of the project approval process that surrounds it. As R. Benson and T. Bugnitz drily note in their book *From Business Strategy to IT Action*, 'Considerable management energy is spent prioritising and dealing with the politics of project selection.' Perhaps you have been involved with gaining approval for a new project: working out what is likely to impress/delight the approval board, researching possible revenue and ROI, creating a killer PowerPoint presentation or documentation... Or perhaps you have been trying to get IT resource allocated to your business unit or project and have needed to build a case as to why you deserve the investment...

Much energy goes into such work, but it is often misdirected or wasted. All too often, IT developers are excluded from the process altogether. You probably have your own horror stories of how figures were massaged to make them look more attractive, or about the politicking and horse-trading done to ensure one project attracted funding. You're not alone! A whole host of IT and management writers line up to lament the flaws in the process.

At best, you hear complaints like that of Susan Cramm in her book *8 Things We Hate About IT*, 'Leaders treat IT business cases as a bureaucratic hurdle'. At worst you hear the kind of devastating critique made by Jack Keen in *Making Technology Investments Profitable*: 'Experience indicates that over 50 per cent of funded IT

Hi there

projects still use no formal ROI guidance. Over 80 per cent of the business cases… seen are flawed in overt or subtle ways. Shiny shoes, a tap dance, and a few discreet offline conversations continue to grease the skids of approvals of favoured projects. Political influence lurks at every turn.'

If you have experienced such political manoeuvrings, on how many occasions have you seen people held responsible when actual returns turned out not to meet those in the original proposal? Were questions asked, or did the whole project simply quietly slide into oblivion? In the sections below we'll look at how our decisions ought to be made.

The micro: requirement-/feature-level prioritisation within projects and teams

"A solution's design is the results of hundreds, if not thousands of design decisions related to capabilities, requirements and functionality. If these design choices are not primarily selected based on 'contribution to value', then the solution itself cannot hope to be an ROI triumph."

Keen, J., 2011. *Making Technology Investments Profitable*.

The macro level decisions about a project's priority are based on assumptions. As we start work on the project we often begin to prove or disprove some of these assumptions in a way that might fundamentally change our original idea.

Suppose we are building a children's playground. We have decided to build an area for teenagers, and a separate area for toddlers – we allocate half of the budget to each area. As we make our plans we discover that the extra safety features required for toddler swings and slides are going to make the equipment more expensive. This new information may have an impact on what we do. We could create a smaller toddler's area, or we might want to take money from the teenager's area budget in order to fulfil our original toddler play-scheme.

Agile methods ask the team to improve the accuracy of prioritisation through their in-depth understanding of the activity. This is important because otherwise a top-down priority list can fail to take account of knowledge learned at the front-line. In Agile, of course, this knowledge comes from customer feedback, because the developed software has been shown to users (actual or proxy) at the end of each iteration. By using real feedback, the team is able to stay focused on the features that are most highly valued. It also matters because priority does not rest on a single measure alone. Collecting the car from the garage may be a more valuable task than switching the washing machine on, but switching the washing machine on may be more time sensitive (in order to have clean clothes), and anyway it only takes a few seconds.

We make this kind of decision effortlessly in our own lives, but organisations trying to impose standardised processes can misdirect employees' efforts. As Don Reinertsen comments in *The Principles of Product Development Flow*: 'Many companies make the mistake of ranking their projects in priority order and telling the entire organization to support projects based on this ranking. This can lead to an unimportant task on a high priority project displacing a high priority task on a lower priority project.'

To return to our playground example, the directive might be that teenagers need something to do after school and that therefore their playground is the most important. If we had begun by setting up a single swing and climbing frame, we might be able to discover something important from actual use. Perhaps parents and toddlers are most likely to arrive in the middle of the day, while teenagers are more interested in hanging out and chatting at the end of the day. Thus we might focus our next build on equipment we think both groups of users could enjoy – seating areas, roundabouts and see-saws, saving the specialist equipment build for a later increment.

Figure 2.   Children's playground

Learning allows us to challenge assumptions we make about value, risk and cost. The challenges may cause us to revaluate a decision we made earlier – a technology may become cheaper, making its use more attractive; a market can expand or shrink. These changes are often ignored by organisations that continue to refer to the original proposal as if it were divine revelation.

Hi there

Case Study – *37 Signals and the epicentre*

The first principle that 37 Signals give in their company description is this:

> ## USEFUL IS FOREVER
>
> Bells and whistles wear off, but usefulness never does. We build useful software that does just what you need and nothing you don't.

It's a principle they apply during development as well. Jason Fried wrote a blogpost regarding the development of a new feature for Backpack (now merged with Basecamp).

They call their approach 'epicentre design'. They ask: What's the one thing we need before anything else? What's the core? What's the absolute essential bit of functionality that must exist before anything else can exist? When you've discovered that, you've found the epicentre.

The 'most requested' feature from customers was a calendar function. The team were considering various features, iCal subscriptions, color coding, email reminders/alarms, dragging and dropping events, subscribing to other people's calendars, attaching Backpack pages to calendar items, etc. Exciting and useful as all of these might be, none felt like the epicentre.

The calendar could work without any of these nice-to-have add-ons. What it couldn't manage without was the ability to add events. Without this, a calendar feature simply wouldn't exist. As Fried points out, once the epicentre was delivered, the team had also delivered the largest chunk of value.

1.3. Managing supply

Having a lengthy list of projects or features you want, carefully ordered and prioritised is only useful if you have the capacity to deal with them over time. If you know that you can only do ten things in the next five years, then worrying about ordering anything after that on the list is a waste of time, because in five years time your priorities will almost certainly have changed. Indeed, they will change because of what you learn from doing the first ten things. Prioritising without capacity is as useful as dreaming about how you are going to spend your lottery winnings, before your ticket wins.

Hi there

Activity 1 – The difference between a push and a pull system

This activity will take approximately 30 minutes. We will use a mock production line to show the difference between a push-based system and a pull-based one. You can do this activity on your own, but it's fun to involve a couple of other people and you can use it as a warm-up at a team meeting.

Preparation:

To run this activity you will need:

- A dice.
- Printed and cut-out assets for this activity – assets are available in the Additional Resources section of the valueflowquality.com website.
- Glue stick.

To build a T-shirt:

You should set the process to make T-shirts up as follows:

1. First station colours in the T-shirt body using a black pen.
2. Second station draws a happy face on the yellow circle to make the logo.
3. Third station sticks the logo.

| Starting point | First station | Second station | Third station |

To run the activity:

Round 1

1. Roll the dice and build T-shirts based on number rolled using the method above. Record your manufacturing number.
2. Roll the dice again to see how many T-shirts are bought. Record your sales and left-over stock.
3. Repeat both steps three times, recording the results.

Round 2

1. Take 2 of the T-shirts manufactured in the previous round. These are your new inventory.
2. Roll the die to see how many T-shirts you can sell.
3. If the inventory in the system drops below two T-shirts, make enough T-shirts to top the inventory up to two and fulfil the sales demand.
4. Repeat three times, recording your results.

Hi there

Commentary:

You will have seen that in the second round, even though some inventory remained in the system at the end of the round, it was at a controlled level. You should also see that a pull system matches supply against demand better than a push system. Since almost all systems experience ups and downs in demand, a buffer is necessary to cope with this variation.

Discuss with your group whether your software/product development process is push or pull. Are there changes you would like to make? Would you like to change the whole system or just aspects, such as coping with variation? How might you effect the change you think necessary? What benefits would need to show in order to persuade others to sign off on the change?

Hi there

A pull system balances demand against capacity. Rather than a business owner handing a list of projects to a development department, which is a push system; in a pull system the development department signals, 'we have capacity to work on another project'. This signal then pulls in a project or feature from a pre-ordered list. In IT the pace of demand tends to be relentless, so the risk of producing large quantities of work that no-one wants is not as worrying as it is for t-shirt manufacturers. But if work in progress builds up, heavy congestion results, clogging up the system and thereby slowing how much work actually gets finished.

> *"Lean guidance dictates that limiting work to the team's capacity is fundamental to efficient workflow. This environment is created by forming dedicated teams that pull work from a prioritized backlog and focus on completing it before starting new work."*
> Shalloway, A., Beaver, G., Trott, J., *Lean-Agile Software Development: Achieving Enterprise Agility*

Note that a pull system doesn't tell you how to order the list. The work on prioritising or ordering still has to be done, but it doesn't always have to be done continuously. David Anderson in his book *Kanban* describes a team in India switching to a pull system for work coming from several different units in Microsoft. Instead of a manager constantly checking the entire backlog and reprioritising every item within it, the development team signalled how many items they would work on. If three, then the business sponsors selected the five most important items — which they usually knew by instinct and gut feel as much as by careful analysis — and argued it out until they had decided which three took priority this month. The rest of the list didn't need close scrutiny.

Getting work flowing through the system in a reliable timely manner, is the most important first step. No matter how brilliant the work that has gone into identifying a valuable opportunity, it will be wasted unless you can actually deliver something to take advantage of the opportunity. Once this is being achieved, prioritising effectively increases value.

While a pull system is how Kanban and Lean function, it is not the only method of working. The opposite — a push system — still functions perfectly well, as long as we do not fall into the trap of trying to do everything on our list of valuable projects. Don Reinertsen neatly summarises the way most companies work: 'take the projects that "must" be done and divide the available people into them.' With admirable restraint, he goes on to discuss a better way: 'rank order your projects. Then take your most important project and assign as many people to it as it can effectively engage.'

Hi there

2 BUT EVERYONE PRIORITISES! SO WHAT'S WRONG?

It may sound as if we are stating the irritatingly obvious. However dysfunctional you occasionally feel your organisation may be, managers probably don't wander in with tasks written on pieces of card, throw them in the air and say 'everyone pick something at random to work on!'

Activity 2 – How does your organisation prioritise in its product/software development lifecycle?

This activity will take 30 minutes, and involve a group of your peers; 5 or 6 is ideal. In preparation, collect a list of features from the project backlog and discover what their relative priority is – do they have a figure, a value, or a label?

Part 1 (10 minutes): Lead a discussion that walks through the product/software development lifecycle within your organisation. Start from when an idea is first conceived and identify and list any points in the process at which prioritisation decisions are taken.

Part 2 (10 minutes): Select 2 of the points identified and discuss each of the prioritisation methods used in detail. Identify who's involved in the making the decision, what information it's based upon and, in the opinion of the group, the pros and cons of the decision making process.

Part 3 (10 minutes): Read through the order of the features in your project backlog. Does it conform to the prioritisation methods discussed? What other factors might have affected the actual ordering of the tasks? Save this list for Activity 3.

Commentary:

Despite organisations having a set method or technique, in reality we often find that 'alternative' prioritisation methods are in play. Perhaps personal pressure has been brought to bear to escalate an item, perhaps there are external factors which the prioritisation technique employed does not take into account – for example, regulatory changes or linked dependencies.

Hi there

Probably you have been involved with projects where lots of work has been done to try and improve understanding of the potential value, cost of delay and risk, but the results have been disappointing or just plain wrong. You sometimes feel the wrong project gets given priority, or two projects go ahead that clash with one another, or duplicate work that nobody knew about ... The process as it stands in most companies is wasteful and uncoordinated.

This doesn't mean that people aren't working really hard trying to prioritise what to work on or invest in. After all, the right decision may mean the future health or collapse of the organisation – the stakes don't get much higher than that! No wonder then, like bridge players anxiously counting how many trumps remain in play, we expend enormous amounts of effort on the decision. Benson lists the outcomes that managers believe depend on a right decision:

- 'Creating better investments, alternatives – or, in IT terms, creating better ideas for development projects.

- Choosing the right investments and projects from the alternatives.

- Eliminating non-performing and poorly performing existing IT resources from current spending.

- Improving the performance of the remaining existing IT resources.

- Implementing and following through on the right investments and performance improvements.'

The right decision, however, can come at an unacceptable cost.

