banvien
Trusted
Technology
Partner

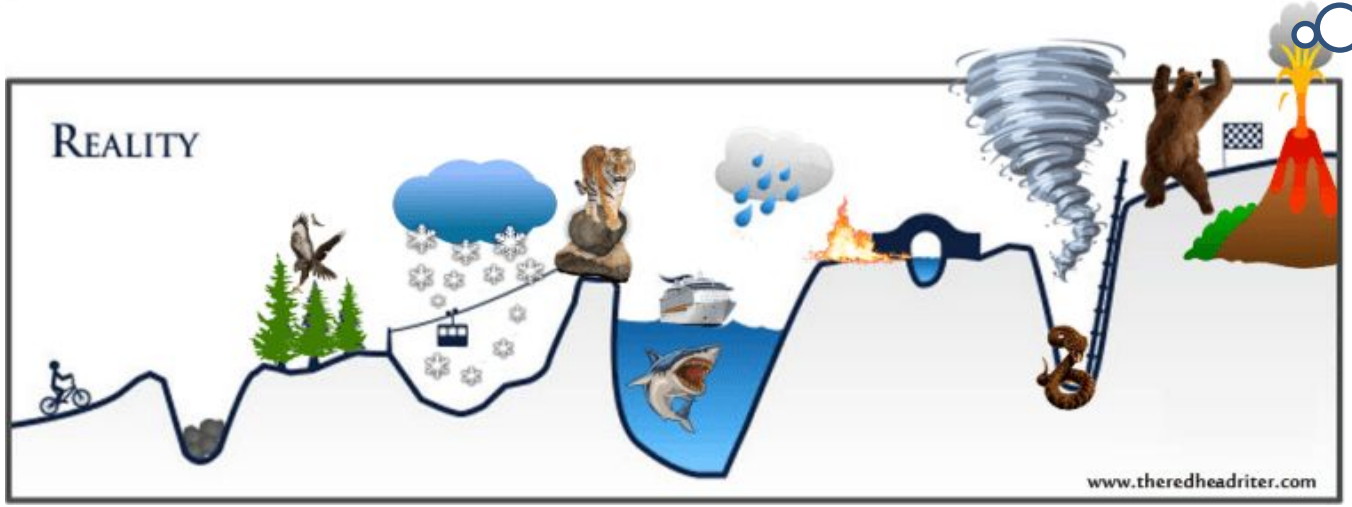
Agile - Scrum

Introducing Agile

WHAT DO WE MEAN BY AGILE DEVELOPMENT?

- Many people when asked to define agile development believe is a way to cut corners, write less documentation and reduce administration and management overhead.
  - This is not the case! Running an agile project can be more effort...

- Agile software development assumes that software development is a complex process that is hard to define and subject to change.
  - Agile software development embraces this fact to ensure:
    - Incremental development
    - Continual review
    - Improved customer satisfaction
  - There are many different agile processes such as Kanban or Scrum



Expecting change from the start and using a method that embraces this can enable us to make a better product.

WHAT DO WE MEAN BY BIG DESIGN UP FRONT? (BDUF)

- Traditionally software development has followed the big design up front model.
  - This is often tied into a **Waterfall** type software process where implementation can
  - only begin after detailed functional requirements and design documentation has been reviewed and approved.
  - Such documentation can be time consuming to make, hard to review effectively and difficult to for users to visualise the end product.
  - However some people believe this is worth the effort as avoiding defects and issues at the early stages of the project same money.
  - It is much more expensive to fix issues in the latter parts of the project.

THREE SIMPLE TRUTHS – EFFECTIVENESS OF REQUIREMENTS

- Problems with the BDUF projects come from 3 main issues. These issues always exist in all projects no matter how much effort we spend up front to avoid them:

It is impossible to gather all the requirements at the beginning of the project.

Whatever requirements you do gather are guaranteed to change.

There will always be more to do than time and money will allow.

Realising this from the project start should enable us to run projects in a way to limit these problems. Agile software methods, help us in this case.

KEEPING AGILE CONCEPTS SIMPLE

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

KEEPING AGILE CONCEPTS SIMPLE

1. Deliver something of value every week
   - *The fastest way to build customer confidence is to deliver working software.*

2. Break big problems down into smaller problems
   - Increments are short so you cannot do everything. This means complex problems must be broken down.

3. Make sure software you are delivering works
   - If you deliver software of value every week it must be tested during the increment it was developed in.

KEEPING AGILE CONCEPTS SIMPLE

4. You go looking for feedback
   - Regularly stop and ask the customer for feedback on the software.

5. Change the plan if change occurs
   - Things change on projects and when they do you must react quickly. Changing your plans is critical if changes occur.

6. Become accountable
   - When delivering something every week and showing the customer we become accountable. We are responsible for quality, schedule, expectations and budget.

Agile teams

HOW TO MAKE AN AGILE TEAM?

- Roles are harder to define on agile teams. Developers on the team are not just programmers. They must do everything that is needed to do the job.
- Analysis, Design, Implementation and Testing are all continuous activities that happen concurrently.
- Therefore these activities cannot happen in isolation. The people doing this work must work together daily throughout the project.
- There is one team that are accountable for the work.
- Quality is the team responsibility.
- There should be no “passing the responsibility”. E.g. “How did the QA team miss that bug?”

HOW TO MAKE AN AGILE TEAM?

- Team Location
  - If possible sit your team closely together in the same office.
    - In this situation questions are answered fast.
    - Problems fixed quickly when found.
    - Less problems interacting with each other.
    - Trust and mutual respect can be built quickly.

Introducing types of Agile methodologies

AGILE METHODOLOGIES

- There are numerous different agile software development methodologies, here is a selection from Wikipedia:
  - Adaptive software development (ASD)
  - Agile modeling
  - Agile unified process (AUP)
  - Disciplined agile delivery
  - Dynamic systems development method (DSDM)
  - Extreme programming (XP)
  - Feature-driven development (FDD)
  - Lean software development
  - Kanban
  - Rapid application development (RAD)
  - Scrum
  - Scrumban

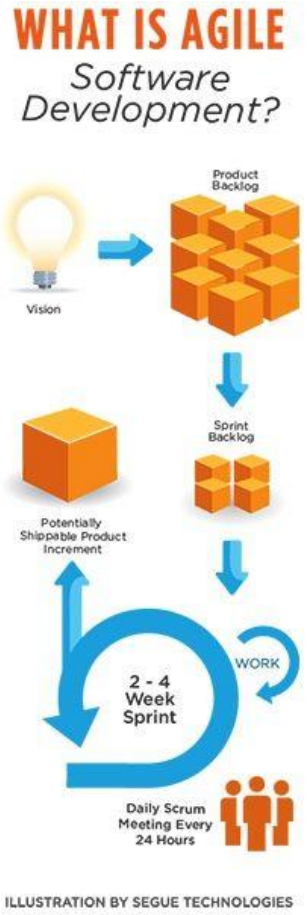
Introducing Scrum

The term '**Scrum**' was first used by Hirotaka Takeuchi and Ikujiro Nonaka in their ground-breaking 1986 paper, "The New New Product Development Game."

