AGILITY IN MIND

# Brilliant Agile Project Management

UK:+44 (0)330 043 0143
US: +1-888-203-4358
better@agilityinmind.com
agilityinmind.com

The Coach House
48 New Park Street
Devizes, Wiltshire
SN10 1DS
United Kingdom

Agility in Mind Limited
Registered in England & Wales
Number 7289974


# *agile* project management

PEARSON

At Pearson, we believe in learning – all kinds of learning for all kinds of people. Whether it's at home, in the classroom or in the workplace, learning is the key to improving our life chances.

That's why we're working with leading authors to bring you the latest thinking and best practices, so you can get better at the things that are important to you. You can learn on the page or on the move, and with content that's always crafted to help you understand quickly and apply what you've learned.

If you want to upgrade your personal skills or accelerate your career, become a more effective leader or more powerful communicator, discover new opportunities or simply find more inspiration, we can help you make progress in your work and life.

Pearson is the world's leading learning company. Our portfolio includes the Financial Times and our education business, Pearson International.

Every day our work helps learning flourish, and wherever learning flourishes, so do people.

To learn more, please visit us at **www.pearson.com/uk**


# *agile* project management

## A practical guide to using Agile, Scrum and Kanban

**Rob Cole and Edward Scotcher**

PEARSON

Harlow, England • London • New York • Boston • San Francisco • Toronto • Sydney • Auckland • Singapore • Hong Kong
Tokyo • Seoul • Taipei • New Delhi • Cape Town • São Paulo • Mexico City • Madrid • Amsterdam • Munich • Paris • Milan

**Pearson Education Limited**
Edinburgh Gate
Harlow CM20 2JE
United Kingdom
Tel: +44 (0)1279 623623
Web: www.pearson.com/uk

First published 2015 (print and electronic)



ISBN: 978–1-292–06356–0 (print)
978–1-292–06358–4 (PDF)
978–1-292–06357–7 (eText)
978–1-292–06359–1 (ePub)

*British Library Cataloguing-in-Publication Data*
Cole, Rob, author.
Brilliant Agile project management : a practical guide to using Agile, Scrum and Kanban / Rob Cole and Edward Scotcher.
pages cm.—(Brillian)
Includes index.
ISBN 978-1-292-06356-0 (pbk.)
1. Agile software development. 2. Scrum (Computer software development)
3. Just-in-time systems. 4. Computer software—Development. 5. Information technology projects—Management. I. Scotcher, Edward, author. II. Title.
QA76.76.D47C6434 2015
005.1--dc23

2015034100

10 9 8 7 6 5 4 3 2 1
19 18 17 16 15

Cartoons by Ken Pyne

Print edition typeset in 10 pts Plantin MT Pro by 76
Print edition printed in Great Britain by Henry Ling Ltd, at the Dorset Press, Dorchester, Dorset

NOTE THAT ANY PAGE CROSS REFERENCES REFER TO THE PRINT EDITION

For Alfie and Cissie, Ella and Alice

For Nkeiru, Lisa, Amara and Joe

# Contents

# About the authors

**Rob Cole** is a project management consultant with over 20 years of experience. He specialises in project troubleshooting and mentoring. Rob has been involved in the Agile community from its earliest days and is a practising Scrum Master.

Rob can be contacted at: **robacole@btopenworld.com**

**Edward Scotcher** is a leading agile product manager, project manager, trainer and coach. He specialises in helping organisations, teams and individuals adopt Agile in a practical and sustainable way.

Ed can be contacted at: **edward.scotcher@agilityinmind.co.uk**

CHAPTER 3

# Getting ready: preparing to be *agile*

## Introduction

Many traditional project management methodologies are based on dotting all the i's and crossing all the t's before setting off. Project requirements are drafted and redrafted, reviewed, revised, revamped and examined from every possible angle before any *real* work is done – whereas there's a view in some circles that *agile* is all about making it up as you go along, starting off with a vague idea and then winging it.

Nothing could be further from the truth. Yes it's true that the first goal is to get feedback quickly on a basic release. Yes, it's also true that subsequent deliveries build on that foundation. And yes it's very true that there's plenty of flexibility along the way to make adjustments or even completely change track if necessary. However, this is done in a considered way based on a vision of the final destination and always for good *business reasons*.

