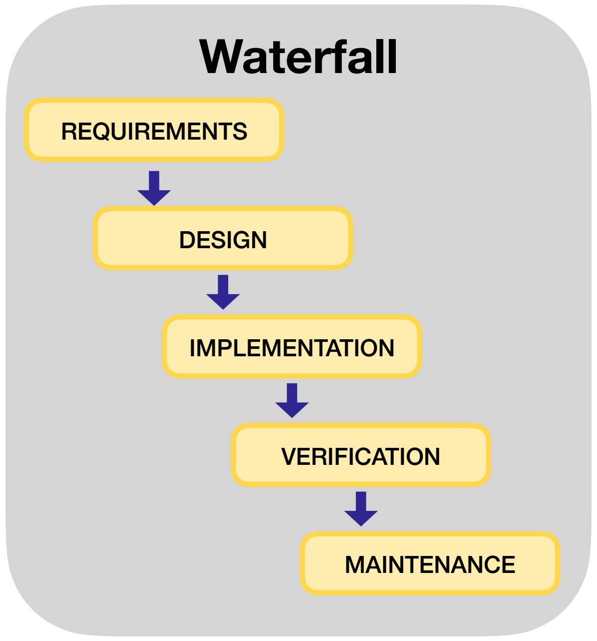
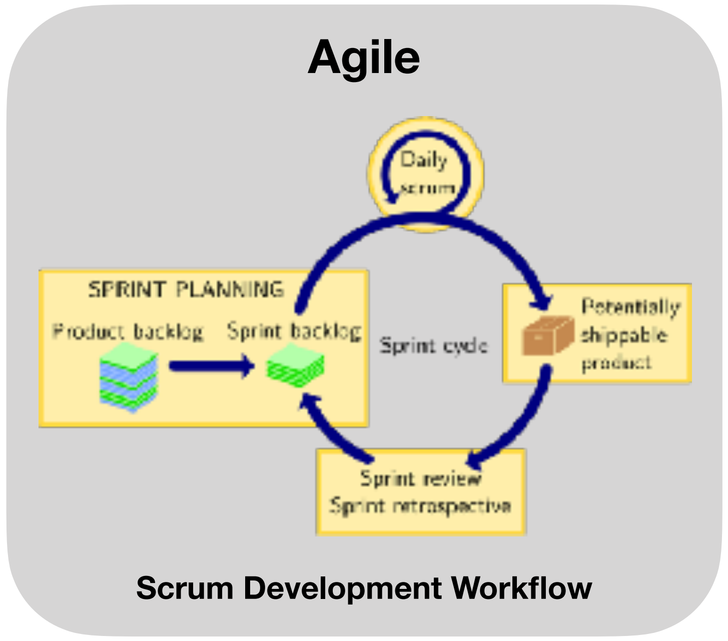
Scrum Development Framework
CS179N

Development Methodologies

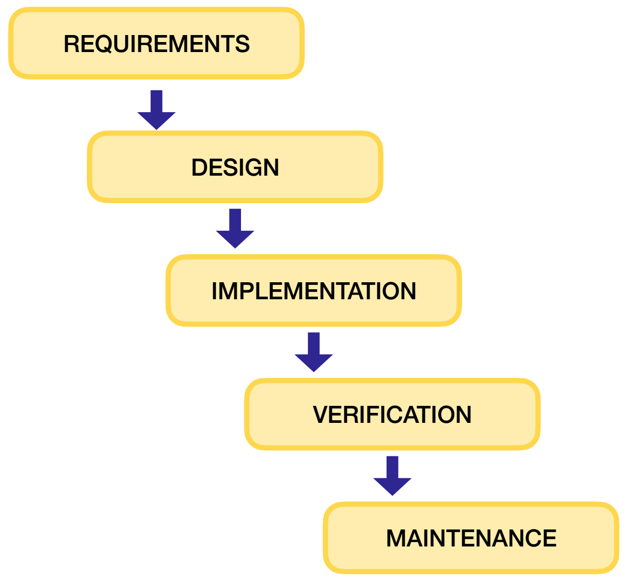




Left: corresponds to a “heavy-handed” management style. Predictive method. Know in advance what features and tasks are planned along lifetime of product development. A lot invested in early-stage analyses. Difficult to change course. Testing is a separate stage.

Right: corresponds to a “lightweight” management style
agile: requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). … Requirements and design are held to be emergent. [Wikipedia]
other agile styles: Extreme Programming
Testing is completed with development.

Waterfall

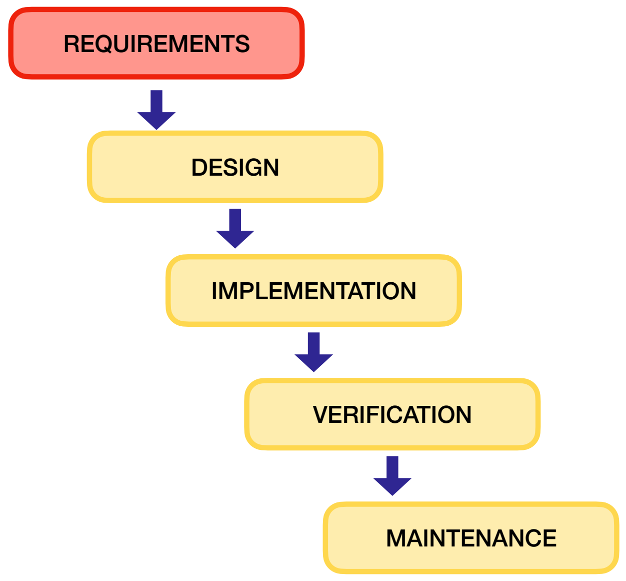


- Caricature of a rigid, inflexible, sequential process
- Progress flows largely in one direction
- Pedagogical example of how not to do software development

- term “waterfall” is usually used to describe a critical view of a commonly used software development practice
- 1985, US DoD required similar process of contractors

https://en.wikipedia.org/wiki/Waterfall_model

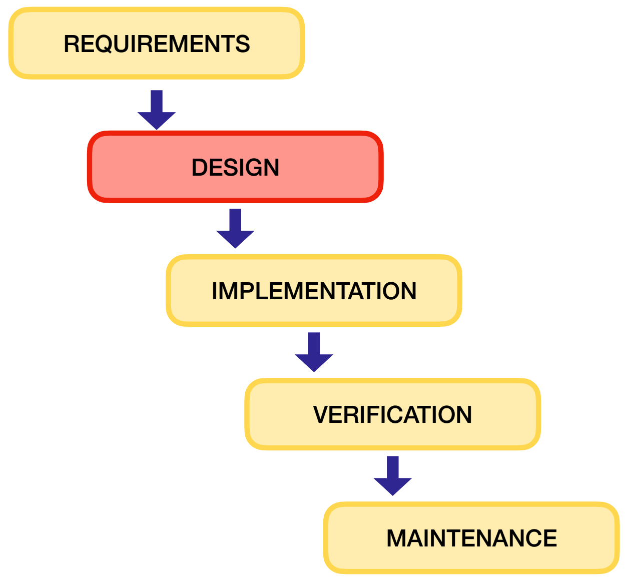
Waterfall



- Gather and analyze requirements from customers and stakeholders
- Results in a product requirements document