They borrowed the name from the game of rugby to stress the importance of working as a team in complex product development. ... That core was **named** '**Scrum**' by the authors.

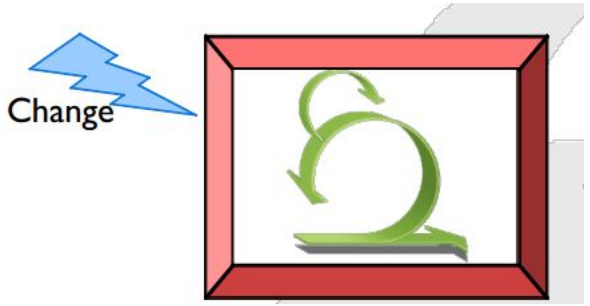
SCRUM

- Scrum is an agile software development process.
- The project is split into short cycles called “Sprints”.
- In these Sprints work is developed that should be ready for deployment.
  - All developments are documented, implemented and tested within each Sprint.
- Meaning at the end of each Sprint the software is working and can be demonstrated.
- The software is never in a “broken” state and always ready for review.

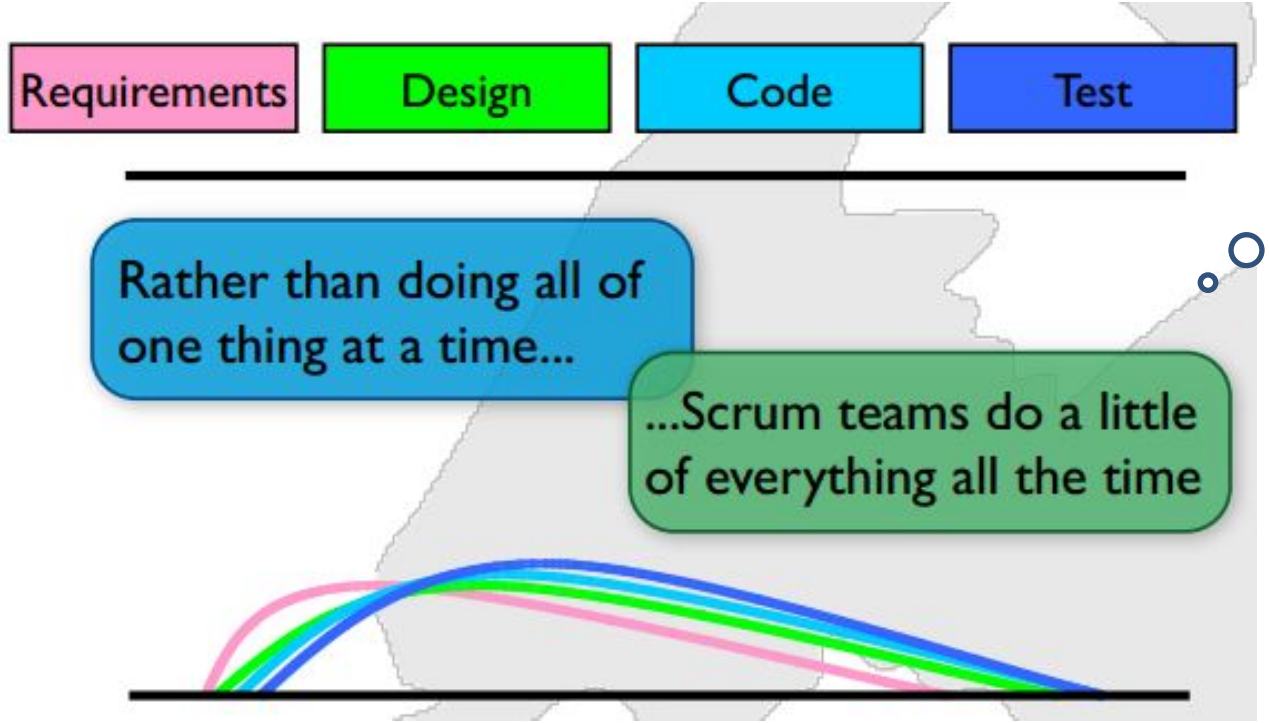


SPRINTS

- Sprints are suggested to be between 2 and 4 weeks long.
- Duration should be constant and not change. This gives constant rhythm to the development.
- Developers are most efficient when the work is identified up front, they do one job and are not interrupted this will give best efficiencies.
- So, plan Sprint durations around how long you can keep change out of a sprint.