Philip II of Spain ruled one of the largest empires the world has ever known: the Iberian peninsula, large parts of Italy and the Netherlands, and South America (far more desirable than North America, which could be left for the envious English and French to squabble over). Philip appointed governors for his far-flung lands, but he insisted on making all the important decisions personally. That meant an avalanche of paperwork as he looked into each matter, consulted various councils and then made his decision. Philip was the ultimate micro-manager. He even planned the Armada's attack on England – in advance and from hundreds of miles away. This meant Philip's orders (sensible as they were) gave no option for his commander to take account of changing conditions – the English use of fireships, an enormous storm... The Armada was defeated.

Figure 3.   Philip II of Spain

Philip knew that his decision was important – so he spent a great deal of time and effort in trying to ensure that it was the right one. Unfortunately in doing so he lost sight of what really mattered – achieving the right outcome.

This is a common problem in many organisations today. So much effort can go into the process of prioritising and making a decision, that the overall value for the organization can be reduced either through delay or through a dissipation of the resource required into several 'priorities'. That is – in many cases – the work has not actually resulted in a decision at all!

A survey in 2011 by Booz and Company of 1,800 executives uncovered the following:

- Most executives (64%) report they have too many conflicting priorities.

- The majority of executives (56%) say that allocating resources in a way that really supports the strategy is a significant challenge, especially as companies chase a wide set of growth initiatives.

- 81% admit that their growth initiatives lead to waste, at least some of the time.

- Nearly half (47%) say their company's way of creating value is not well understood by employees or customers.

The report by Paul Leinwand and Cesari Mainardi concludes: 'these symptoms stem from companies' incoherence — their strong tendency to chase growth initiative after unrelated growth initiative, often with very little success. ... [A]s a ... priority list grows, the company's revenue growth in fact declines relative to its peers.'

Just like Philip trying to fight wars with most of Europe and consolidate his power in newly acquired territories, a lack of focus means resource is over-stretched. Philip II felt that everything was important. He couldn't prioritise – and that meant doing nothing well. Add to this a focus on the decision process not the desired result... let's just say that Philip II of Spain is not remembered as one of the world's great rulers.

Hi there

3   SO HOW CAN WE DO IT BETTER?

The Agile Manifesto unambiguously states:

FIRST PRINCIPLE OF THE AGILE MANIFESTO
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software...

Figure 4.   First principle of Agile Manifesto

If you are directly delivering software to an external customer the answer to what you should do first may be more straight-forward – what does the customer want most/first? To help you determine this you will need a feedback loop that helps tell you if you've got the right answer. In order to make this as inexpensive and quick as possible you need to deliver in small increments. If any of that is worrying you, then check out the sessions on Feedback and Delivering In Increments.

But what if you are delivering software for different internal customers? What if you are delivering software and services to every department and project, all of whom are shouting that they need it now, and why on earth can't you deliver what they want... what then?

In *From Business Strategy to IT Action*, R. Benson and T. Bugnitz state one typical cause of this conflict: 'most companies carry out planning, prioritisation decisions, budgets, performance management, and so forth, in silos or stovepipes.' This is especially problematic when it comes to IT, where IT may be a resource required across different projects each owned within a differing silo. Any lack of co-ordination will stretch IT harder than anyone else, perhaps imperilling not just one project, but all of them. It's hard because each of those business units or silos has different priorities – priorities which may actually be in conflict with one another.

At a macro level, it is the job of the C-suite and senior management to ensure that each department has goals that contribute towards satisfying the customer; that conflicts are raised, debated and resolved. This doesn't always happen, or doesn't happen perfectly. As Mike Cohn ruefully notes in *Agile Estimating and Planning*, 'Product owners are often given the vague and mostly useless advice of "prioritise on business value."'

What is 'business value'? Booz and Company's survey quoted previously recognised that nearly half of those surveyed seemed to sense a disconnect between the way the company saw value, and the way employees and customers saw it. That's pretty serious, because the only value that will translate to revenue depends upon customers.

Hi there

Activity 3 – How do we define business value?

This activity will take 50 minutes. You'll need to involve a group of your peers – 5 or 6 people in total is ideal.

The objective is to agree how to quantify business value. You may already be using a technique for this in your work. This exercise is an opportunity to revisit an important aspect of your work: ensuring that you're delivering the highest value to your customer.

Part 1 (5 minutes):

Using your list from Activity 2, run through the points at which prioritisation decisions are taken and the measures and methods used to take them.

Part 2 (15 minutes):

Examine some alternative measures:

Value: What is the likely revenue/cost-saving that will result from the project? If this is hard to define, think about what the alternative would be. Can you quantify the alternative and say the project will be at least equal to this?

Urgency: How urgent is the project? What will happen if you launch late or early? What will happen if you do not launch certain components?

Cost: How long will the project take? How much internal resource will it consume (development hours)? Are there any other direct costs associated with it?

New Knowledge: Is this testing a new process, market area or technology from which the organisation might derive valuable knowledge or learning?

Part 3 (10 minutes):

Which measures do you think would help improve your prioritisation methods? Can you agree a definition for 'business value' in your organisation?

Part 4 (20 minutes and going forwards):

Run through the ordered list of features/tasks from Activity 2 or macro level projects, and re-prioritise the list using your new measure of business value. Does this give a different order? Can you use the new order going forwards?

Hi there
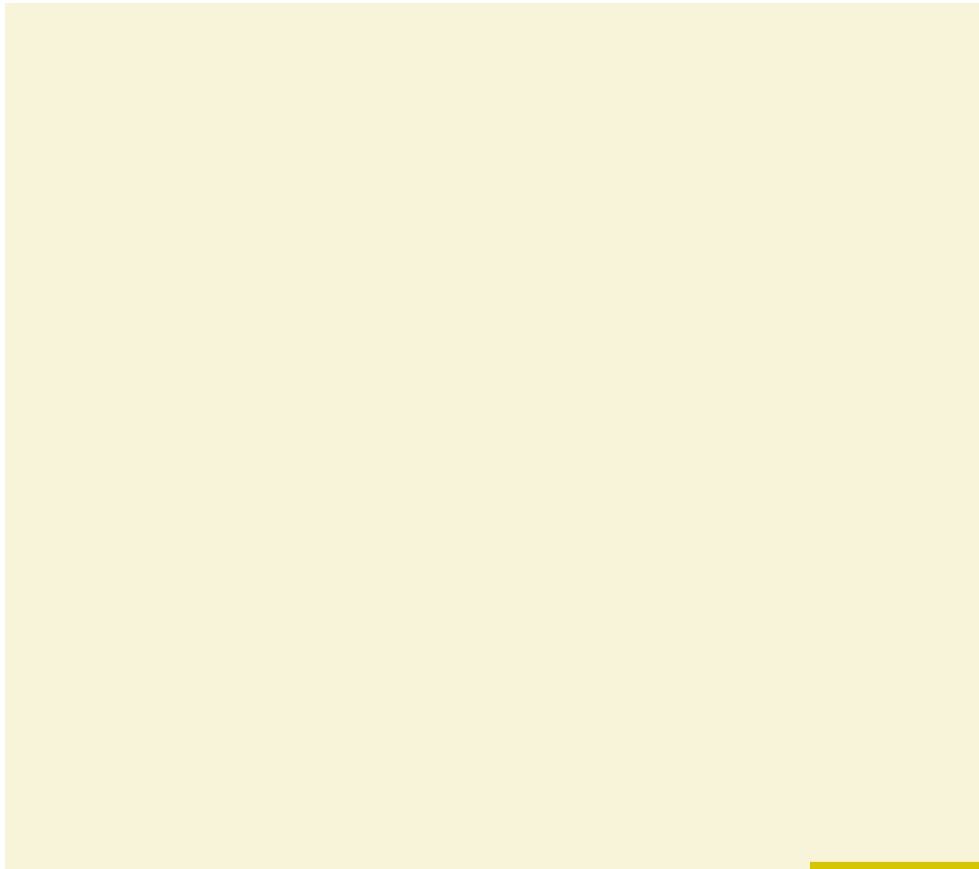
We – the VFQ team – may believe that spending time putting in graphs is terribly important to how impressive a course this is, but if you the customer/user don't care about them, then we have wasted time that could have been better spent. It is not that graphs are bad, simply that they may not be as desirable to you as pictures, or readable font, or the bibliography for further reading. Delivering soon is also important. The team needs to choose where to spend its time.

For those who see IT as a pure 'service function', it may seem surprising that IT managers need to assume the role of helping to map business value onto external customer value. However, the wise investment of a scarce resource demands an ability to proactively take on major business-wide decisions and focus on the external customer. This can actually help avoid conflict and game-playing because it changes the dynamic of the conversation from 'This is what I want, how much is it going to *cost* and *when* will I get it' to 'OK. So I'm competing with others based on how *valuable* my idea is.'

When prioritising a list of features for a product, the most important opinion is that of the customer. Mike Cohn describes an all too familiar story: 'the end of the project approaching, the team scrambles to meet the schedule by dropping features. Because there was no attempt to work on features in order of priority, some of the features dropped are of greater value than those that are delivered.'

Hi there

Activity 4 – Creating value for whom?

Review the output from Activity 3 and the ways that you and your colleagues thought that business value is or could be measured. How many of these measures directly mention your customer?

In that respect, understanding the value that you are creating for your customer is important. Work through the relevant points from Activity 3 and establish whether there is any way you could understand the measure from a customer perspective.

Look for a customer or typical user with whom you have a friendly relationship. Call them up and ask which feature on the list sounds the most important to them. If they have time, ask them to select a rough order for the features, selecting which they would like to be delivered first. Compare this with your own list and note any differences.

Commentary:

In the Understanding Your Customer session we describe the fact that organisations sometimes lose sight of their customer, and that the term can be misused. We're quite clear in our definition: customers are external to your organisation and have a choice over whether to use your product or service or not. If you can create something that your customers perceive to be of value, making them choose your product or service, they will introduce a greater amount of value to the system.

Hi there

Let's be clear: the process of prioritising doesn't tell you which department was right when they told you that their project was more important, nor which customer was right when they claimed that this feature was more desirable than that. But what we can tell you is that if you don't pick one then you won't get either done - and that is worse.

3.1.   Three techniques to help prioritise

If you ever find yourself at an IT dinner and conversation seems to be waning, ask loudly 'so what's the difference between ordering, prioritising and selecting? They're just three different words for the same thing, right?' Then sit back and wait for the decibels to rise.

Why people get into semantic arguments on this, we're not quite sure. Some people like to say that ordering permits you to reorder as new ideas are added, while sequencing does not. Others say the opposite. Nor are we going to argue if you want to swap the words about... as long as you use the tools to help you, that's fine by us.

1. Ordering

Sometimes tasks are sequential: I must go out and buy eggs before I make breakfast pancakes. Sometimes they are unconnected: I must go to the dentist and I have to buy eggs. I could buy eggs on my way to the dentist or on my way back. Does it matter? It might, if buying eggs delays me so that I miss my dentist appointment.

Many tasks in product development are not strictly sequential. One developer can be working on a log-in feature while a second developer is working on a stock check feature. If you only have one developer you might need to decide which feature is more important right away. This insists on a choice – you can't just say 'but they're both equally important'. They may well be, but one still

Figure 5.   Breakfast pancakes

has to be ordered before the other. By changing the order of our tasks we can often make a big difference to the value that we deliver – indeed, Jeff Sutherland has been quoted as saying that we can double ROI by reordering the backlog.

Just in case you think we don't have enough synonyms in this section, we should add that you may also hear the term sequencing as in 'sequence to market'. Don Reinertsen uses it, as does Roland Cuellar, author of *Agile Metrics*. Using the word sequence may help because it allows product management to move beyond the concept that everything is high priority by stating, yes, that may be true, but let's determine which high priority item comes next.

What level of detail do you need to have to make ordering an objective, rather than a subjective, decision? It may not be hard when you're deciding between two features, but when you are trying to select from hundreds, potentially thousands, the issue could be difficult.

Hi there

Tim Lister and Tom DeMarco have a pragmatic view in their book *Waltzing with Bears*: 'Getting your stakeholders to do incremental value projections is going to be like pulling teeth. It's a nontrivial amount of work… You can expect almost any stakeholder to object strenuously to the effort and to assure you in the most earnest tones that "It's all necessary to support core functionality. Honest." This is the same tired, old value-is-equally-distributed line, but don't expect to convince them of that.'