It's very easy to confuse flexibility with a scattergun approach but *agile* is a far more considered approach to project delivery than traditional methods, not the other way around. Less effort is put in before setting off and it's more wisely invested. There's a world of difference in what gets done first and instead of trying to pin everything down, *agile* puts in place a framework that delivers early and builds steadily from there.

*Agile* ensures the end goal is defined up front and an enabling infrastructure is put in place for getting there but worries little about the fine detail of the journey before setting off.

Luck is where opportunity meets preparation.

Denzel Washington

## Defining the vision

Without a vision, the people will perish. Moses said that about 4,000 years ago and from the looks of it most project managers still don't know what he was getting at. If we don't know where we're going, we won't get there and this is especially true with projects. That's because people will make all sorts of assumptions about what it is we're trying to do, interpret things differently and drive towards different results. Having a clear vision is essential.

Unfortunately many visions go the way of bad *mission statements* and seem to say much but once put under the microscope reveal very little:

- We want to offer unparalleled quality.
- We aim to put our customers first and deliver value.
- We will be the best at what we do and loved by everyone everywhere.

Nice sentiments but ultimately useless. You can't deliver against them. You can't use them to know when you've done the business.

### *Defining a project vision*

- ✔ What is the name of the project or product you are making?
- ✔ Who is it for?
- ✔ When will it be done by?
- ✔ What will it do?
- ✔ What will it not do?
- ✔ What benefit does your business get for doing it?
- ✔ What benefit does your customer get by using it?

*Write the vision in a way that your Mum or Dad can understand what you're planning to do.*

A vision should be a tool that you can use. You can use it to communicate intent. You can use it to explain what you are doing and what you are *not* doing. You can use it to prioritise against. A vision for a project should be less of a strategic mission statement for the business and more of a tactical, practical device to help you stay focused. A vision must be open to change and get updated *when* it goes stale, which it surely will.

As an example of how to use a vision, let's imagine that we're an existing company that wants to start selling organic vegetable boxes to our restaurateur customers. A vision can help us define exactly what that means:

> By the end of September, Project VegBox will develop a new iOS app that lets our customers order from a premium range of 10 veg box products for their restaurants, to be delivered with their regular orders. By offering this, our business gets to grow the veg department by creating a new product line and our customers have an easy and convenient way to get organic veg at wholesale prices.

The vision explains what we are planning to do. It's quite specific about the target group, platform, and product – plus when the project will be done by. Most importantly, it talks about the value delivered to both the business and the end customer.

**brilliant example**

A family day out doesn't need to be planned with military precision. It's more than enough to agree on the basics: the destination, what to take and how to get there. There's no need to agree a minute-by-minute itinerary before setting off and limited value in considering every possible

what-if scenario. The main thing to get right is whether you're going to a theme park, heading for a relaxing day on the beach or a day shopping.

*If the end destination is agreed, the rest is just about the logistics of getting there.*

## Driven by business value

Too many projects start off with a half-baked notion. Someone in a position of influence or with a budget to burn comes up with a brainwave and before anybody knows it there is an unstoppable juggernaut heading for who knows where. Of course, projects are never openly acknowledged as whims, but don't assume the foundations are solid just on the back of a persuasive senior manager or the enthusiasm of the company clever clogs. Carry out due diligence and never assume that a sensible investment decision has been made.

*Agile* is totally focused on delivering *business value*. From the start of any project and all along the way, the business team will know exactly what they're getting for their money.

**brilliant definition**

People say they want to *deliver value* so often that it can become almost meaningless. What does *value* mean? Think in terms of *benefit*. What *benefit* does this bring to the customer or the business? The most successful projects provide business value to both.

*Agile* doesn't dictate the definition of *business value* and to a large degree beauty is in the eye of the beholder; it provides a framework for ensuring the business think through what it wants for its hard-earned cash without in any way dictating to it what constitutes a wise investment. *Agile* facilitates and enables

sensible decisions and nothing more. It takes the horse to water but doesn't force it to drink.

Every delivery, every feature, every nuance must be described in business-speak. Gone are the days of a person or persons unknown defining a list of requirements in technobabble or a foreign language the business doesn't fully understand before lighting the blue touch paper and retiring to a safe place. Long gone are the days of the business putting its blind faith in people they hardly know. With *agile* the business describes what it wants and then works within the project team to ensure the vision is delivered exactly as requested.