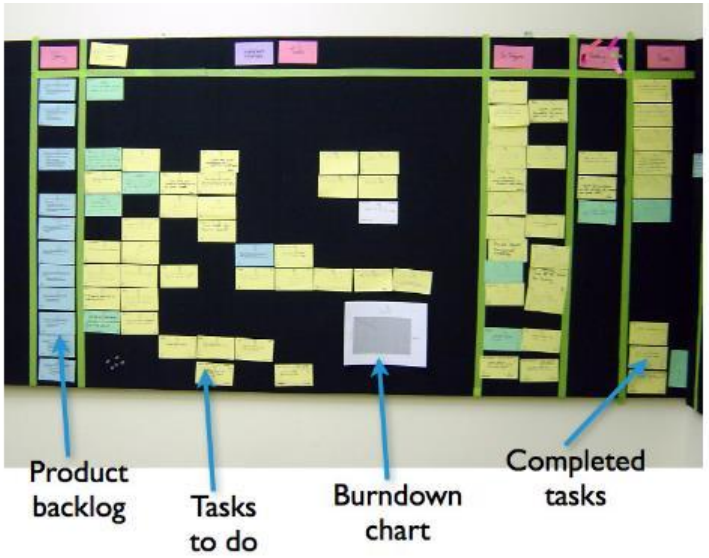


SEQUENTIAL DEVELOPMENT VERSUS SPRINT DEVELOPMENT



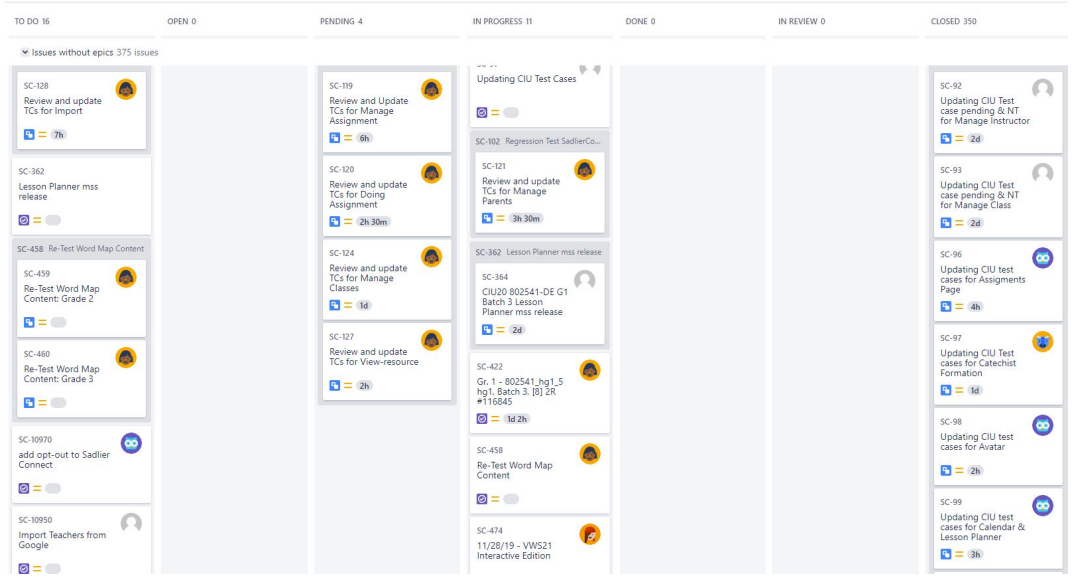
SCRUM BOARD

- Your project will have a Scrum board where you can visualize the status of the project.
- Before agile software tools were available teams often use post-it notes or cards on a whiteboard with the tasks written on them.
- Everything is visible to the team.
- It is easy to discuss and move tasks.
- The status easy to update.
- So the information is always current.



SCRUM BOARD

- JIRA has a built in Scrum board feature. It means everyone on the team can easily see their tasks and the overview of tasks.
- We can add quick filters to easily jump between views of the board.



BACKLOG

- The product backlog is the total desired work for the project. They are requirements.
- Ideally they are written to show priority and value to stakeholders and users of the product.
- If the backlog does not already exist, it can be initially created in the product planning meeting. (Ideally before)
- Prioritized by the Product Owner. (More on roles later) Reprioritized at the start of each sprint.
- Product Backlog requirements are chosen for development within the Sprint Planning meeting.
- The Product Owner continuously manages the backlog. Items can come in and out of the backlog depending on circumstance and priorities change. (Backlog grooming)

USE CASES OR “USER STORIES”

- User stories are often referred to as the work packages for the Scrum process.
- These are delimited functional stories or packages of work.
- E.g.
  - As this type of “person” I want to do “something” so I need “this” feature to do it.
  - As a Teacher I want a report to show the score of my students in SadlierConnect.
- These stories will be implemented in the sprints.
- It is important for each story to define what is meant by done. i.e. the end of sprint target.

DEFINITION OF DONE

- It is important that the team has a definition of “done”.
- This will be the target for all work taking place within a sprint.
  - E.g. for SadlierConnect this means:
    - Design reviewed.
    - Code completed.
      - Code created, reviewed and committed.
    - Unit tests created, committed and passing.
    - Developer functional tests created and passing.

DEFINITION OF DONE

- The definition of done is different depending on the project being worked on.
- E.g. The definition of done for sample software would be very different than that defined for product software.
- This definition should be defined in the Project Plan and agreed with the quality team.
- The Definition of Done applies to all items in the project.
- We must ensure items are only classified as completed when the Definition of Done condition is met.

- This avoids a build up of technical debt that would need to be completed before a release.

EPICS

- We use the term “epic” for a work product that will take multiple sprints to complete.
- An epic can be considered a large user story.
- An epic has sub-issues which can then be placed on each sprint during sprint planning.
- In addition the business value is not delivered until all parts of the epic are completed.

SCRUM FRAMEWORK

Scrum Team

Product Owner
Scrum Master
Development Team

Scrum Events

The Sprint
Sprint Planning
Sprint Review
Sprint Retrospective
Daily Scrum

Scrum Artifacts

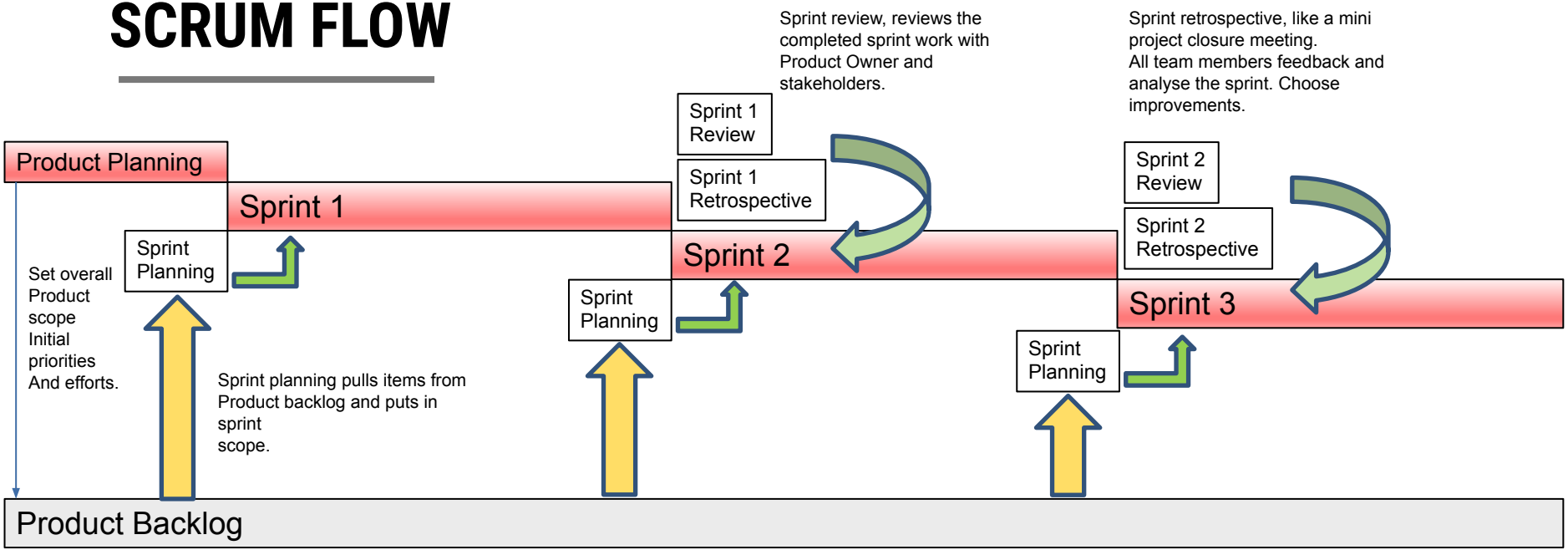
Product Backlog
Sprint Backlog
Increment