They go on to point out that a rougher method of value ordering works just as effectively: 'The very fact that you will be implementing incrementally gives you a level for extracting rank-ordering instructions from even the most unwilling stakeholders. After all, some parts of the system will necessarily have to be implemented after some others. Some pieces will be first and some will be last. If you throw that back in your stakeholders' laps, they will leap to tell you about order of implementation.'

We'll go into the debate about level of detail and using numbers to help prioritise in more detail below.

2. Selection (just in time priority)

This is the method we described above when explaining how a pull system works. Rather than requiring the entire backlog to be ordered, pull works on a 'just in time' sense of priority. Because we know how long work will take to flow through the system we have a fairly short horizon against which to plan (which should make ordering much easier). If the work you input today will be completed in 30 days time (assuming that is our average lead time), and we have three slots available to 'pull' work into, then the question becomes relatively simple. What three pieces of functionality do you most want to see implemented in 30 days? This helps focus the mind, but it does not, of course, stop disagreement or subjective mistakes.

David Anderson discusses several methods for dealing with this in his book, from simple negotiation, through whoever shouts loudest to a 'weighted' vote. The real problem is that none of these methods are focused on the real customer and what the customer values. This means they don't actually help with the objective prioritisation and decision-making that we are aiming for.

What they do help with is speedy decision-making (the benefits of which we've discussed above), and they can also help ensure that the risk is felt by the decision-maker. For example, the tendency is for a department to insist that 'this feature is really, really important'. The development team might bust a gut to produce the feature, only to discover that it was not that important after all… The ways around this which Anderson discusses are either 'weighted choice' (if you pay 50% of the team's budget, you might get to choose the priority 50% of the time) or 'classes of service' essentially a way of saying that the faster you want something done, the more you are charged. We'll go on to discuss classes of service as a technique further in this session.

Selection as we have described it remains about prioritising the tasks, but the prioritisation is done 'just in time'. In theory this should cut down on the amount work required if you are performing major calculations on the value of each feature or requirement, but it's not a solution in its own right. When dealing with more complex decisions across numerous projects or with huge numbers of requirements, flaws start to show.

3. Prioritisation

We've said it's essential, we've said people try to do it, but we do it badly and we've said that the value to customers is what counts... this is all true, but prioritising features is still not easy. A customer doesn't always know what's valuable until they're using it. Or they may just want everything, or they may have their own confusion about internal priorities. This can make it hard for a customer to send you back a neatly typed and reordered to do list – preferably with a dotted line to represent the point at which they don't mind dropping features against timely delivery. Product managers find it hard too. After all, they may have signed off on a project to deliver X amount of value – it's quite hard to ask them to say 'oh well I don't mind losing some of that value'.

If you want to give a figure that quantifies value, it's even harder – especially if you want to break it down to a granular level. While the first person to stick a camera on a mobile phone knew this was adding value to the product offering, the ability to make a phone call was simply standard. Nowadays, having a camera on your phone might also be regarded as standard, necessary to keep up with the competition, rather than adding unique value. Leffingwell and Widrig in *Agile Software Requirements*, ask 'How does one quantify the impact of keeping market share, one feature at a time?'

Mike Cohn, in *Agile Estimating and Planning*, looks at four factors to help prioritise new capabilities:

1.   The financial value of having the features.

2.   The cost of developing (and perhaps supporting) the new features.

3.   The amount and significance of learning and new knowledge created by developing the features.

4.   The amount of risk removed by developing the features.

Not unnaturally, we have a tendency to focus on the first two, but monetary return is not the only one. Updating your architecture to better support your products might not add a lot of value by itself, but the risk of not doing it might be extreme.

Hi there

Value

DeMarco and Lister wrote a succinct rule of thumb: 'when benefits are not quantified at all, assume there aren't any.'

We need to know how much we expect to make or save from the introduction of a feature. Of course this can be hard to quantify, but however flawed your financial assumption, it is better than none. What is good design worth, for example? It may be hard to give a perfectly accurate answer, but since we know that products rated 'stylish', 'attractive' or 'easy to use' have a certain sales advantage, we can at least work towards putting a value against the cost of employing a good designer.

Cost

This is often much easier to know and it has, rightly, a major impact on a feature's priority. Cost has two elements – direct cost and cost of time. It may sound obvious, but sometimes companies can treat their internal resource as if it were free – refusing to invest a small amount of money in automating a process that would save days of a developer's time because they refuse to see that time as value. In general, organisations spend more of their time on working out the cost of a feature than they ever do on what value it will deliver – arguably this is entirely the wrong way round.

New knowledge

We learn as we do things, and this knowledge is valuable to us. If we didn't ever try to do anything new, we wouldn't advance. Sometimes our efforts to find the right solution can look like waste (why didn't you just get it right the first time?), but learning that a path is wrong or an idea is impossible is also valuable. As Mike Cohn insists, 'It is important that this effort be acknowledged and considered fundamental to the project.'

Risk

Everything we do includes some risk. The cheapest, least risky kind of business would be one that didn't do anything – but then it wouldn't be a business. Risk can be defined as anything which might derail the project or limit its success. Risk is not confined to the project alone – it could impact on the wider organisation as well.

Often the high-risk features are also the ones with the greatest potential for value – something unique or new to set you apart from the competition. These may be at odds with the bread and butter of the product, the features that will produce the returns reliably and swiftly. It can be tempting to fill the pipeline with safe projects whose value feels assured, ignoring the riskier, but potentially game-changing proposals. Often this is a recipe for stagnation and decline. That is why neither risk nor value alone is sufficient to determine priority; the two must be balanced.

4 PRIORITY: WITH NUMBERS OR WITHOUT NUMBERS

We have already touched on this debate when quoting Lister and DeMarco on how to break down value across features. Do you prioritise with numbers, or without? Do you assign a quantified value or Cost of Delay as a numerical figure to each item? 'We believe project X will deliver $50,000 a month and it has a Cost of Delay of $60,000 per month (lost sales plus decline in overall time in market).' Or do you choose to give an item a label? 'Project X is very important and urgent; project Y is quite important but not very urgent.' These labels can be based on gut instinct or on fairly sophisticated estimating, but you are not attempting to quantify them exactly.

People are quite often naturally drawn into one camp or the other, often based upon their confidence or otherwise with numbers.

Figure 6.   Number-crunching pleasure!

'Those number-crunchers,' mutter the label crew, 'they spend so much time finessing their estimates they don't get any actual work done. By the time they've calculated a Cost of Delay they already ARE delayed.'

'Pretty important? Pretty important?' sniff the numbers crowd. 'What kind of a way is gut feel to run a business? Why not just pick a project at random if you're not going to bring some objectivity to the process.'

The Emergn View

On the whole we feel that where you can quantify a number, however vague or flawed your assumptions, you will do better to do so. But we acknowledge that one can become too tied up in the entire business of checking your numbers – so much so that you may forget the process only tangentially adds value to the project. Also numbers can feel like solid and unassailable ground – 'my application is worth £400,000, my projection says so. If customers aren't buying it, marketing must have done a rubbish job in explaining why it's so cool.' Numbers are only as good as the assumptions you based them on – you must check, adapt and rework as new learning comes to light.

4.1.   With numbers

The approvals process exists to insure we invest money and resource wisely. To make it through a proposal rarely consists of: 'Hey, I've got a great idea – I think our customers will love it!' Even if you can justify your belief that the idea will delight your customers, those on the approvals board (perhaps the CEO and definitely the CFO), are likely to ask what this means in terms of hard revenue figures. Will customers buy more or pay more? How much profit will your idea add to the company's bottom line? Once you have identified a benefit, there should really be a way to quantify it.

It's worth quoting Denne and Cleland-Huang at some length from *Software by Numbers*: 'Finding that figure isn't as easy as it might seem. But it may be one of the most important decisions to be made in the lifecycle of this software development project; one so crucial that it could actually halt the project completely or cause it to be stillborn. This is neither the time nor the place for the developer to fade into the background. It's often here that the success or failure of an application software development project is ultimately determined. And it's not as though financial approval is going away or becoming less important. Over 80% of IT managers surveyed in 2001 reported that the importance of ROI has increased compared to previous years.'

To some extent then, quantifying the value of your idea in terms of real numbers is simply an inevitable part of the way we do business. But their use is not simply about jumping through hoops. Every product development decision is an economic choice with direct consequences for the profit and loss of the company.

Understanding this properly is the way to manage the trade-offs that we need to make around cost, speed and quality. Indeed, with what Don Reinertsen calls 'an economic framework', we can ease decision-making since many choices can be expressed as decision rules.

Let's posit a simple one: if the cost of delay is more than two times greater than the cost of the resource required to avoid that delay, the team is empowered to incur that cost to acquire the resource (which might be extra staff, for example, or it might be investment in automation). That means you have devolved power to the team, which speeds up decisions, but you can still be confident of overall control because you set the framework within which those decisions are made.

Reinertsen also points out that we do not need perfect and exact calculations: 'Even imperfect decision rules improve decision making. These decision rules are critically important because they permit us to quickly process the small, perishable, economic decisions that arrive continuously and randomly.'

Quantifying benefits

How do we begin to calculate value? Mike Cohn suggests considering the financial value of an idea over a period of time – the next few months, or perhaps even years. This is important because it will help you crystallise whether this value is time-sensitive – for example, if we launch our software next month, we will pick up on customers at the Agile Conference, but if we launch it later, then we won't. This in turn helps to calculate what for Reinertsen is the most important piece of the economic framework: Cost of Delay. We have mentioned it several times in this session and we shall go on to explore the idea more fully below.

So that's easy. We ask how much each project (or increment within a project) is worth in terms of revenue or how much it will helps us save, and then we pick the most valuable ones.

There are two problems with the approach. Firstly, when you are creating something where the market is well understood, you may have quite firm assumptions; with a new market, you may have very little idea. The big wins (and the big failures) often exist in this 'hard to quantify' area.

Secondly, as Clayton Christensen puts it in *Innovation Killers*, 'project teams generally know how good the projections need to look in order to win funding [or gain prioritization], and it only takes a nanosecond to tweak an assumption and run another full scenario to get a faltering project over the hurdle rate.'

No matter how sophisticated the model and spreadsheet we create to crank out the business case, it is still based on assumptions. All too often these get forgotten, and organisations and teams act as if the figures they have generated are real. If reality turns out not to meet expectations, rather than going back to examine which assumptions were flawed and asking if they could have been tested sooner, the entire process is quietly brushed under the carpet, while team members look around for something external to blame.

Hi there

Imagine a typical proposal: by automating this process, we will be able to stop using the manual system which employs two clerks at $20,000 p.a. each. That's a saving of $40,000 against a cost of $5,000 to develop the automation. This sounds great. Everyone signs off on it. But when the H.R. manager finds out, she loses her temper! The assumption has failed to take into account the costs of running a full redundancy consultation period, let alone any severance payments that may be due to these staff. The automation may still be the right thing to do, but it was based on partial assumptions.

What are the answers?

- **Transparency** – not only do people need to know what the rules are for getting through the approvals process, but assumptions need to be made visible. This means taking responsibility for those assumptions – which is not the same as living in fear of the CFO bearing down on your team shouting 'you said it would make $50,000 and it's barely covered the cost of the coffee you drink!'

- **Test quick, test cheap** – the only way to deal with those assumptions (for which you are responsible, remember?) is to start testing them as early and as cheaply as you can. This is where the agile advice of 'fail early, fail often' comes from. It's fine to be wrong about how much a project could be worth, but the faster you find that out, the sooner you can kill the project.

Activity 5 – Understanding the assumptions that your business case is built on

This activity requires up to an hour of preparation time and a 30 minute group session.

In order to understand the value delivered by a project, most organisations perform a Cost Benefit Analysis of some description. Within that analysis lie some assumptions. Those assumptions tend to suggest that a certain change will yield a rate of return.

Example assumptions:

Does your initial revenue per month assume you launch with the full product, or does it take account of a beta version?

