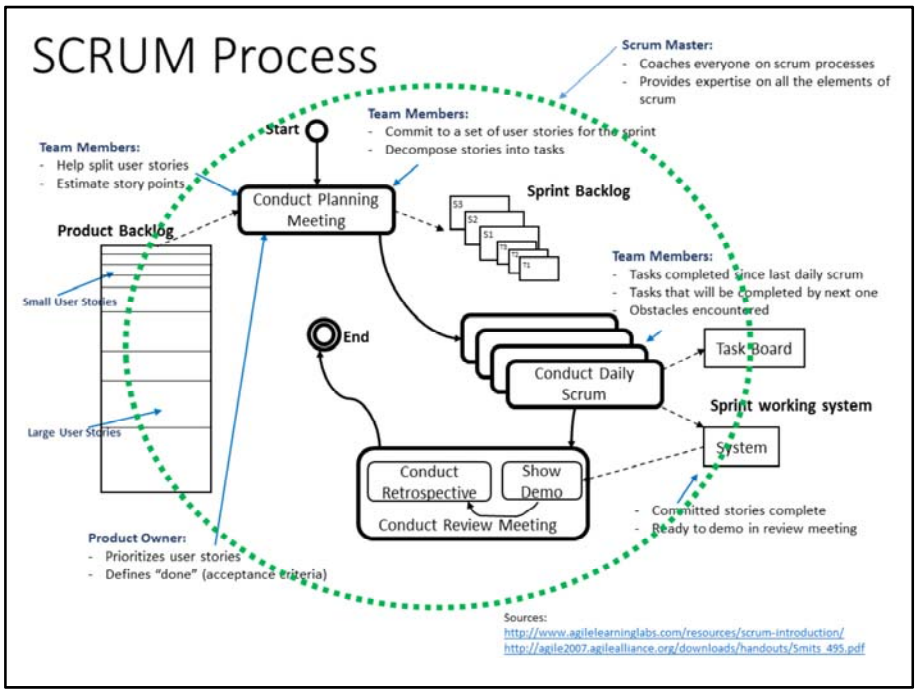


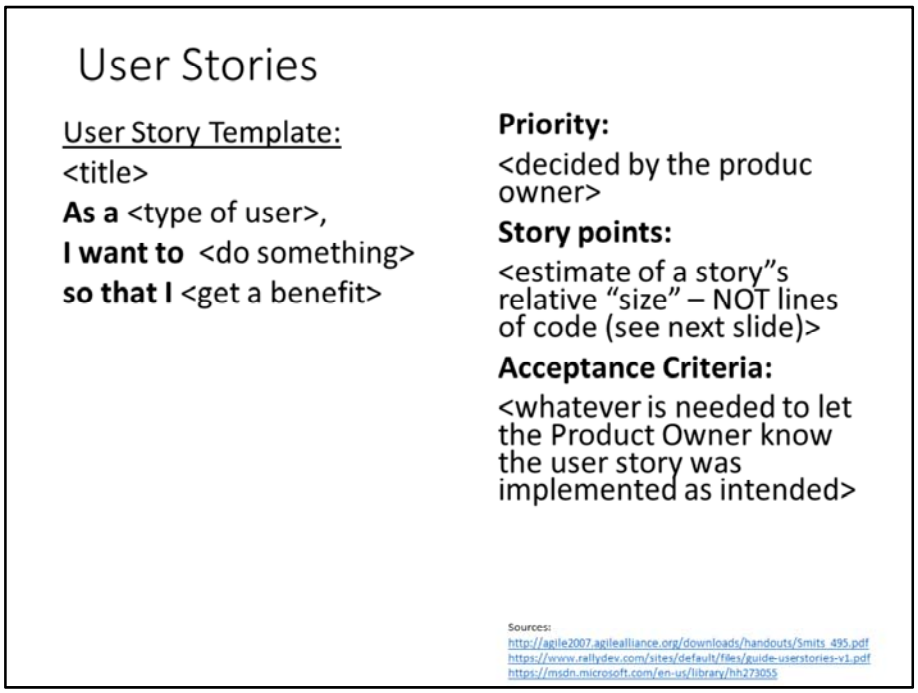

There are 3 main activities during each Scrum sprint:

- **A planning meeting where:**
  - the Product Owner prioritizes user stories in the product backlog that need to be implemented during the sprint
  - the Team commits to the stories they will implement during the sprint
  - the Team Members decompose the committed stories into tasks with estimation times (or at "grooming" meeting)
  - the Team Members split user stories that are still large, and estimate their story points – a comparative effort estimate
  - Team Members and the Product Owner develop acceptance criteria for each user story so that everyone understands what is needed to declare the story "done" - "Definition of Done" means acceptance criteria are met
  - The **input** to the planning meeting is the **product backlog**, a prioritized set of user stories.
  - The **outcome** of the planning meeting is the **sprint backlog** – a set of user stories and all the tasks needed to complete them that the team has committed to complete for the sprint.
- **Daily Scrum meetings where:**
  - Each member of the team states:
    - what tasks they accomplished since the last scrum
    - what tasks they plan to accomplish before the next scrum
    - any obstacles they have encountered; other team members volunteer to help if they are working on a lower priority user story
    - The **input** artefacts to the daily scrum and its **output** artefacts are the states of each task in the sprint: to do, doing, and done. Shown on a task board.

