EVERYTHING YOU'VE HEARD ABOUT AGILE DEVELOPMENT IS WRONG

Simon O'Toole

Australian Astronomical Observatory

Australian Government
Department of Industry, Innovation and Science

SOFTWARE DEVELOPMENT QUIZ

- What is the Waterfall model of software development?
- What are the advantages and disadvantages of Waterfall?
- What is Software Prototyping?
- What is RAD or Rapid Application Development?
- What is Agile software development?
- Name three different Agile methodologies
- What software development methods do you use?
- Do you have any experience with Agile methods?
- If yes, were they good or bad?

A BRIEF HISTORY OF SOFTWARE DEVELOPMENT

- In the beginning, there was binary code
- 1960s – Systems Development Life Cycle
- 1970s – Software prototyping
- 1970s – Waterfall model
- 1980s – Structured systems analysis and design method (Waterfall style)
- 1990s – Rapid Application Development, various flavours of Agile (Scrum, XP, DSDM)
- 2001 – Agile Manifesto; this is the first use of the term Agile, previously “Lightweight”

WHAT AGILE IS…

- Agile is “a set of principles where requirements and solutions evolve through the collaboration of self-organizing cross-functional teams”
- Agile is a method to develop software incrementally with a strong focus on meeting (changing) user requirements via regular releases and testing
- Agile expects and thrives on changing requirements
- Agile is a series of processes that lead more often to software that people want
- Agile is, in some ways, common sense, but with a pragmatic approach
- Agile is a philosophy and a mindset.

… AND WHAT IT IS NOT

- Agile does not mean there's no need for documentation or planning (in fact, planning is fundamental to its success!)
- Agile methodologies are not something that only software teams use, they need to be adopted by product owners and management
- Agile is **NOT** the solution to all existing problems in software development
- Agile is also **NOT** a religion, although some people may make it seem that way

BUT, BUT, AGILE AND SCRUM SUCKS!!

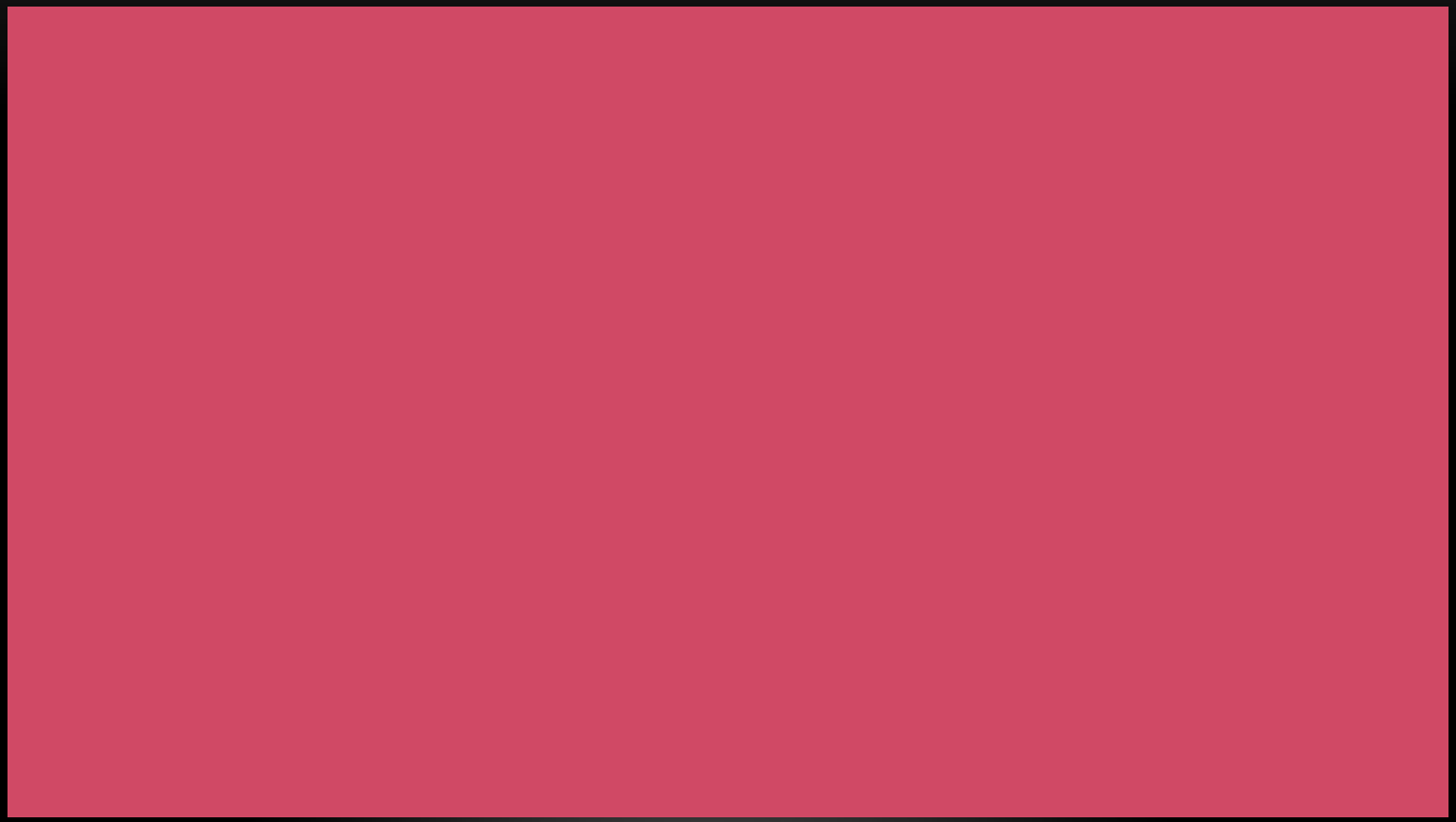
Only if not done correctly!

