# Scrum

## Scrum is a framework, not a method

- Scrum is a framework of practice tied together by a small set of clarity roles
- Difference between method and framework
  - Method implies “one size fits all” approach
  - A framework offers a more flexible platforms from which a variety of approaches can be derived depending of environment
- Scrum is
  - Lightweight
  - Simple to understand
  - Difficult to master

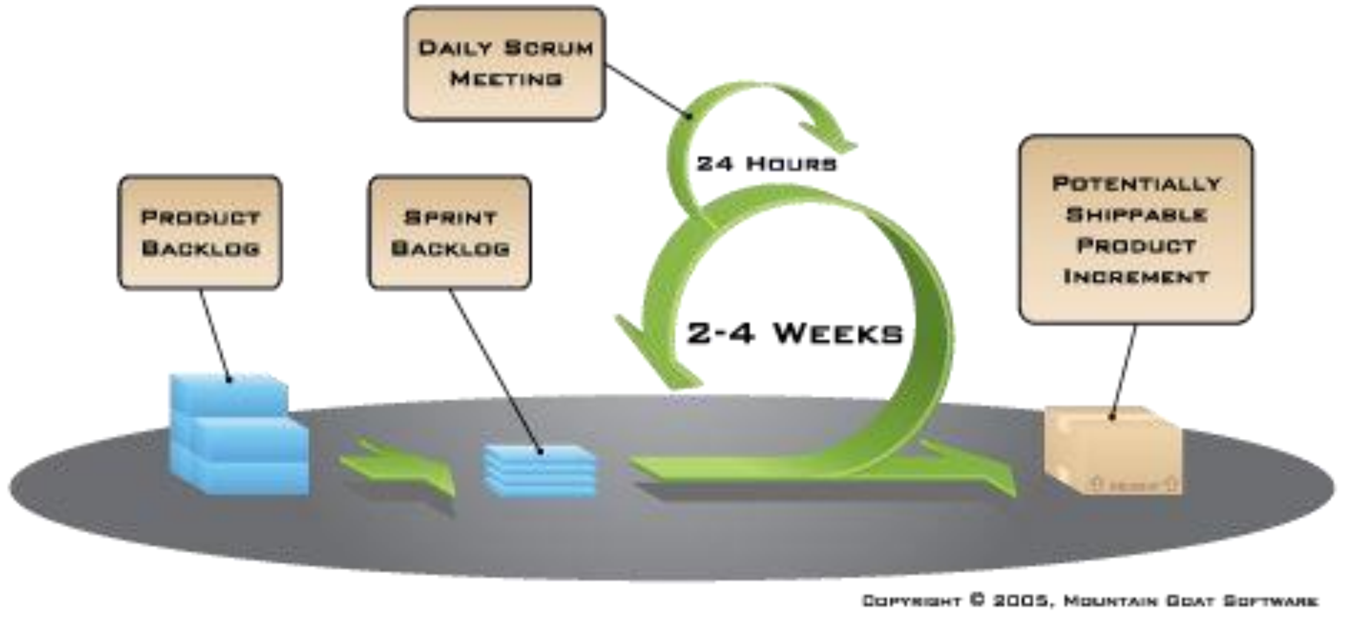
## Scrum elements

- Scrum framework consists of Scrum Team and their associated roles, events, artifact and rules
- Scrum employs iterative, incremental approach to optimize predictability and control risk
- Founded on empirical process
- Three pillars uphold every implementation of Scrum
  - Transparency
  - Inspection
  - Adaption
- Scrum prescribe 4 events for inspection and adaption
  - Sprint planning
  - Daily scrum
  - Sprint Review
  - Sprint Retrospective

## Scrum characteristics

- Self-organizing teams
- Product progress in series of “sprint”
- Requirement are captured as item in a list of product backlog
- No specific engineering practice prescribed

## Put it all together


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

## Sprint

- Heart of a Scrum is Sprint
  - Time-box one month or a less
  - During which potentially shippable product increment is created
  - Sprint consist Planning, Daily Scrum, Dev work, Review and Retrospective meetings
  - Sprint has a Goal
  - Sprint limit risk on one month of coast
  - New sprint starts immediately after the conclusion of previous Sprint
  - During Sprint
    - No scope changes
    - Quality goals do not decrease
    - Scope may be clarified between PO and Dev Team