Not only that, the business will define exactly what's needed as a *minimum* to get going and every additional chunk needed from there on in. The first delivery may not have many bells and whistles but will be usable, deliver value and it will arrive sooner rather than later. It will be crystal clear that for an investment of X then Y gets delivered initially and ditto for every subsequent delivery down the line.

**brilliant tip**

If you don't know what it is you're building (the vision), what benefit it will bring (value proposition) or who it's for (end user proposition), then you can have the best experts in the world and yet never deliver anything worthwhile.

## Building the project team

Many people stress themselves silly by building a high-quality project team, finding the best people they can to get the job done – people with proven experience, experts in their fields – and then fail to give them adequate business and leadership.

Talk about putting the cart before the horse! The importance of getting the right level of business involvement – one empowered individual who understands the business vision – is not just practical and pragmatic, but pure common sense.

With *agile* this person is known most commonly as the *Product Owner* but there are variations on the theme. *Product Management* is worthy of a *Brilliant* book in its own right but for the sake of getting started let's keep it simple. A Product Owner represents the needs of the business and the users. Product Owners live, breathe and dream about the product and what it will be. They know *what* they want even if they don't know *how* it will be done. They are leaders, able to make decisions quickly and stand by them even in the face of opposition.

The *agile* team is a diverse, cross-functional group of individuals that has the ability and authority to deliver the vision on behalf of the business. Put simply, between them they have everything they need to get the job done properly. The Product Owner leads the way in terms of the business vision but it's very much a team effort. The team consists of people who have an *agile* mind-set, who are not afraid of change and don't need to use process and bureaucracy as a crutch to get by. Confident decision makers with a self-starter, can-do attitude are the best for this.

Collectively the team must buy into the vision and co-own all aspects of the project delivery. If that isn't the case, there'll be big trouble ahead.

## Creating a backlog

Once there is a practical vision in place and the business value is established, the next step is for the project team to pin down in more detail what's required. At the heart of any project is this type of requirements list and with *agile* this is known as

### *Ways to get off on the wrong foot*

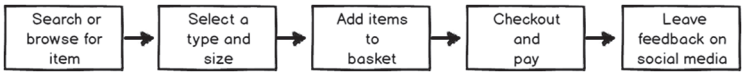
- ✔ Proclaim *agile* as the only answer – either join the gang or hit the highway.
- ✔ Announce that it's all obvious – any idiot can pick it up, so no training is required.
- ✔ Declare *agile* is infallible – failures will be solely down to personal inadequacies.
- ✔ Dictate unreasonable targets and deadlines – explain that nothing is impossible in this brave new *agile* world.

the *product backlog*. It replaces the traditional, detailed, requirements-specification type approach and is in the form of a shopping list of ideas that's meaningful to the business. Items on the backlog are always user-centric even if they have a technical slant. The litmus test is that they make sense to pretty much anyone.

A sensible place to start is for the project team to dig into the vision statement as a group – to make sure everyone understands it, its scope and what it's helping us to conceptualise. Making sure all parties are on the same page from the start is much easier than trying to fix a broken project two-thirds of the way through the process. The diversity of the team is important, as they need to think about the project from all different angles. If necessary, specialists can be drafted in to help out.

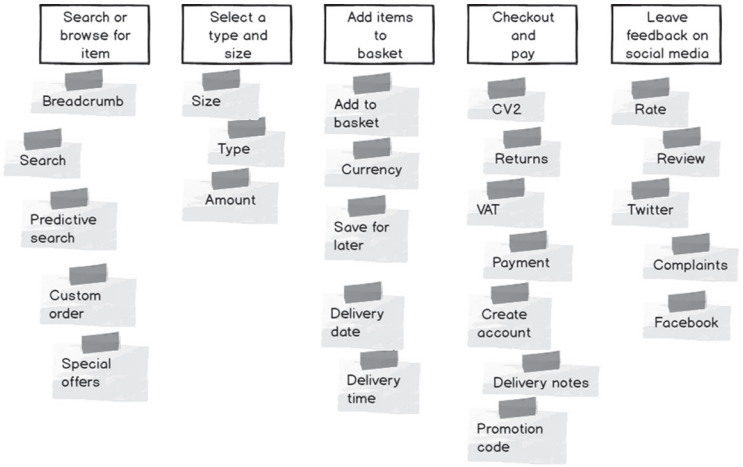
## Define primary functionality

The aim is to produce a summary list of what's needed to deliver the project vision. There are several ways to do this and our favourite approach is to think about each step of the customer journey to produce a *workflow*.