- Agile **will** fail if:
  - The software team is not fully behind its use
  - Its adherents are only superficially engaged with the ideas behind it
- As more people start using a method, there are more beginners and therefore fewer people who understand it well
  - The perception that it is no good increases
  - Great discussion by Ron Jeffries: http://ronjeffries.com/articles/016-09ff/victim/

http://www.agilemanifesto.org

THE AGILE MANIFESTO

THE FOUR AGILE VALUES EXPLAINED



THE 12 PRINCIPLES OF AGILE DEVELOPMENT

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within the development team is face-to-face conversation.

THE 12 PRINCIPLES OF AGILE DEVELOPMENT

7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done — is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

FLAVOURS OF AGILE

Include the following:

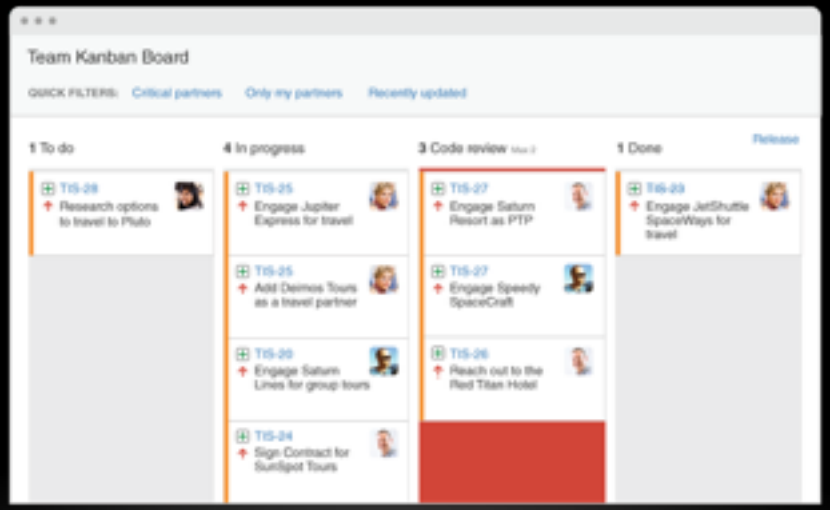
- Scrum
- Extreme Programming (XP)
- Dynamic Systems Development Method (DSDM)