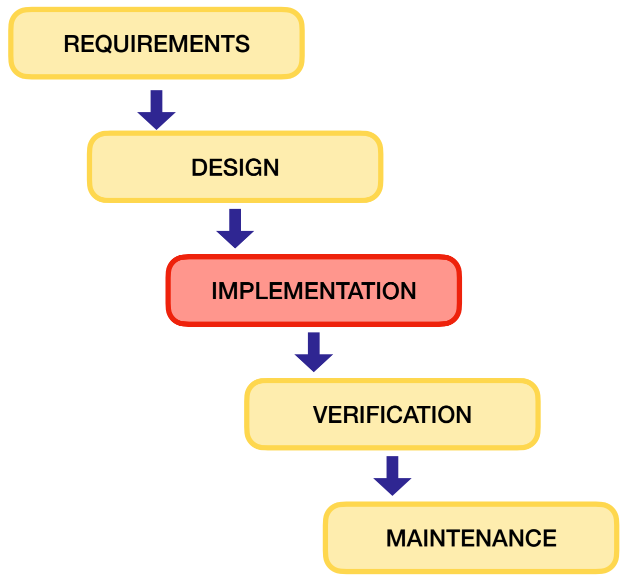
- establish the needs of “stakeholders” that are to be addressed by the software capabilities required by users
- requirements imposed by contracts, standards, specifications, etc.

Waterfall



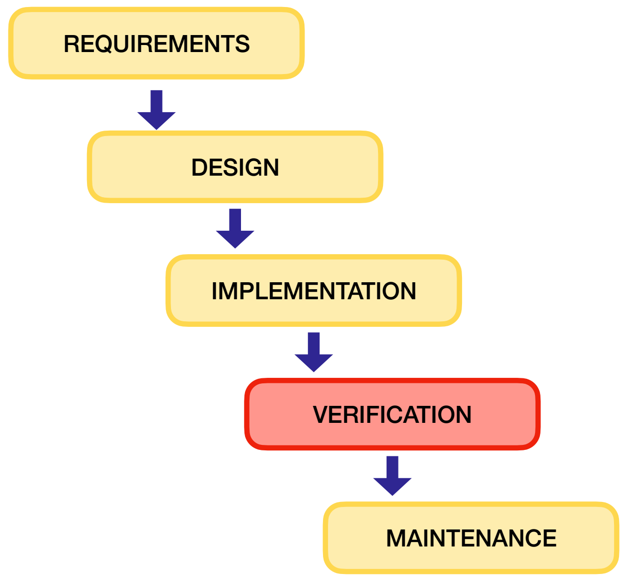
- Determine high-level software architecture
- Create specifications of software for subsequent implementation

Waterfall



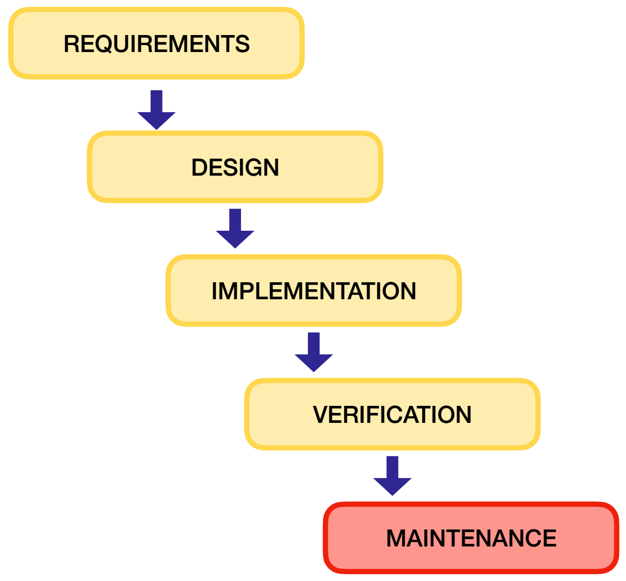
- Design is implemented in code
- Unit testing
- Integration of software components

