Software Engineering

- What is it?
- How is it different from
  - Computer science?
  - Programming?
  - Computer engineering?
  - Other kinds of engineering?
- Is there such a thing?
  - See, for instance:
    Programmers: Stop Calling Yourselves Engineers
    (https://www.theatlantic.com/technology/archive/2015/11/programmers-should-not-call-themselves-engineers/414271/)
    The Atlantic magazine, November 2015

Goal of Software Engineering

Create **software "products"**

- Within budget
- On time
- Meet customers' needs
- Easy to modify (adaptable to future needs)

By **using best practices**

- Use a defined process
- Apply appropriate techniques in workflows including
  - Requirements
  - Analysis
  - Design
  - Implementation
  - Test

Managing Software Development

- Software systems are developed
  - by teams
  - using a structured process
    - programming (“coding”) only small part
- Management aims to satisfy the goals of
  - the developing organization
    - e.g. gain knowledge, market share, make a profit
  - the customer organization
    - e.g. gain business control, flexibility; new capabilities

The Most Important Factor

- We will learn a lot about
  - Design principles
  - Software processes
  - Development methods
  - Tools supporting software products and processes
- The most important success factor:
  - people (working in teams)

Practical Wisdom

- Being a good engineer (lawyer, doctor, teacher, banker) requires practical wisdom:
  - **Doing the right thing** (for the needs of the client)
  - **in the right way** (quality product/service)
  - **for the right reasons** (following good practices)

  cf. *Aristotle* and/or the TED talks by *Barry Schwartz*:
  - https://www.ted.com/talks/barry_schwartz_on_our_loss_of_wisdom
  - https://www.ted.com/talks/barry_schwartz_using_our_practical_wisdom

CMPS115 Approach

Practice SCRUM project management

So, what is SCRUM?

Great Scrummage

Scrum (Scrummage): restart of play after ball goes out of play or a minor infraction

SCRUM: a “Lightweight” SW Process

(“light-weight” as opposed to “heavy-handed”)

- Small teams
- Incremental development
- Time-boxed scheduling
- Adaptive and agile

The Agile Manifesto

“Software engineers of the world unite!”

Not quite. But somewhat.

It’s a statement of values.

Question: What did the Agile Manifesto respond to?

Answer: stay tuned.

The Agile Manifesto

Value this	over	that
Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

Source: www.agilemanifesto.org

Scrum Characteristics

- One of the “agile processes”
- Small teams (< 10 people)
- Product progresses in a series of 2 to 4 week long “sprints”
- Visible, useful increments
- Requirements are captured as *user stories*
- “product backlog”, a prioritized list of user stories: input to SCRUM process
- No specific engineering practices prescribed

SCRUM History

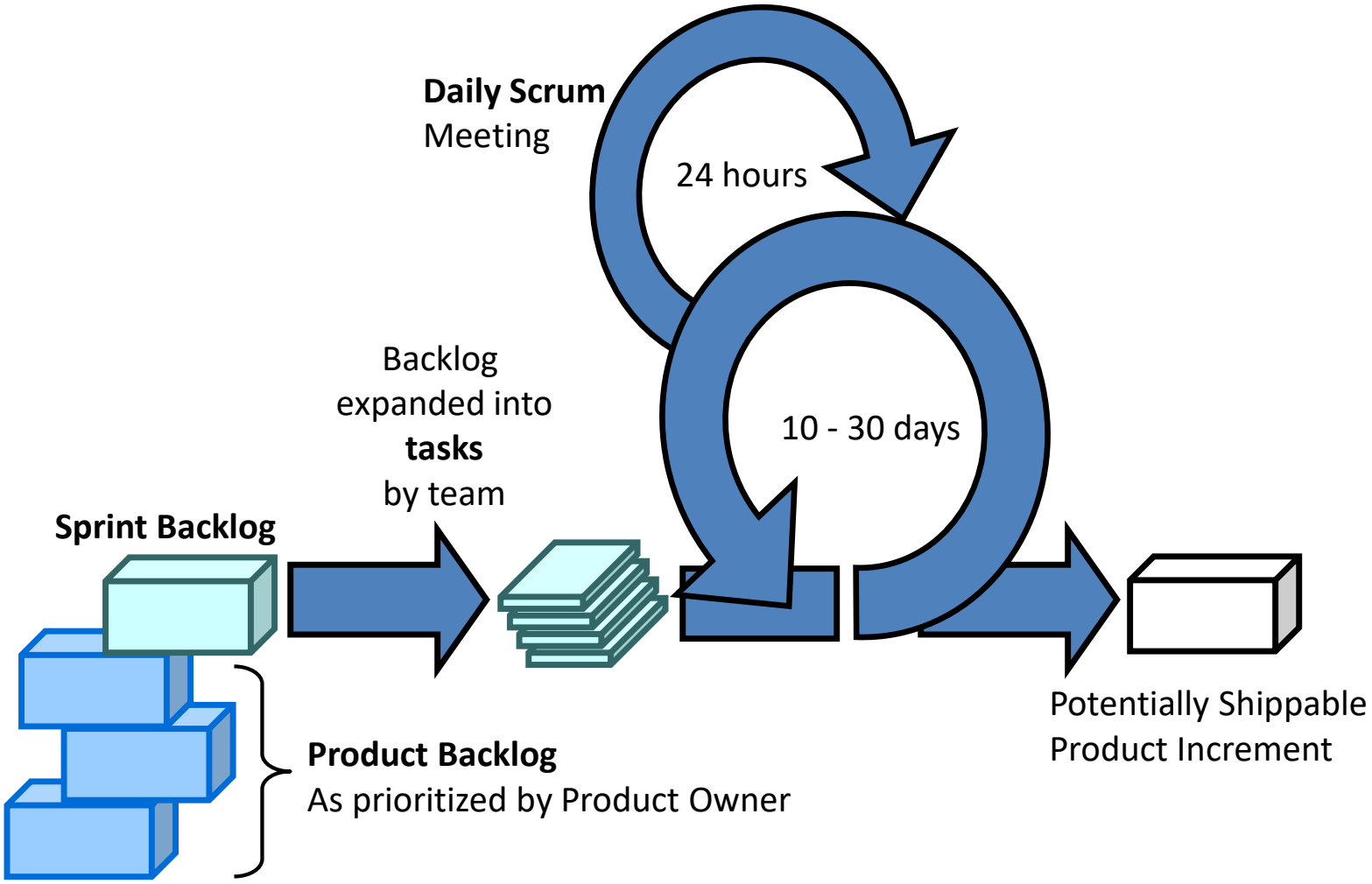
- 1986
  - Harvard Business Review paper by Hirotaka Takeuchi and Ikujiro Nonaka
  - Described new holistic approach to product development
  - Used game of Rugby as analogy
    - Team “tries to go the distance as a unit, passing the ball back and forth”
- 1991
  - Book *Wicked Problems, Righteous Solutions* by Peter DeGrace, Leslie Stahl
  - Used “scrum” to refer to approach described in Takeuchi/Nonaka
- Early 90’s
  - Independent development of scrum methodology by Ken Schwaber (Advanced Development Methods) and Jeff Sutherland, John Scumniotales, and Jeff McKenna (Easel Corporation)
- 1995
  - Sutherland and Schwaber presented paper describing Scrum at Business Object Design and Implementation workshop at OOPSLA ’95
- 2001
  - Schwaber collaborates with Mike Beedle to write book *Agile Software Development with Scrum*

