Delivery Lead Playbook

RKM Agile Consulting Ltd
https://rkmagile.co.uk

What is a Delivery Lead?

A Coach

A Problem Solver

Shielding team from unnecessary noise

A Facilitator

Identifying & helping to remove single points of failure

Being pragmatic & open minded to how the team will work & not over prescriptive

Encouraging the sharing of knowledge inside & outside of the team

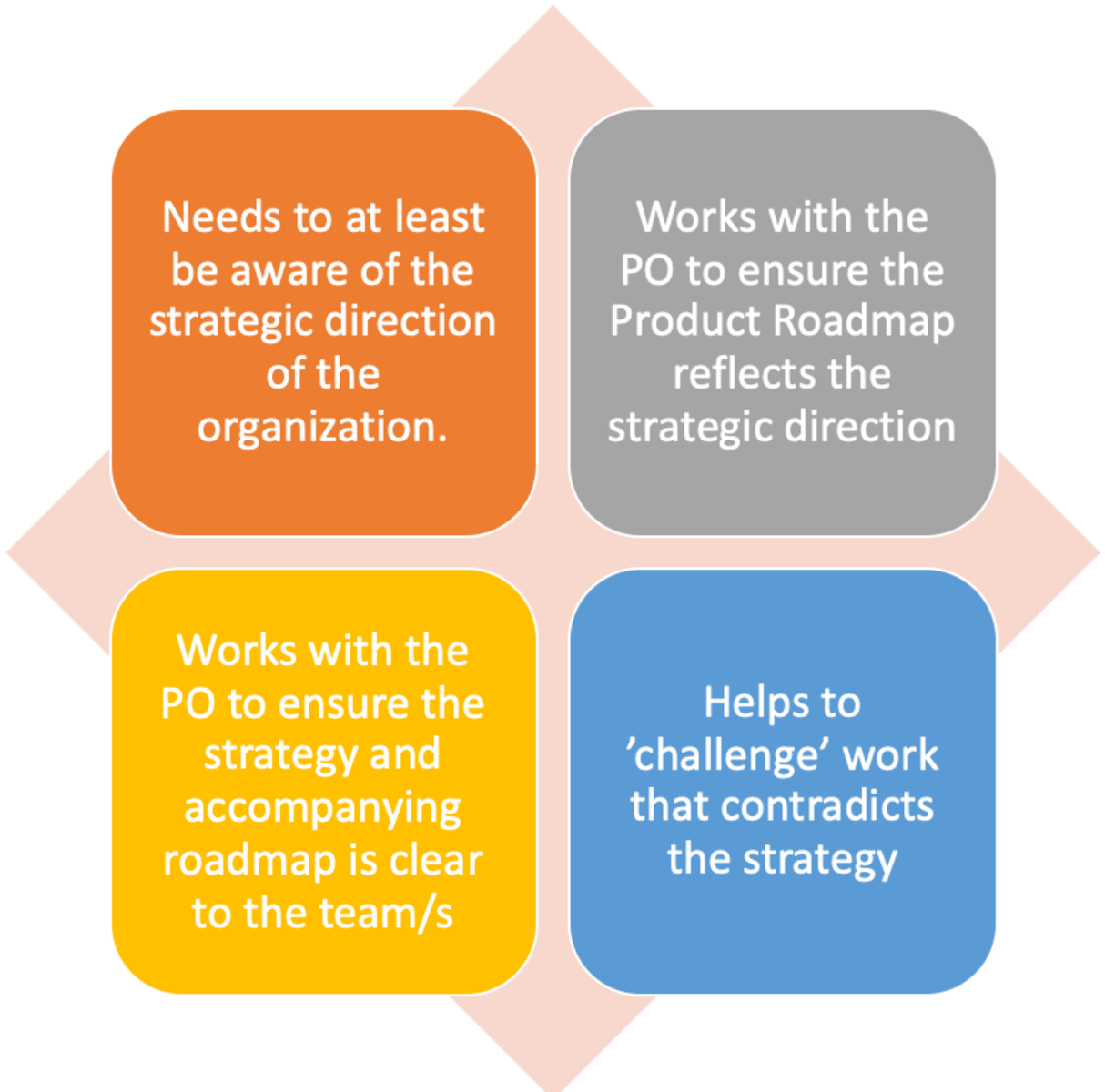
'Servant Leader'