PARTICIPANTS IN A SCRUM PROCESS

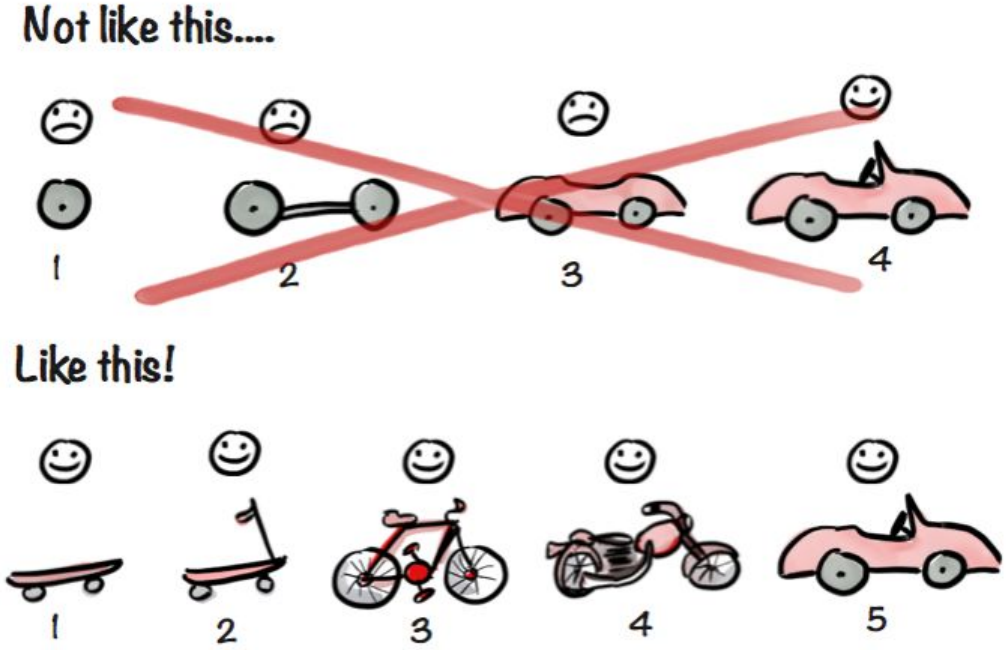
Customer or Stakeholder

Product Owner	Scrum master	Everyone else
Defines the features of the product – improve user stories	Responsible for day to day running of Scrum process and values	Cross functional teams – all the skills required to deliver
Makes scope versus schedule decisions	Removes issues facing the team	Self-organizing – can prioritize their sprint tasks and plan them
Responsible for project budget issues	Coaches the team to get best performance	Should be fully allocated to project
Prioritizes the backlog	Helps improve productivity anyway possible	Changes only made to the team in-between sprints
Adjust features and priority every sprint	Enables close team cooperation across all roles	
Accept or reject the work products	Shield the team from external factors	

SCRUM FLOW


Sprint review, reviews the completed sprint work with Product Owner and stakeholders.
Sprint retrospective, like a mini project closure meeting.
All team members feedback and analyse the sprint. Choose improvements.
Sprint 1 Review
Sprint 1 Retrospective
Sprint 2 Review
Sprint 2 Retrospective
Product Planning
Sprint 1
Sprint 2
Sprint 3
Sprint Planning
Sprint Planning
Sprint Planning
Set overall Product scope Initial priorities And efforts.
Sprint planning pulls items from Product backlog and puts in sprint scope.
Product Backlog

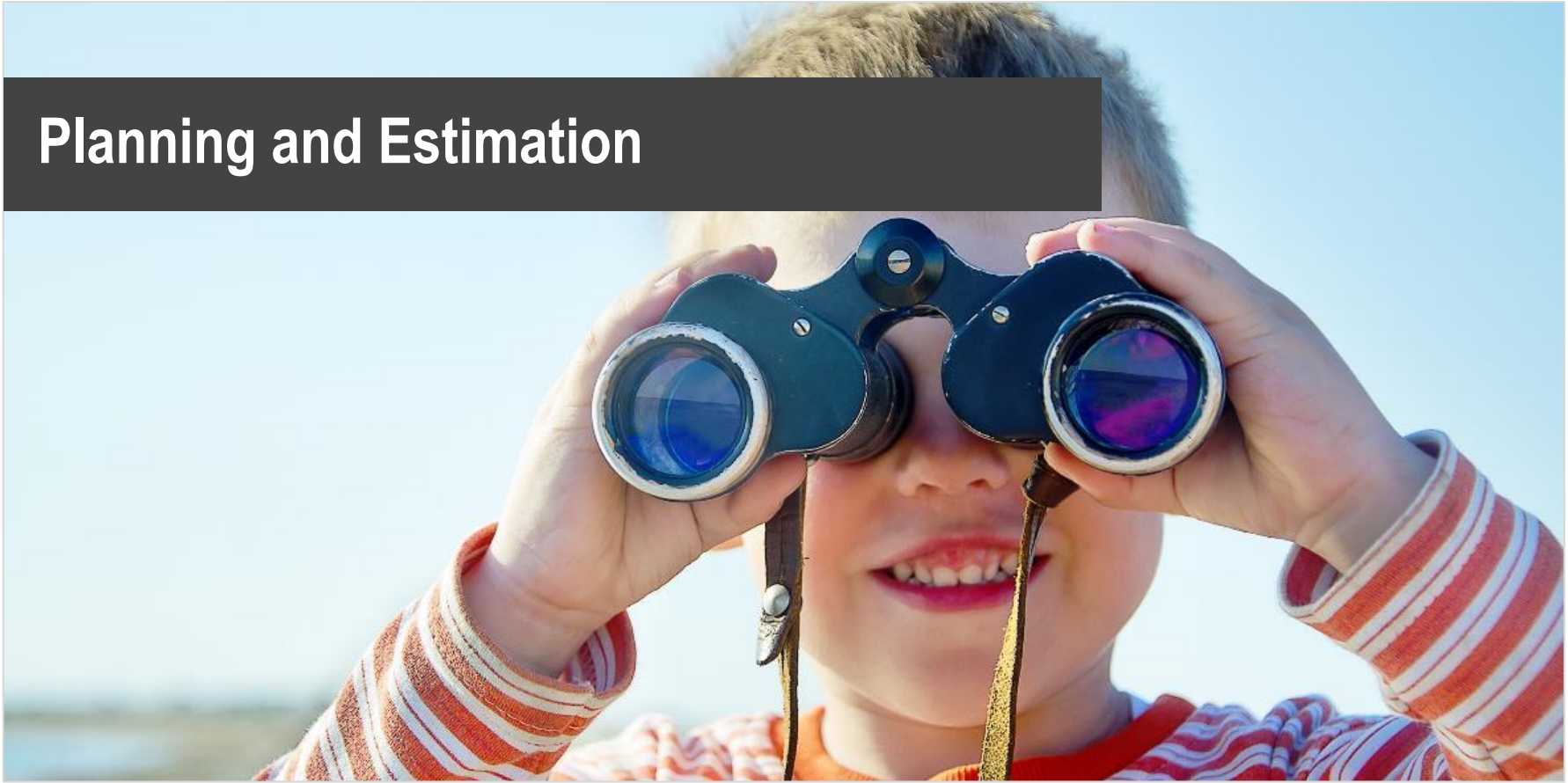
USING THE ITERATIVE APPROACH TO CREATE A STRONGER PRODUCT WE ACTUALLY WANT…



TEAM SIZES

- Ideally Scrum works best with team sizes between 5 and 9.
- A few engineers more than this can still be achieved.
- Too many and the key meetings take too long and efficiency is reduced.
- For teams bigger than this teams can be separated into functional sub-teams.
  - Think of this as teams of teams.
- For smaller teams a full Scrum process or framework is unlikely to work well.