Useful Material

View (on YouTube) *Agile in Practice* (3-5min each)

- Stand-Ups (Daily Scrum)
- Scrum board
- Frequent Small Releases
- Story cards/user stories
- MoSCoW
- Planning poker
- … more …

SCRUM Process Overview

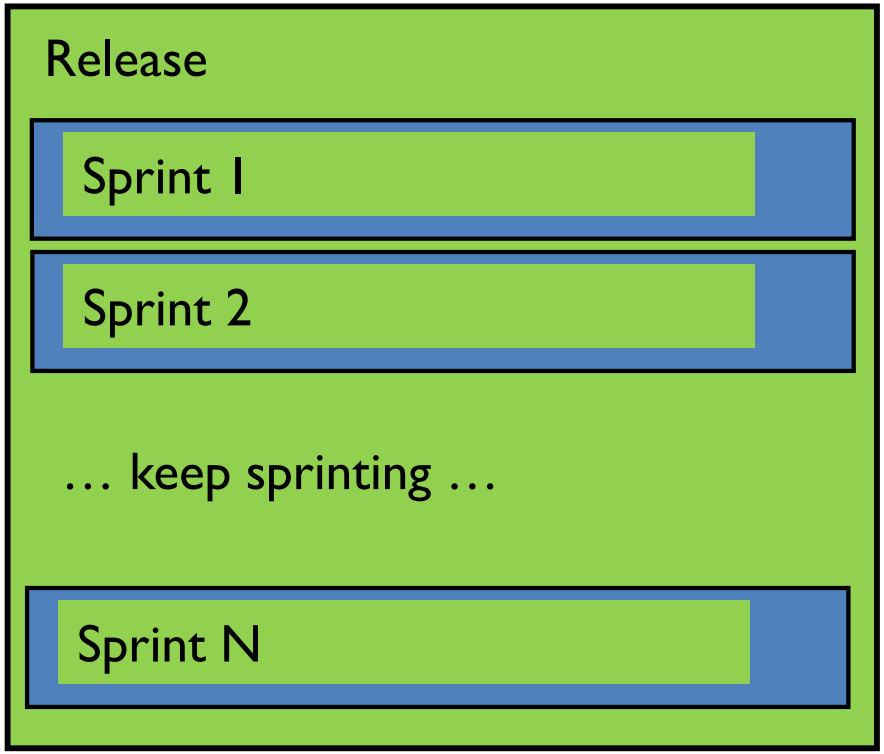


Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Release

Release: (transfer to customer of) a shippable product increment

- Produced iteratively by a sequence of *Sprints*



CMPS115:

One release at the end of quarter, produced in three Sprints

Sprints

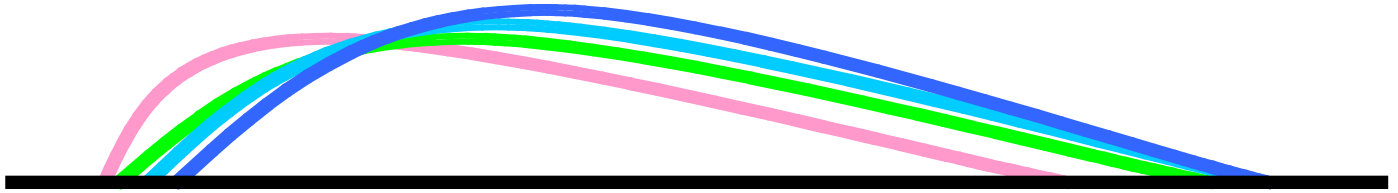
- Scrum projects make progress in a series of *Sprints*
  - Analogous to iterative-and-increment life-cycle model
- Typical duration is 2–4 weeks or a calendar month at most
- Sprint is *time-boxed*:
  it ends when time is up, not when Sprint goal is reached
- A constant duration leads to a better rhythm
- Product is designed, coded, and tested during the sprint

Process Linearization in Time

Requirements | Design | Code | Test

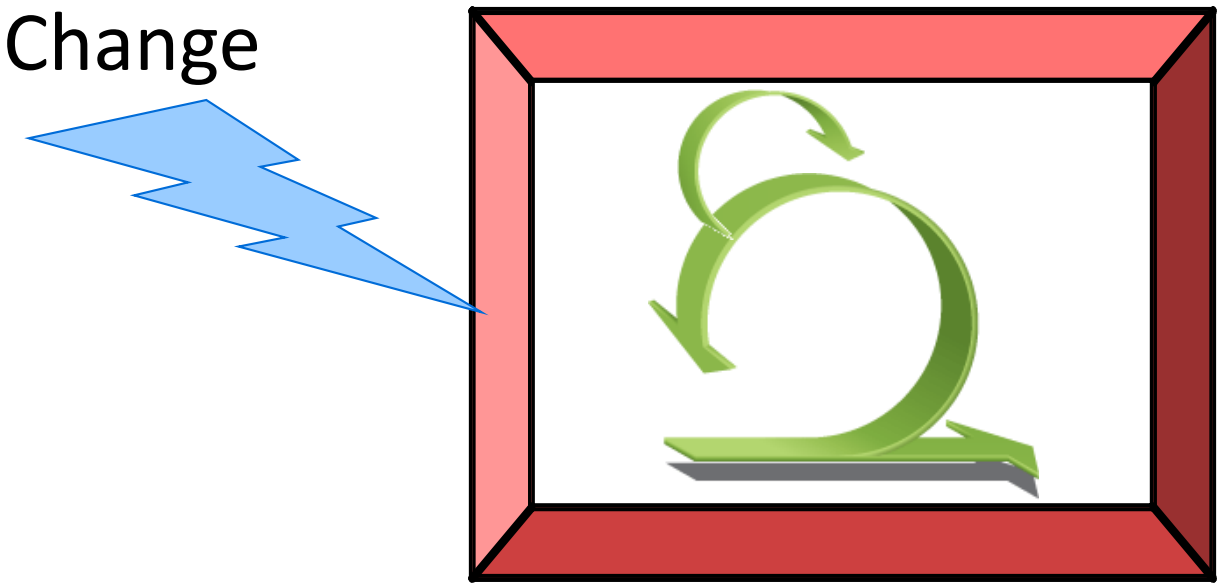
Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time