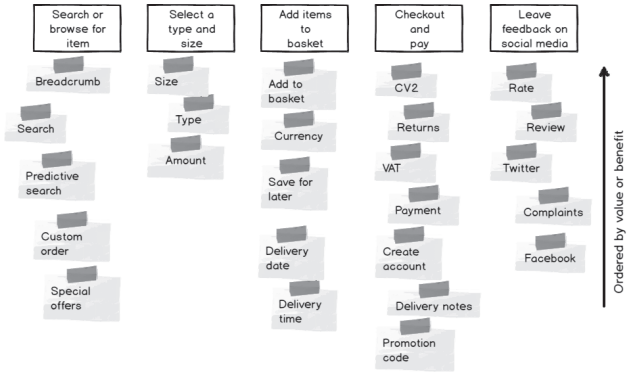

## Produce feature groups

Once the workflow is mapped out, gather together ideas about what's actually *done* within each step in the journey. Added together these items deliver the functionality of the step and are often referred to as *feature groups.* Some of the items will be absolutely essential and some nice-to-haves. To begin with get all the thoughts down.



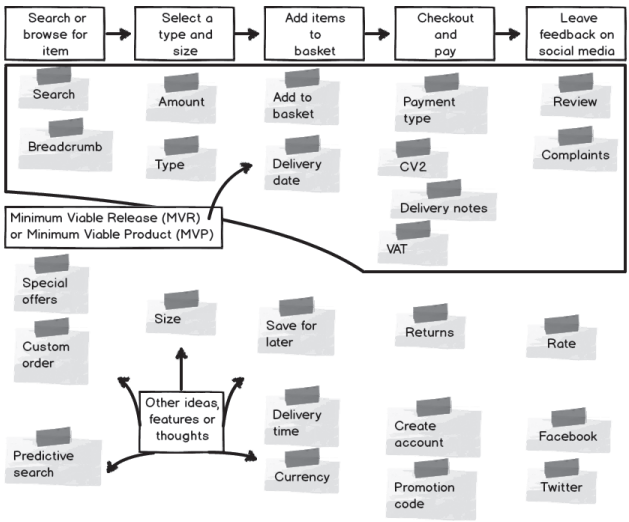

## Prioritising the features

Using the value statement in the project vision and common business sense prioritise the ideas, in descending order with the most valuable item at the top of each list.




## Identify the first delivery

Once that's done think about whether each step in the customer journey is vital from Day 1 and what's the most valuable chunk of ideas within those steps. This selection process can be challenging and at the end of the day a matter of opinion, but the business, or whoever represents the needs of the business, is the best judge of all this. The end result is the minimum the project must achieve to deliver a useable outcome. This is usually referred to as the *minimum viable product (MVP)* or the *minimum viable release (MVR)*.



One of the biggest hooks for *agile* working is to get fast, meaningful feedback from the end customers and they need something tangible to provide an opinion about. It's not possible to get meaningful feedback on any new under-the-bonnet techie infrastructure, for example, but it is possible to get feedback on a new VegBox order form. This would naturally be part of the MVP.

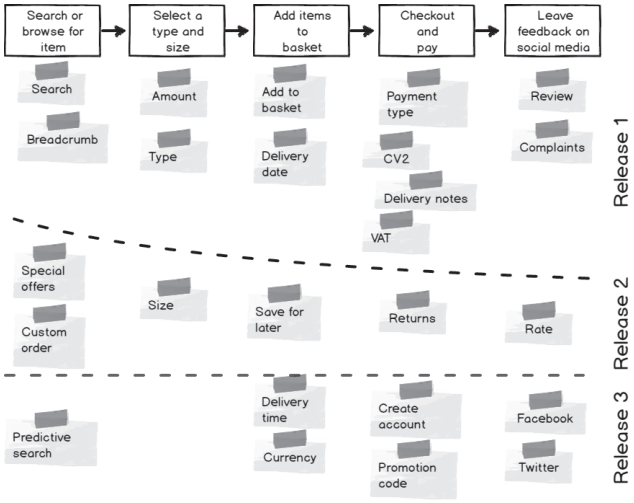
We need to be aware that the more there is in an MVP, the longer it will take to get feedback, whereas if we put too little in it, there will not be enough information to get feedback on. You have to strike the balance between return and risk – there's no golden rule. Try to find the point where you get good feedback on something useful, something that will help you make informed decisions. It can pay dividends to carry out market analysis beforehand.