Waterfall

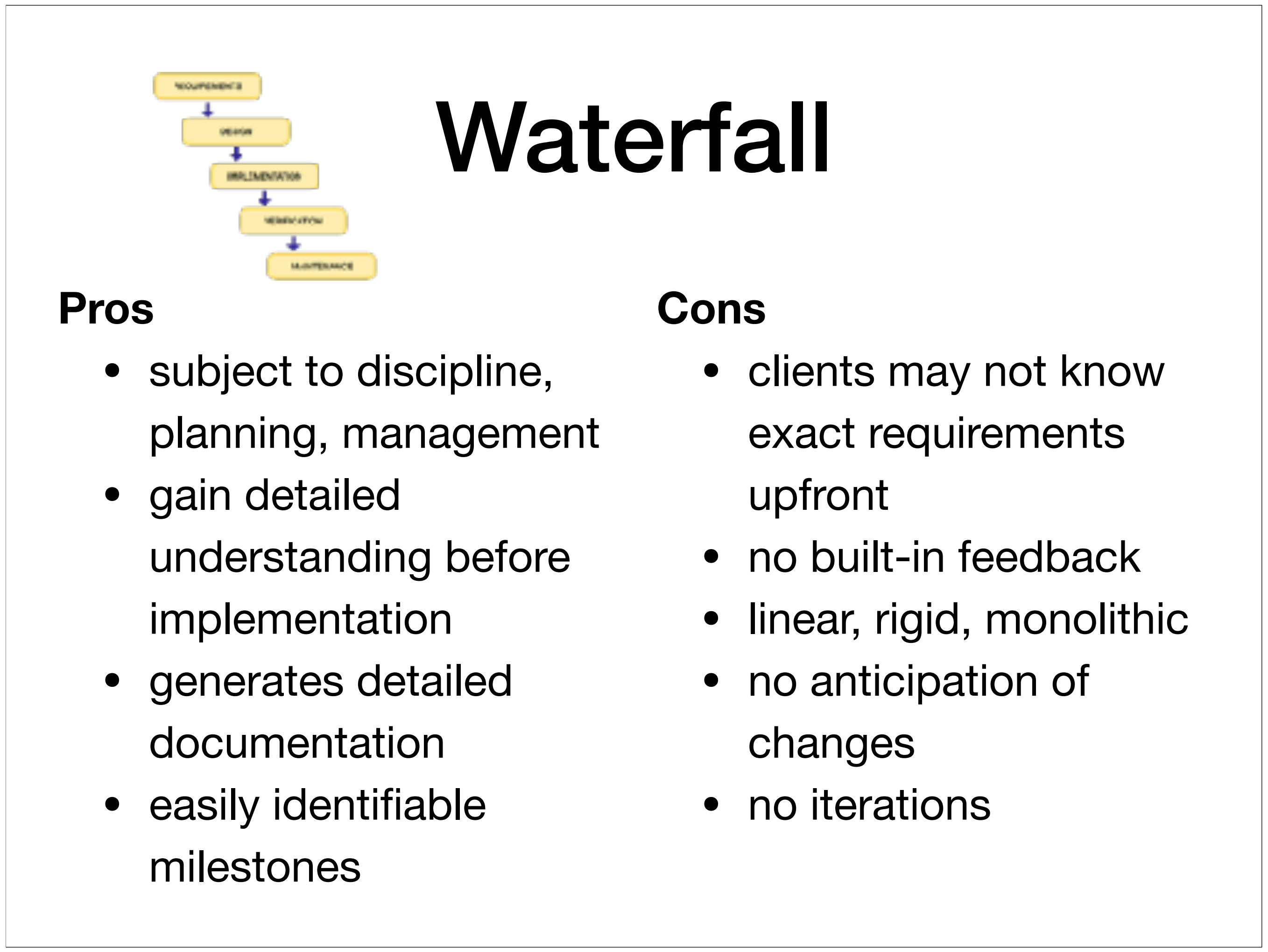


- Comprehensive testing of integrated system
- Systematic discovery and debugging of defects

Waterfall



- Deploy system into customer environment
- Customer support
- Correct faults or improve performance



Waterfall

Pros

- subject to discipline, planning, management
- gain detailed understanding before implementation
- generates detailed documentation
- easily identifiable milestones

Cons

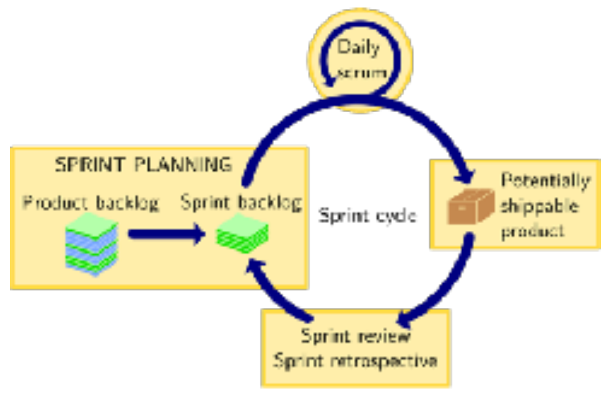
- clients may not know exact requirements upfront
- no built-in feedback
- linear, rigid, monolithic
- no anticipation of changes
- no iterations

- pros: without detailed documentation, knowledge of software can be lost
- documentation supports training of new team members or new team
- US DoD, now has stated preference against waterfall-type methodologies, starting with MIL-STD-498, which encourages evolutionary acquisition and Iterative and Incremental Development.

(See https://en.wikipedia.org/wiki/Waterfall_model, Supporting vs. Criticism)

Agile Software Development

- Adaptive planning
- Respond quickly
  - New opportunities
  - Competition
- Get feedback quickly
  - Customers
  - Team members



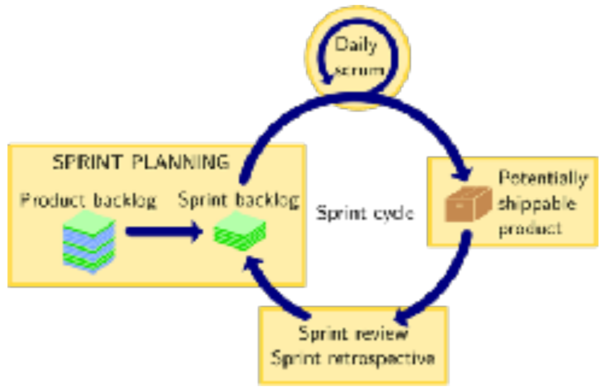
Scrum Development Workflow

For more background, see *The Agile Manifesto,* 2001

Get feedback as quickly as possible from customers and team members.
Delay planning, adapt plans as you go.
Respond quickly to new opportunities and competition.
Identifying and fixing problems early saves time and money

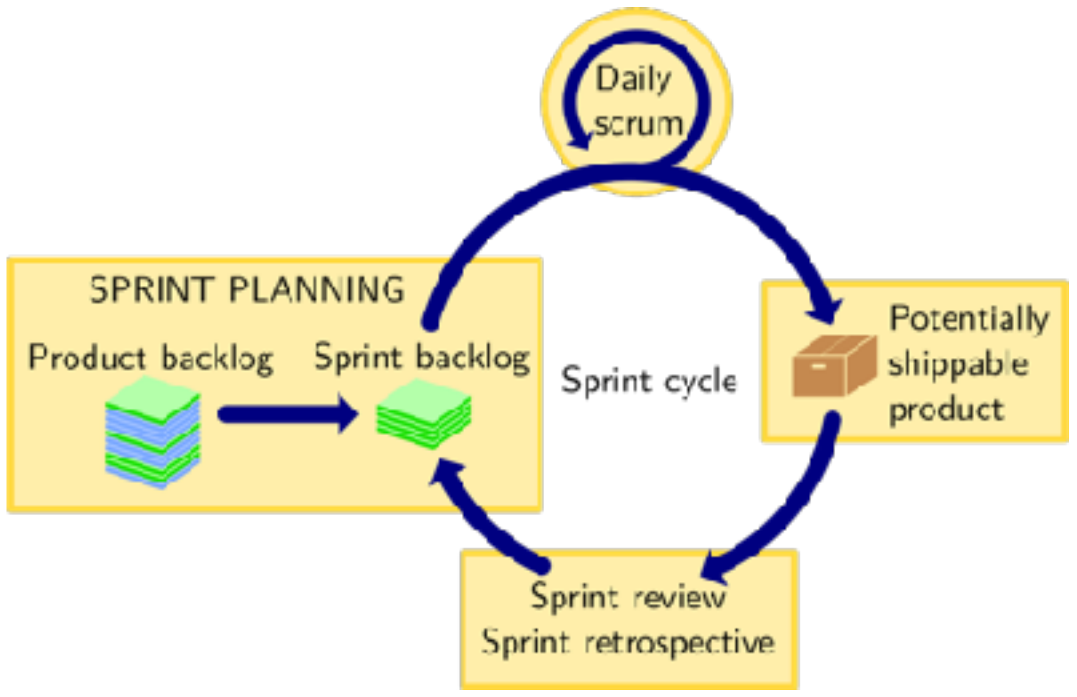
The Scrum Development Framework

- Agile
- Small teams
  - Cross-functional
  - Working closely together
- Sprints
  - Short workflow cycles
  - “Potentially-shippable” product



Based around small teams and short work cycles
Teams are self-contained and have all skills needed to do the job
Small team size facilitates communication
Short cycles mean more frequent opportunities for feedback and identifying problems
After each cycle, product is in a working state ("potentially-shippable")