NOTE: Kanban is not an Agile methodology, but is often used in conjunction with other methods

Many teams choose the aspects of each that best suits their skills and experience

ASIDE: KANBAN BOARDS

- Kanban was developed by Toyota in 1940s to manage their production processes
- Similar to the “Just In Time” (JIT) process
- Matches amount of Work in Progress to the team’s capacity
- A Kanban Board is used to visualise these workflows

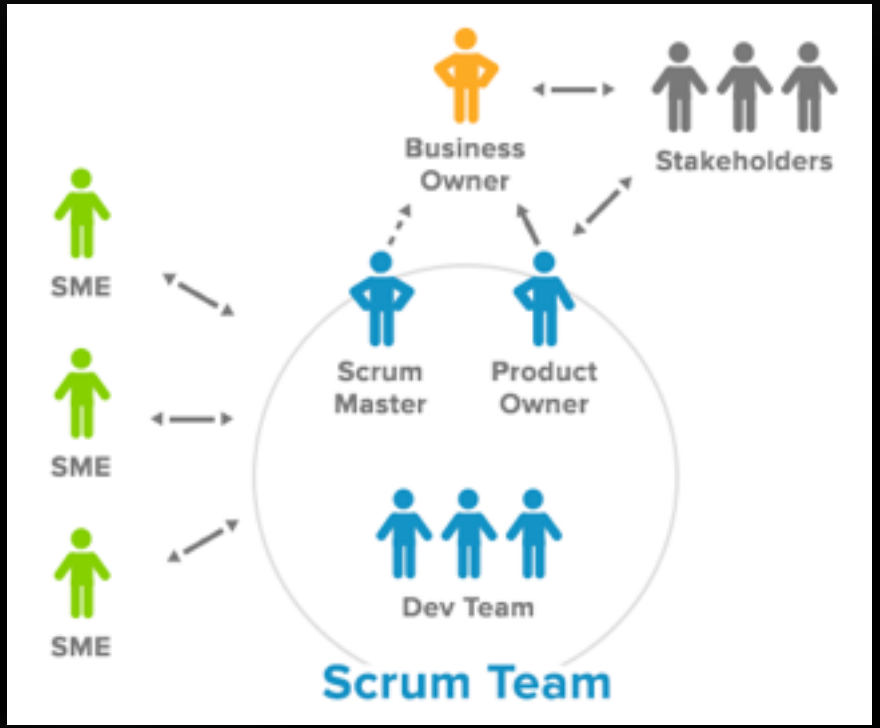


https://www.atlassian.com/agile/kanban

TEAM STRUCTURE

Scrum:

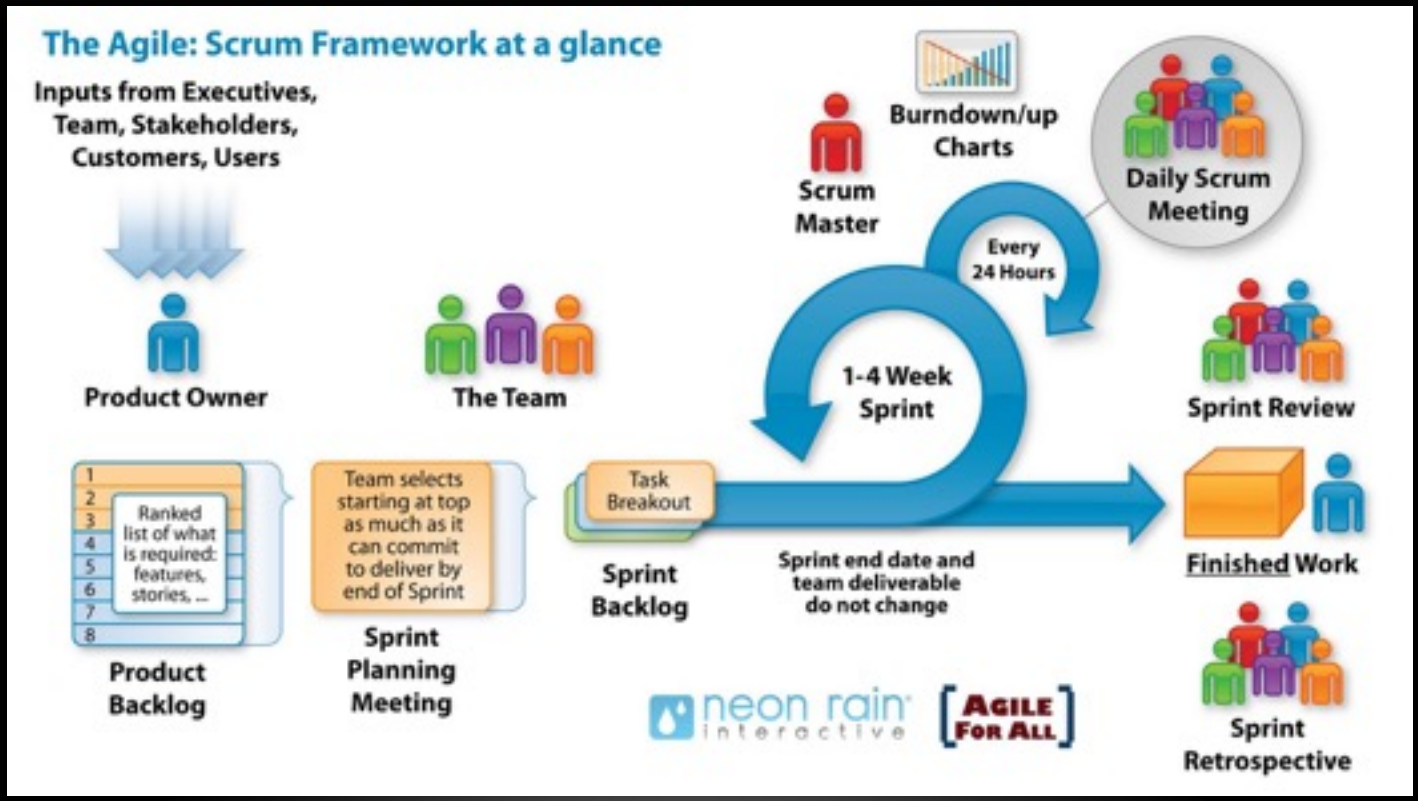
- Product Owner
- Scrum Master
- Developers
- (Subject Matter Expert)
- (Business Owner)
- (Stakeholders)



http://www.mendix.com

SCRUM

- Usually used in software development, though this is changing



http://www.neonrain.com

TEAM STRUCTURE

DSDM:

- Project Level
- Solutions Development Level
- Supporting Level
- Team members can have multiple roles