Beware of *loaded* terms too. For some the word *release* means a publicly available product. For others it means something to see and test out on a closed audience. Remember, it's possible to release in little bits to a closed group and then, once this has built up, do a proper public release. Don't make assumptions that everyone means the same thing! No one approach is right, so pick the one that works best for you.

## Adding features

Once the MVP or MVR or whatever you want to call the first delivery is out there, the *real* fun begins. Additional functionality or even specific features can be delivered in bite-sized chunks or packaged up into bigger releases. This is called *incremental delivery*.

Businesses love incremental delivery. No more waiting for years and years for one huge delivery containing every imaginable bell and whistle. The *agile* delivery preference is for little and often. There is a balancing act here once again but the business decides *what* and *when*. The smallest possible delivery is one solitary feature that can be validated through practical use.




## Getting more information

So far the items in the first release, our MVP, are very lightweight and we'll need to get more detail on them. The reason they're high level at this point is because we're trying to formulate ideas and it's wasteful to elaborate requirements that may not go on to be used. That was more than enough to define the big picture and agree the MVP but not enough to get on with the work itself.

The next step is to get more information on everything earmarked for the first release. The classic tool for capturing further details is the *user story*.

### Tell me a story

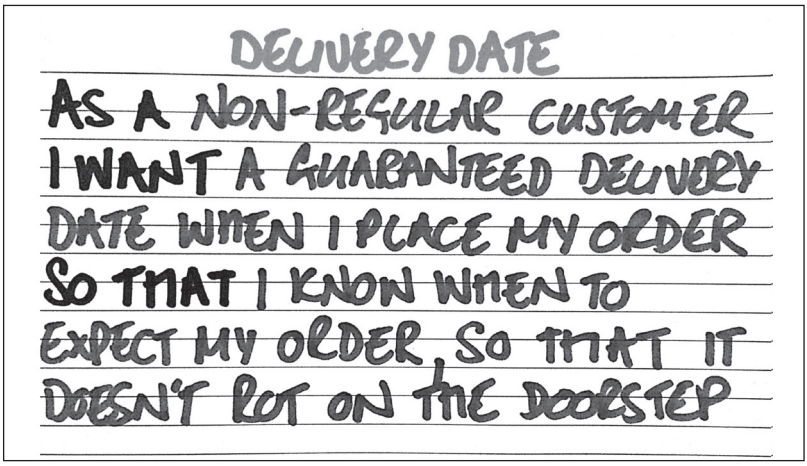
User stories are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple format:

As a <type of user>, I want <some goal> so that <reason>.

| | |
|---|---|
| Title | Assign a clear and concise title. This is a great way to summarise, index and search for stories. |
| As a <type of user> | We need to know who the end user will be. Who this feature is for. |
| I want <some goal> | What is the functionality the end user wants? Describe the 'what' not the 'how'. |
| So that <reason> | What is the reason for needing this feature (with some kind of business or customer benefit here)? |

A user story is a very high-level definition, containing just enough information so that the team can produce a reasonable estimate of the effort to do the actual work. User stories are often written on index cards or sticky notes, stored and arranged on walls or wherever there's a space to facilitate planning and further discussion. They shift the focus from writing reams about features to discussing them.

A user story isn't a requirement. It is defined as *a reminder to have a conversation* and these discussions are often more important than whatever's written. It is these dialogues that



spark the most important thinking points about the *requirement*. Remember, anything written today that won't get started on for a while can go stale. However, if we just gather enough detail to have a *meaningful* conversation, the reminder stays fresh and time isn't wasted penning copious detail. Win–win.