## Scrum framework

Roles

- Product Owner
- Scrum Master
- Team

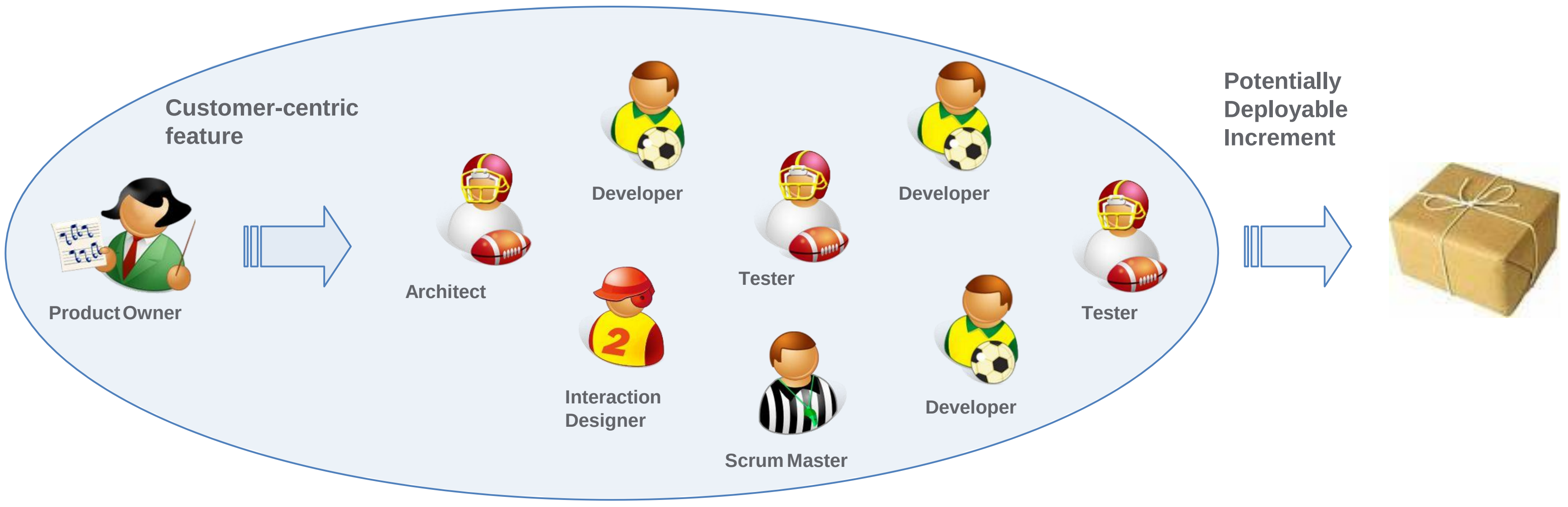
Ceremonies

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artifacts

- Product backlog
- Sprint backlog

# Scrum Team



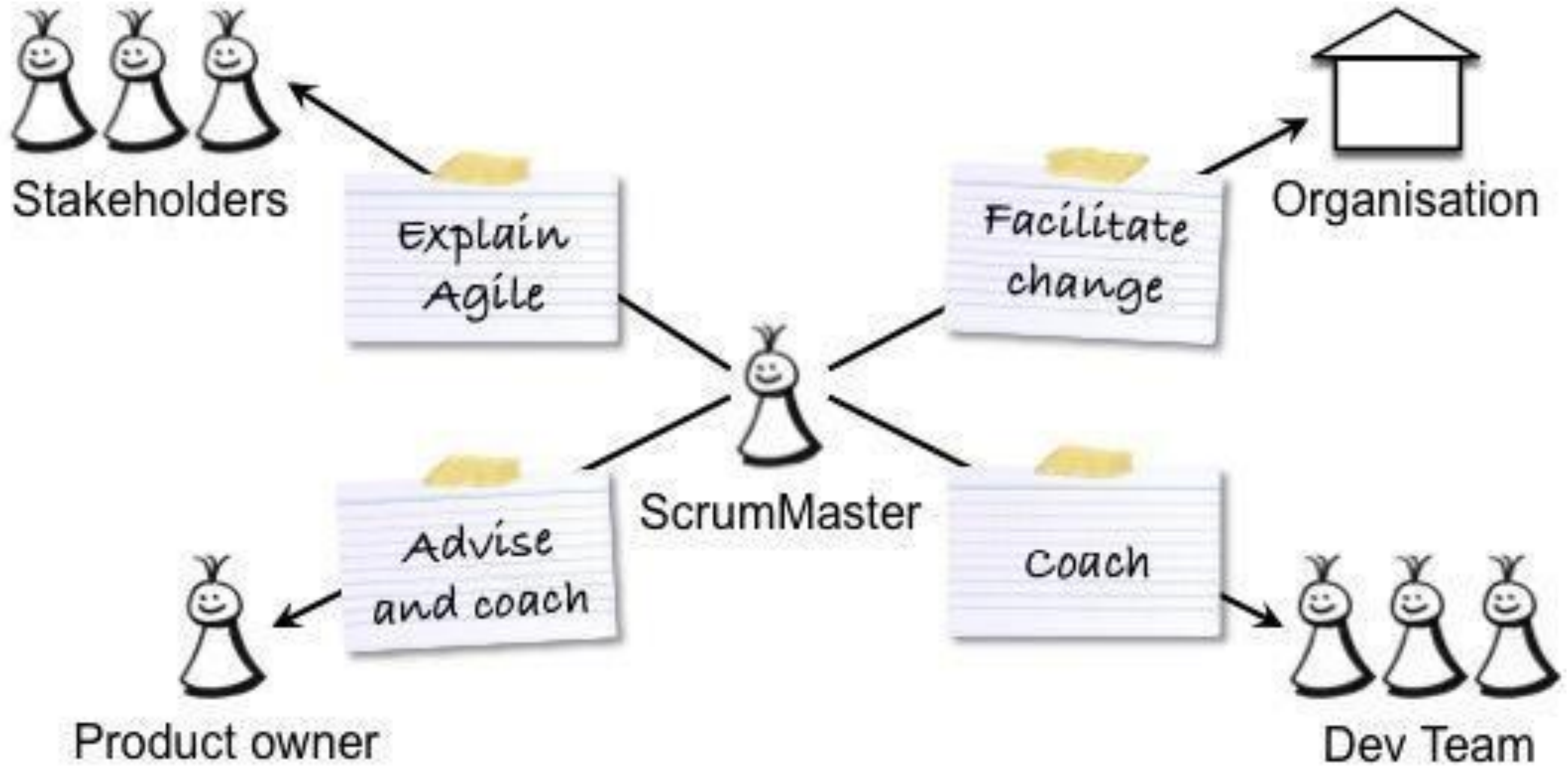
- Scrum Team consists of Product Owner, Dev team and Scrum Master
- Scrum Team
    - Self-organizing
    - Cross-functional
    - Delivers products iteratively/incrementally