http://www.dsdm.org

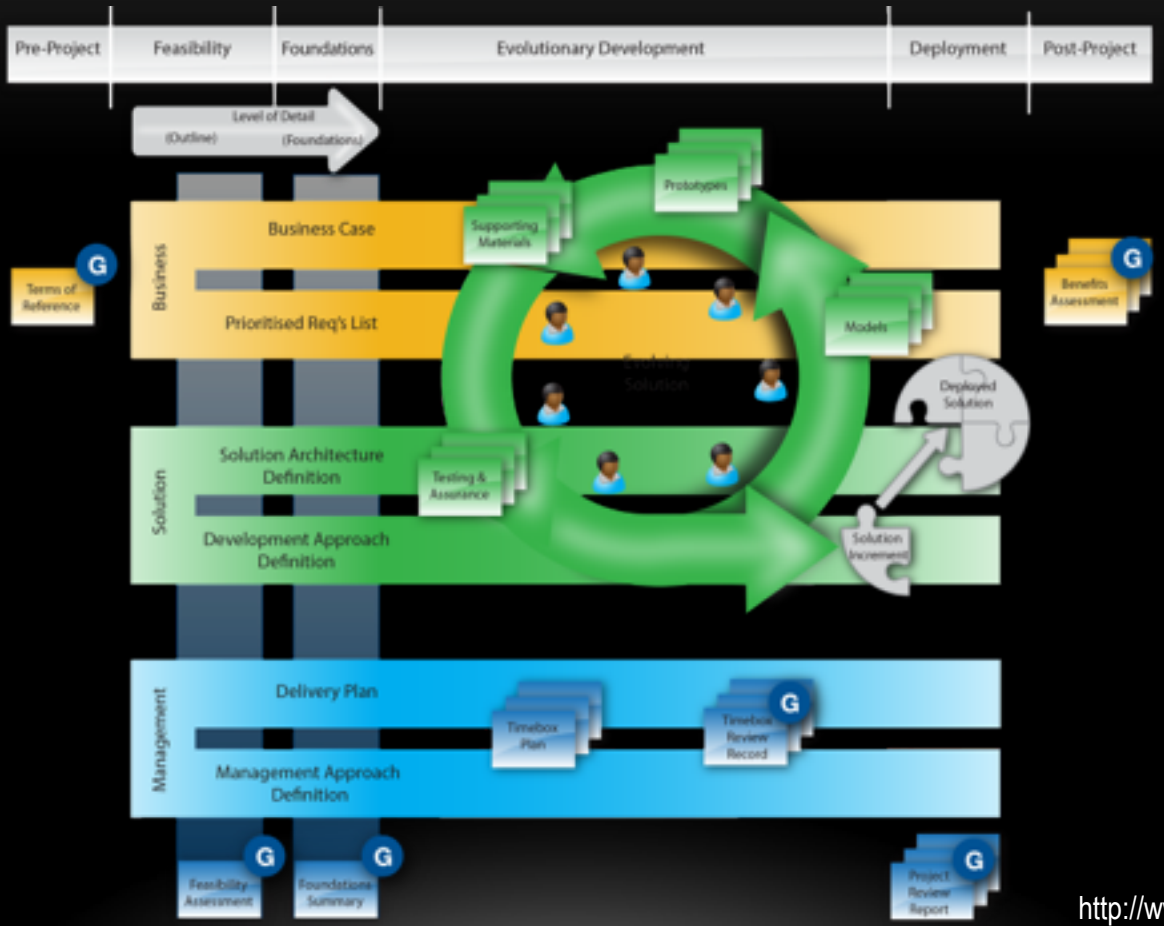
DYNAMIC SYSTEMS DEVELOPMENT METHOD

- The DSDM philosophy is

"best business value emerges when projects are aligned to clear business goals, deliver frequently and involve the collaboration of motivated and empowered people."

- This is achieved when all stakeholders:
  - Understand and buy into the business vision and objectives
  - Are empowered to make decisions within their area of expertise
  - Collaborate to deliver a fit for purpose business solution
  - Collaborate to deliver to agreed timescales in accordance with business priorities
  - Accept that change is inevitable as the understanding of the solution grows over time
- DSDM can be implemented at Whole of Project Level, and not only for software

DYNAMIC SYSTEMS DEVELOPMENT METHOD



http://www.dsdm.org

AGILE PROJECT INITIATION

Project Initiation includes these steps:

1. Find/Write Project Brief
2. Form the team
3. Initial Plan
4. Initial Requirements
5. Initial Infrastructure

Once the team is formed, the remaining steps are typically done in a sprint/timebox.

SCOPE VS REQUIREMENTS

- Traditionally, project scope is defined as:

  The overall definition of what the project is supposed to accomplish, and a specific description of what the end result should be or accomplish.