By upgrading a piece of software, you assume that you will avoid an increase in maintenance charges, but the upgrade may require new training amongst your staff and this cost may not be included.

Preparation (1 hour): Review previous business cases and uncover the assumptions they contain. You can do this on your own, or within a group.

Once you've identified some assumptions, pick a couple that are of interest to you and organise a session to brainstorm ways that your organisation could have tested the assumptions cheaply and quickly.

Tests for example assumptions:

Try a 'willingness-to-buy' test using a prototype with a few typical customers.

Find out what the skills of the end users are. Ask what training they required for the last upgrade.

Group session (30 minutes): Divide into two groups, each taking one of the assumptions you identified. Explain the problem: how to test this assumption quickly and cheaply to ensure that our predictions are right. Make it clear that the objective of the session is to generate as many ideas as possible. As the session progresses, be mindful not to allow critical appraisal of ideas – the session should really be about collaborative idea generation – no holds barred.

Distribute Post-it notes or similar among the audience and ask people to note down their ideas and stick them up on a wall. As you do so, announce them to the rest of the audience – this should allow other people to take inspiration from the ideas. Once the pace of new ideas has slowed up, announce an extra minute for the final ideas. Once the time has expired spend some time grouping the ideas that have been generated. Decide which idea you will progress as a group and decide on the steps necessary to use this test for a current or future project.

Commentary:

In an ideal world, all assumptions in business cases would be tested cheaply and quickly with the intended customers so that a guarantee could be placed on the return. In extreme circumstances, this can be very difficult. That is no reason for assumptions to go untested – just asking questions of different groups and about past projects can provide insights. Often the simplest thing to do is reveal assumptions – making them as transparent as possible – so that everyone can challenge them and in doing so gain a greater understanding.

Hi there

4.2.   Without numbers

If you are plucking numbers out of the air, or your calculations risk taking more time than the code itself, or even if you are worried about teams obsessing over random metrics rather than the ideas that stand behind them... you can manage without numbers.

Richard Dawkins in his book *The Selfish Gene*, argues that our genes are extremely good at influencing our brains to make highly accurate, lightning calculations. If your child falls into a river, you will almost certainly jump in and attempt a rescue – even in the face of significant risk to yourself. If your sibling's child falls into a river you are still extremely likely to jump in. If a stranger's child falls into a river, the odds on you jumping in decrease; for an unknown adult, the odds fall further. Dawkins would claim that your genes have done an extremely nice calculation that balances risk to you against the chance of saving someone who shares a significant number of your genes (50% for your child, 25% for your sibling's child, under 0.5% on average for a stranger's child if you happen to be of the same race/nationality).

This does not mean that you stand hesitating on the bank doing mental arithmatic. No – your gut instinct has made you act in one way or the other long before. But your gut instinct has weighed the odds up with a more worrying accuracy than we might like to admit.

The same is often true in software development. Developers may not estimate how long each defect will take to fix (thereby giving a cost that can be balanced against the value of the fix), but they will classify bugs based on how critical they are, which is sufficient to set the priority. You instinctively know that the error your customer's CEO is complaining about is more important than a bug which occurs only when trying to order 38 tonnes of cotton wool on the third Tuesday of the month.
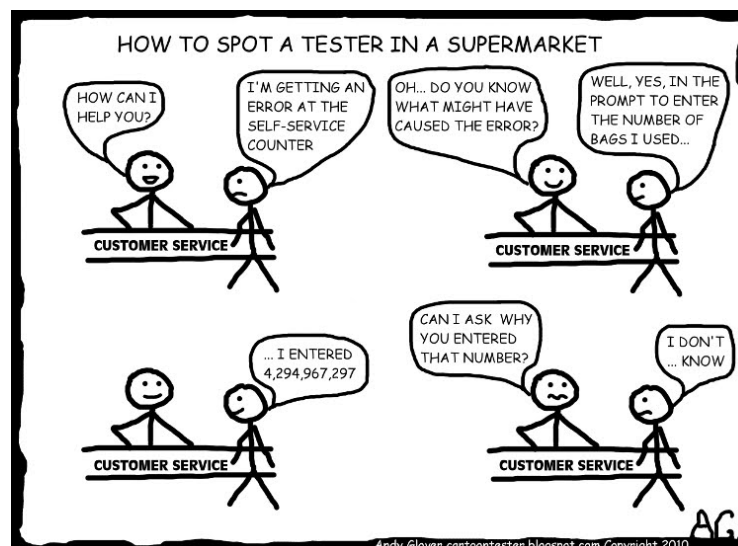

Figure 7.   The relative importance of errors

There are some highly respected figures in the industry who feel that calculating ROI and other value metrics is simply a waste of time because of how many, and how vague the assumptions that lie behind these figures are. Among the most forthright critics is Rothman, whose book *Manage Your Project Portfolio* calls ROI 'a lie or, at the least, fun with numbers.'

Gaining really reliable ROI figures requires robust research, like a 'willingness-to-pay study'. Making this sufficiently real to be meaningful normally requires large blocks of functionality. Either this means doing much of the work before the test which is meant to justify the work (beautifully circular logic), or proving the desirability of features on paper which may end up so radically different when built that the research is now invalid.

Luke Hohman on practicalmarketing.com discusses the attachment that is normal for us to have to our ideas (which includes the work we've done on research and business plans). He concludes that the capacity to respond to change is compromised when 'asking a product manager who just invested a lot of time and money in her market research to reshuffle the backlog based on new information. They'll be torn. The market research says they should stick to the plan. The new information says it should change. The greater the investment in the market research, the more challenging it is to make the change.'

His point is valid, but it simply goes back to the solution we gave earlier – your testing must be early and it must be cheap. An insightful blog post by S. Anthony on the Harvard Business Review points out, 'While corporate leaders sometimes think that they lack good ideas, more often their real challenge is to quickly separate good ideas from bad ones.'

The point is not to have a water-tight ROI figure, but one which you are prepared to test and revise as new information comes your way. A willingness-to-pay study costs money, especially if done for the 100 people normally considered a minimum statistically significant sample in market research. Asking ten friends if they would buy it is free. It doesn't mean your information is perfect or unassailable – just that it's probably better than asking no one.

Hi there

5 TECHNIQUES TO HELP YOU PRIORITISE

You've probably come across some of these already in your working life. We will provide a brief overview of each technique along with its main advantages and disadvantages. Finally we have assessed how well they fit in with the principles of ordering, selection and numbers described above, which should help you decide in which situations they are most likely to be of use.

Activity 6 – Preparing for prioritisation

In the following activities you will have an opportunity to experience, practice and reflect on various prioritisation techniques, their advantages and disadvantages. To make this experience more meaningful and relevant this activity asks you to work with real data from one of your projects.

This activity will take 30 minutes and prepare you for all the following activities. You can do it on your own, but you might find advantages in involving some of your colleagues – three or four other people.

Once you have selected the project that you wish to focus on – either a current one, or one that is yet to start – you need to review the project and identify 30 discreet requirements of a similar size (size is quite important – but don't labour over the decision too much).

For each of the items note what the requirement is, and why it is required on an index card. Give the requirements a number and record this on the sheet in the Appendix as the 'requirement identifier'. This is not a priority order as yet, but it will help in each of the subsequent activities.

Hi there

5.1.   MoSCoW (DSDM)

Why have one acronym when you can have two? The 'MUST, SHOULD, COULD, WON'T', is a technique to help you prioritise using words rather than numbers which was originally included in the Dynamic Systems Development Method (DSDM) techniques, designed to assist companies adopt Agile methods. Despite this, the tool has become popular in many places – including organisations still wedded to sequential or waterfall development.

MoSCoW analysis requires you to separate all your requirements into four categories: Must, Should, Could and Won't.

Let's imagine we're organising a party:

Must: every requirement under this heading must be implemented in order for the solution to be considered a success.

*We **must** have: a venue with professional lighting and sound equipment, guests, security.*

Should: these items are high priority and should be included in the solution if possible. They are often critical, but it might be possible to satisfy the requirement in other ways if strictly necessary.

*We **should** have: a licensed bar (if strictly necessary we could tell guests to bring their own alcohol), a DJ (our backup plan is to play a playlist from our laptop through the professional sound system).*

Could: describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.

*We **could** have: decoration, parking area, cloakroom.*

Won't: represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future.

*This time round we **won't** have: entertainers, party favours, a chill-out room.*

You can probably guess some of the problems. Because the process does not enforce a strict order, it enables stakeholders to avoid some of the most difficult questions. It's all too easy to see the 'must' list fill up, while the only items in the 'won't' list were ones that no one really wanted anyway. The development team can be equally at fault, overestimating the difficulty or complexity of implementing certain requirements in order to influence the prioritisation process. Jean Tabaka, in her book *Collaboration Explained*, recommends 'Balancing MoSCoW by the Product Owner or customer with technical estimates that clearly bound a timebox scope may help a team better grade its stories or requirements.'

Hi there

Summary

In this section, we've introduced the MoSCoW method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
No	Probably not	Usually not

Pros and cons

Pros	Cons
Easy – you can sit around the table as a team and work on this together	No actual order is created, which does not assist with pull system or increment planning
Simple, meaningful language can increase the engagement of everyone on the team	New teams tend to fill the 'must' category because stakeholders deem everything important
Clear criteria for each category should assist the team	Criteria are often ignored
	Highly subjective, encouraging politicking from all quarters

Case Study *– Adapting MoSCoW*

Sometimes customers have a constraint around time, or an insistence on a specific feature. On this occasion, the customer had a strict constraint around budget. The brief to the consultancy Thoughtworks was simple – create as much functionality as you can within a certain amount of money. That meant that prioritisation was absolutely key – because the consultancy needed to add value from day one, knowing that at whatever point development stopped, the company should be left with a valuable and usable piece of software.

Chris Johnston, a developer on the team, described using a very short iteration of just one week so that the team could gather feedback fast to ensure they never strayed far off track. Because they were travelling at speed, the team used a form of MoSCoW with the customer. The broad categories felt simple and easy to the customer, which made the process speedy. Unlike in the traditional use of MoSCoW tool, because they were working in such small iterations, the team planned only to work on the 'must' items. These were the user stories without which it made no sense to continue with the project – the absolute core functionality.

The team then gave each story a rough estimation of cost. Rather than doing this as an exact $ figure, the team kept things simple by using a relative points system. Thus a 5-point story was more effort than a 1-point story, and a 1-point story represented the least amount of development time of all the stories on the table. This was not yet matched to actual development time, just effort.

At the start of each week the team thus had a pool of stories they knew were extremely important (valuable) to the client, graded by effort. The business analysts and developers could also help group together stories that were dependent upon one another or sequential as an aid to prioritising within the iteration. In consultation with the customer, the team could pick stories whose points matched the team's velocity.

There was one further refinement – the team used another non-financial measure in assessing priority: risk. The team prioritised the risky stories or ideas, because these were the items on which feedback was the most crucial. Other teams might have been tempted to concentrate on elements they knew they could deliver in order to provide a pre-determined solid value, but Thoughtworks realised that risky elements should be developed, deployed and evaluated by the customer earlier. The more feedback they gained, the more the team would learn, making it more likely that risky items would stop being 'risky' and become successful.

The team were deploying software every week for the customer to try out. That meant the customer was engaged in actively re-prioritising items each week, moving ideas from 'Should' or 'Could' into or out of 'Must'. There was little worry of customers over-stuffing the 'Must' category, because they were confident they could re-prioritise or add more in a week's time. It also meant the customer felt true ownership of the software and confidence in its delivery. This combination of circumstances and tools – strict budget constraint, short feedback loops powered by fast iterations, a clear understanding of what constituted 'value', and frequent deployment of small working increments – was the reason that the MoSCoW method worked so well for this team.

Hi there

Activity 7 – Using MoSCoW

This activity will take 30 minutes and use the requirements that you identified and captured in Activity 6. Gather a few colleagues – 5 or 6 should be enough.

Part 1 (20 minutes):

Prioritise the requirements using the MoSCoW method. Note the order that the requirements end up in using the table provided in the Appendix.