# Product Owner - PO

- **Maximizing** the **value** of the **product**
- **Optimizing** the **value** of the **work** the Development Team does
- Managing the **Product Backlog**
- **No changes** that endanger the Sprint Goal

## Scrum Master (SM)

- Responsible for ensuring Scrum is understood and enacted, and that the Scrum Team adheres to Scrum theory, practices, and rules
- Keeps a Dev Team working at its highest level of productivity by facilitating Dev



## Development Team

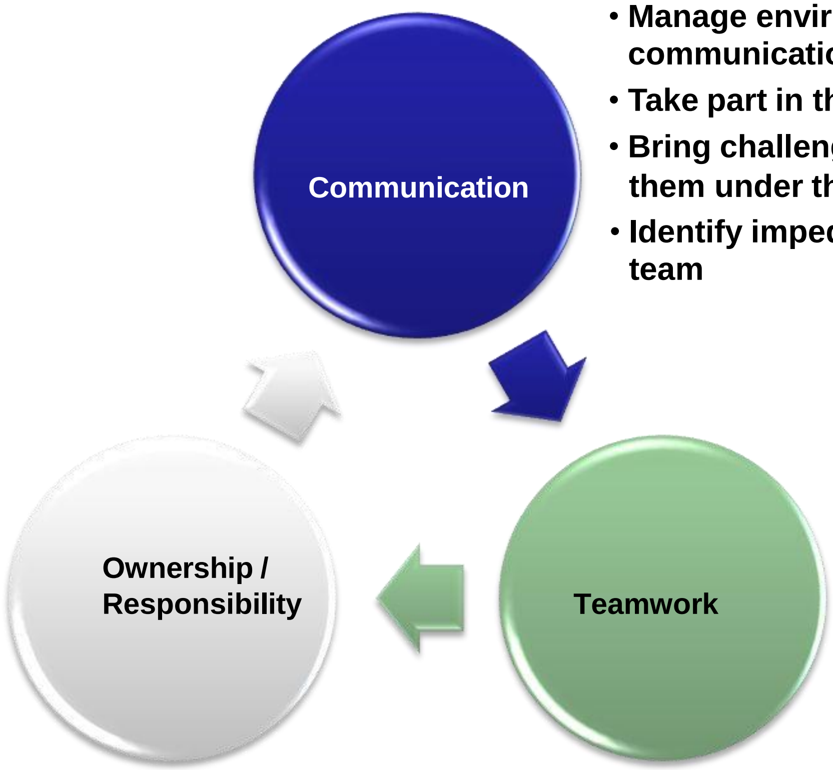
- Development team consists of professionals who do the work of delivering a potentially shippable increment of “Done” product at the end of each Sprint

self-organising

cross-functional

long-living

# Development Team members



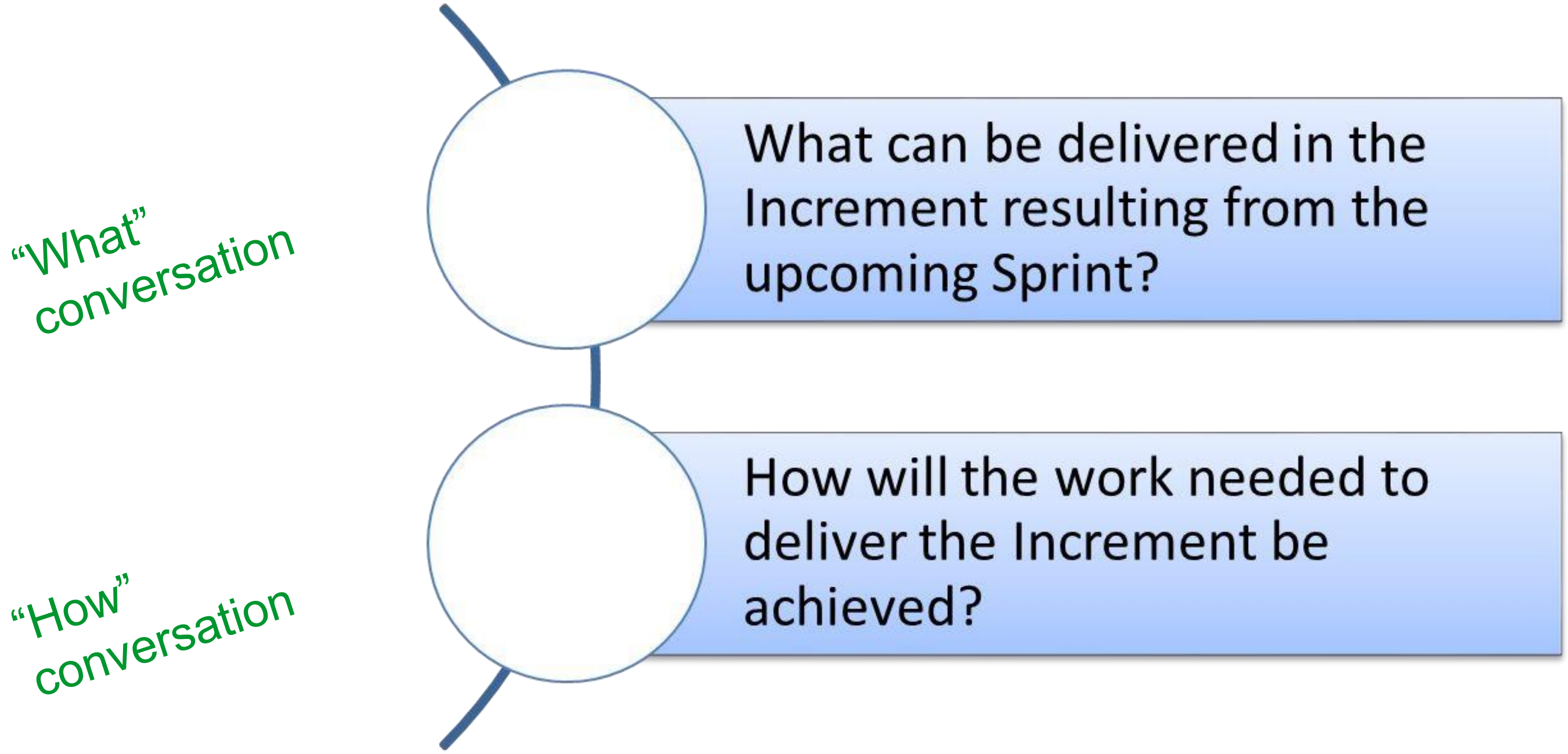
- Value face to face communication
- Manage environment to promote communication and teamwork
- Take part in the Daily Scrums
- Bring challenges into the open, not sweeping them under the carpet
- Identify impediments and share them with the team