  It is a subset of the scope of the product under development

- The main difference with Agile project scope is that change is seen as inevitable in Agile, so the system facilitates change, rather than combats it
- Agile projects start with a Project Brief, which is sometimes (but usually not!) written before the project starts

USER STORIES

- The Project Brief provides the context in which the User Stories are written.
- User Stories are usually brief descriptions of the intended software behaviour from the user's perspective.
- They are often expressed in the form

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

- User Stories make up the high level project requirements, which in turn defines the project scope
- Anyone in the team can write a User Story
- A large User Story is often referred to as an Epic.
- The initial list of User Stories/Epics makes up the Product Backlog

PRIORITISATION

How do you prioritise tasks in your projects?

- Needs Based Analysis?
- Planning Game (from Extreme Programming)?
- MoSCoW prioritisation?
- Something else?
- No priorities, just GO!

MOSCOW PRIORITISATION

MoSCoW stands for:

- M – Must Have

  the "Minimum Usable Subset" or Minimum Viable Product

- S – Should Have

  the Product Owner/Business Visionary & Sponsor expects to have these

- C – Could Have

  these features would be nice to have

- W – Won't Have This Time

  these features are not appropriate or necessary for this timebox

CASE STUDY: RED SKIES OBSERVATORY

RSO 8m telescope: new spectrograph (see handout)

Red Skies Observatory is building a new high-resolution spectrograph for their 8m telescope. The system is fibre-fed and is designed for very high velocity precision measurements using several different simultaneous calibration sources. While the instrument is being built, little thought has been given to software, beyond a basic idea of what the scientists would like to be able to do. There is a Science Team (from all around the world) associated with the project, as well as the Instrument Team (mainly at RSO). There is also a team working on the RSO Data Archive, who will host the data coming from the telescope.

https://cloudstor.aarnet.edu.au/plus/index.php/s/WJ1CaKEvFr6utqg

PRIORITISATION EXERCISE

In groups of 5-7 people:

- Determine all the requirements of the RSO spectrograph software based on the specification
- Write out each requirement as a User Story
- Use MoSCoW prioritisation to determine the Minimum Viable Product along with the priorities for each User Story
- The Observatory Directory wants the system to begin end-to-end commissioning in 6 months

TIME-BOXING AND SPRINTS

Different flavours of Agile use different terms for effectively the same thing:

- Scrum: Sprints
- DSDM/AgilePM: Timeboxes

DSDM Timebox

HOW DOES A SPRINT/TIMEBOX WORK?

- Kick Off: ~4-8 hours
  - Set up and allocate tasks
  - Prioritisation of tasks within timebox
- Iterative Development: 1-4 weeks
  - Daily Stand Ups
    - Short (<20 minute) meeting to discuss status and flag issues
    - Rough guide: 2 mins/member plus 2 min wrap
  - Team members solve issues outside the Stand Up
  - Build AND Test
- Close Out / Sprint Retrospective: 4-6 hours

WHAT IF YOU DON'T COMPLETE THE TASKS?

- Some sprints/timeboxes will not be successful, for a variety of reasons
- If this occurs, examine the reasons in the close-out/retrospective
- Agile is about teams, so apportioning blame on individuals is NOT appropriate
- The TEAM needs to learn, possibly re-evaluating time estimation and priorities
- Return the User Story or Stories to the Product Backlog and move onto another
- Come back to User Story later

HOW DO YOU PLAN A TIMEBOX?