Core Responsibilities

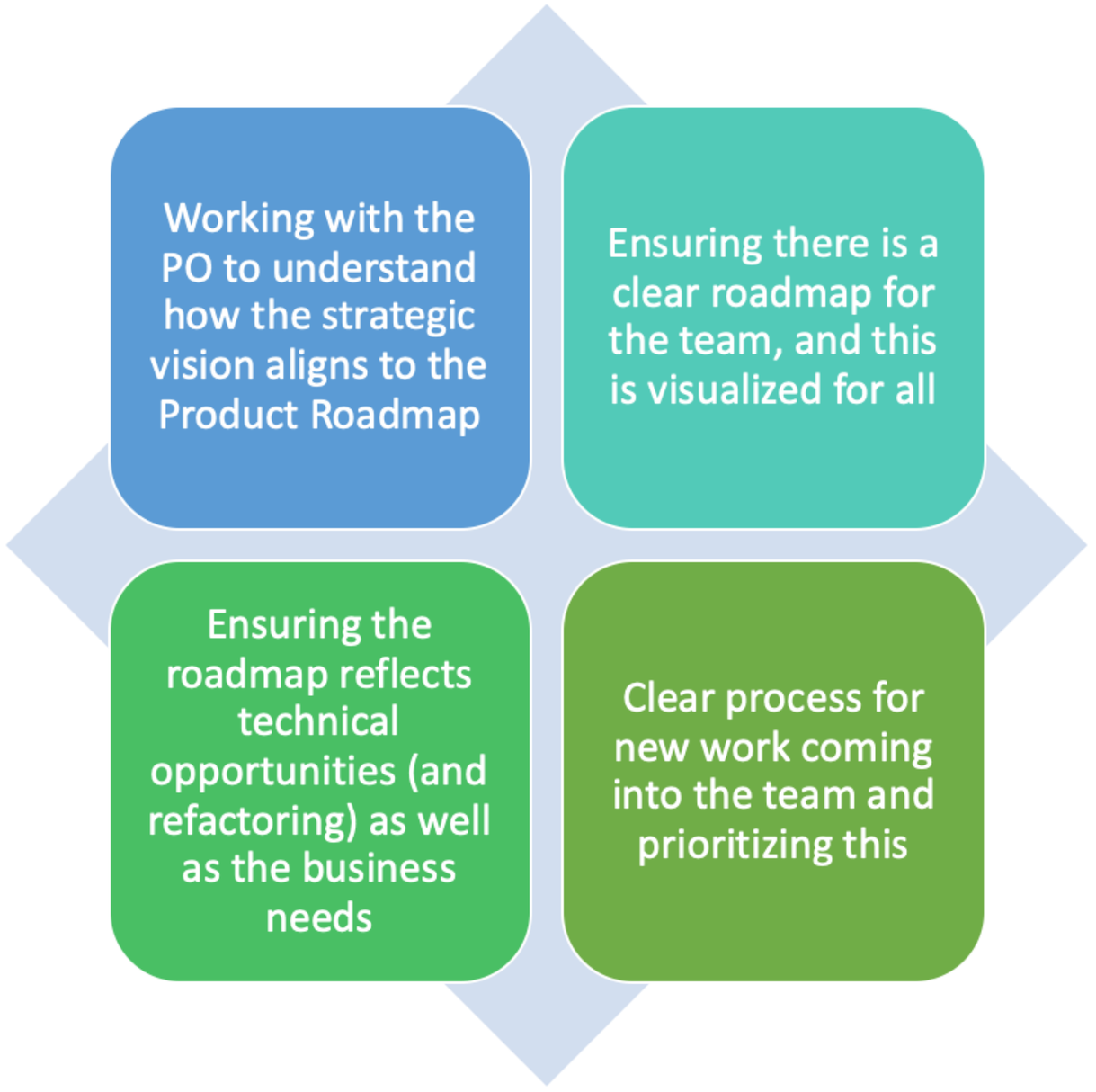
How the Delivery Lead contributes to the lifecycle of a product