Source: “The New New Product Development Game” by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

No Requirements changes during Sprint

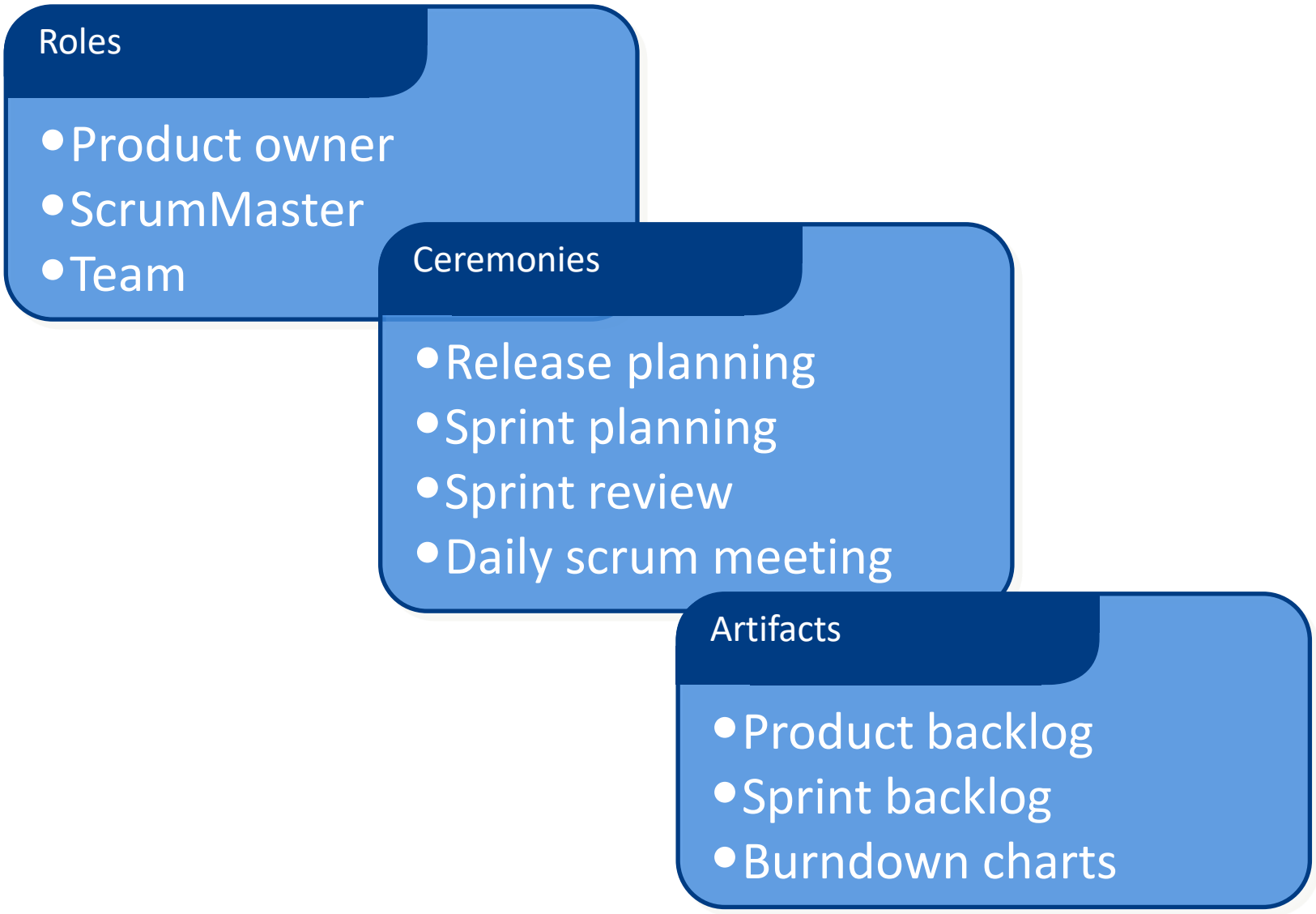


Sprint durations: limited by how long change requests can be held off

Sprint structure:

- Planning session
- Execution
- Sprint review

SCRUM Framework



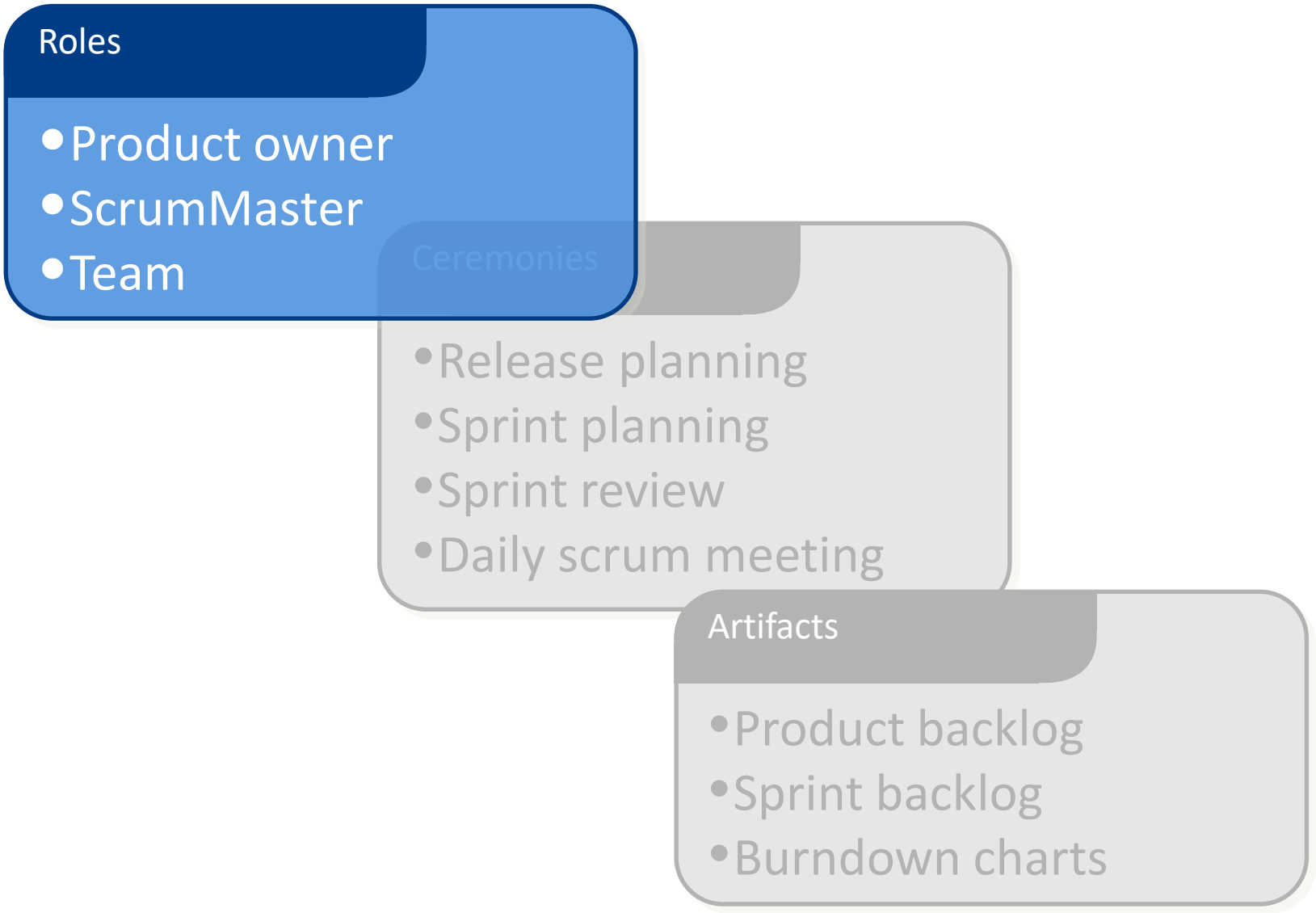
Helpful Material: Agile Overview

- The Scrum Framework (10 minutes)
  - Introduction to Scrum
  - Lyssa Adkins, Scrum Coach
  - https://www.youtube.com/watch?v=_BWbaZs1M_8

- Introduction to Scrum in 7 minutes
  - Steve Stedman
  - Roles, Ceremonies/Practices, Artifacts
  - https://www.youtube.com/watch?v=9TycLR0TqFA

- Scrum in 6 minutes
  - ScrumStudy
  - https://www.youtube.com/watch?v=aP3TBpWWwJ8

SCRUM Framework: Roles



Product Owner

- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every **increment**, as needed
- Accept or reject work result

Important note:

In CMPS115, Product Owner Role slightly modified;
stay tuned.

SCRUM Master

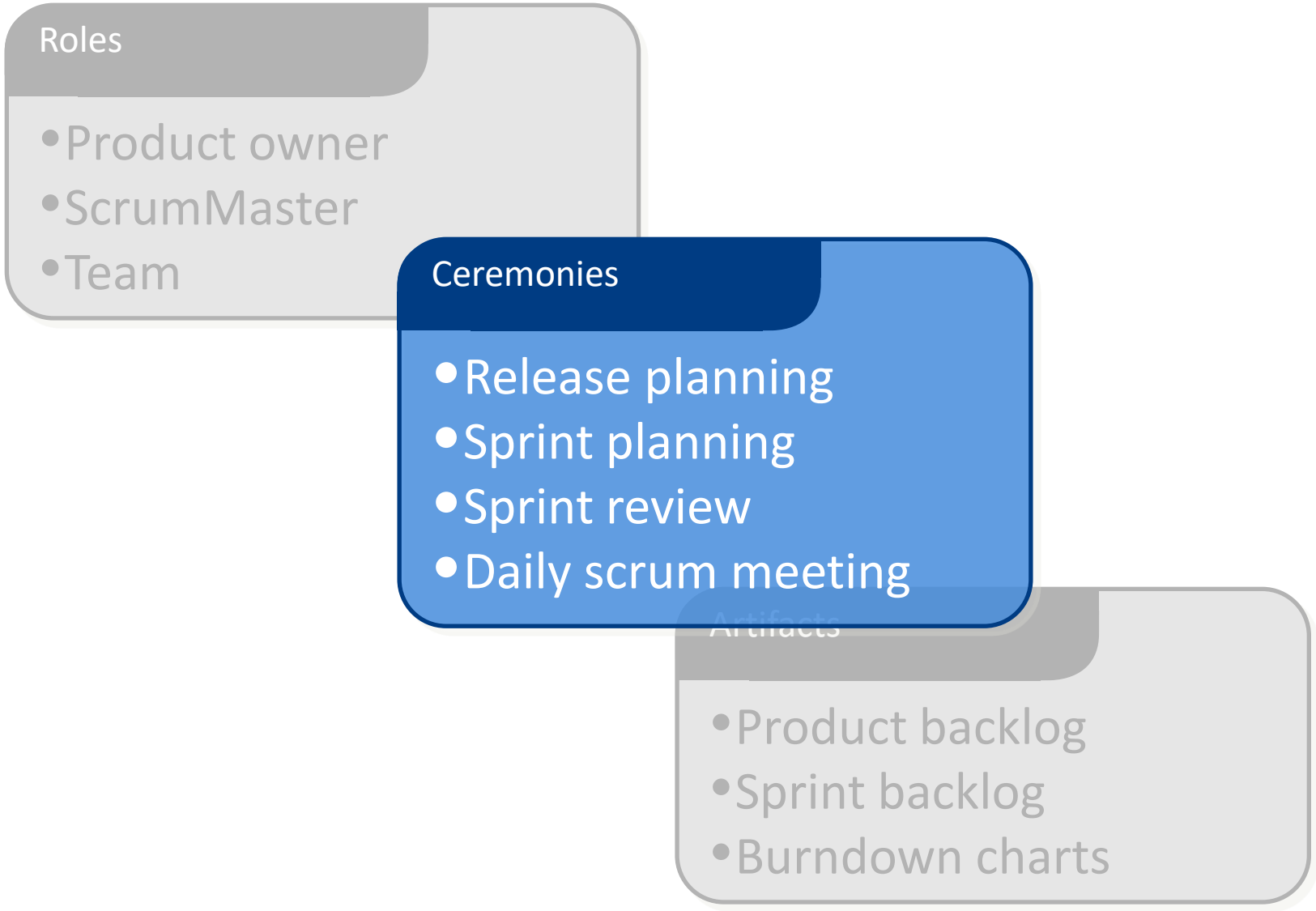
- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensures that the team is fully functional and productive
- Enables close cooperation across all roles and functions
- Shields the team from external interferences