## Pin down acceptance criteria

*How do we know when we are done?* It's a key question and the first one that should be asked when starting a conversation prompted by a user story. In order to work efficiently, it's important to know when to stop. What are the boundaries of the work? How much do we do for it to be accepted? How will we avoid over-egging or gold plating the requirements? For a user story to be truly done it needs *acceptance criteria* – the prerequisites that have to be met for a story to be assessed as complete.

Acceptance criteria can take many forms, from simple *conditions of satisfaction* all the way through to rigorous and very exact checks. For the purposes of getting *agile*, simple binary statements written in plain language are the best place to start. Let's take the *delivery date user story* a little further by writing some simple acceptance tests for it:

- The delivery date will always be the next working day.
- The delivery date will be on a Monday if the order is placed on a Saturday.
- The delivery date cut-off time for orders will be 3pm.
- There will be email confirmation of the delivery date.
- All product lines have the same delivery date rules.
- We can never specify a time of delivery, only the day.
- The user can leave a note for the delivery driver.
- The anticipated delivery day will be shown on the screen when ordering.

Acceptance criteria written collaboratively by the team is the most likely way to cover all the angles. Led by the *Product Owner* or business representative, the team can talk through the stories, remove ambiguity and pin down the end game. These sessions are the best way to bring about team alignment and they don't need to be long, or laborious. They can be done *just in time* to start work; the aim is to start with the end in mind!

As a by-product, acceptance criteria provide a useful measurement to report progress against. Frequently, business confidence is undermined though being vague: '*I think we're nearly done*' or '*I feel we're on track*'. So these checks are an aid to being more precise by providing specific and measurable milestones: *we're halfway through the acceptance criteria*. This type of gauge will be easily achievable if the checks are properly formatted and alarm bells should be ringing if not.

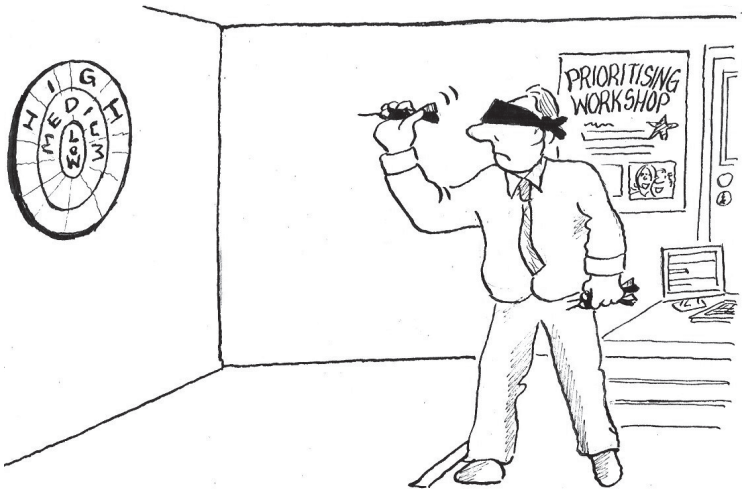
## Splitting stories

Sometimes, too much of a *good conversation* generates an over-abundance of material. Don't worry, this is a good thing – don't stop the dialogue, capture it all. Some ideas may be not appropriate for the story you're working through or may be too advanced for it. Not to worry, as once captured they can be filtered. Some of it can be added to other more appropriate stories.

Others may call for a new story to be written and this is known as *splitting a story*. A common example is where the Product Owner sees some of the acceptance criteria as unnecessary for the time being and wants to create a new story for the extended features – to be reminded to talk about them in the future. Once the new story is written, it can just be prioritised into the backlog along with everything else.

Keeping user stories to a manageable size is important. The more complex a story is, the more risk of something going horribly

wrong. Huge reams of acceptance criteria are an indicator that a story has gone that way and *must* be split. There are times when this needs to happen multiple times and it's a legitimate way of breaking work down into reasonable-sized work packages.




## Yes, size matters

Now we have a vision, a backlog, an MVP and some well-written user stories complete with acceptance criteria. Great stuff. The next step to ask then is: *how much work is all that?* Traditionally project managers or specialist estimators carry out this task and then throw their predictions over the fence. But on an *agile* project the team, the people actually doing the work, produce the estimates. Apart from the huge benefit of more reliable projections, there's the advantage of getting team buy-in.

Of course the size of each user story is required to predict when the MVP will initially be delivered and when the subsequent other features. But this is also a way of being forewarned

of potential problems with individual pieces of work – head scratching, big intakes of breath and shaking heads are sure-fire signs of trouble ahead.