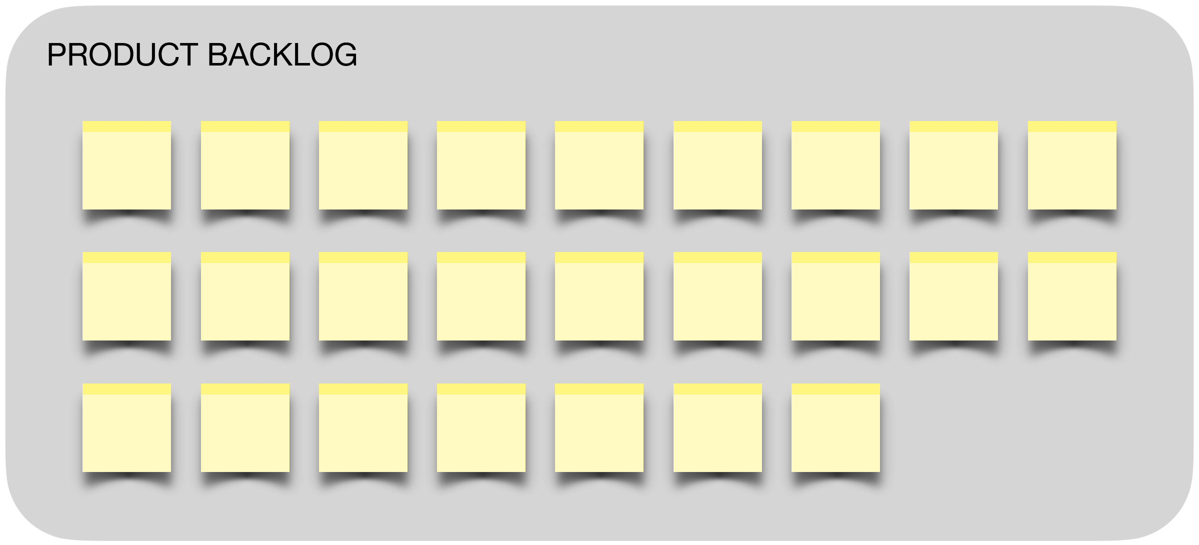
Scrum Workflow



Sprint cycle. Basic unit of work planning and completion. Relatively short duration, typically 1 week to 1 month in length.
Sprint planning. Choose the most important tasks to do during the next sprint cycle. Sprint Team decides how much it can do during the sprint.
Daily scrum. Frequent status updates to maintain communication and identify problems before they derail progress on the sprint.
Sprint review. Demonstrate completed results from the sprint and non-completed items. Review product backlog and progress to completion.
Sprint retrospective. Identify problems within the team, such as personality conflicts, external influences, inadequate resources, etc.

Product Backlog

- Each feature is in the form of a “user story”
- Living document - requirements and design evolve



- feature wishlist - what would make this product great?
- instead of a traditional requirements spec
- a living document
    - changed throughout the project
    - rather than freezing requirements early on (as in waterfall), requirements can evolve iteratively
        - maximize customer value and minimize developer effort
- each entry adds value to customer
- granularity of items not uniform - only those to be implemented in next sprints defined in greater details, and others coarse-grained
- Good documentation: https://www.scrum-institute.org/The_Scrum_Product_Backlog.php

Product Backlog Items

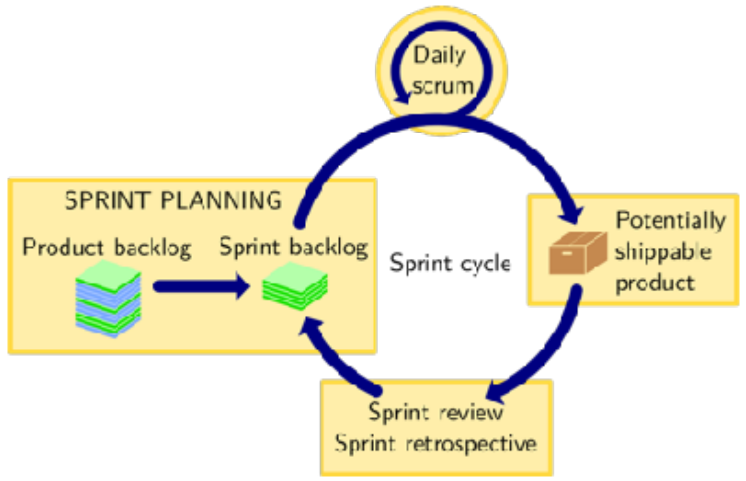
- Each item is a **user story**

As a <role>, I want <feature>, so that <benefit>

- Each item has estimated **story points**
- Each item has **priority**
- Can be edited and refined as desired

story points:
Proxy for amount of effort (time)
Easier to estimate relative effort than absolute effort
Estimates improve over life of project

Sprints



- Short cycle, 2-4 weeks
- Implement subset of backlog items
- At end of sprint, want a complete product increment
- New features should be demonstrable

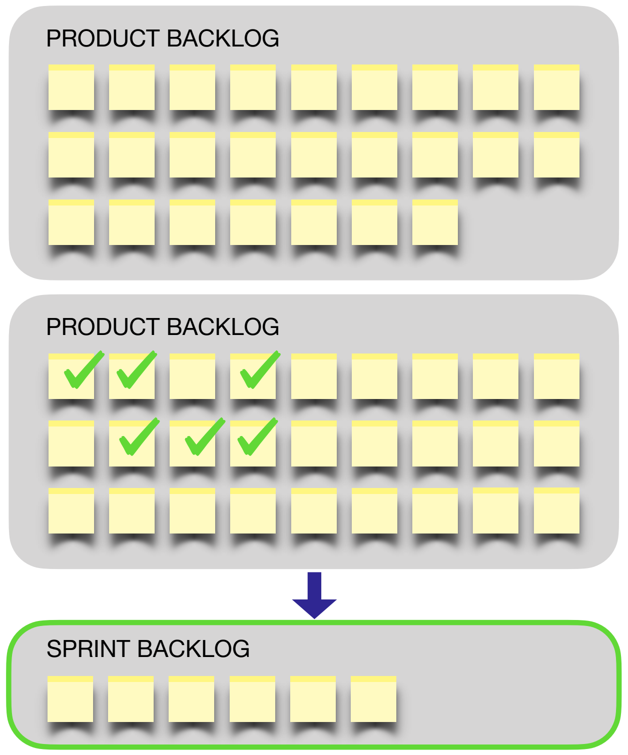
Each sprint follows a defined process
starts with planning meeting
end of sprint - review meeting - check if all items completed
retrospective - see if process can be improved
short daily standup meeting - “daily scrum”