NB: stay tuned for CMPS115 interpretation

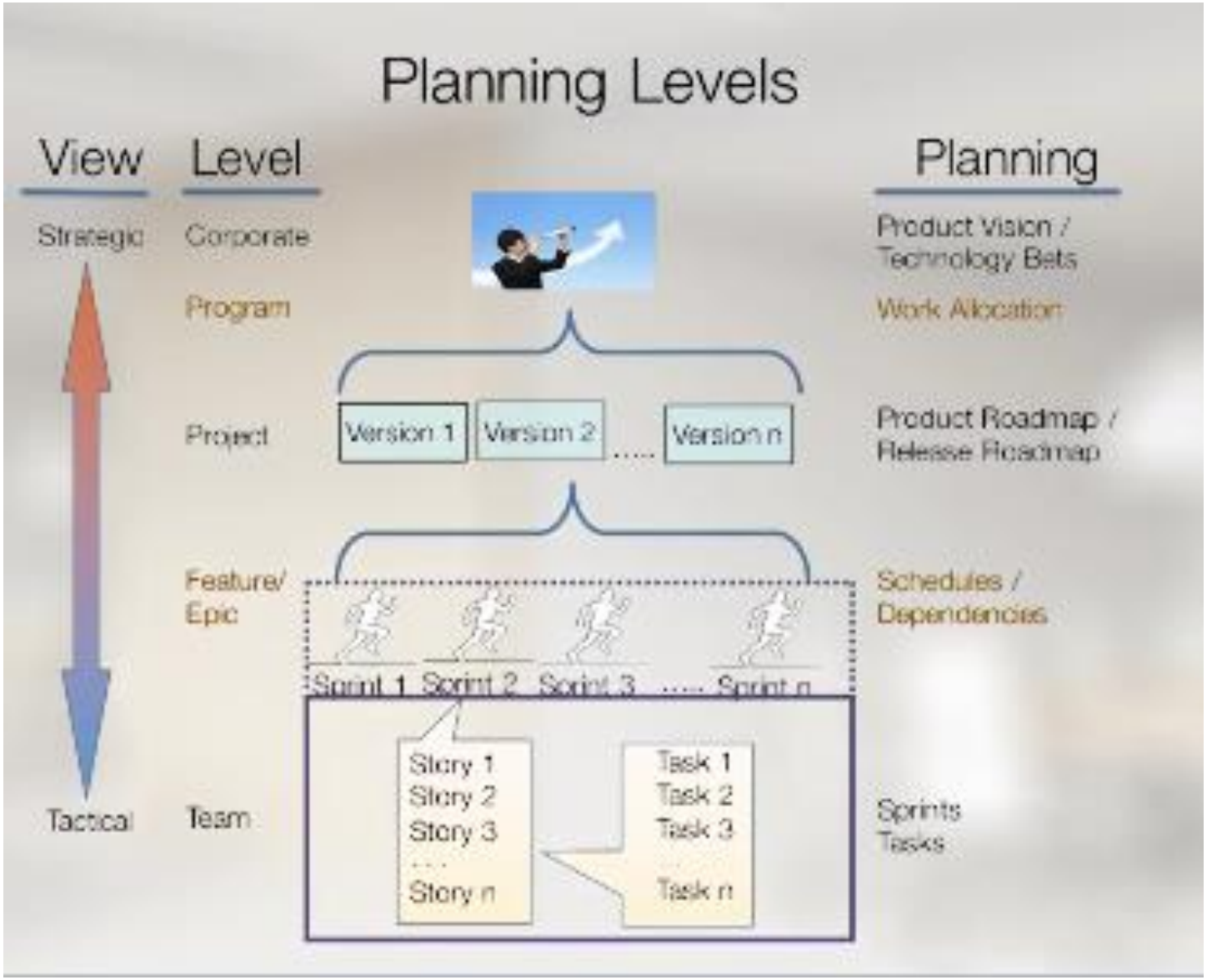
The Team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - Some exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles (rare in actual practice)
- Membership should change only between sprints

Scrum Framework: Ceremonies



Agile Planning and Estimation



Release

Release

- Major milestone of a software project
  - A new (version of) a software product/system
  - Transfer from development team to customer/users
- Contains a set of *product features*
  - As specified by *user stories*

Release Planning

- Determine the set of user stories to include
  - **Analyze** project vision/concept
  - **Decompose** concept/vision into a set of user stories
  - **Prioritize** the user stories
  - **Estimate** in **story points** difficulty of implementing each user story

Release Plan

- Result of Release Planning
- *Input* to Sprint Planning process

User Stories

User Story

- Specifies a product feature
- Technique for eliciting and documenting software requirements
- Captures one user/system interaction by specifying
  - User role
  - Goal to be achieved
  - Reason/motivation for the interaction

User stories are closely related to *Use Cases*

- Term “use case” shunned by SCRUM faithful

User Story Format

- User story format
  - As a {user role}, I want {goal} [so that {reason}]
  - Examples:
    - As an **employee**, I **need access to my evaluation** so that **I know what I can improve**.
    - As a **player**, I **need to pick up game world objects** so that **I can collect food and ammunition**.
    - As a **product tester**, I **need access to internal database state** so that **I can determine if product works properly**.
- Class exercise developing a few user stories for product

Attributes of User Story: INVEST

INVEST conditions

- **I**ndependent
  - Free of implementation dependencies on other stories
  - otherwise combine user stories (debatable)
- **N**egotiable
  - Useful as basis for discussion between stakeholders/team if in backlog
- **V**aluable
  - Communicates value to user and to team
- **E**stimatable
  - Possible to estimate effort to implement user story
- **S**ized appropriately
  - Need to be small enough to fit into a Sprint
- **T**estable
  - It must be possible to verify that a user story has been implemented.

Prioritize User Stories for Release

- **Prioritize** user stories as part of release planning
- It's a cop-out to say "everything is equally important"
  - Better to be explicit about the order of implementation
- What do priorities mean?
  - A user story with highest priority is implemented first
  - A user story with lowest priority is implemented last
  - Lower priority items might **never be implemented**
    - If there is a feature you really want to see in the project, need to ensure it has a high priority
- Product Owner has ultimate authority over setting priorities

MoSCoW

- Must have
- Should have
- Could have
- Won’t have

Watch MoSCoW Agile in Practice

- https://www.youtube.com/watch?v=QfZo9cxnQgY

Estimating “Size” of User Stories

- Development effort needed for User Stories
  - Measured in *Story Points*
- Story points are abstract units

Estimating “Size” of User Stories

Story Point

- Team specific
- (abstract) Unit of design/implementation effort
  - Not person-months/years/hours
  - Relative measure
  - Avoids arguments
- Super-linear, discrete scale
  - Super-linear: uncertainty increases with bigger tasks
  - Discrete: each number denotes a range
  - Less uncertainty with many small tasks than few large tasks

Story Point Ranges

8 intuitive degrees of difficulty map to corresponding Fibonacci number; to wit:

Points	**Intuitive ("T-shirt") size**
0	freebie, already done
1	extra small
2	small
3	medium
5	large
8	extra large
13	XXL
20/21	XXXL (huge)

- Not magic numbers; teams can choose as they see fit
- Key property: values represent "ballpark" ranges; no sense quibbling over +/- 1
- **Your** Poker Deck (next) based on **your** numbers

Estimation Exercise

1 point:

Effort of walking from Thimann Lecture Hall to Science & Engineering Library

Estimate: effort of walking from Thimann to

- Engineering 2
- base of campus (intersection Bay and High)
- Downtown (Bookshop Santa Cruz)

How do your estimates compare with an estimate of distances?

Planning Poker

"Loser does the work?". No!