Planning and Estimation

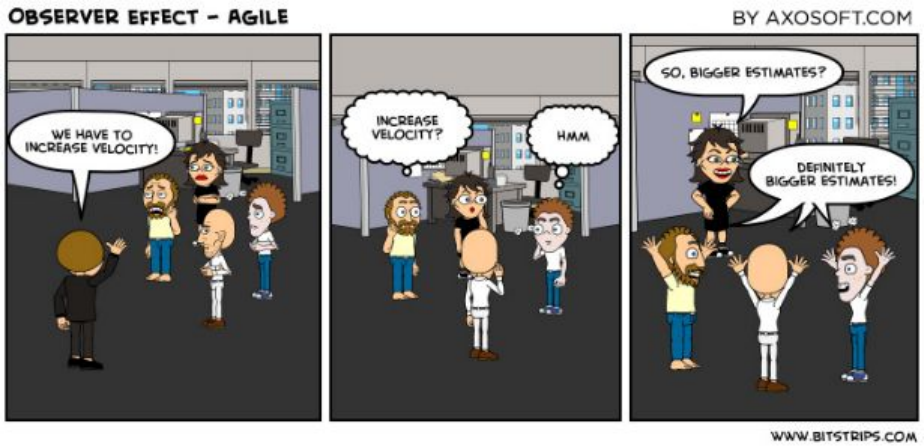


SIMPLE AGILE PLANNING

- In simple terms, agile planning is nothing more than measuring the speed a team can turn user stories into working, production-ready software.
- We then use this measurement to find out the estimated completion date.

SPEED = VELOCITY

- The team's velocity refers to the amount of work the team can achieve during a single sprint.
- This is an important metric to measure to gauge efficiency and improve planning.
- Never measure individual performance.
- Use as a guide – over a period of time.



VELOCITY

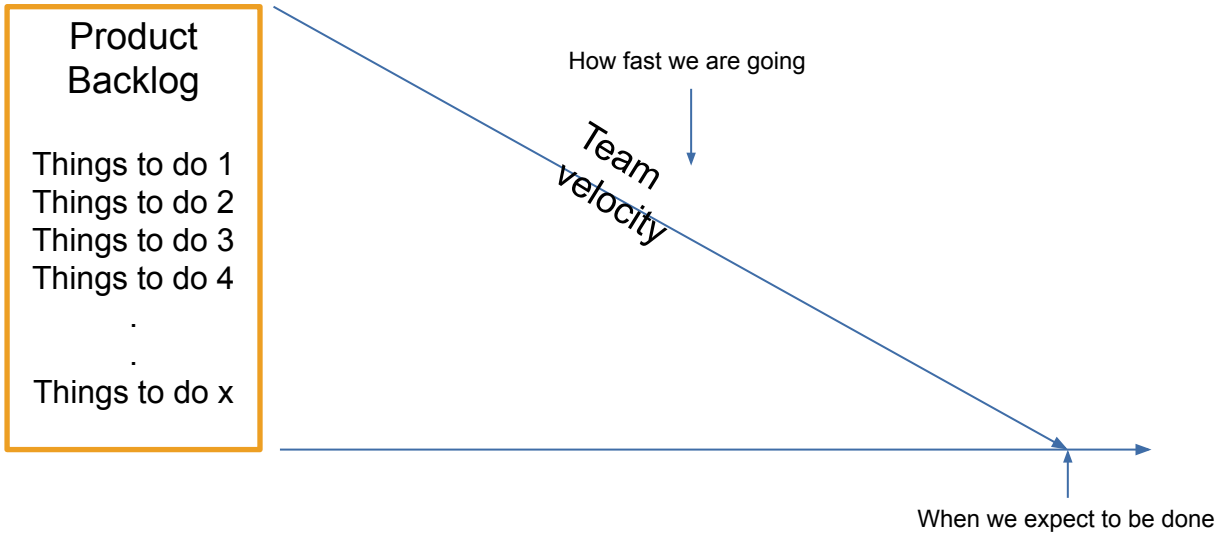
- Velocity only counts completed work effort.
- Completed work is that which satisfies the definition of done.
- Partly completed work is never counted towards the velocity.
- This is why when planning it is better to break tasks down into smaller parts.
- There is then less risk you will not deliver any functionality in a sprint.

If a larger task cannot be completed. We can deliver part of the issue and split the issue into more than one part.

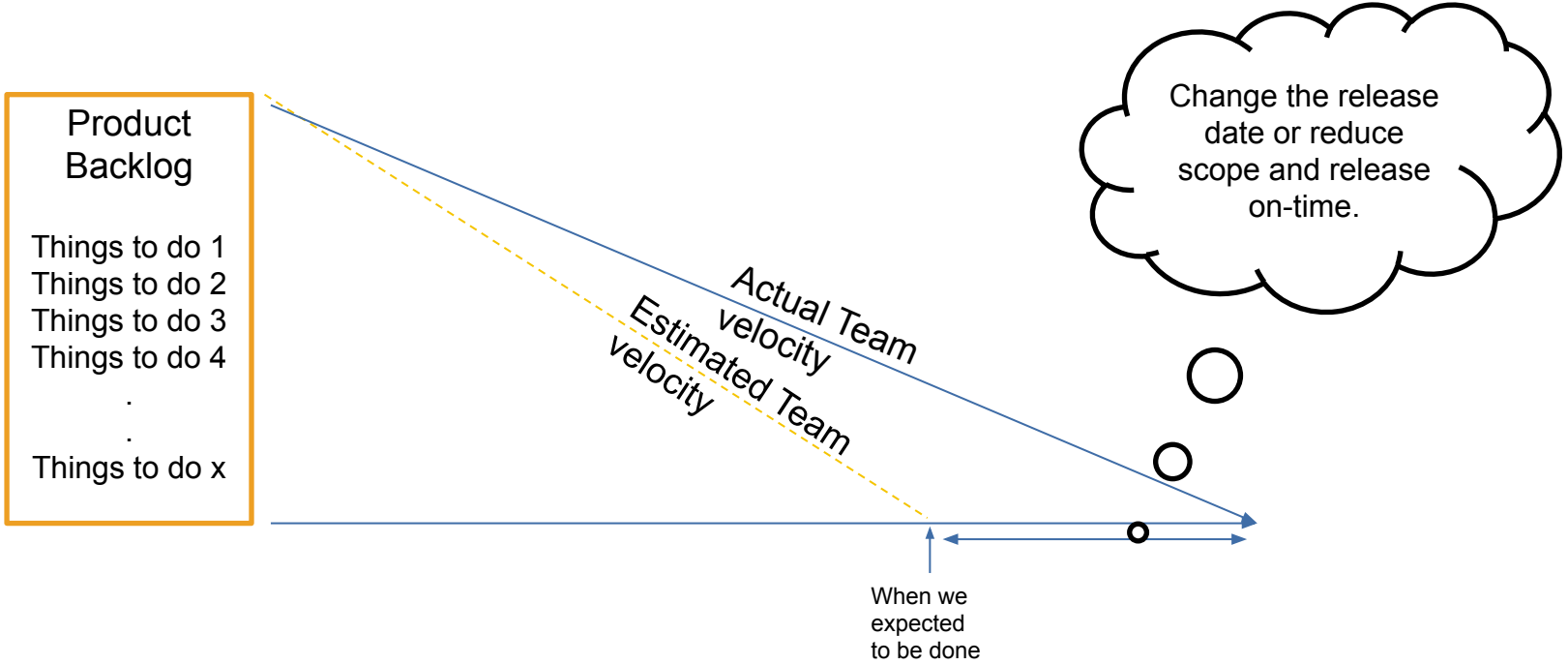
We get some velocity for the half finished issue.