- Take ownership of all the work the team commits to
- Follow up actions from the Sprint Retrospectives, look to continually improve
- Meet with Product Owner, Users and Stakeholders frequently to clarify tasks and monitor progress

- Solve problems rather than blame others
- Ask for assistance or guidance when required
- Aware of what the other Team Members are doing
- Help other Team Members, professionally and personally

# Sprint planning

- Work to be performed in the Sprint is planned at the Sprint Planning meeting
- **Time-boxed** to a maximum of 8 hours for 1-month Sprint
    - SM teaches the Scrum Team to keep Sprint planning within the time-box
- SM ensures that the event takes place and attendants understand its purpose


"What" conversation
What can be delivered in the Increment resulting from the upcoming Sprint?
"How" conversation
How will the work needed to deliver the Increment be achieved?

## Daily scrum

- Daily Scrum is a **15-minute time-boxed event** for the Development Team to **synchronize activities** and **create a plan** for the next 24 hours
- Held at the **same time and place each day** to reduce complexity
- Dev Team uses Daily Scrum to **inspect progress toward the Sprint Goal** and to inspect how progress is trending toward completing the work in Sprint
- Daily Scrum optimizes the probability that Dev Team will meet the Sprint Goal
- **Scrum Master ensures** that the Development Team has the meeting, but the **Development Team is responsible** for conducting the Daily Scrum
- **Input** to meeting should be how the team is doing toward meeting Sprint Goal
- **Output** should be a new or revised plan that optimizes the team’s efforts in meeting the Sprint Goal.

# Sprint review

- Sprint Review is held at the end of the Sprint to **inspect the Increment** and **adapt the Product Backlog** if needed
- Sprint Review is 4 hour time-boxed meeting for 1 month Sprints.
    - Scrum Master teaches all to keep it within the time-box
- SM ensures that event takes place and attendants understand its purpose.
- *Team is allowed to demonstrate only those User stories that are* ***truly done***
- Result of Sprint Review:
    - A revised Product Backlog that defines probable PBIs for next Sprint, which should optimise value of work people are doing

## Sprint retrospective

- Sprint Retrospective is an opportunity for the Scrum Team to **inspect itself** and **create a plan for improvements** to be enacted during the next Sprint
- Sprint Retrospective is 3 hour time-boxed meeting for 1 month Sprints
  - SM teaches all to keep it within the time-box
- SM ensures that the event takes place and attendants understand its purpose