Technique for consensus estimate of user story effort

- Each team member has deck of cards
  - One card for each intuitive size, showing corresponding number of points
- Product owner picks and explains a User Story
- Team discusses implementation effort needed
- Then:
  - Each team member puts the card face down that shows private estimation of effort
  - All cards are turned at once
  - Repeat until convergence: all cards show same number

Agreeing on Estimates

Agile practice:

Planning poker

- https://www.youtube.com/watch?v=0FbnCWWg_NY

Calibrating Estimates

- Estimating user stories is difficult, especially when a team is not experienced
  - Accuracy improves with experience over time
  - Compare estimates with actual performance; adjust
- For a team's first estimate:
  - Pick a "small" user story that all can agree on
  - estimate that first
  - Alternately, pick one that is small, large, and medium in size, and estimate those first, to get a sense of the range
- Once the team has estimated three or more items
  - Revisit the estimates, to ensure the team agrees with the relative size of the estimates of the items
  - This helps calibrate the scale used by the team
- Note that different teams might have different scales
  - That's OK, so long as each team is internally consistent

Assigning User Stories to Sprint

- Assign User Stories to Sprint as part of release planning
  - Need an idea how many story points team can implement during Sprint
    - Start with a good faith guess
    - Revise based on experience
- NB: Sprint goals set in release planning are estimates, *not* commitments
  - During Sprint planning, *user stories* are decomposed into *tasks*
  - Task effort levels **are** commitments

Targets, Estimates, Commitments

- Target
  - Release date is a target
  - Usually not controlled by development team
- Estimate
  - Story points are estimates
  - Value ranges based on preliminary analysis and experience
- Commitments
  - During Sprint planning, user stories decomposed into tasks
  - Task effort levels **are** commitments;
    i.e. agreements between team members and customer

Helpful Material: Agile Practices (1)

- Agile in Practice
  - http://www.agileacademy.com.au/agile/KnowledgeHub
- Story Cards/User Stories
  - https://www.youtube.com/watch?v=LGeDZmrWwsw
- MoSCoW (Prioritization)
  - https://www.youtube.com/watch?v=QfZo9cxnQgY
- Planning Poker ("Size" Estimation)
  - Agile in Practice
  - https://www.youtube.com/watch?v=0FbnCWWg_NY

Output of Release Planning

At the end of release planning:

- A **prioritized list** of **user stories**, with implementation time **estimated in story points**, organized into **Sprints**.

Plan for Release #1	
User Stories, Priority ordered	Story Points
Sprint 1	
1. As {role}, I …	5
2. As {role}, I …	2
…	
Sprint 2	
…	

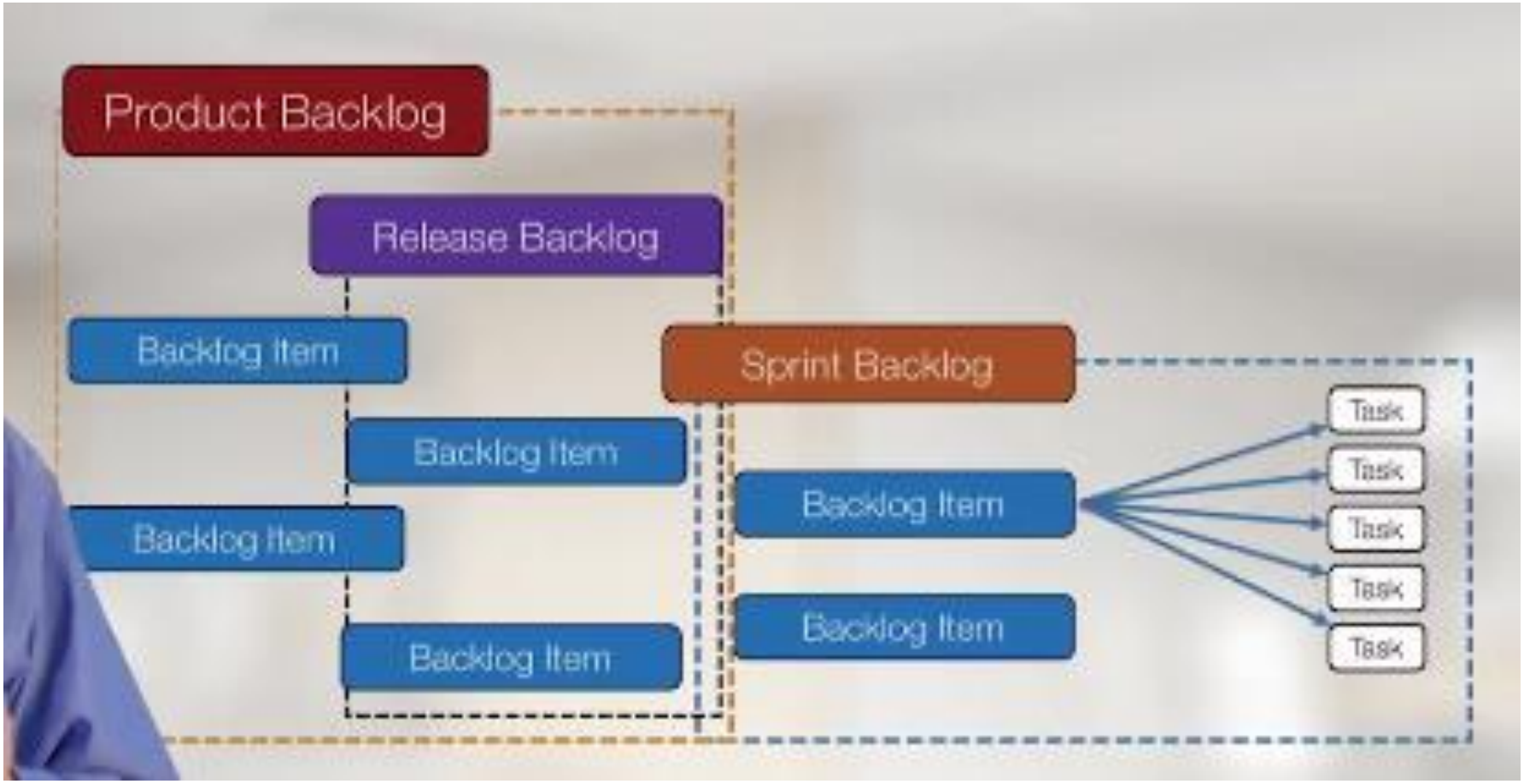
Product Backlog

- All of the user stories that have not yet been implemented form the **product backlog**
- For a given release, some user stories will be grouped into planned Sprints. Others will not, but may be placed into future Sprints (or may be dropped).