- Strategic – what are the goals and how does the team align
- Product – the roadmap, its organization and priortisation
- The team backlog – the next set of things to focus on for the team and how relates to the bigger backlog
- The delivery – dev and qa
- The pathway to live/deployments
- How you support it when live

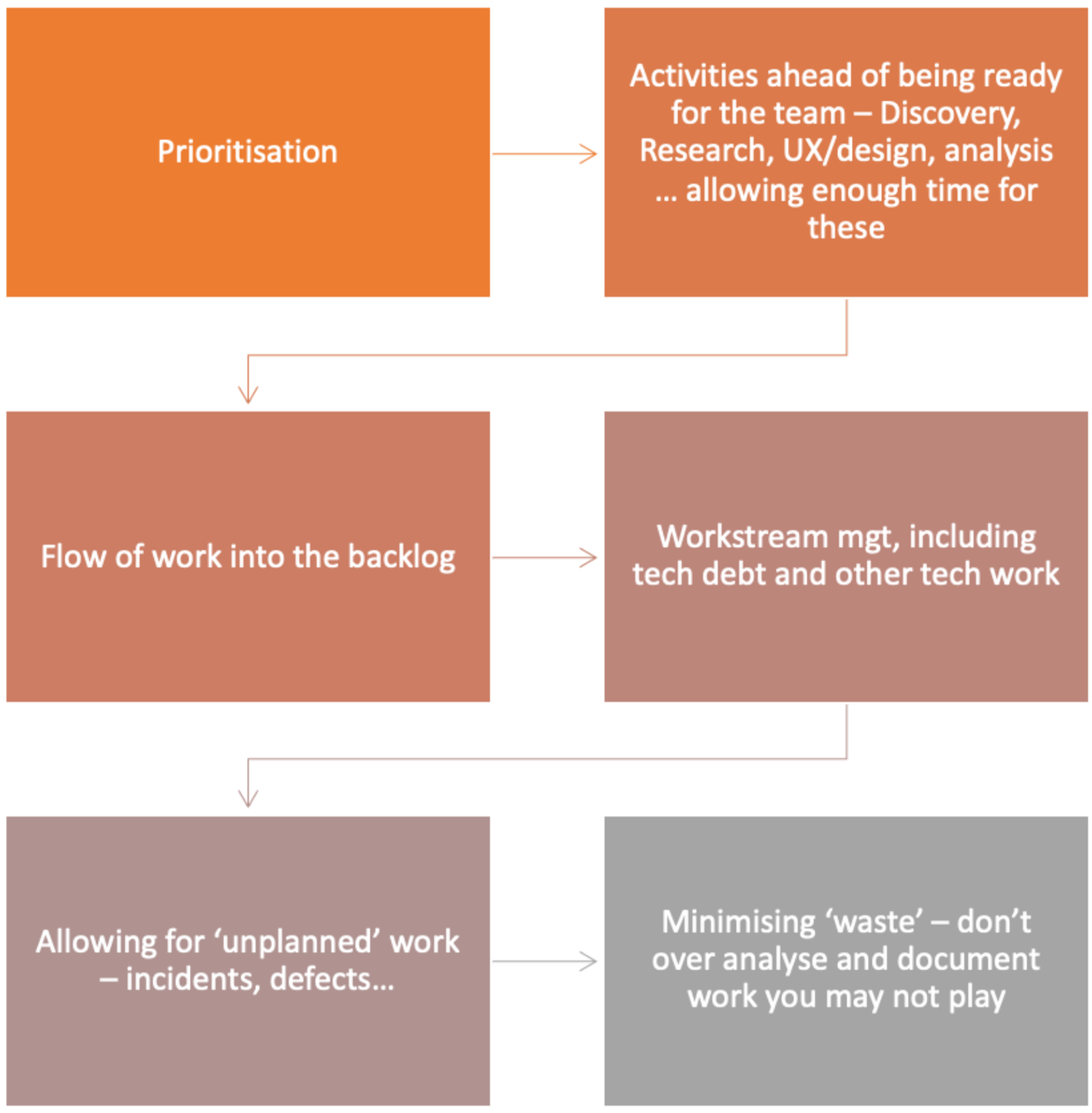
How the Delivery Lead contributes to Strategy