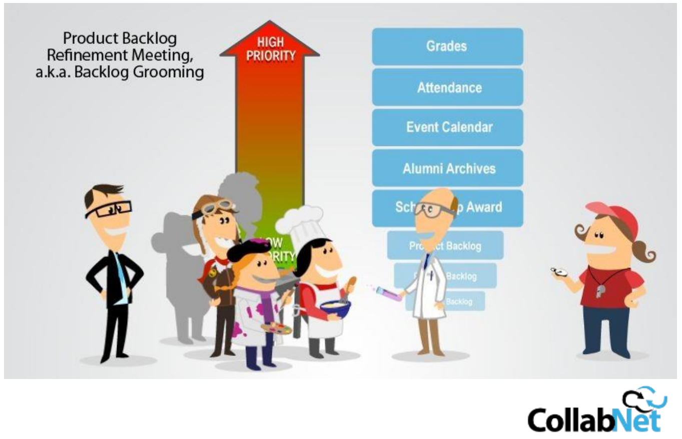
## Sprint retrospective

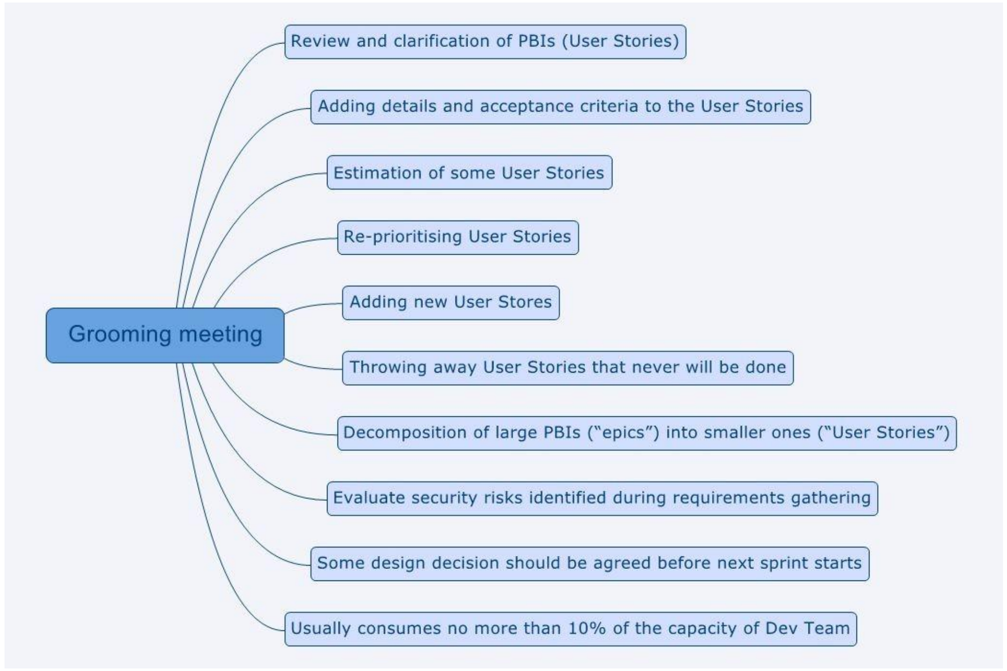
- The **purpose** of the Sprint Retrospective is to:

  - Inspect how the last Sprint went with regards to people, relationships, process, tools
  - Identify and order the major items that went well and potential improvements
  - Create a plan for implementing improvements to the way Scrum Team does its work

# Product Backlog Refinement (Grooming) Meeting

- Product backlog refinement is **ongoing process** in which PO & Dev Team collaborate on the details of PBIs

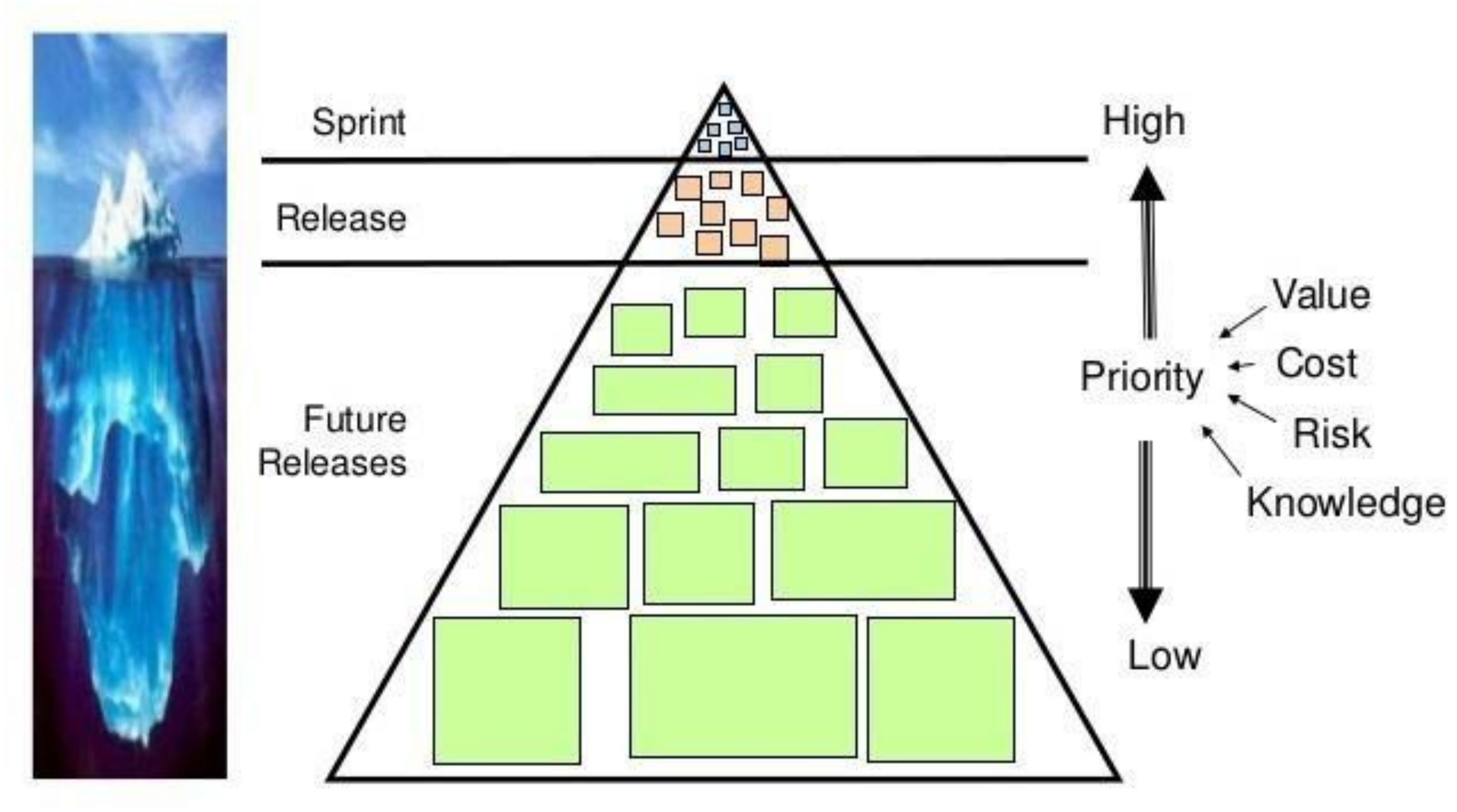



Grooming meeting
Review and clarification of PBIs (User Stories)
Adding details and acceptance criteria to the User Stories
Estimation of some User Stories
Re-prioritising User Stories
Adding new User Stores
Throwing away User Stories that never will be done
Decomposition of large PBIs ("epics") into smaller ones ("User Stories")
Evaluate security risks identified during requirements gathering
Some design decision should be agreed before next sprint starts
Usually consumes no more than 10% of the capacity of Dev Team