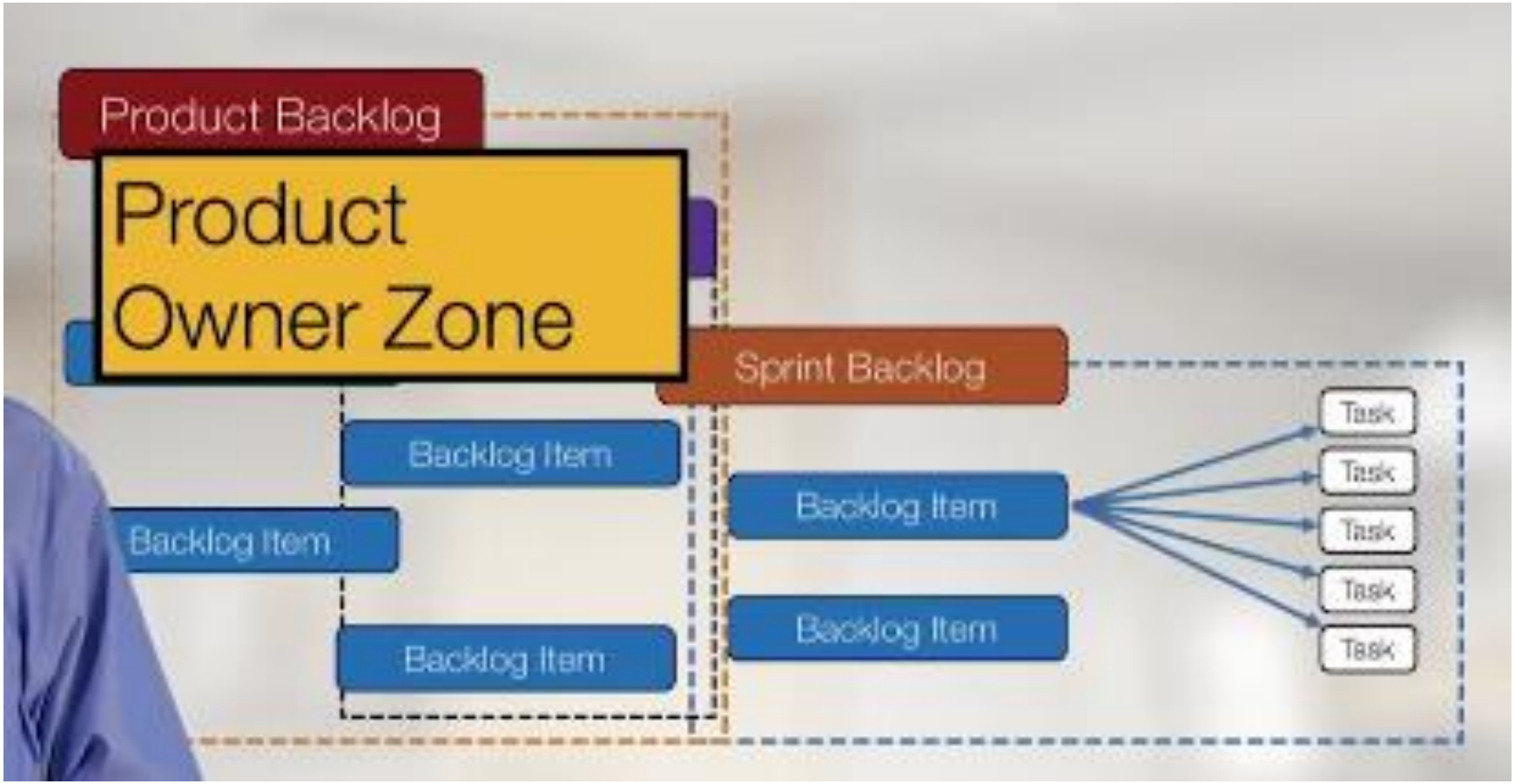
- Product backlog = user stories assigned in current release + all unassigned user stories

- That is, the release plan is a subset of the product backlog intended for the current release

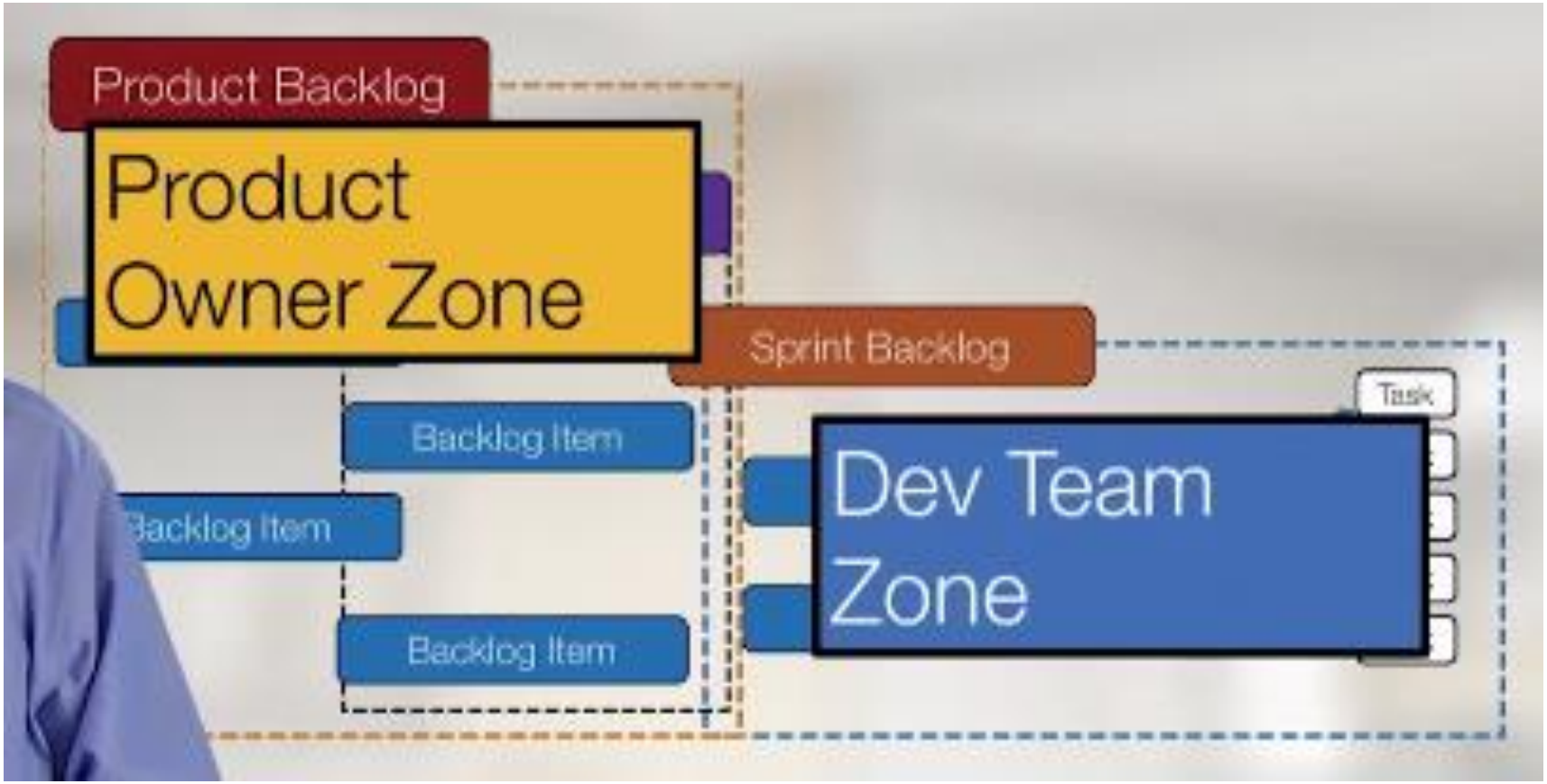
Product Backlog Views (1)