Note that the **outcome at the end of the sprint, before the review meeting** is a working system that implements all the committed stories of the sprint.

- **A Review meeting where:**
  - The team demos to all interested parties what they have accomplished over the sprint – the user stories they have implemented
  - The team holds a private retrospective of what worked and didn't, and decides how to improve the next sprint
  - The **input** to the demo portion of the meeting is the working system with committed sprint user stories implemented and fully tested
  - The **output** of the retrospective portion of the meeting is a set of improvements that will be made to the next sprint

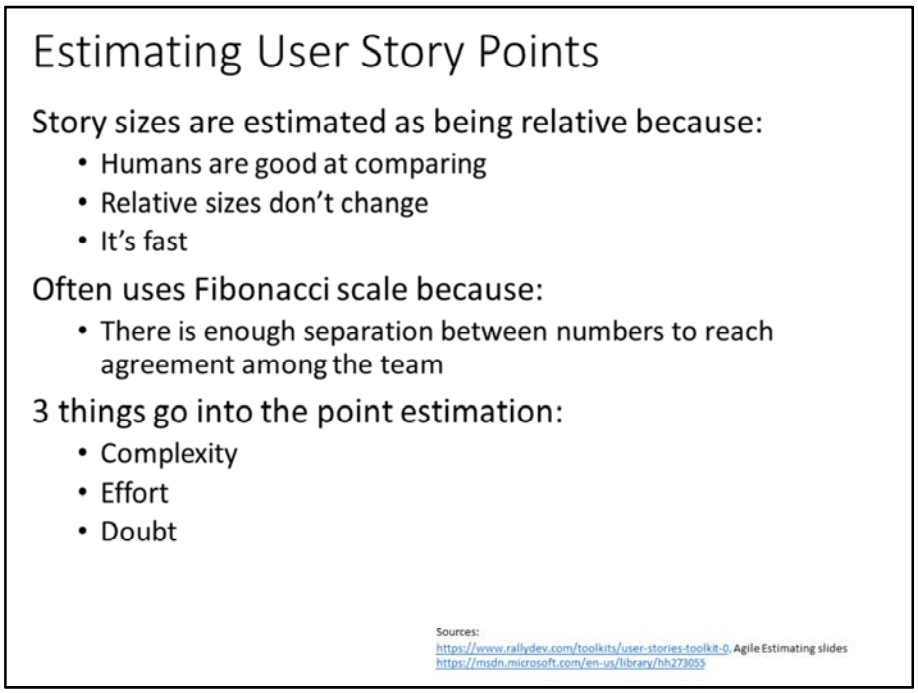
**The Scrum Master provides coaching and expertise at all points of the process and for all the artefacts of the process.**




A user story must always include its acceptance criteria. The acceptance criteria must be defined so that a simple pass/fail can be assigned to each element of the criteria. The product owner assigns priorities to user stories. The team members assign story points to the story. Initially, a story probably isn't well-understood and can be pretty large. In this case the story point estimation is tentative. As the user story moves up in prioritization, it needs to be split into smaller, well-understood stories. Smaller stories are easier to estimate so the estimate becomes less tentative. By the time a user story rises to the top of the product backlog to be included in the next sprint, the story must be small enough that tasks to implement it (including its acceptance criteria) can be identified, and the time to complete the task can be well estimated.

Sometimes there is too much unknown about some part of a user story. In this situation the team can define a "Spike".
From: http://agiledictionary.com/page/4/

**Spike**
A story or task aimed at answering a question or gathering information, rather than at producing shippable product. Sometimes a user story is generated that cannot be estimated until the development team does some actual work to resolve a technical question or a design problem. The solution is to create a "spike," which is a story whose purpose is to provide the answer or solution. Like any other story or task, the spike is then given an estimate and included in the sprint backlog.




Adapted from: http://agiledictionary.com/page/3/

**Planning Poker**

An estimating tool in Agile, Planning Poker is a structured game used to reach group consensus while estimating tasks. It is called poker because it does indeed involve a deck of cards.