How the Delivery Lead contributes to Product

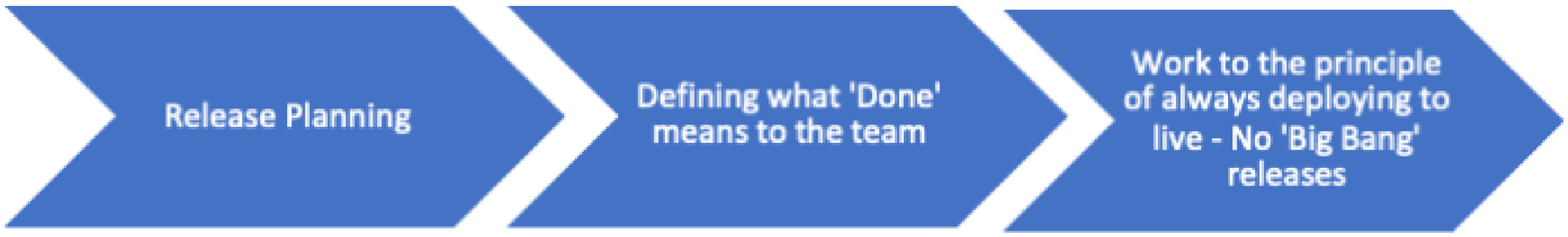


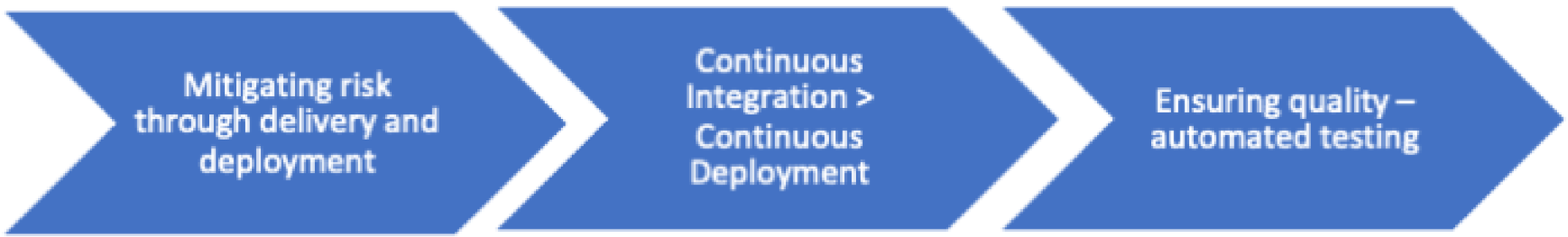
How the Delivery Lead contributes to the Team Backlog



How the Delivery Lead contributes to the Delivery

How the Delivery Lead contributes to the Pathway to Live





Supporting the Product

Ensuring have the ability to measure the change and make objective decisions

The ability to visualize live system performance

Being able to react to issues – and fix these through effective workstream and capacity management

Taking the subjectivity out of what you do

Using data: quantitive and qualitive to identify improvements

Team feedback: Retros

Culture of Try, learn, adapt…

Fostering the agile community

Knowledge sharing in the team and wider community

Each function has its own group meet

Cross functional meets based on areas of interest

Need to be regular and scheduled

Need to have an agenda to stay current and be of value

Setting some form of 'Terms of reference' so the group have defined their purpose is always beneficial

The communities should cover:

Development, QA, Analysis and Product, Delivery Mgt

Should look at encouraging Dev Ops involvement – they are are active and valuable part of the tech community and sharing knowledge and practices can be hugely beneficial

Key Ceremonies

Backlog Planning

Owned by the PO - The DL should be involved and also making sure these sessions are taking place. The PO - working with the DL - will assess the next priorities and focus for the team

Sprint Planning

Set up and facilitate planning for a 2 week period - the team will assess how much work they feel they can achieve in that period. The DL will support this by utilising metrics from previous sprints & looking at capacity in the team

Retros

The DL will ensure a regular retro will take place to provide the forum for the team to identify and discuss the areas that aren't working so well and things they wish to do to improve working. The DL will also ensure that any actions identified are recorded, assigned and actioned