Sprint Planning

Start with
product backlog

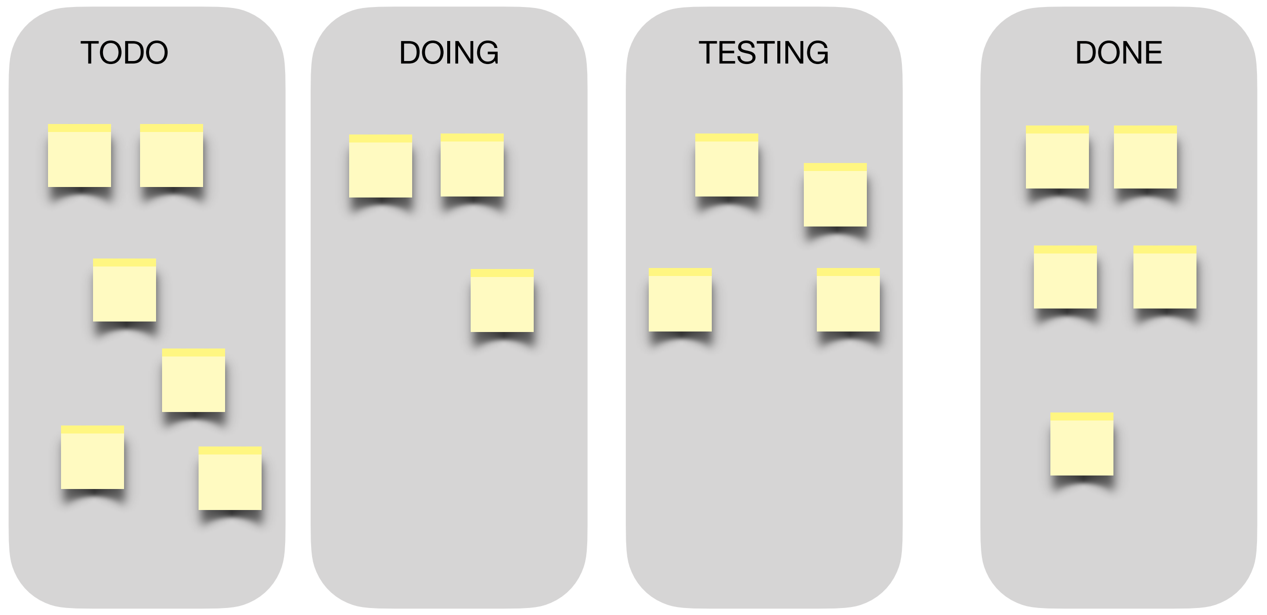
Choose user
stories to include
in this sprint

Defines the
sprint backlog



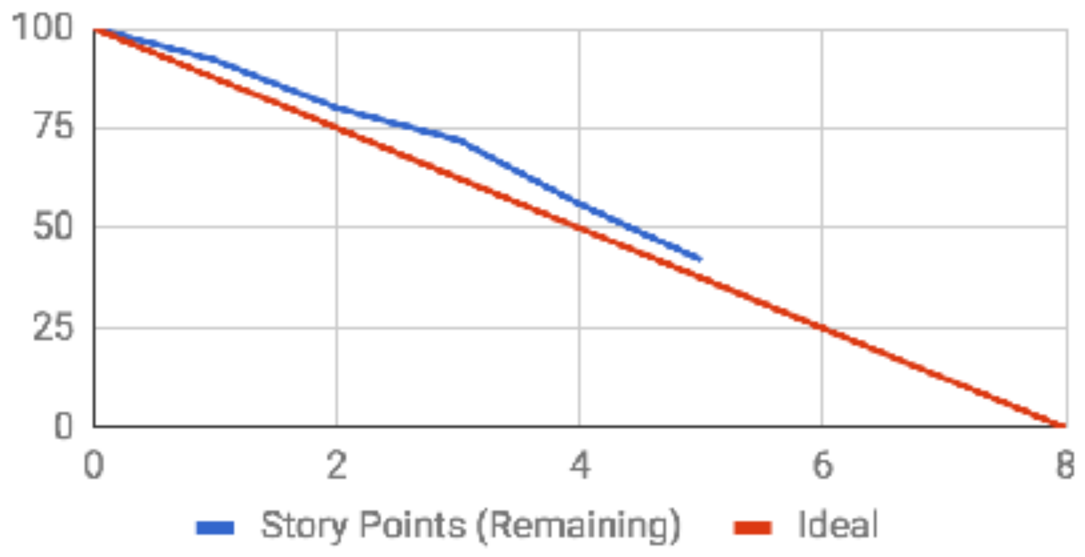
some backlog items got refined during sprint planning

Sprint Backlog



This is a visual representation of the state of the work, progress, etc.
This is called a “Kanban board” - could also have other categories as needed. e.g., blocked

Burndown Chart



Are we on track to finish in time?

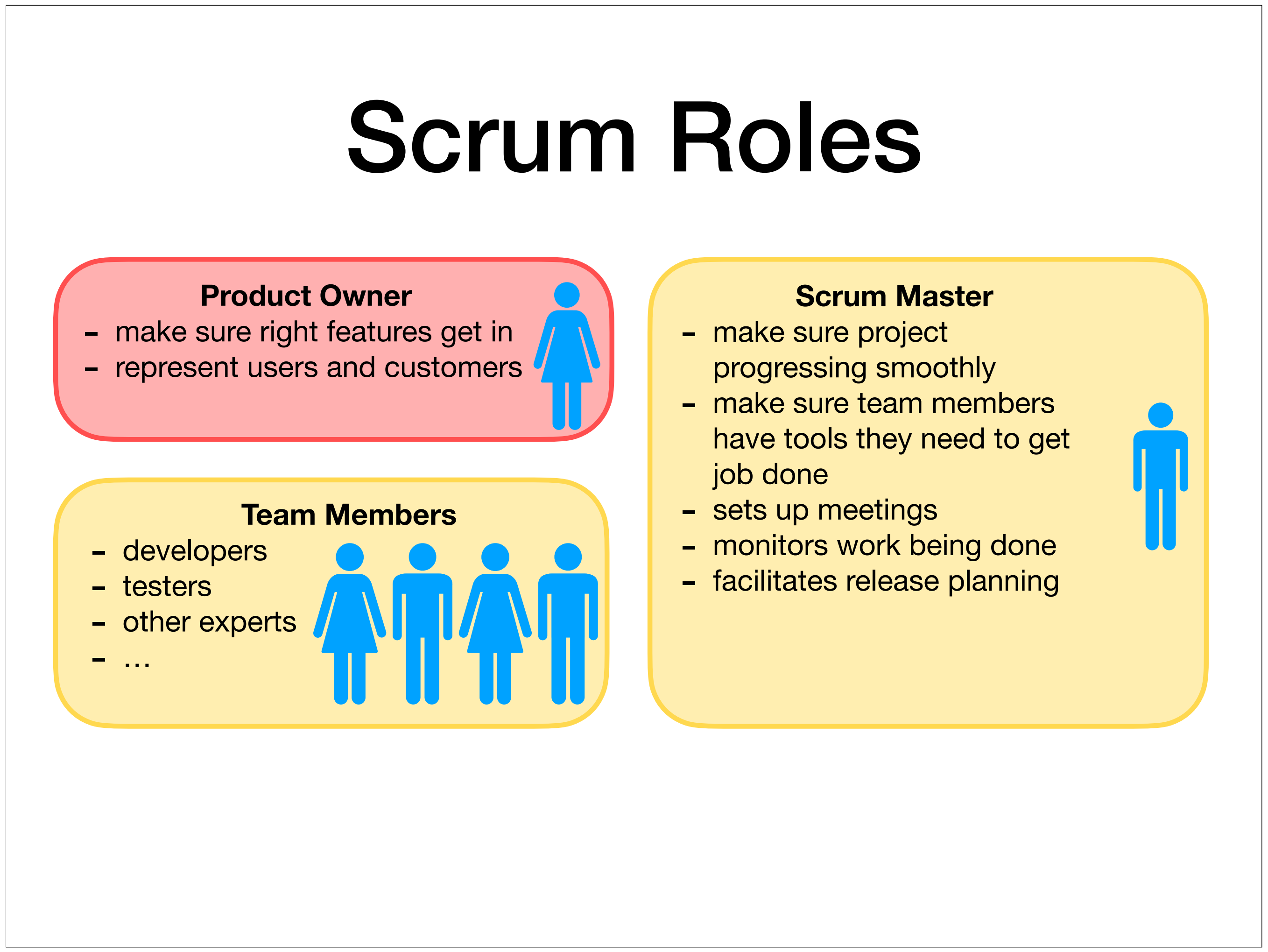
Visual indicator of progress towards completion.
Tracks amount of completed/outstanding work compared with the amount of time remaining to complete it.
Work may be measured in time or story points, which are a proxy for relative time.
Total story points may go up or down, such as due to adjusting requirements or updating effort estimates.