Part 2 (10 minutes):

Discuss the merits of the method you used with your colleagues. How were you able to quantify your decisions between one category or another? Did you notice anything particular in terms of how decisions were made?

5.2.   Classes of service

Classes of service is an extremely simple idea to grasp because we're used to it already in many different industries. In the far-off olden days when cameras took pictures with film, you had to have your photographs developed. You could walk into a shop, hand over your film and pick a class of service. Did you want it the next day, or in a one-hour express service? Naturally, you paid a premium for the express. You could also post off your film for the cheapest service of all, which usually took a week or so.

Courier companies, Royal Mail, Amazon… most delivery services adopt a form of classes of service, all of which boils down to a simple rule – the faster you want it, the more you pay.

IT can implement exactly the same rules, whether dealing with external customers or their organisation's business units. The manager jumping up and down foaming at the mouth over how crucially important his particular requirement is, can calm down astonishingly quickly when told he can certainly have the requirement fulfilled by tomorrow… as long as he pays treble.

Figure 8.   A courier in a hurry!

Money doesn't have to play a part in the decision, although it is very effective. The idea is to ensure that those making the requests take responsibility for the pressure they put on the development process. It recognises something very simple – that not all work has the same value, or the same time-pressure. David Anderson puts it rather more formally, 'Classes of service provide us a convenient way of classifying work to provide acceptable levels of customer satisfaction at an economically optimal cost.'

There should be no more than six classes of service, but generally four is sufficient. The rules for each need to be transparent and clear, including how many requests may be permitted in each class of service at any one time, their impact on work in progress limits, and the estimation requirements involved.

Example classes

It is common to have an 'expedite' class. This is the highest priority, it jumps the queue and is worked on first. Some companies even permit expedite requests to break work in progress limits. Note that, this does not have to be assigned by value – indeed expedite is often selected because of risk, as a fix for a project already launched, for example, with an embarrassing, public mistake.

Next would come a 'fixed-delivery' class. Just as many delivery companies ask you to pay extra if you wish to book a certain time-slot, so guaranteeing a delivery schedule to take account of some external time-pressure (a customer's launch or a contractual obligation) may require its own class of service.

Hi there

'Standard' class is the most common. These items are important and need to hit an agreed level of service (perhaps we expect to see them pass through the workflow within 20 days).

Intangible is not the same as unimportant... indeed, these items may become extremely critical at a future date, but they are not critical to the project right this second. They might include updating systems, regulation compliance or a longer-term goal.

Rules are required to ensure that the backlog does not get filled up with expedite requests. To ensure this only one expedite request is allowed at any one time. There should also be fixed numbers created for fixed delivery, standard and intangible items. This way the team can ensure that they work on enough 'intangibles' – since these will generally become critical at some point, when they might overwhelm the team.

Since there is only one 'expedite' request permitted in any given time-frame, business sponsors have to argue the matter out between themselves to decide which item will gain priority. As long as the system is transparent, clearly understood and respected, this should result in the most valuable items gaining priority.

Classes of service save time, because they encourage a self-organised, pull system. Rather than a manager needing to check the status of everything and then informing developers what they should be working on next, as a slot opens up, developers know to scan the backlog and automatically pull the highest priority class of service. Before pulling a new item, they also need to check if they can help finish anything on the board – this may involve 'swarming' to help colleagues wrestling with a difficult expedite request.

If you were reading carefully, you would have noticed that we said the discussions between business sponsors 'should' result in optimised value. The sponsors are responsible for selecting and debating the prioritisation (which in turn selects the class of service). The method for doing this remains open to the usual concerns – how is it being done? Based on value or risk? With numbers or without? Also, since it relies heavily on debate between business sponsors, the selection risk becoming heavily focused on internal needs and priorities – not on the customer.

Summary

In this section, we've introduced the classes of service method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
A small buffer may be ordered for the team to select from	Yes	Yes, it could be depending on how the business sponsors select classes of service

Pros and cons

Pros	Cons
Fast – once established, Anderson even describes sponsors selecting a new item within two hours of capacity becoming available	How each item is assigned a 'class' remains unresolved and continues to have all the potential pitfalls of poor prioritisation decisions
By being connected to an explicit service level agreement, the system increases trust and predictability	The set up is not easy – clear rules must be established and all stakeholders must agree to a new method of working
Clear criteria for each category should assist the team	

Hi there

Activity 8 – Using classes of services

This activity will take 30 minutes and use the requirements that you identified and captured in Activity 6. Gather a few colleagues – 5 or 6 should be enough.

Part 1 (20 minutes):

Prioritise the requirements using classes of service. In order to do this, imagine the following. You have the standard four classes of service: Expedite, Fixed-Delivery, Standard and Intangible. The Expedite service is limited to 2 items, the Fixed-Delivery 8, the Standard 15 and the Intangible 2.

With your group organise the list of requirements in to the classes of service. You should attempt to stay within the limits but you might find that it is necessary to exceed them in certain circumstances.

Part 2 (10 minutes):

Discuss how the session went. Bear in mind that in Kanban a prioritisation session like this wouldn't occur – it would be done as and when capacity became available. How were decisions made in respect of which class of service an item would be put in? Did you need to exceed the limits at any points? What factors did you consider in order to make the decisions?

Note down the classes of service that the items ended up in, in the table in the Appendix – use a 1 to denote Expedite, 2 for Fixed-Delivery, 3 for Standard and 4 for Intangible.

5.3.   Equity

When business owners are debating who gets to pick the next item to go into development, occasionally a form of 'financial equity' is used – essentially, a form of decision-making where those paying 50% of the budget get to make 50% of the choices. An extension of this idea is one occasionally found in organisation where IT developers are assigned to the various business units. Perhaps finance is given two dedicated developers, marketing four and customer services one based on their projected need, size or budget... The idea behind such a split is admirable – improve IT's ability to meet the business unit's need. It is often popular with the business units themselves who are delighted they no longer need to struggle to gain attention for their problem from an overstretched department. Yet equity often throws up larger problems in the medium term.

As D. Mark writes in his article *Splitting Demand From Supply in IT*, 'It has the unfortunate consequence of fragmenting IT developers across business and functional units, so coordinating and prioritising projects becomes harder... the company as a whole may suffer from high development costs, poorly managed performance, and difficulties deploying cross-group functionality.'

After all, if requests were taking a long time to get through IT, it was probably because they were working on something else – a project, one would hope, of high value to the organisation. While the individual business unit may be delighted to be rid of frustration and delay, the organisation's overall value is likely to suffer.

Summary

In this section, we've introduced the equity method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
No	No	Yes - usually uses some form of counting

Pros and cons

Pros	Cons
Connects IT with individual business units – increasing an individual developer's domain knowledge	An individual's newly acquired knowledge is not shared with a wider development team
Clarity over responsibility for scarce resources is agreed up front	Resource may have been dedicated to lower value areas of the business, reducing ROI overall
	The method does not in itself prioritise either between business units or within each unit's projects

Hi there

5.4.   HiPPO decisions (HiPPOQ)

The acronym (much more attractive than the average management jargon) stands for Highest Paid Person's Opinion. It reflects a not uncommon *de facto* decision-making rule – whatever the CEO, or CFO, or Operations Manager or whoever thinks is the most important feature gains priority. Not many organisations like to admit it, and CEOs sometimes express astonishment at the idea, but we'll happily take bets on you having experienced it at some point in every project you've worked on.

The Highest Paid Person tends to think they know best. They may have the most experience. Apart from anything else, they may argue, they will take ultimate responsibility. Without wishing to infuriate any customers, we're not sure the last part is true. When a product fails in market, the CEO rarely resigns, but it's not uncommon to see the product manager or team lead looking for another job. There are certainly plenty of examples of inspired leaders who 'bet the farm' on their gut instinct and won big. But then again, you tend not to hear about all the failures... The HiPPO tends not to be a formal process of prioritisation – it happens internally. The team knows that the senior person is pushing for feature x, and so consciously or unconsciously, they move feature x up in priority.

Then CEO of OpenTable (and self-confessed HiPPO) Jeff Jordan, rather optimistically wrote that 'data killed the HiPPO Star'. He described how the company used overnight testing to examine their functionality, picking up major problems and making immediate changes. Even a relatively small change in how the site appeared to a non-registered user could increase reservation success rate by 10%. His attitude is in stark contrast to the number of CEOs who insist that they do not influence the prioritisation of a product list but simply find it serendipitous that the teams are in agreement with their own opinion every time.

Microsoft's Experimentation Platform has produced some rather charming hippo toys. As they write: 'We actually love the animal. Squeezing a HiPPO helps relieve stress when a feature decision is made on the basis of opinion alone.'

Summary

In this section, we've introduced the HiPPO decisions method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Maybe (If within Scrum Yes)	Yes	Unlikely

Pros and cons

Pros	Cons
Fast and simple – this is the opposite extreme to the frequent management complaint of 'too many chiefs, not enough Indians'	Command and Control culture where disagreement and healthy debate may be stifled leading to apathy within the team. 'I just do what I'm told, Guv.'
Single point of leadership/ responsibility. This doesn't have to be the CEO, it may be the product owner or project manager	Subjective – may ignore data and normally based on a single opinion

5.5.   Value divided by effort

The most common 'number' method for prioritisation is Benefit Cost Ratio (BCR). It is designed to help balance out the relative priority between differently sized tasks. Is it best to finish a short task that will produce a low amount of revenue or embark on a major piece of work that will take several months but has far greater potential payback? Those wishing to sound less formal in their estimation sometimes refer to the figure as 'bang for your buck' – the more value you yield for less cost, the higher the bang for your buck. To calculate Return On Investment (ROI) simply take the net value and divide it by the cost.

The calculation is nice and simple: we simply divide value by cost – and then express the result as an index or a percentage… take your pick. This at least gives you relative figures to compare features and decide on the priority. When the product launches, you can then compare actual value divided by actual cost to see how well your estimation process worked.

Cost is only one measure – you could also express it as effort or time taken. As a ratio, the idea of relative importance based on value and effort does not have to involve actual figures. A common tool to help decide on value and effort is to gain input from the whole team in an exercise sometimes known as planning poker.

Planning Poker

Every member has a number of cards, one for each feature. The cards have already been labelled with a relative value for the feature – current wisdom suggests using a non-linear series like the Fibonacci series to do so.

The team discuss the feature – how difficult will it be? How much time will it take? They then write an estimate for the number of development days on a card (keeping the score hidden). All the scores are revealed at the same times, prompting a second discussion perhaps with particular attention to the reasons behind outlying estimates (those who thought it would take either a very long time or a very short time).

Hi there

David Anderson recommends it as 'a form of collaborative cooperative game, [it] is highly efficient, as it allows a fairly accurate estimate to be established quickly'. One of the dangers is that in spite of the secret voting system, it can lead to 'groupthink'. The idea works best with relatively small teams focused on a single project rather than trying to estimate between highly divergent or new projects.

Summary

In this section, we've introduced the value divided by effort method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Yes	Yes	Unlikely

Pros and cons

Pros	Cons
Fairly fast, a rough and relative sequencing without using numbers can be as effective as actual figure estimates	Works best with small teams, when really large projects or teams are in play the process can be too time-consuming with high co-ordination costs
	Estimates, whether with numbers or without can be highly subjective and planning poker, as described can lead to groupthink

Hi there

Activity 9 – *Using relative ROI*

This activity will take 30 minutes and use the requirements that you identified and captured in Activity 6. Gather a few colleagues – 5 or 6 should be enough.

Part 1 (20 minutes):

You need to begin by coming up with an estimate for your items, both for value and size.

Start with value. Identify one of the items in your list of 30 that the group feels they understand well. This item will be your baseline – assign it 100 points (write this down on the card as your value estimate). For each of the other items in your list compare them to the other estimates you have made (the baseline item in the first instance). Ask the question "Is this item more valuable, or less valuable?" then ask the question "How much more/less valuable is?" For example, if you think an item is half as valuable as your baseline item (100 points), then the new item is worth 50 points. Repeat this step until the group has estimated the value of each item – don't spend too long estimating – it is intended to be indicative, not precise.