- 
- Scrum Team decides how and when refinement is done (recommendation is for this meeting to happen in the last few days of the sprint)

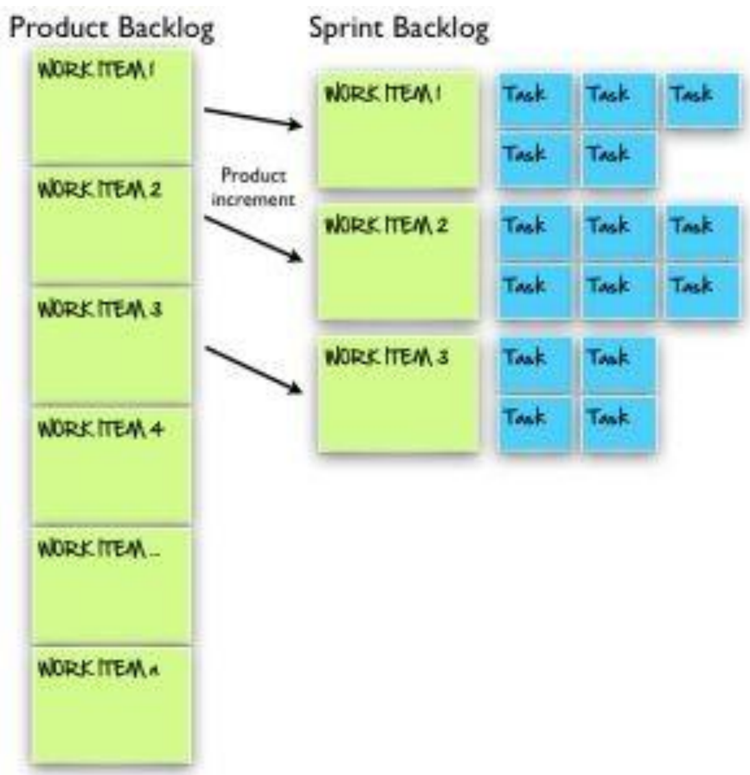
# Product backlog

- Product Backlog is an **ordered list of everything** that might be needed in the product and it is the **single source of requirements**



# Sprint backlog

- Set of PBIs selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal



# Agile planning and estimation

## What makes planning Agile

- Agile planning is more focused on planning than the plan
- Agile planning encourage change, embrace for change
- Agile planning results in plans that are easily changed
- Agile planning is spread throughout the project

## Story points

- Probably the most commonly used estimating unit among Agile teams today
- How long user will take (effort)
- Influenced by complexity, uncertainty, risk, volume of work etc...

## 3 key advantages of estimating in Story points

- Forces the use of relative estimating
  - Studies have shown we’re better at this
- Focuses us on estimating the size, not the duration
  - Time based estimates are not additive
- Puts estimates in units that we can add together
  - We derive duration empirically by seeing how much we complete per iteration

## Technique for Estimating

- Estimate by Analogy
- Disaggregation
- Planning poker

## Estimate by Analogy

- Comparing a user story to others
  - “This story is like that story, so its estimate is what that story’s estimate was.”
- Don’t use a single gold standard
- Triangulate instead
  - Confirm estimates by comparing the story to multiple other stories.
  - Group like-sized stories on table or whiteboard

## Disaggregation

- Breaking a big story into smaller stories or tasks
    - You know how long the smaller tasks take
    - So, disaggregating to something you know lets you estimate something bigger you don't know
- Sometimes very useful
- But disaggregating too far causes problems
    - Forgotten tasks
    - Summing lots of small errors can be big number
- How much effort?
    - A little efforts helps a lot
    - A lot of effort only helps a little more

## Planning Poker

- An iterative approach to estimating
- Steps:
  - Each member of the team is given deck of cards, each card has a valid estimate written on it
  - Customer/Product owner reads a story and it’s discussed briefly
  - Each estimator selects a card that’s his or her estimate
  - Cards are turned over so all can see them
  - Discuss differences (especially outliers)
  - Re-estimate until estimates converge or reach consensus in which all participants agree with an estimate after they have discussed it and understood each other opinions.

## Use the right units

- Can you distinguish a 1-point story from a 2?
  - How about a 17 from an 18?
- Use units that make sense, such as
  - 1, 2, 3, 5, 8, 13 (Fibonacci numbers)
  - 1, 2, 4, 8
- Stay mostly in a 1-10 range
- Use 0, ½ if you want