Scrum Roles

Product Owner
- make sure right features get in
- represent users and customers

Team Members
- developers
- testers
- other experts
- …

Scrum Master
- make sure project progressing smoothly
- make sure team members have tools they need to get job done
- sets up meetings
- monitors work being done
- facilitates release planning



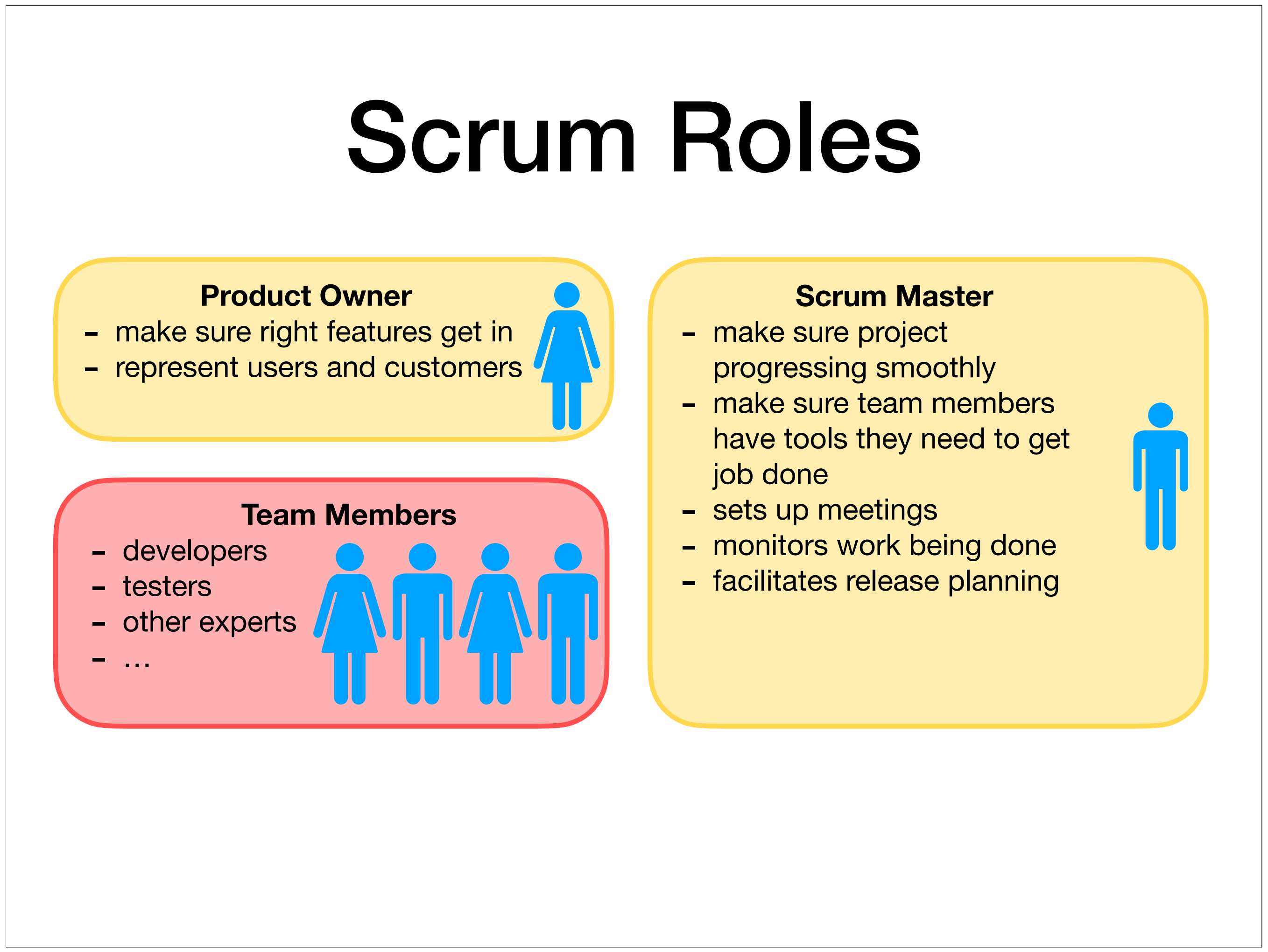
Scrum Roles

Product Owner
- make sure right features get in
- represent users and customers

Team Members
- developers
- testers
- other experts
- …

Scrum Master
- make sure project progressing smoothly
- make sure team members have tools they need to get job done
- sets up meetings
- monitors work being done
- facilitates release planning

Product owner makes sure that the final product will meet the needs of the customers.
Describes and prioritizes backlog items.
Makes sure team understands what is expected for each item.
Decides what features make it in and which do not.