You now need to estimate the effort or time that you think each item will take. Start with a good baseline item – perhaps similar to a task you have carried out before. Estimate all the other items relative to that one.

Finally, for each item identify its ROI – divide the relative value score by its relative size score.

Rank the items according to their ROI – this is their priority. Record the order in the table in the Appendix.

Part 2 (10 minutes):

Discuss this method. How does it compare to the others you have used so far?

Hi there

Case Study – Adapting the measure for maximum benefit

Roy Schilling works as an Agile trainer and coach for a consultancy working with investment banks in the United States. Recently, the Dodd-Frank Reform Act was passed into law with the explicit aim: 'To promote the financial stability of the United States by improving accountability and transparency in the financial system, to end "too big to fail", to protect the American taxpayer by ending bailouts, to protect consumers from abusive financial services practices, and for other purposes.'

Such an act meant changes had to be made to many applications within banks, from large to small, in order to comply with regulation. To prioritise the many features or 'epics' required by such changes, the team used a multi-layered approach to prioritisation – specifically a form of weighted shortest job first. However, since the 'Cost of Delay' for the project was essentially fixed (the date by which regulation must be complied with or the company will incur fines), what really counted was how the team created a refined value/effort figure.

To begin with the team used MoSCoW to help the customer understand the scope of the project. Unlike in traditional usage, the purpose was really to help whittle the potential changes down to the core of necessary issues that would be affected by the new regulation. The team only planned to work on the 'Must' items.

From this pool (pretty large – the number of features were expected to fill a 6-9 month project), the team began to assign value and effort ratings. The team for this piece of work was kept fairly tight – no more than 10 – and included business owners who could provide value estimates and technical developers for the effort or duration estimates. As Roy Schilling freely admits, there is always a subjective element to 'value', but by ensuring the business owners were sitting in a room together, the conversation included its own checks and balances. Rather than assigning a $ value, the team used 't-shirt sizes' ranging from XS up to XXL. This helped make it clear that the value was not only financial, but also balanced risk and other important factors.

Each item would then be given a storypoint value taking into account relative effort, where 1 was the lowest effort of any task on the table. To make the calculation, each t-shirt size would be given a number value (taken from a modified Fibonacci sequence). The prioritisation was then simple – each item was calculated as value/ effort and displayed in order on a spreadsheet. This large prioritised backlog could then be split down into release backlogs with a shorter horizon.

At the end of each release, the team could revisit the prioritisation and check their assumptions for both value and effort, ensuring learning was captured and applied for the following release plan.

An interesting side-benefit of the process was that business owners realised that in order to move features up the priority list, they needed to make them smaller, because high value divided by short duration gave a high priority score. While this may have felt like 'gaming the system', in fact it encouraged a good behaviour – splitting features into smaller increments and focusing on the core element of value.

Hi there

5.6.   Cost benefit analysis (financial measures)

A full cost benefit analysis is a traditional process associated with launching new products or major projects, and forms part of the original business case. The prioritisation measure associated with this is 'tangible discounted cash-flow', a phrase that really trips off the tongue. This impacts on the point at which the project breaks even and how long the payback period is - both crucial measures in gaining approval for and measuring the success of any development project.
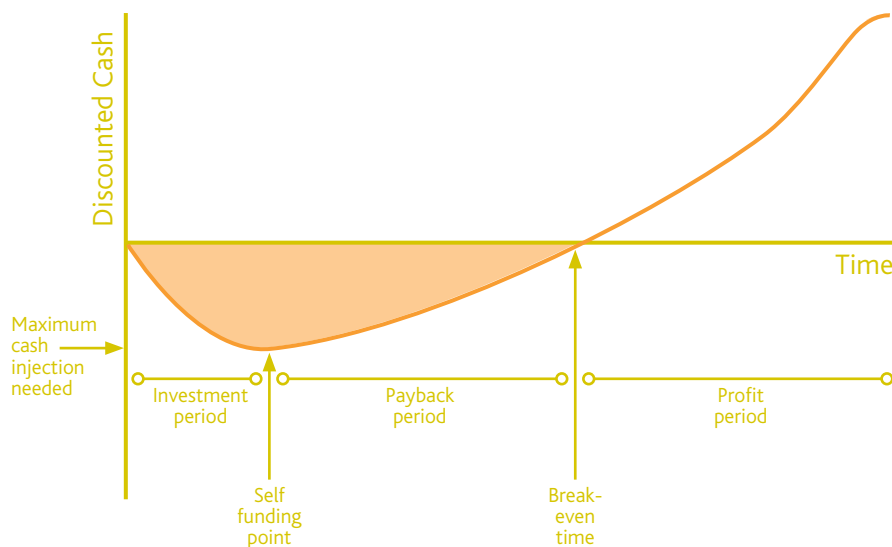

Figure 9.   A successful application development project

You might be congratulating yourself on having worked out the Benefit Cost Ratio, only to find managers shaking their heads and reminding you that since the cashflow is coming in over the next four years, you need to work out the discount rate (what your cost might have brought in if you let it earn interest in a bank, for example). After that you should consider the Net Present Value (NPV), which is what the project's profit would actually be equivalent to in today's money taking into account the discounted cash flow (obviously the higher the better).

If you're struggling with this idea just imagine someone offered you a choice between $20,000 today or $23,000 in ten years time. What should you do? TAKE THE MONEY NOW!!! Apart from inflation, you could put the money in a bank and even at 2% compound interest you would still be the winner. This measure is intended to help you decide between projects offering different sizes of value over different time periods.

From the NPV you can work out how many years the project will take to payback its investment. This is also sometimes known as the break-even point. Your organisation may have a rule regarding how soon a project needs to payback – perhaps within the year; perhaps within three years – but obviously a short payback is more attractive because there are fewer risks associated with it.

Hi there

Next comes the Internal Rate of Return (IRR). The IRR is how high the discount rate (that's an alternative investment rate remember?) needs to be to make the NPV equal zero. In essence, the higher the IRR, the less likely it is that you could achieve the return by any other means – and therefore the more attractive the investment. If the IRR is 28% you are unlikely to find a bank or stock market investment that would offer the same return. If your IRR is 3% when interest rates are at 6% then you need to have some solid reasons for undertaking the investment that are not finance related...

A purist would remind you that even 'intangible measures' (things like employee satisfaction, brand recognition or customer satisfaction) can be translated into a tangible financial measure if you try hard enough. For example, if your employees love you then they are less likely to leave. If you usually lose 25 employees a year and you think that this project is going to make everyone so happy that only the 3 guys coming up to retirement age are going to leave, then there will be an associated saving in recruitment, training and productivity. But good luck with those assumptions...

These are, unquestionably, helpful measures. They allow a comparison between investments. If you are choosing which of two companies to acquire then measures like these constitute due diligence. If you are deciding whether to work on the entertaining animation that pops up when a user inputs an order or the new layout for the registration screen – then IRR is probably overkill.

The criticisms quoted above in our 'without numbers' section are entirely valid. For these calculations to mean anything useful, significant amounts of work need to be done. This risks spending more money on the study than the project is worth, or delaying it so far that the work itself becomes meaningless as the environment changes. Full cost-benefit analyses are undoubtedly the best for ensuring the right decision, but often at the cost of the right result. The more granular the level at which you are making the decision, the more problematic this is likely to be.

There is one further issue. The measures described above do not take into account time sensitivity – or the Cost of Delay. If you are offered a little project to design some animated Christmas cards and a huge project to create an integrated website application – which should you do first? If you can only do one, then you pick the most valuable. But if you want to do both and it's November, then you do the Christmas cards first. As Don Reinertsen writes, 'The total profit of a high ROI project may be less sensitive to schedule delay than that of a low ROI. In such a case, the low ROI project should go first. Overall portfolio ROI adjusted for delay cost is more important than individual project ROI.'

In recent years, *Software by Numbers*, a book by Denne and Cleland-Haung, has helped overturn the idea that a cost benefit analysis is too complex. They encourage teams to return to considering ROI, Internal Rate of Return (IRR) or Net Present Value (NPV), but by looking at how they impact on each minimum marketable feature (MMF). In other words – how these measures can be made to work for smaller projects or increments, as well as for investments so massive that spending six months in a cost-benefit analysis makes perfect economic sense for the company.

Hi there

Summary

In this section, we've introduced the cost benefit analysis method. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Yes	No	Yes

Pros and cons

Pros	Cons
The most accurate, perfect decision and comparison of projects comes from full analysis using the measures briefly touched on (plus one other).	Comparatively slow and expensive.
	All figures can be manipulated by changing the numbers just a little in order to pass the approval hurdles
	The more detailed the analysis the more confidence people have in it – often unjustified if analysis is based on uncertain assumptions

5.7. Incremental Funding Model

Mark Denne and Jane Cleland-Huang point out in their book, *Software by Numbers*, 'financial return is usually the enduring metric of success in software development'. Denne recalls the breakthrough they made when instead of planning their project work to minimise cost and risk, they reprioritised to maximise early delivery of value to the customer. Such re-prioritsation required the marriage of financial measures like ROI with incremental delivery. Despite the validity of the caveats discussed above, the principle of optimising financial value and being able to see that we are doing so, remains sound.

The authors describe their application of traditional financial analysis to incremental delivery as the Incremental Funding Model, or IFM. Nothing makes something sound official like giving it an acronym, while the term 'model' perhaps makes the idea sound rather more ground-breaking than it is. We discuss the idea elsewhere in this course in the Delivering In Increments session, but here we will simply say that the concept is not dissimilar to the way venture capitalists fund 'risky' projects.

Hi there

A small amount of funding is approved to generate results; if these are successful or show potential, more can be granted. Financial feedback forms an equally important guide to development direction as any other kind of feedback.

The way a project delivers value has a major impact on all of the measurements we've discussed above. To give a simple example, if we are delivering revenue this year rather than next year, its NPV is higher. If a feature begins earning revenue earlier then it will pay back earlier, reducing risk and letting us earn revenue for it for longer. When the model is reconsidered this way, NPV and IRR both rise, while payback goes down... all of which makes Incremental Funding Models a more attractive way of building an overall business case for the project.

Of course, breaking an idea up into independent chunks of value to create Minimum Marketable Features (MMFs)is not always easy, nor is deciding exactly which items go into each MMF. In this session though, we assume that you have done this, and that we are now focusing on how to sequence MMFs for optimum value.

Avoiding the greedy approach

If we are deciding between minimum marketable features, the highest value to cost ratio remains the most attractive, surely? That's what's going to push that IRR so high that approvals boards will be falling over themselves to sign off the project.

Of course, this is understandable, but there are other considerations. Reinertsen's point about cost of delay is still valid, so is the need to include intangible elements that may not directly add to an immediate financial return, but add greatly to value – anything from good architecture to improved customer satisfaction. Moreover, even if MMFs are theoretically separate, some may be dependent on an earlier one's design or delivery. For example – if you are creating a banking feature for a mobile device then a small MMF that allows the customer to see the week's banking activity on the account is valuable. A secure log-in does not produce any direct revenue, nor does it save the cost of sending out paper statements, but no banking system will work unless it is secure. All the other MMFs are dependent upon this one.

That example may be obvious, but others can be a harder call. Short-sightedness over such issues affects the project's overall long-term value. For Denne and Cleland-Huang, this is an oversight that can be cured by a deeper look into the financial measures – specifically the truly snappily named Sequence-Adjusted Net Present Value or SANPV. They conclude 'the greedy heuristic might intuitively be taken by customers and developers if they select an MMF without the benefit of a full NPV analysis.'

Summary

In this section, we've introduced the Incremental Funding Model. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Yes	No	Yes

Pros and cons

Pros	Cons
Use the numbers from your business case	Full access to numbers required by the whole team (some information may be sensitive and difficult to obtain)
Encourage early delivery by appreciating the value of increments	Can push projects towards financial focus, ignoring other potentially crucial measures of value
	Difficult and time-consuming

5.8. Weighted look ahead approach

Not to worry, the indefatigable Denne and Cleland-Huang have an alternative calculation (which for some unknown reason they have not chosen to give the acronym WLAA).