Showcase / Demo

The team should be open & transparent with the work they have done. Running regular showcases helps here. If you use sprints then holding these at the end of the sprint is a great natural pint to hold these. The DL ensures these take place and the appropriate people are involved. Often the PO and other team members actually run these but the DL ensures they are ready and take place

RKM Agile Consulting Ltd

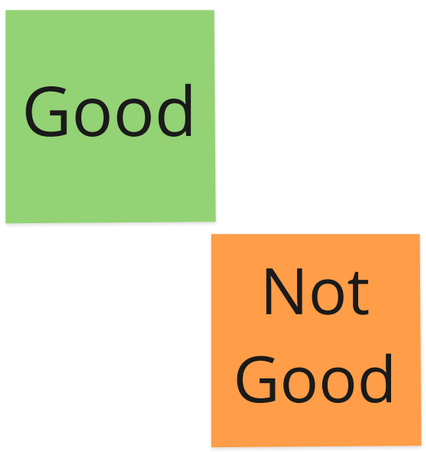
Retrospectives

Typical format:

What's gone well, not so well > Improvements to focus on

Give the team time to write down areas for the 1st 2 areas, then vote on the areas they most want to discuss further and identify improvements

1 hour should be more than enough for a retro, unless you know there are a number of complex issues that need addressing - in which case allow more time (e.g. 1.5 hours). More than this and the quality of the output will likely reduce



Remote Tools:

Its always recommended to keep things as simple as possible

Using an online whiteboard such as MIRO, or Teams also has a board.

Its even possible to use JIRA to record your retros

Retro styles:
There are many different examples and types of retro to try out. Try to find which works best for the team. Sometimes it's worth mixing things up and trying a different format

One resource for retros can be found here: https://trello.com/b/40BwQg57/retrospective-techniques-for-coaches-scrum-masters-and-other-facilitators

Remote vs Face to Face:

With the improvement in the availability and choice of online boards running a retro remotely shouldn't pose a barrier like it used to.

Remote: Consider allowing people to submit ideas ahead of the retro so you have time to order them and focus on the improvements

Ensure the retro adds value:

A retro is only valuable if you identify useful, meaningful improvements and actions and test are actually addressed. Assigning actions as work in your backlog is one option if people are struggling to make time and commit to actions

Sprint Planning

- Define your teams planning and work cadence - typically 2 weeks
- Include whole team (Dev, QA, PO, UX, DL and anyone else working within the team to deliver the work)
- Should have a set of candidate stories for the next sprint - analysed and ready to be discussed with the team. These may already have been 'sized'.
- Understand the capacity in the team (holidays, time not working in the team) and allow for this in planning
- Look at previous metrics from earlier sprints - e.g. no. of stories completed, points completed, work max of story/bug/incident etc..

- Determine how long you will need for the planning. The more work done ahead of the session to have stories ready and reviewed will reduce this. Likely to need between 1 and 2 hours. M ore than this reduces the output from the team and indicates that more pre work is required next time
- BA/PO will pitch each sprint caudate/story. The team will assess the size and will ask questions where needed to clarify. The team will determine if they will bring that story into the next sprint.
- The DL should be challenging if it looks like the team are either over or hugely under committing to the work for the next sprint
- Set a goal for your next sprint - give clarity to everyone what the outcomes will give

Key Artefacts

TEAM WORK MGT (PLANNING AND PROGRESS) – E.G. JIRA

RAID (RISKS, ISSUES, ASSUMPTIONS, DEPENDENCIES)

TEAM PROGRESS – BURN UP/ BURN DOWN

RETRO OUTPUTS

GO LIVE CHECKLIST (OR 'SWITCH ON')

SIZING/REL SIZING

Work Management and tools

There are many tools available and the principles for how you use the tool should be the same, i.e.