There are several estimating techniques well suited to *agile* projects and the following are worth considering:

- **T-shirt sizing.** Assign S, M, L, XL and XXL tags to everything and gauge roughly how much work each size involves. Easy to use and a good starter-for-ten but can lack precision.
- **Story pointing.** Use a Fibonacci scale for all the pieces of work in hand – for example from 1 to 100 points – where 1 is easy-peasy and 100 a raised eyebrow moment. Takes some getting used to but there are many devoted fans.
- **Affinity prioritisation**. Sequence all of the stories in order of relative size – shuffling stories into smallest to largest order – and assign 1 to the smallest and the relative size to the next biggest and so on. Useful for getting to grips with the MVP.

### *The easy road to ropey estimates*

- ✔ Start with an incomplete backlog.
- ✔ Vague *user stories* are a must.
- ✔ Big, complex, multifunctional stories add spice.
- ✔ Be optimistic and hope for the best.
- ✔ If in doubt just guess.
- ✔ Don't worry... estimates are usually wildly wrong anyway.

## Less is more

Outside of the *agile* world there's normally a cat and mouse game played at the start of a project. The business team know

instinctively they have to ask for everything under the sun because there's only one shot at getting *most* of what they need. They also know that when things start to go pear shaped – as they regularly do in some form or other – the deliverables are going to be pared back. So it's better to ask for the kitchen sink to improve the negotiating position.

Project teams know this goes on of course and are happy to have wiggle room built in for when the times get tough. The problem is that when the squeeze hits, many of the bells and whistles have already been delivered and it's too late to recapture that poorly invested effort. When the budget dries up or time runs out, the remaining work includes many non-negotiable essentials: for example, within a house renovation project, having a very high-spec kitchen with all the latest lighting and gizmos when the bathroom is still a bare shell.

Of course it's common sense to start by delivering a barebones solution and build from there but there needs to be an understanding that the plug isn't pulled after initial delivery. There needs to be faith that there will be incremental deliveries from there on, building and honing the final product. Although much depends on trust, the whole ethos of an *agile* project is based on this premise, so nothing can go wrong unless the whole set-up is a total sham – which even *agile* can't insure against.

The *MVP* is the absolute minimum required just to get going. Every feature and nuance is non-negotiable without any nice-to-haves. The litmus test for anything on the first to-do list is that the whole MVP would be unusable without its inclusion. In practice, it doesn't matter too much if a couple of minor bits 'n pieces sneak in as long as the traditional gold plating doesn't happen. The objective is to end up with a lean, mean set of requirements that can be delivered quickly.

Once business teams and customers go through the loop, they immediately get how much better this works for them. The reduced time-to-market is a big winner and the key to

competitive advantage. Plus, in practice it's usually very hard to predict the optimum final outcome and much easier to add to a working product. It is interesting to look back on any nice-to-haves after the first delivery is in place as normally other more important features come into contention. There's nothing stopping the team from having a long list of features waiting for the production line.

**brilliant tip**

Finance teams aren't usually leading the charge to go *agile* but they quickly get on board when the benefits are explained properly. A reduced time-to-market, incremental delivery and an early return on investment is music to their ears. The end to ever-spiralling budgets alone is enough to win them over. Very powerful allies.

## Risk and expectation management

Done properly, there's no risk of *typical* problems occurring on an *agile* project because risk management and mitigation is built into the framework – greater all-round visibility with a diverse team of specialists on hand helps enormously. The main risk with *agile* is going off-piste in some way or another:

- **Don't deviate from tried and tested practices.** Many people try to change too much, too frequently. Stick with the guidelines and make adjustments one at a time. If your changes don't work, drop them!
- **Communicate, communicate and communicate.** Bad communication is the root of all evil. Leaving information out is as misleading as giving bad information. Use the *backlogs* as the focus for regularly having the right conversations at the right time.

- **Avoid large work items.** The larger requirements are, the harder they are to understand. Break down any big items into smaller, more manageable chunks.
- **Keep talking.** The best way to manage *agile* risk is by continually having meaningful conversations with those people around you. Let them all talk!


**tip**

One of the biggest risks of failure is in only paying lip service to the *agile* framework and becoming an ordinary project in disguise.