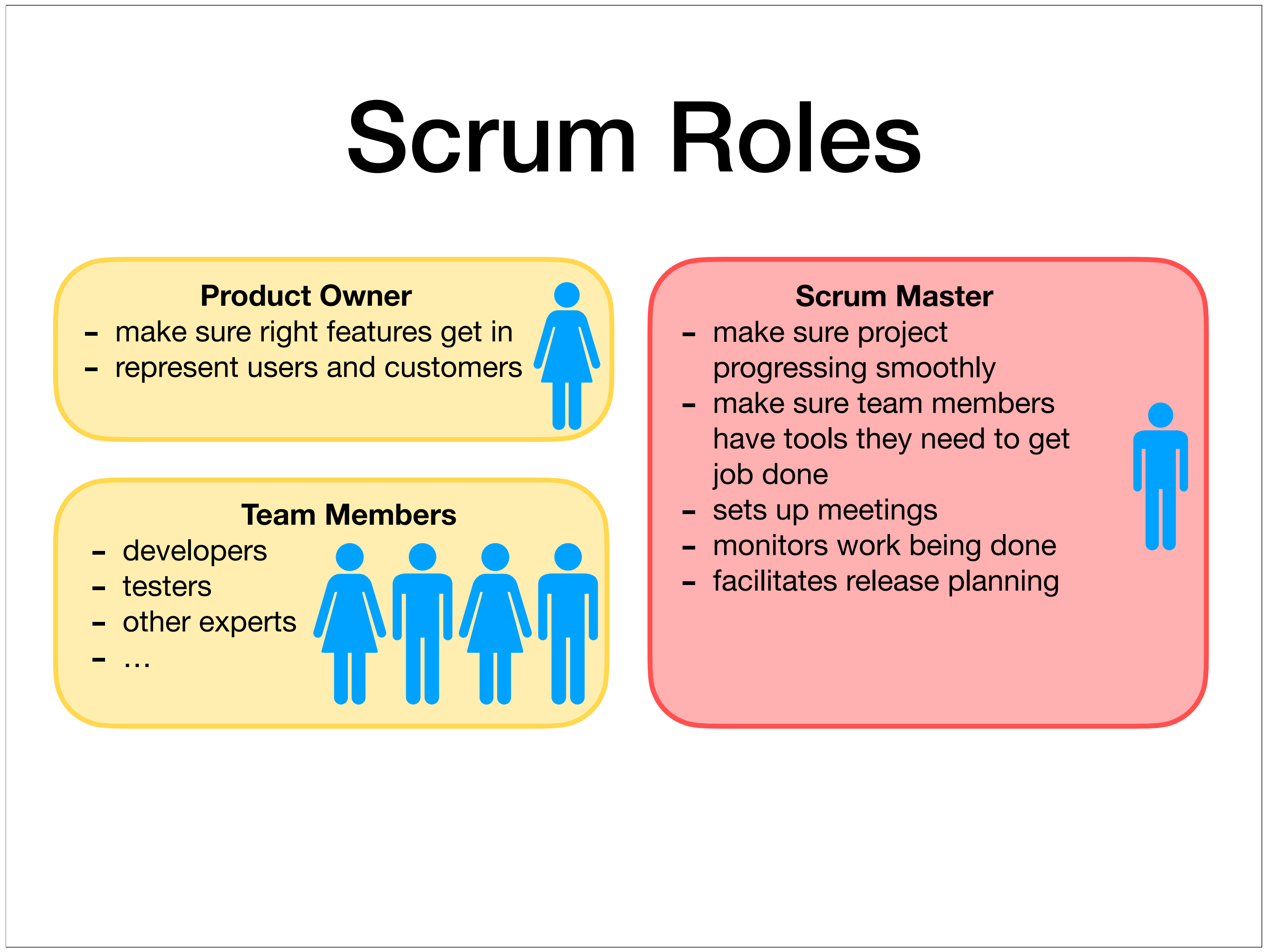
Scrum Roles

Product Owner
- make sure right features get in
- represent users and customers

Team Members
- developers
- testers
- other experts
- …

Scrum Master
- make sure project progressing smoothly
- make sure team members have tools they need to get job done
- sets up meetings
- monitors work being done
- facilitates release planning

Team members do the work.
Programmers, testers, experts, artists, etc.
Self organizing; the team decides for itself how to do the work.
No subgroups; credit and responsibility belongs with the team as a whole.



Scrum Roles

Product Owner
- make sure right features get in
- represent users and customers

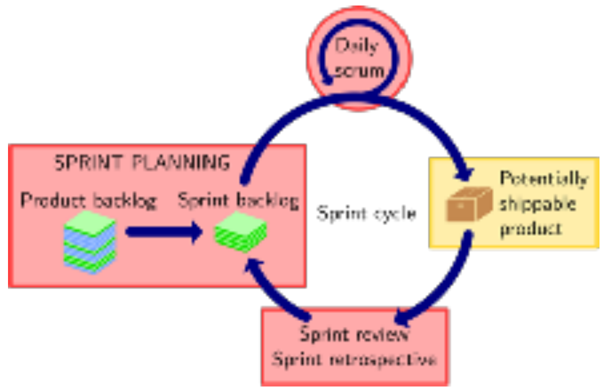
Team Members
- developers
- testers
- other experts
- …

Scrum Master
- make sure project progressing smoothly
- make sure team members have tools they need to get job done
- sets up meetings
- monitors work being done
- facilitates release planning

Sets up meetings and makes sure the scrum process is being followed.
For team:
	Scrum master makes sure the team members can do their jobs efficiently.
	Makes sure resources are available.
	Shields team from outside pressures, moderates interactions with outside.
For owner:
	Helps team understand the needs and goals of the owner
	Helps team understand the need for clear and concise backlog items
	Helps owner understand how to arrange backlog

Scrum Meetings

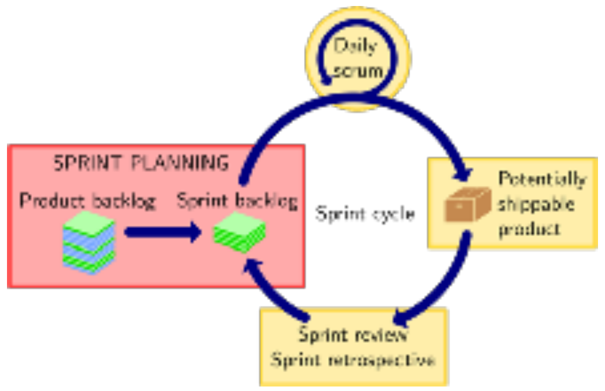
- Ensures communication
  - Planning
  - Feedback
  - Identify problems