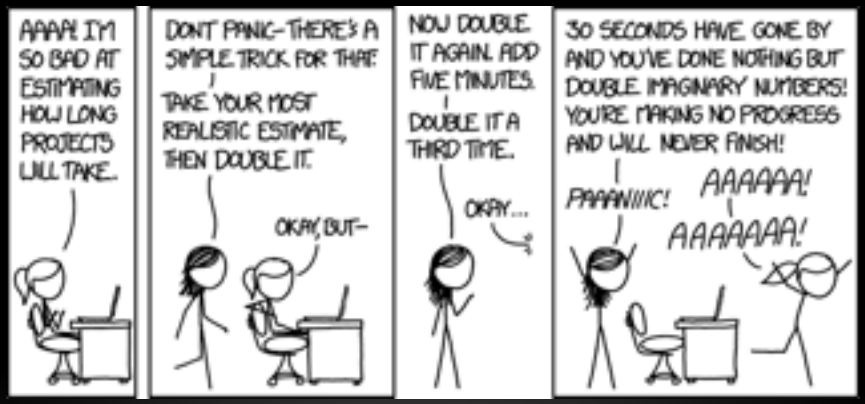
- Take the User Stories/Requirements and break them into smaller tasks as necessary
- Estimate how long each User Story and tasks will take
- Organise them into groups that will take about one to four weeks
- Set out a list of timeboxes/sprints for the Project
- This list is a guide, you will need to re-assess as requirements change

SPLITTING A USER STORY

- There are at least three reasons to split a User Story:
  - The story is an Epic, i.e. to too big to fit in a Timebox
  - Part of the story can't be estimated
  - Part of the story has/is higher priority (more value and/or more risky) than the rest
- Splitting a User Story can allow for more detail to be added
- Allows for more fine grained management of tasks and subtasks
- One User Story/Epic can be done over several timeboxes (at least in DSDM)
- In DSDM this is called an Increment – working software should be delivered after each Increment
- In Scrum you might expect working software after each Sprint

WAYS PEOPLE ESTIMATE SOFTWARE

- Darts.
- Give it to the manager: Managers are more skilled and there is a huge possibility to underestimate the work/feature.
- Ask the expert: Architects or domain experts are experts on their own domain but may not know about the capabilities of the team who will develop the system.
- Without "bothering" the developers. They are busy.