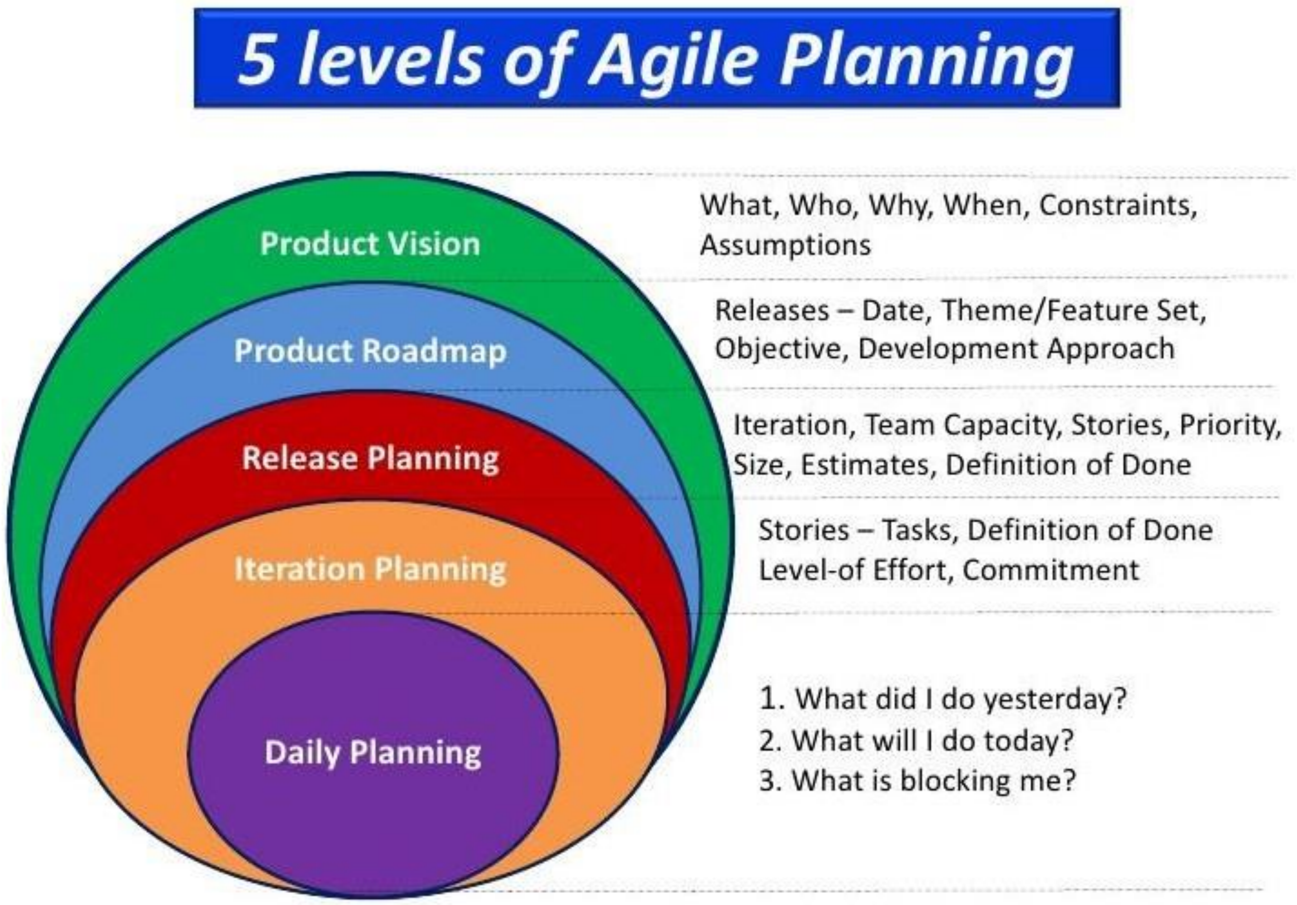
# Estimating PBIs: Why and When to Do It

- Two primary reasons to estimate Product Backlog Items (PBIs)
  - Allow the PO to priorities – estimate represents the cost, without knowing the cost it is difficult to priorities PBI
  - To make long-term projections
- When to create estimates?
  - Product backlog should be estimated early enough that PO can priorities or to answer questions when/how much will be delivered, but not earlier than that.
  - A few days before the end of each sprint (e.g. following that day daily scrum)