But this completed part must still meet definition of done.

SIMPLE AGILE PLANNING

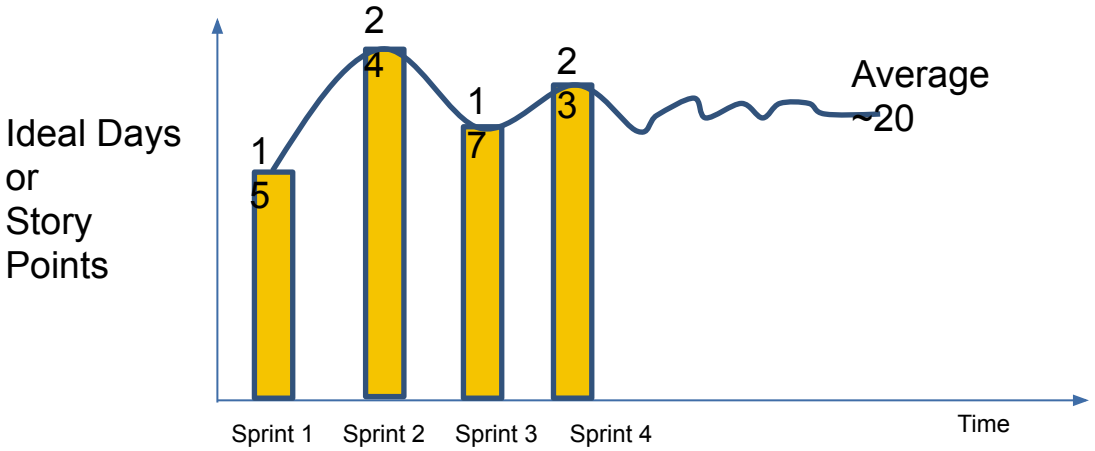


SIMPLE AGILE PLANNING



VELOCITY ‘WILL’ FLUCTUATE

- The team velocity will change from sprint to sprint.
- So it is important to take an average over time. This should be monitored and carefully tracked.
- We use this average for our planning and roadmap estimations.



Sprint Planning

PRODUCT PLANNING

- Almost all products will need an overall scope defined in some kind of Product Outline Specification before execution.
- Some may need this before execution can be agreed.
- Some may need this to judge scope and size of the project team required.
- The Product Owner will have a Product Planning meeting to gauge the overall size and goal of the project. This may not involve detailed planning.
- Agile is suitable for projects where we know some change is inevitable.
- This is the first chance to consider change, support tasks and what you are going to do.

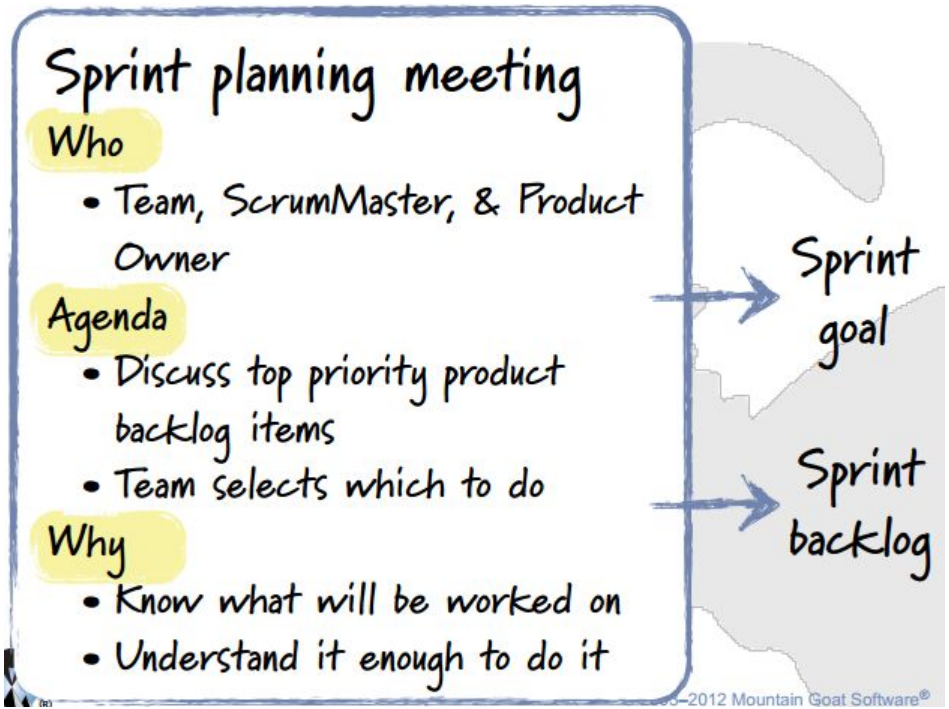
PRODUCT PLANNING

- For initial planning we will estimate the product backlog.
- We then need to consider the risk of this list not being complete and new items being added.
- This will depend on the stakeholders and project being estimated.
- You can use historical information to guide you.
- This will indicate the buffer required to be added to the project for estimation purposes.
- The buffer is a useful tool to enable effective change management.

PRODUCT PLANNING

- To do effective planning we must get estimates for all issues in the backlog as quickly as possible.
- This is achieved during product planning meetings.
- These meetings are similar to Sprint Planning meetings but we go through the backlog and estimate as much as we can.
- The issues remain in the backlog but have a time or story point estimate to complete them.
- This gives an indication of effort for every work item we are intending to fix.
- This is required so we can estimate the time it will take to complete the backlog early in the project.

SPRINT PLANNING MEETING



SPRINT PLANNING MEETING

- Items are pulled from the backlog until the development team thinks it has enough work to fill the sprint.
- Some items may need additional stories and definition. High-level design considered.
- All items should be estimated using days or story points to determine if the target is reasonable for the sprint. Both are perfectly valid.
- This is not just done by the scrum master. (Really important!)
- The team will use its historical velocity as a guide to what can be achieved.

INCREMENT

- The increment is the total work completed within the sprint.
- At the end of a sprint all items must be done according to the team’s definition of done.
- The increment must be in a usable condition regardless of whether the product owner intends to release it.

WHAT IS A STORY POINT AND WHY USE THEM?