First, the authors consider all the different NPV values that could be generated by any sequencing path through the MMFs (described as a strand). Anyone who can cast their mind back to permutations will find this next bit easier, while anyone who enjoyed logic puzzles as a child will find it easier still.

If we have MMFs from A to E that can be combined in any order, then you would have 120 permutations. Fortunately, or perhaps unfortunately, since several MMFs will be dependent on a preceding MMF (which Deene and Cleland-Huang call a precursor), this number reduces depending on the relationships. Without wishing to go through a slightly tedious worked example, take it on trust that you can come up with a number of strands, depending upon these relationships.

Rather than simply selecting the strand with the maximum NPV as the end result, you can improve your selection further by taking into account development time and use a calculation 'negatively weighting each strand according to the number of time periods required to develop it.'

Hi there

If this is giving you a headache, the authors' cheery admission will really delight you: 'As it turns out, the IFM sequencing problems belongs to a category of computer science problems for which there are no known algorithmic solutions. It's necessary to compute and compare all possible combinations. This type of exhaustive computation typically requires an unreasonably long time to run on a computer.'

What, you may ask, is the point of it then? The authors claim, and they are in many ways right, that by applying rules of thumb learned from analysis, we can get pretty close to optimal delivery sequence. It gives you an approximate answer that's good enough for all practical purposes (as the engineer said to the mathematician).

When you recall that for million pound projects the increase in value may be over $300,000 simply for reordering the delivery sequence (and that's on top of the gains we made just by beginning to deliver value in increments as opposed to in a big bang), then sitting down with a spreadsheet starts to feel like something worth spending a few days on.

Summary

In this section, we've introduced the weighted look ahead approach. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Yes	No	Yes

Pros and cons

Pros	Cons
Encourages incremental development and increases value	It's difficult.
	All the caveats about numbers, over-confidence and assumptions apply.

5.9.   Weighted shortest job first (sometimes known as CD3)

If you feel the need for a beer, relax. The good news is that we think there's an even easier approximation on which to rely, which requires far less mathematics. Reinertsen summarises the decision principles as three simple rules:

1. When delay costs are homogeneous, do the shortest job first.

2. When job durations are homogeneous, do the highest-cost-of-delay job first.

3. When job durations and delay costs are not homogeneous, use weighted shortest job first (WSJF).

OK, so we think maybe he could have said it more simply – but we agree with the idea. If both jobs have the same cost of delay then do the shortest first, because the shorter the job, then the sooner you can release the value it represents. If both jobs take the same amount of time, then you should do the one with the highest cost of delay first (even if this job is of lower value).

The third rule is the tricky one – and it's also the most common. Normally every variable is different: your two jobs have different values, take a different amount of time to complete, and have different time sensitivity (Cost of Delay). You are, to return to that favourite cliché, trying to compare apples and pears.

Weighting the priority is a simple calculation. Priority is based on the cost of delay divided by the length of time it takes to do the job.

Note that the calculation requires only the data from one single job – not sequencing paths and strands and lengthy NPV calculations. There's no need to try to create complex algorithms. The calculation will give you an order. It may not be the most perfect, but it will be simple and fast, and is built on fewer interdependent assumptions. In other words, it's focused on the right outcome, not the right decision.

Why duration? Why not cost – the more common measure used in Benefit Cost Ratio and such related measures? Just as Denne and Cleland-Huang recognised in their calculation – time is in some ways more important than money. Time is a resource that can never be recovered. By delivering value early we make overall value improvements that far outweigh the effect of cost alone.

Hi there

Summary

In this section, we've introduced the weighted shortest job first approach. In the tables below, we summarise the main characteristics and consider the pros and cons.

Characteristics

Ordered	Just in time	Uses numbers
Yes	Could be	Yes

Pros and cons

Pros	Cons
The fastest system with numbers that allows you to prioritise while comparing unlike tasks	Takes some time to get up and running – requires teams to calculate Cost of Delay and estimate task duration
Can scale easily to include large numbers of items or MMFs and works at both micro and macro level	Does not reject ideas and relies on having good, profitable ideas to choose between
Maximises ROI when your primary constraint is development team resource	If your constraint is not resource but money and cashflow, then this system could be disastrous.

Hi there

Activity 10 – Using weighted shortest job first (CD3)

This activity will take 60 minutes and use the requirements that you identified and captured in Activity 6. Gather a few colleagues – 5 or 6 should be enough.

Weighted shortest job first requires a little more work per item than the others, but we urge you to undertake this activity – there are some important points of differentiation.

For each of the items in the list you identified you need to calculate the Cost of Delay, then divide that by the estimated duration of the task. The answer can be used to compare the item directly with the others in the set.

Normally you would list the assumptions upon which benefits are based, but we won't for this activity. Secondly, we'll ask you to quote the Cost of Delay as a weekly figure – we prefer this, because it instils a greater sense of urgency, but there are circumstances when you might want to consider quoting the number on a per monthly basis, perhaps even yearly. Finally, don't take too long to compile the benefits case associated with each item, this should be a relatively quick way to get to a number.

Part 1 (30 minutes):

To calculate the business value for each item, first consider 4 different types of benefit:

- Increase Revenue – will the item increase sales or profit margin?

- Protect Revenue – will the item help your company sustain a position it already has in the market?

- Reduce Cost – will the item remove a cost that your organisation is currently incurring?

- Avoid Cost – will the item assist in the avoidance of a cost that your company is not currently incurring but will do in the future?

Your item may have multiple benefits, some in Increase Revenue and some in Reduce Cost, for example. Try and express the benefits for an item as the return over a 3 year period (which is a standard accounting period).

Now we can find the Cost of Delay for an item. Take the total benefits that you

$$\textit{Business Value} \quad = \quad \textit{Revenue Increased + Revenue Protected + Costs Reduced + Costs Avoided}$$

have identified for an item, divide them by 3 to get the benefits for a year and then by 52 to get the Cost of Delay per week.

Now we need to estimate the item's size. Depending on how big or small your

$$\textit{Cost of Delay} \quad = \quad \textit{(Total Benefits / 3) / 52}$$

Hi there

items are you may wish to estimate in hour, days or weeks of effort. It doesn't matter which, as long as you use the same time measure for all the items.

Finally, you can calculate the WSJF (CD3) index. Divide the Cost of Delay number by the Duration.

Repeat this exercise for each of the items in your set so that you have an index

WSJF (CD3) Index = Cost of Delay / Duration

for each item. Now rank the items and note their order in the table in the Appendix.

Part 2 (10 minutes):

Discuss the method with your colleagues. How did it differ to those that you have previously used in the Activities above? Were the conversations you had while doing the prioritising different? Did you notice any patterns emerging during the calculation of benefits cases?

Hi there

Case Study – *Maersk and CD3*

Maersk, the Danish shipping giant, has a single global SAP system that covers accounting and reporting. It also supports the key ordering processes and both the physical and financial supply chains. Changes to the system were necessary to improve responsiveness to customers or for other business needs. The problem was that changes were taking up to four months and nobody was convinced that the right changes were being selected for development. There was a sense that the 'fuzzy front end' was taking so long that projects got stuck in an analysis paralysis. To get around this, managers used other methods to work around the official process, gaining the ear of senior figures to circumvent controls, for example.

The company needed a common standard that would improve prioritisation trade-offs and ensure that IT was delivering the maximum business value. Crucially, the prioritisation method needed to focus the team on improving cycle time, delivering faster than the current four months. The team also knew that the measure needed to work independently of any specific methodology or process. SAP systems, for example, do not lend themselves particularly easily to Agile, partly because of a difficulty with incorporating small incremental changes, but mostly because it is not easy for SAP specialists to become truly cross-functional.

For these reasons, Maersk decided to use a measure that would 'put a price tag on time' – Cost of Delay (CoD). Whether projects had a value peak affected by late entry or not, the team were able to show the positive impact on revenue when launching early. They began estimating these benefits, both revenue and costs saved and making assumptions where necessary (and recording the assumptions to be checked in the future) in order to reach an actual economic $ value. For example, if a planned change was to improve invoice accuracy the team, worked to give accuracy a $ value. In this instance they estimated that more customers would pay on time – worth an additional $4 million per annum. They also estimated fewer 'chase' calls would be required by employees, a saving in labour cost of $100,000. Thus the CoD of one week was the total benefit p.a. / 52 weeks. In this example, that would equate to a CoD of $78,846 per week.

Now the team needed to translate CoD figure into a comparative figure to deal with the different duration of each project. As we know, this is: cost of delay divided by duration.

The new method made it far easier to deal with difficult trade-offs. The decision-making was visible and transparent, using data that everyone could understand and measure, taking some of the subjective passion out of prioritisation debates. It provided a clear-cut business case that the Board could get behind and it focused the attention of all those involved on value (not just cost-cutting).

Occasionally there would be genuinely special cases – times when strategy might dictate a priority even if the figures did not quite support it. Maersk called this 'Prioritisation in the Real World'. CD3 provided the standard, and any decision outside of this framework was the exception, not the rule.

Hi there

6 CONCLUSION

We hope we've made it clear that prioritisation is important. Done well it will ensure great value for your organisation. That should make your bosses, shareholders and customers happy, it should keep your company's finances healthy – and that should increase your own job security (and financial health), plus provide the satisfaction of a job well done.

We've discussed many different ways of helping you prioritise, but they all rest on common foundations if they are to succeed.

The prioritisation recipe

When cooking dinner the success of your dish depends on several things.

Ingredients

If you put mouldy cheese and canned spinach into your homemade pasta dough, your ravioli is not going to taste as good as if you used the finest ricotta and fresh spinach gently wilted with a little olive oil and sea salt.

In prioritisation the right selection ingredients are:

1. Knowing exactly what 'the best business value' means. This is not only financial.

2. Objective methods – in other words, every idea or project is judged fairly.

3. Transparent methods that are visible across the whole organisation.

4. A timely manner. Spending too long on making the perfect decision could diminish your chance of getting the right outcome.

Figure 10.   The finest ravioli

Technique

You've prepared your beautiful ravioli – now you need to be sure your cooking technique is right (lightly poaching them until they float to the surface). Too long or too short a time in the water will ruin them.

With prioritisation you also need good techniques for the best result:

1. Keep testing your assumptions. Your decisions are only as good as the basis on which you made them.

2. Don't let numbers become the hurdle rate which will tempt you into tweaking them for false results.

Behaviour

You could be the most technically-brilliant cook in the world, but there are some behaviours just as important as talent: keeping your kitchen clean so you don't give everyone food poisoning, presenting your dish attractively, tasting your food…

Here are the essential behaviours you need in IT prioritisation:

1. Ordering is what matters most. Never mind the value ratio to cost or time or anything else. Get the items into a priority order and start testing and then changing the order based on real data.

2. Don't see value and figures as a waste of time belonging to those bean counters in Finance. Value calculations are helpful for many types of decision and IT managers should own the data in order to robustly manage the process in which they are supposed to be expert.

3. Try to quantify your 'gut instinct' – apart from any actual accuracy you achieve, this may help when comparing what actually happened to what you thought or hoped would happen. You will learn for the next time.

4. Remember that using numbers gets easier over time. Even if data-gathering feels time-consuming and calculations seem difficult, the rigour of your thinking will be improved, while a feedback loop ensures learning. Eventually, you build up sufficient internal understanding to actually make well-informed decisions quickly – it may seem like gut instinct to others, but in fact it is the result of inner decision rules based on steadily acquired knowledge and experience. Just beware – this is what plenty of HiPPOs think…

Finally, we'd like to speak briefly about the opposite to value and prioritisation. Whether ordered with numbers or without, people make decisions in our organisations. They can (and do) ignore excellent business cases in favour of something risky but thrilling; they can turn down enticing projects because they don't like something they can't even articulate about the idea. People are not always rational. We know consumers don't always buy what is 'best' for them, often they buy what's 'cool' – perhaps the most intangible of all intangibles. Nor do investors always invest based purely on financial value – sometimes they support something just because they like it.

When Rayner Unwin read a lengthy typescript from an obscure academic called J. R. R. Tolkien, he wrote to his father that the book was a work of genius but might lose them £1,000 – a sum which would have probably bankrupted the publishing house. His father telegraphed back a famous reply, 'If you believe it is a work of genius, then you may lose a thousand pounds.'