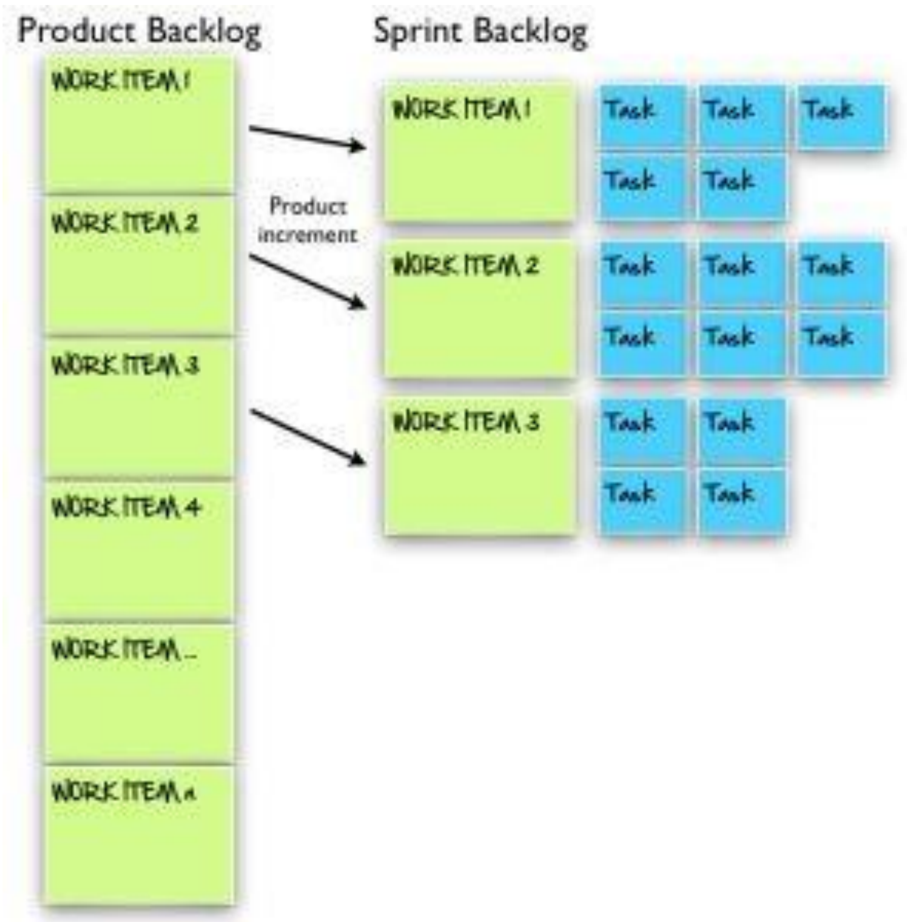
## Overcoming a Fear of Estimating

- Create safety around estimating
  - managers, SM, PO should make sure that giving estimates is comfortable.
  - Ask for honest estimates (without buffer)
- Describe how team will benefit from estimating
  - Establishes the team’s credibility
- Use estimates only for things that team approves of
  - Never use the team’s estimates against the team

# Agile planning


5 levels of Agile Planning
Product Vision
Product Roadmap
Release Planning
Iteration Planning
Daily Planning
What, Who, Why, When, Constraints, Assumptions
Releases – Date, Theme/Feature Set, Objective, Development Approach
Iteration, Team Capacity, Stories, Priority, Size, Estimates, Definition of Done
Stories – Tasks, Definition of Done Level-of Effort, Commitment
1. What did I do yesterday?
2. What will I do today?
3. What is blocking me?

# Iteration planning



## Two approaches

- Velocity-driven iteration planning
  - “We finished 15 story points last time, let’s plan on 15 story points this time.”
  - Very unreliable in what will be accomplished during an iteration
  - Velocity is mostly useful over the long term
- Commitment-driven iteration planning
  - More likely to lead to realistic iteration commitments
  - Discuss the highest priority item on the product backlog
  - Decompose it into tasks
  - Estimate each task - Whole team estimates each task
  - Ask ourselves, “Can we commit to this?”
    - If yes, see if we can add another backlog item
    - If not, remove this item but see if we can add another smaller one

## Team Velocity

- A measure of the amount of work completed per Sprint
    - Based on the previous Sprints
    - Can be used to forecast what might be completed in the future Sprints
- To do a release plan, you need to know or have an estimate of velocity
- Three ways to get velocity:
    - Use historical averages
    - Run 2-3 iterations and see what you get
    - Forecast it

## Release Planning

- Purpose
  - To answer questions such as: How much will be done by 30 June? When can we ship with this set of features? How many people or teams should be on this project?
- Inputs:
  - Relative priority of backlog items
  - The size estimate given to each backlog item
  - The length of the project
  - The team’s velocity (Story Points delivered per Sprint)

# Release plan

## Updating the release plan

- Revisit the release plan at the end of every iteration
- Update it based on:
  - Current understanding of velocity
  - Current prioritization of the product backlog
- This should be a very short and sweet process
- Use multiple views of observed velocity

# Why Agile planning works

- Re-planning occurs frequently.
- Estimate size, derive duration.
- Plans are made at different levels, with different level of precision.
- Plans are based on features, not tasks.
- Small user stories keep work flowing.
- Work in process is eliminated every iteration.
- Tracking is at the team level, not individual level.
- Uncertainty is acknowledged and plan for.
- Uncertainty is expressed in either the functionality or the date

# User Stories

## User stories

- User stories are probably the most popular agile technique to capture product functionality:
    - Working with user stories is easy. But telling effective stories can be hard.

- User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

- A story should be small enough to be coded and tested within an iteration—ideally just a few days.
- When a story is too large, it is called an epic. Backlog items tend to start as epics when they are lower priority. For release planning, epics should be broken down into smaller chunks.

## How Do I Write User Stories?

- *As a <user type>, I want to <function> so that <benefit>*

Examples:

- As a consumer, I want shopping cart functionality to easily purchase items online.
- As an executive, I want to generate a report to understand which departments need to improve their productivity

## Detailing a User Story

- Acceptance criteria
  - Condition of satisfaction
  - The conditions of satisfaction is simply a high-level acceptance test that will be true after the agile user story is complete

- As a vice president of marketing, I want to select a holiday season to be used when reviewing the performance of past advertising campaigns so that I can identify profitable ones.
  - Make sure it works with major retail holidays: Christmas, Easter, President’s Day, Mother’s Day, Father’s Day, Labor Day, New Year’s Day.
  - Support holidays that span two calendar years (none span three).
  - Holiday seasons can be set from one holiday to the next (such as Thanksgiving to Christmas).
  - Holiday seasons can be set to be a number of days prior to the holiday.

## User Stories and Planning

- Here are two areas where user stories affect the planning process on agile projects
  - Scheduling (every story has priority in product backlog; the implication is that the priority assigned to a story affects when the work will be done to implement that requirement)
  - Estimating (Developers are responsible for estimating the effort required to implement the things which they will work on; The implication is that because you can only do so much work in an iteration, the size of the work items (including stories), affect when those work items will be addressed)

## Korisni linkovi

- http://www.agilemodeling.com/artifacts/userStory.htm#sthash.swpouj1t.dpuf
- https://www.mountaingoatsoftware.com/
- http://freescrumtraining.org/training/?gclid=Cj0KEQiA1b7CBRDjmIPL4u-Zy6gBEiQAsJhTMHcRd8vDcBmaFYMZ5iA0aca4Pk01izMCr9_F5XYyoIIaArIa8P8HAQ

¿ Questions ?