- We can used days as an estimation technique and also story points.
- We have settled on using story points.
- These are an arbitrary relative estimate that considers size and complexity of the task.
- The values to select are far enough apart to make the selection easier.
- It is common for teams to use a modified Fibonacci sequence. But you can use anything..

Disadvantage! Using story points makes estimation comparison across project teams very difficult.

WHAT IS A STORY POINT AND WHY USE THEM?

- Why use them?
  - Asking engineers to estimate in days is time consuming and not necessarily accurate.
  - Estimating in days and size in KLOC gives infinite choices to the engineer. Making it more time consuming.
  - Story points allow fast estimations that can be near as accurate as day estimations.
    - Over time the estimations get more accurate.
    - Estimations are made by comparing against similar jobs you have done before.
      - To start I would suggest choosing 3 or 4 previous tasks and giving them a story point value.
    - We have been using story points for a few projects now and the estimation is no less accurate than before but much faster to obtain.

STORY POINT EXAMPLE

- Can you tell me how many kilograms a Labrador dog weighs?

STORY POINT EXAMPLE

- This example may be for fun but it shows why it is easier to judge relative complexity and size of tasks rather than the detail.
- We do not need the very detailed estimate and often these type of estimates are soon out of date as change comes anyway.
- Story points or Ideal Days allow fast effective estimation that is accurate.

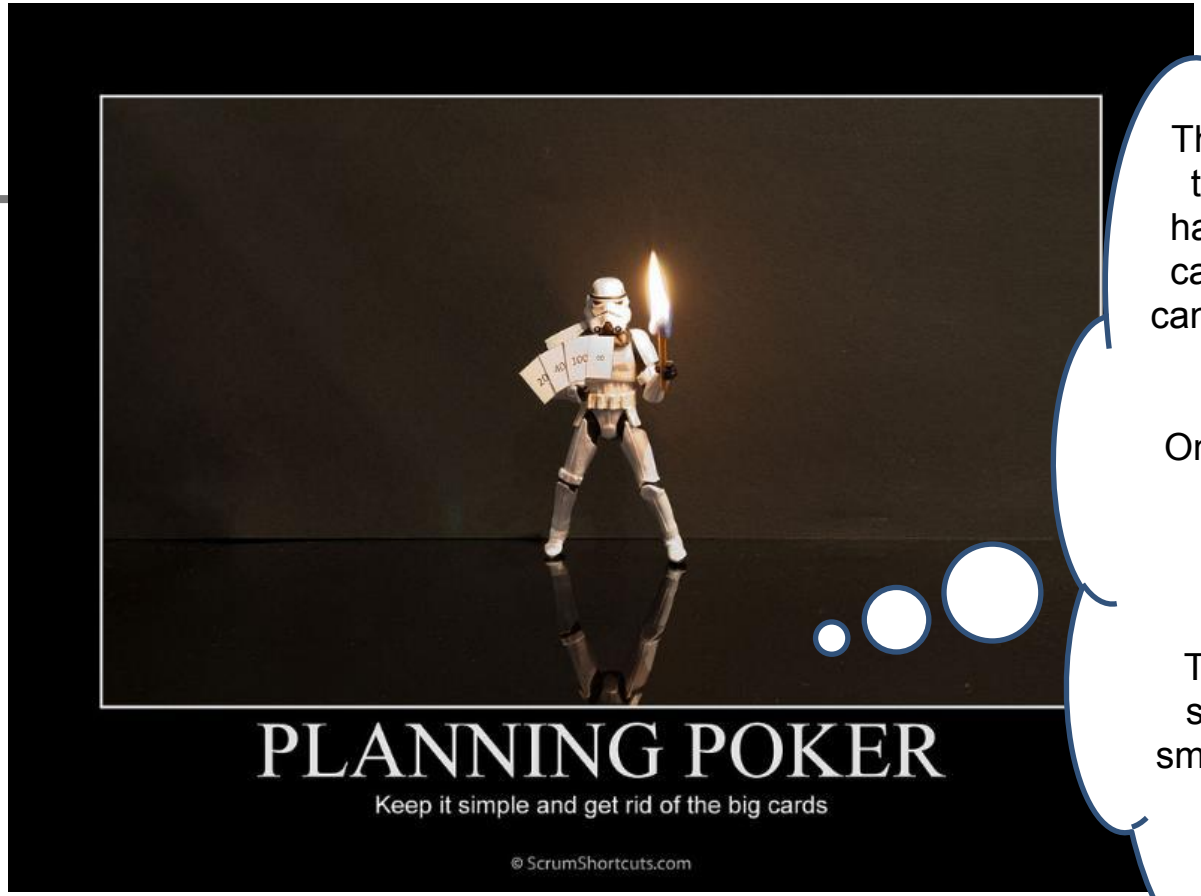
BUT WHAT IF WE WANT TO USE DAYS?

- We can use the term ‘ideal days’ or ‘perfect days’ instead of story points.
- They can be used in exactly the same way as story points.
- Still use the same modified Fibonacci sequence to make selection easier.

An ideal day is a perfect day with no interruptions and you can work totally focused on the task.

PLANNING POKER

- In the Sprint planning meeting you can also use planning poker cards.
- Each team member has their own planning cards.
- The most knowledgeable team member for the task describes the particular user story we are discussing.
- Each team member then selects a planning poker card.
- All team members then show the cards at the same time.
- This avoids the “Delphi” effect where one member can influence others.



This is important – there is no point having big number cards for work that cannot be completed in the sprint.

Once you know the team velocity remove the big cards.

This will promote splitting jobs into smaller manageable chucks.

Keep your card a secret so you don't influence other team members.

If you have an "expert" on the team this can influence more junior members.

PLANNING POKER

User Story
1

When there are high and low estimations these are discussed and the reasons for the estimation differences understood. Then the team decide what to do.

SPRINT PLANNING

- *I mentioned before that running an agile process can be more effort and work.*
- One main reason is that the effort consumed by planning is much bigger.
- The development team should be solely focused on the tasks set out in the Sprint Planning.

- It is easy to do insufficient planning in an agile process.
- Some metrics suggest that a successful agile team should be utilizing 20% of their time on planning activities.
- It is not just sprint meetings – but product meetings, examining the backlog, triaging new issues. The effort for the Product Manager and Scrum Master is much higher than before.

SPRINT PLANNING MEETING

- At this time the Product Owner will manage the priority of items and the remaining work in the backlog.
- Important to continually check and consider remaining work in backlog to judge if the overall project is on track.

PLANNING PRIORITY

- We should ensure those user stories with the highest business value and importance to the customer are developed and delivered first.
- Anything could happen at any point on the project that causes an issue. (Project cancellation, funding reduced, etc)
  - So we must prove the projects worth as quickly as possible by delivering the highest value items.
  - Having the stakeholder involved in this process ensures we understand their needs and requests.
  - It also avoids spending a lot of time on “nice to have” features when we have not delivered the basic features first.