Game Play: Each card in the deck is inscribed with a number in the Fibonacci sequence, in which each number is equal to the sum of the previous two: 1, 2, 3, 5, 8, 13, 21, etc. Team members each choose a card that represents their best guess as to the difficulty level of the user story, and everyone turns over their cards at once. The high and low cards get the floor to argue their cases, and after that, the game is repeated, the theory being that the players' estimates will converge through rounds of structured discussion. A Project Manager or Scrum Master may serve as moderator.

In CS 314, you can use the game to estimate complexity, effort, and doubt individually and then add them up to obtain story points, or you can just estimate difficulty level.

## Decomposing a User Story into Tasks

- Tasks should be small enough that the estimated time needed for them should be 1-10 hours
- Break the user story into as many tasks as needed so that if all are completed the acceptance criteria will be met.
  - Defining test data, designing test approaches (e.g. unit test, GUI tests, integration tests, …) must be included as tasks.
- Sometimes "non-functional" requirements (e.g. performance or security-related) are included as split user stories and have their own set of related tasks.
- Regularly occurring tasks related to acceptance criteria for every user story can basis of the "definition of done" (e.g. all tests pass and there are no outstanding defects).

Sources:
https://www.rallydev.com/toolkits/iteration-planning-toolkit, Testing in the Iteration slides

From http://agiledictionary.com/:

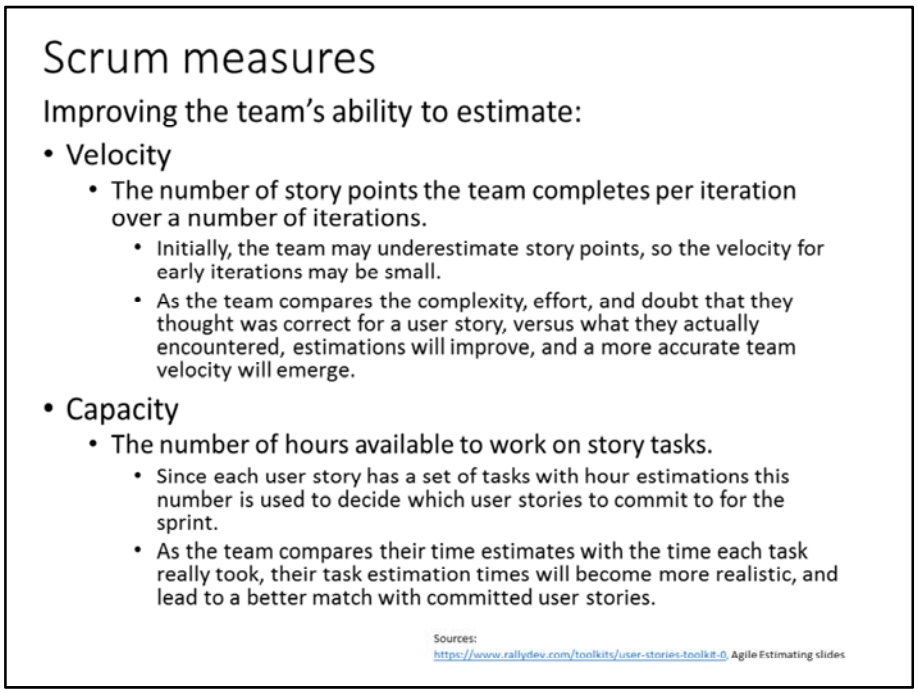
**Definition of Done**

Also: Task Complete Definition, Punch List

A Team's universally agreed-upon criteria for what makes a unit of work "potentially shippable." This checklist of steps that must be completed for each unit of work (e.g., task or user story) may include items like "documentation created," "code review completed," "all tests created and passing," etc. The Definition of Done usually takes the form of an information radiator, being posted prominently in the team's work space.

A well-crafted Definition of Done may prevent the accumulation of technical debt that naturally arises when team members define done loosely and colloquially.

In Extreme Programming, the Definition of Done may be referred to as Task Complete Definition, a Punch List, or a Binary Milestone.




From http://agiledictionary.com/page/5/:

**Velocity**

The rate at which an Agile completes work, used not to measure progress per se, but to accurately estimate the team's capacity for future iterations and guide the team and product owner in planning upcoming iterations.

In the first iteration, a team has no velocity; they take a flying guess at how much work they will complete. But from iteration two onward, velocity is derived from the amount of work the team completed in the previous iteration or iterations.

Different teams use different units of measure for estimating work and determining velocity, some of the most common being: story points (Scrum), craft units (XP), ideal days or hours. The team may determine its velocity by averaging the amount of work completed to date over the number of iterations completed (work/time), or simply by taking the amount of work completed in the previous iteration and carrying it forward.

The data used to determine a team's velocity may be documented on the project Burn Down and Burn Up charts.