In the final analysis, a compelling vision will encourage teams to achieve success even better than a fully worked-out business case. As the venture capitalist Anthony Tjan wrote, 'A large part of successful execution is rational prioritization and careful planning. But the spirit of excellence in execution gets its energy from conviction… Entrepreneurs know that they go into situations with the statistical odds stacked against them, but rationality is overwhelmed by conviction in the possibility of succeeding.'

Hi there

Making priority choices between projects or MMFs or features does not depend on an algorithm. It depends on a skilful balancing act between speed, accuracy and setting decisions up so that you can learn from them and apply the learning. Of the three, the focus on feedback is the most important.

Five things to try this week:

- Write down the assumptions that underlie a prioritisation decision or the numbers that go into it. Share them with the team. Make sure you keep them for future comparison.

- Set up a test for one of your assumptions or an element of your assumptions. Keep it small and simple.

- Have a go at quantifying the Cost of Delay and thus CD3 for a new project.

- Reframe a couple of project estimates according to differing measures of value – share any insights with the team.

- Check the epicentre of design on your current project. Have you nailed it? Have you already achieved it? Can you stop here?

Learning outcomes

Now you have completed this session, you will understand:

Where there is a mismatch between supply and demand, prioritisation is a necessity

- IT is busier than ever and needs to invest its resources wisely

- We need to maximise business value both at the macro and the micro level

- To compare different types of project we need measures that permit meaningful comparisons

Strategies devoted to managing demand

- Priority is not based only on organisation-wide priorities, urgency can be equally important

- Identifying the key customer priority for each project ('epicentre' design)

- Need to change prioritisation as learning develops of what is the real demand

Strategies devoted to managing supply

- Prioritisation requires capacity to ensure work is flowing

- The difference for prioritisation between pull and push systems

- Do not try to do everything on the priority list by dividing your capacity, instead focus as much capacity as possible on getting the most valuable done first

Problems with how we prioritise

- Decision rules are often not transparent and as a result are open to manipulation

- So much focus is placed on the 'right decision' that it takes too long, adding to cycle time

- With an external customer, priority is often easier to decide

- With internal customers, it's important to map business value onto external customer value

- Priorities decided by functional silo are often in conflict with one another, pulling an organisation in different directions

How to unpick 'business value'

- Value – both revenue and cost-saving value should be considered, plus any possible time sensitivity. Identify benefits either direct or by considering the cost of the alternative

- Cost – both direct cost and the cost of your internal resource (development time) should be measured

- Risk – risk impacts both on the project and potentially the wider organisation. Since risk is often closely linked to potential high value, both must be balanced

- Learning – learning has a value both for the future and for the organisation

Differences between ordering, selection and prioritisation

- Tasks in IT rarely have to be strictly sequential

- Deciding on a different order can radically alter the profitability of a project

- Order necessitates choice – one must be done before another

- Selection based on just in time priority, its connection to 'pull', classes of service and weighted equity

- Prioritisation using a combination of the measures of 'business value' discussed above

- Level of detail required for objective choice

When to use numbers or approximate 'labels'

- Detailed plans and predictions (especially when tied down with numbers) can give a false sense of security

- Numbers are only as good as the assumptions that lie behind them. These must be made visible and checked as knowledge improves

- Occasionally too much effort and time can be required to uncover sound figures

- Figures can be manipulated inappropriately to pass known approval hurdles

- Providing a sound economic framework makes decision rules easier

- Assign numerical values where possible, ensure assumptions are recorded and test early and frequently

- Where assumptions are vague, in new markets or for short or urgent projects, relative 'labelling' can work just as well

Hi there

Common prioritisation techniques, their advantages and disadvantages, when to use them and the type of flow process they suit best

- MoSCoW – provides a simple and inclusive way of scoping out vision at a broad level. Provide clear criteria and monitor how many features can go into each box

- Classes of service – quick to use and ideal for pull systems. Their use side-steps the question of objective business value

- Equity – dividing IT resource between departments may be popular but it rarely contributes to end-to-end value

- HiPPO decisions (HiPPOQ) – often exists in practice and has the virtue of speed and single point of responsibility

- Value divided by effort – simple Benefit Cost Ratio calculation can also be expressed as a ratio, involving the team through planning games

- Cost benefit analysis – series of financial calculations, which often take too long and are based on dubious assumptions when broken down at a feature-by-feature basis

- Incremental Funding Model – for large projects the potential increase in profitability can make a compelling case for looking at these measures without becoming too granular

- Weighted look ahead approach - considering the order in which increments can be delivered based on their dependencies can impact on profitability

- Weighted shortest job first (CD3) – the simplest calculation to compare differing projects taking into account variations in size, value and urgency (Cost of Delay Divided by Duration - CD3)

The non-rational decisions that affect prioritisation

- People make decisions, and people's decision-making is not always rational

BIBLIOGRAPHY

Anderson, D., 2010. *Kanban: Successful Evolutionary Change For Your Technology Business*. Blue Hole Press.

Anthony, S., 2011. *3 Ways To Prioritize A Long List Of Ideas*. HBR Blog Network, [blog] 4 April. Available at: http://blogs.hbr.org/anthony/2011/04/three_ways_to_ prioritize_a_lon.html [Accessed 2 April 2012].

Benson, R., Bugnitz, T., 2004. *From Business Strategy to IT Action: Right Decisions for a Better Bottom Line*. Wiley.

Christensen, C., 2010. *Innovation Killers: How Financial Tools Destroy Your Capacity to Do New Things*. Harvard Business Press.

CIO.com, 2003. *Staying Focused on Strategic Value Key to Prioritizing IT Projects*. [online] Available at: <http://www.cio.com/article/29846/Staying_Focused_on_ Strategic_Value_Key_to_Prioritizing_IT_Projects>. [Accessed 2 April 2012].

Cohn, M., 2006. *Agile Estimating and Planning*. Prentice Hall.

Cramm, S., 2010. *8 Things We Hate About IT: How to Move Beyond the Frustrations to Form a New Partnership with IT*. Harvard Business School Press.

Cramm, S., 2010. *How IT-Smart Is Your Organization?* [blog] 16 February, Available at: <http://blogs.hbr.org/hbr/cramm/2010/02/how-itsmart-is-your-organizati. html>. [Accessed 21 March 2012].

Cuellar, R., Augustine, S., 2009. *Agile Metrics for Senior Managers and Executives*. Available at: <http://www.slideshare.net/VersionOne/2009-04agilemetrics>. [Accessed 14 August 2012].

Dawkins, R., 2006. *The Selfish Gene: 30th Anniversary Edition*. Oxford University Press.

Denne, M., Cleland-Huang, J., 2003. *Software by Numbers: Low-Risk High-Return Development*. Prentice Hall.

DeMarco, T., Lister, T., 2003. *Waltzing with Bears: Managing Risk on Software Projects*. Dorset House Publishing.

Harvard Business Review, 2009. *Make Better Decisions*. [online] Available at: <http://hbr.org/2009/11/make-better-decisions/ar/1>. [Accessed 27 July 2012].

Harvard Business Review, 2002. *Six IT Decisions Your IT People Shouldn't Make*. [online] Available at: <http://hbr.org/2002/11/six-it-decisions-your-it-people- shouldnt-make/ar/1>. [Accessed 21 March 2012].

InfoQ, 2011. *Product Backlog Ordering, Sequence for Success*. [online] Available at: <http://www.infoq.com/news/2011/12/ProductBacklogOrderSeqToSuccess>. [Accessed 2 April 2012].

Hi there

Jordan, J., 2012. *Data Killed the HiPPO Star*. Andreessen Horowitz, [blog] 21 February, Available at: <http://jeff.a16z.com/2012/02/21/data-killed-the-hippo-star/>. [Accessed 27 July 2012].

Keen, J., 2011. *Making Technology Investments Profitable: ROI Roadmap from Business Case to Value Realization*. John Wiley & Sons.

Leffingwell, D., Widrig, D., 2010. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise.* Addison Wesley.

Leinwand, P., Mainardi, C., 2011. *Stop Chasing Too Many Priorities.* HBR Blog Network, [blog] 14 April. Available at: <http://blogs.hbr.org/cs/2011/04/stop_chasing_too_many_prioriti.html>. [Accessed 2 April 2012].

Manifesto for Agile Software Development, 2001. *Agile Manifesto*. [online] Available at <http://agilemanifesto.org>. [Accessed 21 March 2012].

Mark, D., Rau, D., 2006. *Splitting Demand From Supply in IT*. McKinsey. [online] <http://www.mckinseyquarterly.com/Splitting_demand_from_supply_in_IT_1849>. [Accessed 14 August 2012].

McDonald, M., 2009. *Change The Game If IT Demand Is Greater Than Supply.* Gartner Blog Network, [blog] 5 June. Available at: <http://blogs.gartner.com/mark_mcdonald/2009/06/05/change-the-game-if-it-demand-is-greater-than-supply/>. [Accessed 16 January 2012].

Net Objectives, 2010. *Estimating Business Value*. [online] Available at: <http://www.netobjectives.com/blogs/estimating-business-value>. [Accessed 10 April 2012].

Pragmatic Marketing, 2008. *Why Prioritising Your Agile Backlog For ROI Doesn't Work.* [online] Available at: <http://www.pragmaticmarketing.com/publications/topics/08/why-prioritizing-your-agile-backlog-for-roi-doesnt-work>. [Accessed 11 April 2012].

Reinertsen, D., 2009. *The Principles of Product Development Flow: Second Generation Lean Product Development.* Celeritas.

Ries, E., 2011. *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Portfolio Penguin.

Ross, W., Weill, P., 2004. *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Harvard Business School Press.

Rothman, J., 2009. *Manage Your Project Portfolio: Increase Your Capacity And Finish More Projects.* The Pragmatic Bookshelf.

Scrum Alliance, 2011. *It's Ordered - Not Prioritized*. [online] Available at: <http://scrumalliance.org/articles/367-its-ordered--not-prioritized>. [Accessed 2 April 2012].

Hi there

Shalloway, A., Beaver, G., Trott, J., 2009. *Lean-Agile Software Development: Achieving Enterprise Agility*. Addison-Wesley.

Smith, P., Reinertsen, D., 1998. *Developing Products In Half The Time: New Rules, New Tools*. John Wiley & Sons.

Stevens, D, 2010. *Kanban: What Are Classes Of Service And Why Should You Care?* Dennis Stevens ... Enabling The Agile Enterprise, [blog] 14 April, Available at: <http://www.dennisstevens.com/2010/06/14/kanban-what-are-classes-of-service-and-why-should-you-care/>. [Accessed 2 April 2012].

Tabaka, J., 2006. *Collaboration Explained: Facilitation Skills for Software Project Leaders: Facilitation Skills for Collaborative Leaders*. Addison-Wesley Professional.

Tjan, A., 2010. *The Power of Ignorance*. HBR Blog Network, [blog] 9 August, Available at: <http://blogs.hbr.org/tjan/2010/08/the-power-of-ignorance.html>.

The Experimentation Platform, 2006. *HiPPO FAQ*. [online] Available at: <http://www.exp-platform.com/Pages/HiPPO_explained.aspx>. [Accessed 27 July 2012].

Womack, J., Jones, D., 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Simon & Schuster.

Yuce, O., 2012. *Improving Prioritisation: Introducing Cost Of Delay And CD3*. [online] Available at: <http://softwarepassion.se/speakers/2012/Ozlem_Yuce/Presentations/Improve_Prioritisation.pdf>. [Accessed 10 October 2012].

Hi there

APPENDIX

Results of prioritisation activities

Use the table below to record the order of each requirement (as identified in Activity 6) when ordered in Activity 7 through to Activity 10.

Requirement Identifier	MoSCoW (Activity 7)	Classes Of Service (Activity 8)	Relative ROI (Activity 9)	Weighted Shortest Job First (CD3) (Activity 10)
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				

Hi there

Hi there

www.emergn.com

www.valueflowquality.com

Hi there