- Set up a project that works for your team
- Keep things simple - the more complex the workflow the less likely people will be able to easily use it
- Use the reporting function of the tool to track metrics that are meaningful to you and the team, e.g.
    - Burn Up (great for teams operating with kanban and supporting existing live products) or Burn Down (Burn downs are great for showing sprint progress)
    - Cumulative flow (helps to identify potential bottlenecks
    - Velocity reporting tracking work you have committed to vs work completed)
- Build your reporting dashboards, showing the useful data that others in the team will benefit from, e.g.:
    - Bug counts
    - Average time in status (useful in helping the team determine if stories have been sliced to the right level, or looking for potential bottlenecks

Risks, Issues and Depenencies management

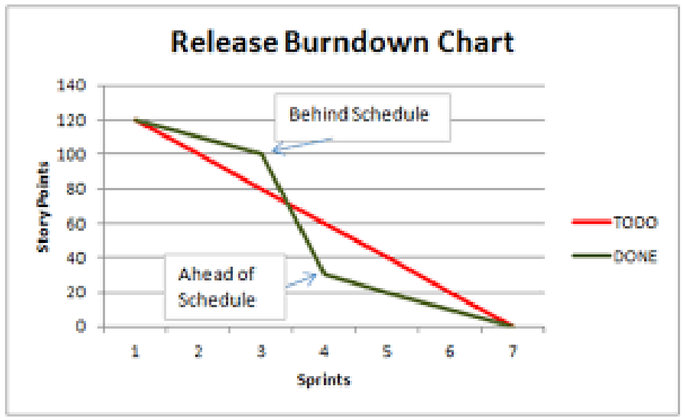
Having a way to log and track your key risks and to priories these and agree mitigation plans – its always helpful to consider the likelihood and impact in how you prioritise these and action them

Dependency management is key to the role. Identfying these, logging them and looking at how to resolve them is key. Ideally your strategy will be to remove the dependency

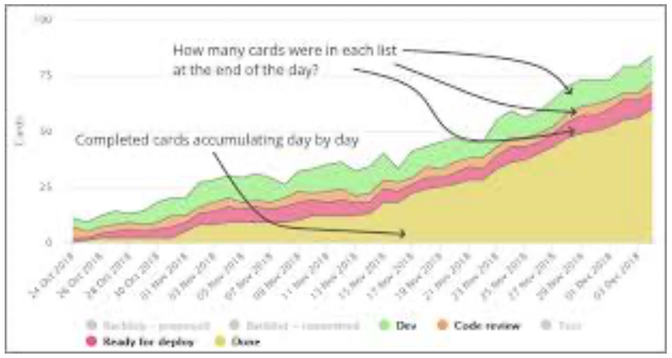
Tracking Progress

Tracking and reporting progress is key to helping you and the team understand if you are on track or not. They can also help highlight impediments and areas to tackle

The Burn down is great for quickly visualising progress towards the end goal and show scope changes and impact - this can help control scope!



The Cumulative flow - more used in Kanban but can still be used to show efficiency. The aim is to keep consistently thin slices for work in progress whilst highlighting increases in work done - can also highlight where you may have issues (e.g. spending too long in dev or QA wait queues)



Tracking defects is important - are you seeing in crises here - why/what are the root causes to tackle?

Team Health - how the team feel is really important - track this and looks for trends/areas to tackle. A healthy/happy team will be more effective

Retro Outputs

Always document the output for all to see and clearly assign people to own actions

Keeping the output is important - we can look back for any systemic trends/areas that keep coming up

Always refer to the previous actions and progress towards these in each retro. If the team aren't closing off actions look for reason why - maybe assign as work on the board!

If working remotely, look for a good online whiteboard such as MIRO

The Go Live Checklist

We still need to plan how we release work. Most of the time this is an individual story that we can release directly to live and the customer sees it straight away. Sometimes we need to wait until a number of stories are ready before we switch on to customers

Don't confuse 'Go Live' with having a Big Bang deployment. Always aim to release to live for your definition of done. Use feature switches to hide from the customer until ready to switch on.

Checklists are useful where you have a significant piece of functionality or project that (even though it may be in live but not visible) you still need to ensure it is ready to be switched on to customers.

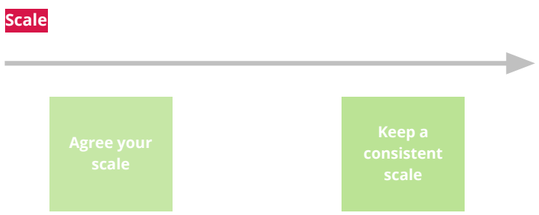
Example Checklist

This is just an example of the areas to consider. ~Not all will be applicable and other considerations may need to be added depending upon the work.