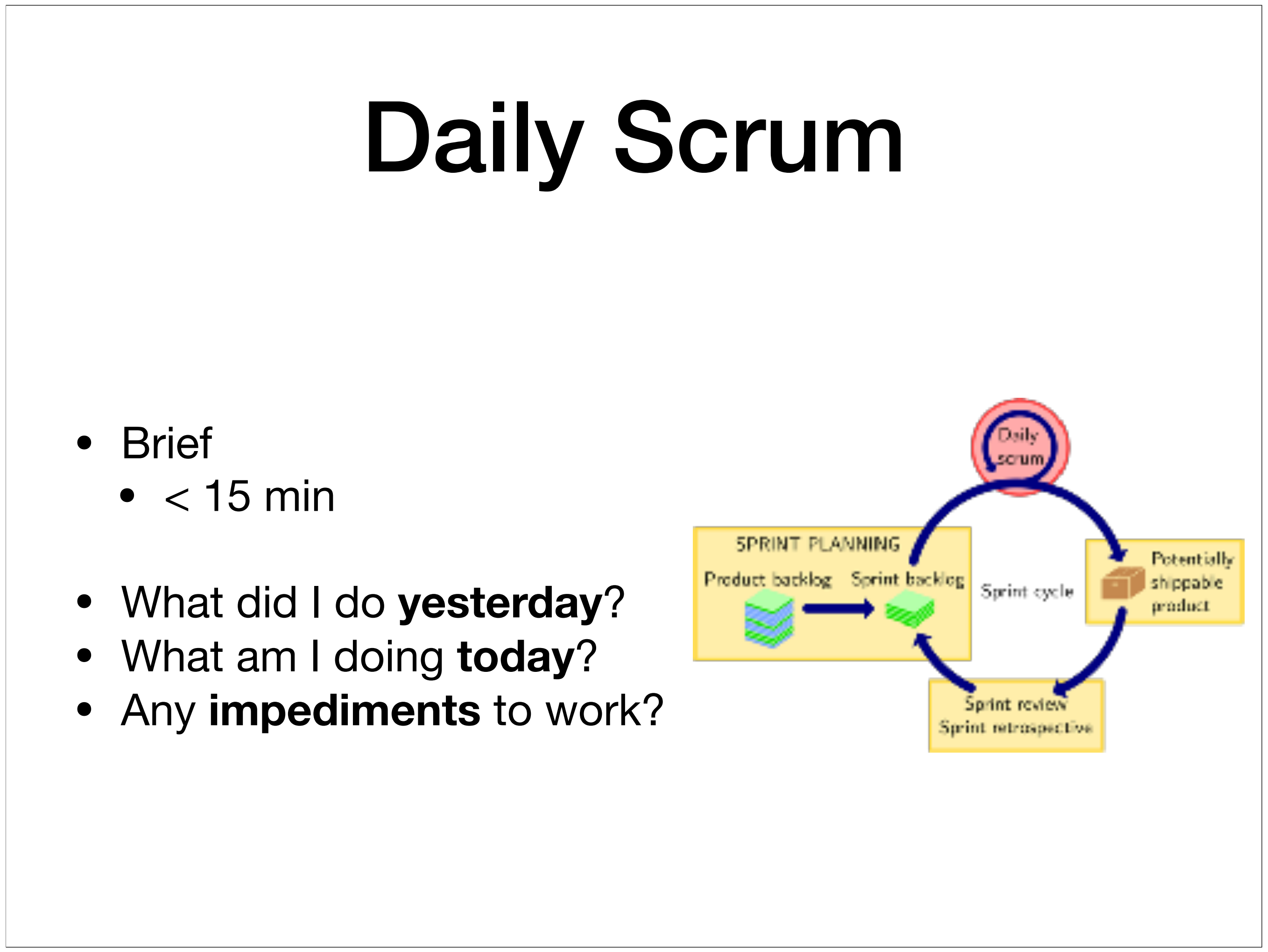
Because everyone loves meetings. :)

Sprint Planning

- Plan next sprint goal
- Select from **product backlog**
- Add to **sprint backlog**
- Forecast, not commitment



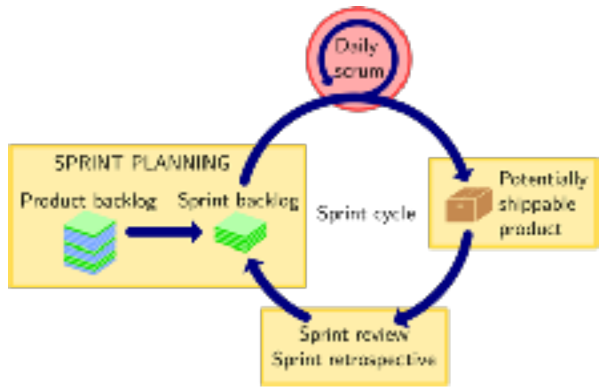
Product owner prioritizes items, but the team alone decides which items and how much it will select for the upcoming sprint.
Team takes items from the product backlog, which becomes the sprint backlog.
The time frame of the sprint is fixed, not the amount of work accomplished. Sprint goals may not all be achieved during sprint.



Daily Scrum

- Brief
  - < 15 min

- What did I do **yesterday**?
- What am I doing **today**?
- Any **impediments** to work?



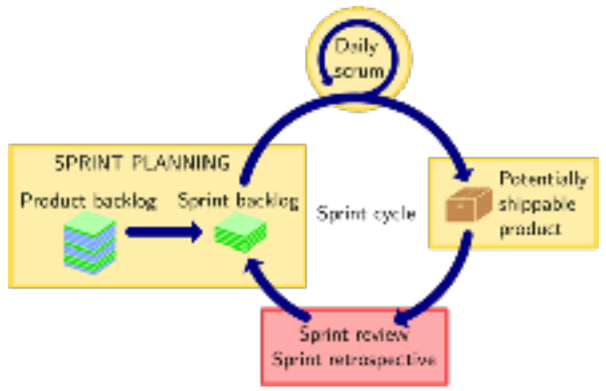
Daily scrum provides frequent feedback, ideally every day.
Efficient communication to facilitate efficient work
For this class, probably not every day, since most team members will not be able to work on the project every day.
Over chat, in lab.
Very short, not more than 15 minutes. Often "stand up"

- What did I do **yesterday**?
- What am I doing **today**?
- Any **impediments** to work?

Not a status meeting.

Sprint Review

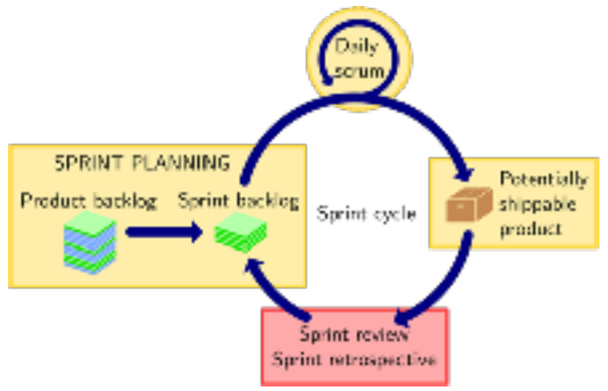
- Team members
  - Demo work
    - What is (not) done?
  - What went well?
    - Problems, resolutions?
- Product owner
  - Product backlog
    - Status (on schedule?)
    - Adapt if needed



Review results of the sprint. Demonstrate items that were completed. Identify what was not completed.
Discuss what went well. Discuss what problems came up, what was done about them.
Adapt product backlog items as needed.
Owner describes state of product backlog, estimated completion date (on schedule?)

Sprint Retrospective

- Improve team performance
  - What is working?
  - What needs improvement?



Reflection on the team's efficiency, not the sprint contents.
What can be done to make the team more efficient?
What is making it less efficient than it could be? Interpersonal dynamics? Missing resources?

Resources

Short video introduction:

https://www.youtube.com/watch?v=XU0llRltyFM&list=PLR-Qzc9Bn-gdymXF6zr_3o8d0fi9Zrs2Z

Free, short book:

https://storage.googleapis.com/scrum-institute/Scrum_Revealed_by_International_Scrum_Institute.pdf

Other:

https://www.scrum-institute.org/The_Scrum_Product_Backlog.php