The Daily Stand-up

DAILY SCRUM MEETING

- Often what people think Scrum is all about…
- Daily meeting – held at the same time every day. Ideally in the morning.
- All team members must join this meeting.
- No more than 15 minutes.
- If it is taking longer then there are too many people or the agenda is not being adhered to.
- Stand-up meeting – sounds silly but keeps the meeting brief.
- Not for problem solving or discussing the details of issues.

DAILY SCRUM MEETING

- Each team member answers 3 questions:
  - What you did yesterday
  - What you are going to do today
  - Is anything blocking your way?

- If there are blocking issues then a separate meeting should be setup to discuss.
- Any issues should not be discussed in the daily scrum meeting.

- The daily scrum may seem annoying at first but totally changes the project pace.
- Team members don’t wait for the weekly meeting to discuss important matters.


Recycle Bin
Death Star Missile System 2.0
Not Started
In Progress
Ready for Test
Done
DAILY STAND-UPS
No matter how heavy your armour is, this is a stand-up not a sit-down
© ScrumShortcuts.com

DAILY SCRUM MEETING

- Don’t record detailed minutes. Only remember important actions to setup other meetings.
- The purpose of the standup is not for the Scrum Master – it is for all team members to be aware of what is going on. (So pay attention and stay off your phone!)
- It is not a planning meeting.
- It is not a technical discussion.
  - If these are required request another meeting to be setup with the people needed.

DAILY SCRUM MEETING

- The stand-up is primarily to share information and coordinate activities in the team.
- However…
- Team members publicly committing to “What they are going to do today” dramatically increases the chances of it getting done!

Continuous Improvement

RETROSPECTIVE MEETING

- Meeting held at the end of the Sprint to review what happened.
- Product Owner, Scrum master and team invited.
- In our case we also invite group manager and QA representatives.
- Like a mini project close meeting.
- Aim for no longer than 60 minutes in length.

RETROSPECTIVE MEETING

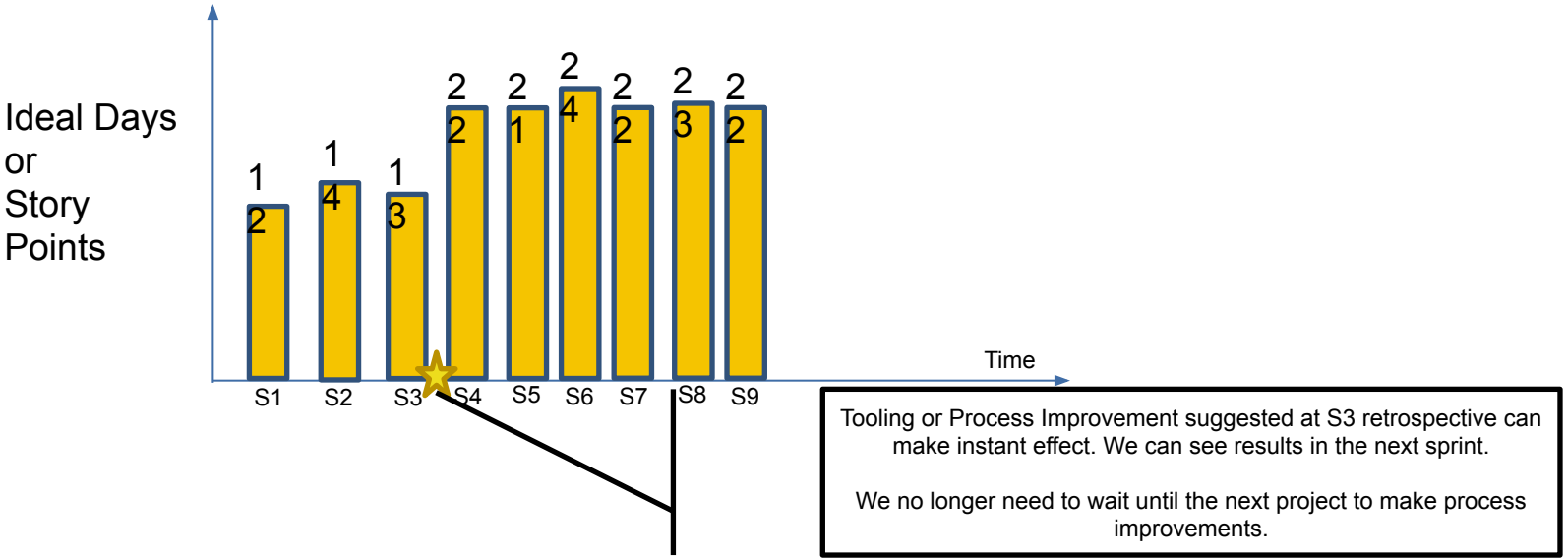
- Review what went well, what did not go well and what we would like to improve.
- Driven by team – feedback collated before the meeting.
  - We capture it in Confluence. All team members add things to agenda.
- Each team member asked what they would like to improve.
- The team chooses one item to focus on and implement in the next sprint.

Remember this is a team retrospective. We understand and believe all team members did the best job they could, given what they knew at the time, the resources available and the situation we were in.

So no individual blame should be discussed.

RETROSPECTIVE MEETING

- Improvements suggested during respective meetings can be implemented quickly and the effect monitored and reviewed.



RETROSPECTIVE MEETING

- Use your Sprint Metrics as a discussion item in the retrospective.
- They are a useful indicator of what went on in the Sprint.
- Some example metrics could be:
  - Velocity over time
  - % of items not completed or removed from the sprint
  - % of Sprint scope change
  - Number of reopened issues from verification
- Automate the gathering of these metrics using scripting. (Python)

RETROSPECTIVE MEETING

- At the beginning of the project hold a retrospective after each sprint.
- It is important to review regularly if the project is a new activity.
- As time goes on this can be relaxed and reduced.
- We try and do this at least every other sprint.

- However some teams use this meeting to show and tell something new if there is no need for a retrospective. This is a good idea.

Looking for customer feedback

PRODUCT REVIEW MEETING

- The team presents what was accomplished in the sprint.
- Typically would take the form of a demonstration of new features or underlying architecture.
- Think of it as a “show and tell” activity.
- Informal meeting.
  - Limit preparation to less than 2 hours. Do not use slides!
- Ideally the whole team participates.
- Invite all stakeholders – open invitation to interested parties.

PRODUCT REVIEW MEETING

- This meeting is critical for Scrum to work well and give benefits.
- It is important we look for feedback on our work – be proactive!

- It is crucial that feedback is recorded and fed back into the next Sprint Planning activity and updating the product backlog.

WANT TO READ MORE?

- Interesting read with interesting examples of Scrum in practice: (here)
- Very simple book – useful as reference: (here)

Some decent websites:

- The Scrum Guide by Scrum.org. (Online Book)
  http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
- Scrum Alliance http://www.scrumalliance.org
- Jeff Sutherland http://scrum.jeffsutherland.com
- Mountain Goat Software - Mike Cohn's Blog
  http://www.mountaingoatsoftware.com/blog

banvien Trusted Technology Partner

Thank You!!