AREA	REQUIREMENT	UPDATES	APPLICABLE JIRA TICKET	WHO OWNS	STATUS
Security	Pen Test completed & required actions completed				
	Security Sign off (if applicable)				
	Privacy /GDPR Sign off (if applicable)				
Legal	Lega/regulatory texts & links in place & signed off				
System Performance	Performance Testing signed off				
Copy	Copy created				
Search Engine & Social Media Optimisation	SEO tagging in place, tested & signed off				
	Paid Searches ok & signed off				
	Any Facebook tagging required in place/tested				
Analytics	Analytics in place, tested and signed off				
	Benchmark analytics (if applicable) agreed and in place				
Communications	Agreed comms channels in place				
Roll out plan	Agreed audience				
	Dates/Times agreed				
	Any additional final checkpoint – 'Go/No Go'/sign off				
Service monitoring	Logging, monitoring and Alerting in place				
Training	Training materials in place				
	Support teams trained/updated				
Customer Services	Training materials provided Understand the new journey to level able to support customers				

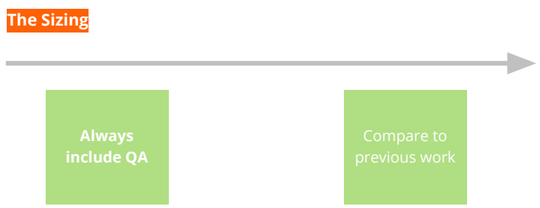
Sizing work

Sizing of stories
Its important to maintain a consistent approach to sizing - whatever scale you come up with, maintain this. Successful sizing of work requires teams to compare the story in question with other stories. When you already have delivered work to compare against this can be relatively easy, but when this isn't the case it can prove more challenging.



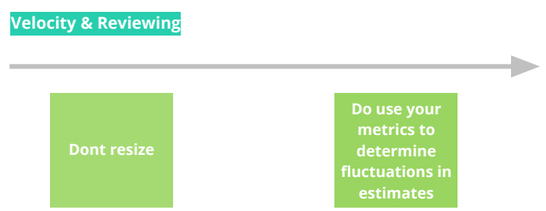
Relative Sizing
A quick and simple way to size work involves Relative Sizing. You may have a list of EPICs or even features. You will probably have 1 that you are able to break out further into stories - often the first one you will work on and the one you know most about. Using this as your baseline - you can very quickly work out the size of your backlog.

Sizing of stories
Choose you scale for sizing - e.g. Fibannaci
Always include development and QA effort in the size
Try to base either on complexity or on idea of size - but steer clear of thinking in terms of time
Compare to previous stories e.g. do we think story x is smaller/less complex, the same, larger..



Relative Sizing
1. Working through the EPICS, relative size them - e.g. EPIC 2 is bigger than EPIC 1, EPIC 3 is smaller than EPIC 1 etc.
2. Try to naturally group these into 3 to 4 groups
3. When finished apply T-shirt sizes > S, M, L, Xl
4. Break out 1 EPIC into stories. Size each using Fibonacci.
5. Add up the num per of points
6. Apply the following scale to your other EPICS: Sml > Med = 2xSml > Lrg = 2xMed
7. Now Total up all your EPICS

Sizing of stories
Don't be tempted to revise the estimate later
Size of team should not affect the estimate



Determine your expected velocity
1. When you don't know your velocity (new project, team etc..)
2. Size a selection of stories with team. Make a note of size but don't make visible on the story yet
3. Would recommend enough stories to cover a couple of EPICS if feasible
4. Now ask the team to pick a selection of stories that they feel they could achieve the team definition of done in a 2 week sprint
5. Note down the points total
6. Now place the stories back in the pile and ask the team to do the same again but being careful to not just select the same cards - again note down the total points
7. Do this between 3 and 5 times
8. Add up all points tallies and divide by no. of time you ran this exercise - this will give you average points/velocity
9. Caution - this is just an estimate and not scientific at this point - you will need to carefully monitor progress and adjust velocity accordingly (and therefore planning). Use of burn down or burn up will complement!

Online Collaboration tools - examples

White board	Instant Messaging	Meetings and comms
Miro - whiteboard	Slack	Teams

Work Management
JIRA
Target Process