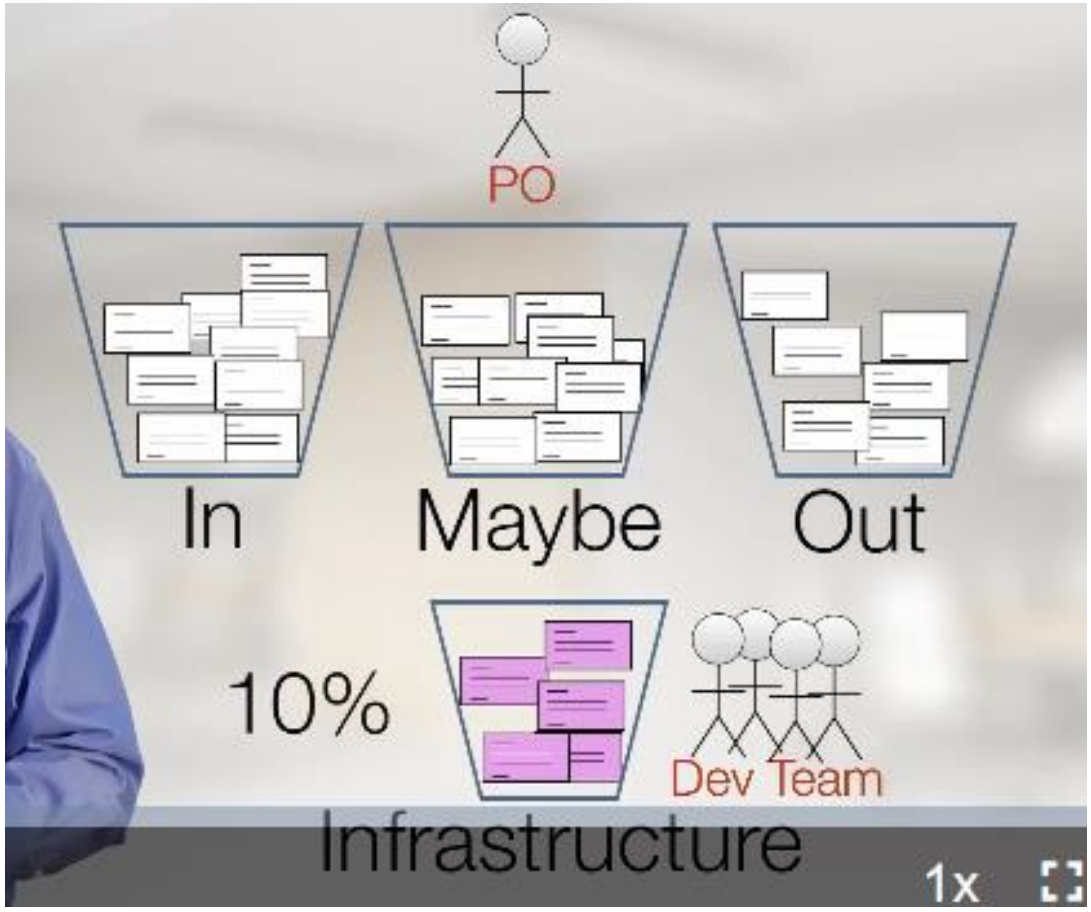
Product Backlog Views (2)



Product Backlog Views (2)



Product Backlog: Infrastructure



- At the beginning of project, infrastructure portion may be larger
- For class project: include learning (APIs, technologies, languages)

Keep some time in reserve

- Don’t allocate 100% of your available time
- Initially, allocate 85% of expected available time
  - Keep 15% in reserve
- Buffer for oversights and unexpected complications

Study Questions

- Be able to describe role of Scrum Master, Product Owner in ideal Scrum, and for CS 115
- Define a user story, and know the template for a user story
- Describe how to play planning poker
- What is a story point? What is the value of having a story point range?
- What are the INVEST criteria for story points?
- What are the outputs of release planning?
- What is the product backlog? How does this relate to the release plan?
- What is a sprint?
- What is the relationship of release to sprint?
- Why do we prioritize user stories? What does high priority and low priority mean?

Scrum Roles: CMPS115 Adaptations

- Adjustments to
  - Roles
  - Assignment of roles
- Typically, in cmps115
  - "start-ups"
    - Team conceives of product idea
    - Business value is uncertain
  - Collaboration with other UCSC researchers
    - External customer

Scrum Roles: CMPS115 Adaptations

Class limitations

- Each team “owns” their project design
- Product owner must
  - be a single person
    - To ensure effective decision making regarding feature priorities, feature inclusion and exclusion
  - Participate in all Release and Sprint planning meetings
- Usually no external customer
- Professor/TAs are stakeholders/partial product owners
  - Need to evaluate projects
  - Cannot be present in all Release and Sprint planning meetings
    - Due to lack of scalability
- Every team member will play ‘special role’ for at least part of team project

Product Owner in CMPS115

- Each team appoints one member as Product Owner, typically
  - "owns" the project design
  - Or at least entrusted with authority to make design trade-offs
  - Remains in Product Owner role for entire project
    - Changes at start of sprints possible, if necessary
- Product Owner is a 'special role'
  - written about in project reflection essay
- Professor/TAs retain right to modify Product Owner decisions
  - E.g. feature priorities, feature cut/save decisions
  - Unlikely to exercise this right often

Scrum Master in CMPS115

- Each team member must be a Scrum Master for at least one Sprint (except for Product Owner)
  - Scrum Master appointed at beginning of Sprint
  - Appointment lasts for entire Spring
    - In large teams, Scrum Master role may be shared by two members
- Scrum Master is 'special role'
  - Part of individual performance evaluation
  - Written about in project reflection essay
- Scrum Master responsibilities
  - Ensure team practices Scrum
  - Maintain Scrum task board
  - Maintain burndown/burnup chart
  - Provide each week detailed feedback on activities of team members