https://xkcd.com/1658/

TIME ESTIMATION

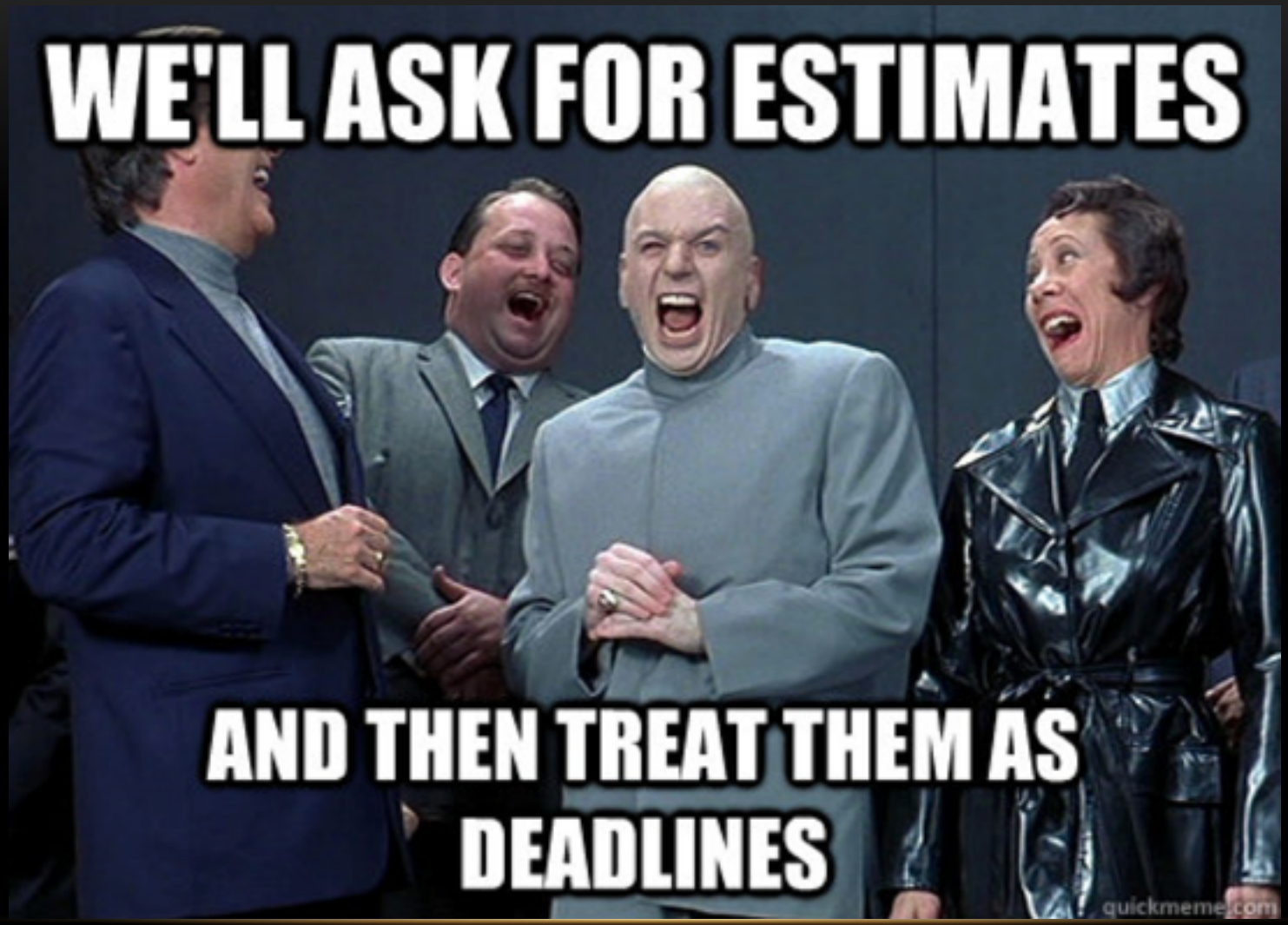
- Examples of actual ways people estimate time:
  - Based on previous effort

  Experience goes a long way, however new tasks present new and unforeseen challenges.

  - Planning Poker

  A way to arrive at a consensus view of time.
- Story points as a “measurement” of time

These are used to measure the effort required to implement a User Story. In simple terms its a number that tells the team how hard the Story is. Hard could be related to complexity, unknowns and effort.


WE'LL ASK FOR ESTIMATES
AND THEN TREAT THEM AS DEADLINES
quickmeme.com

PLANNING POKER

- Planning poker is used to avoid the influence of others when estimating time for a User Story
- It involves everyone in the team, with a moderator, writing their estimate on cards and then displaying them all together
- Extreme estimators are given a chance to justify their choices; *this should be moderated*
- The process is repeated until a consensus is reached

BUILD YOUR OWN SPRINT

- Choose one or more of the User Stories you derived for the Red Skies Observatory prioritisation exercise
- Develop a series of reasonable tasks that achieve the goal of the User Story or Stories
- Estimate the time for each task using Planning Poker techniques

IN SUMMARY

- Agile development is a great methodology for building software and software components so that it meets the users needs
- It can also be used more broadly as a project management tool
- Mix and match the components of whichever Agile methodology works for you
- Agile methods may not be suited to all projects overall (e.g. instrument building), but may be suitable to some parts (e.g. software control and/or data reduction pipelines)

MORAL TO THE STORY

Use what works for your team, but focus on both what is being delivered and to whom it is being delivered.

FURTHER READING

- The Scrum Alliance https://www.scrumalliance.org/
- The DSDM Consortium: https://www.dsdm.org/resources/dsdm-handbooks/the-dsdm-agile-project-framework-2014-onwards
- Defense Against The Dark Arts (of Scrum): http://ronjeffries.com/articles/016-09ff/defense/
- Agile Project Scope http://itsadeliverything.com/agile-project-scope and links within