*Be wary of sheep in wolf's clothing.*

## Managing the backlog

It's hard to over-emphasise how important the backlog is. It is the cornerstone of your project. However, a good backlog can go bad very quickly if it's left unattended. The backlog *has* to be living, breathing and attention-seeking. Used well they're brilliant at helping to demystify what is coming up, helping communication with others, reducing risk and managing expectations. If they are neglected, they become a time-consuming distraction that sends people off course fast. Keep the *backlog* up to date!

At the start of a project, a backlog is usually full of functional requirements and features written as user stories. As the project moves on it will become filled with other items and a user story is just one form of *product backlog item*. Backlogs need to make all work visible and that includes faults, non-functional requirements, improvements, enhancements, new feature requests – everything. Get them all written down and blend them in with everything else, ordering them by *business value* just as always.

The backlog belongs to the Product Owner, who remains accountable for it at all times. As such they should be constantly *refining* it by using it as the main focal point for discussions with all interested parties. Keeping the backlog up to date is hard work, but the benefit pays off through the visibility it provides and the conversations it initiates. Transparency builds trust on projects, and by far the best way to achieve this is to make your continually refined backlog visible to all.

The best way to make sure that your backlog is up to date is to get the team and Product Owner to look at it every day. This can be part of a daily routine or part of any exercise where the team talk about what they're doing. Sometimes, the Product Owner spends a lot of time refining the backlog, sometimes just minutes. The important thing is that it's being used and referred to regularly.

**brilliant tip**

A static backlog is a sure-fire sign of trouble. Keep an eye out for prolonged inactivity.

## Creating the right environment

Success on an *agile* project is more about the individuals and the interactions between them working than anything else. It's hard to get people working together efficiently and their environment is crucial in promoting effective communications. The most successful *agile* teams are product focused, sit together and have easy access to their Product Owner. Plus they're in sight of both the team task board and most crucially the *product backlog*. A team that sits together has fewer obstacles in the way of communicating, interacting or even just building rapport by chatting about the weekend.

Let's be realistic though. It's easy to say we should all sit together, laugh, work, be funny and good looking – but in reality people make long commutes, companies have offices in different cities and partnerships with off-site teams too. Whatever the circumstances, visibility, transparency, communication and interactions collaboration is key and quite often we need help to achieve this. Video conferencing, electronic task boards on giant touch-screen TVs, conference calling are never brilliant but *always* better than nothing. If, and it's a big if, there's a well-maintained and refined practical backlog, these tools can work.

There's no excuse for the team being uncertain about their objectives, what's coming up next and the part they have to play in it *whatever the physical circumstances*. If a team isn't communicating, it means something is seriously wrong and the bitter truth is teams usually perform badly as a consequence of a bad environment. We agree there's undisputed value in good communications and where there's a will there's a way.

If you don't know where you're going, you'll end up someplace else.

Yogi Berra

## The final word

Starting a project with a muddled vision plays right into the hands of the anti-*agile* brigade because, if so, it really will be a case of making it up as you go along. Defining a target of substance sets the scene for everything that follows. Even the best teams in the world can't deliver successfully if they're blindfolded from the beginning. Starting with the end in mind is harder than it sounds but it's non-negotiable.

Once that foundation stone is in place it's not all plain sailing through. Building a rock-steady *backlog* is the next key target, but luckily it's not an onerous task creating an exhaustive,

detailed list of requirements. No need to disappear for a lengthy period of time and produce reams of documentation supporting a gold-plated monolith. Knocking the backlog into shape is a relatively quick, shared experience and – can you believe it – great fun.

Finally, on an *agile* project the first delivery milestone isn't a life sentence. The initial target will be set at the bare minimum to get the business going and nothing more. This will happen quickly with the promise that from then on the deliveries will come fast and furious. No more promises of *mañana* for things to begin kicking into gear. Watch the business purring and getting ready to revel in a brave new *agile* world.

**brilliant recap**

- Start with a specific and *meaningful* vision; no waffle or vague targets.
- Business value is everything, so develop a shared understanding of what it really means.
- *Love thy backlog.* Develop a top quality backlog with rock-solid stories that can be understood by everyone.
- Pin down that MVP! Everything rests on getting it right and out there quickly.
- Always works out the *acceptance criteria* up front; start with the end in mind.