# EECE 310

Scrum

# Learning Goals

- Describe the commonalities of agile software development.
- Describe the software process described by Scrum.
- Decide when to apply which process model.

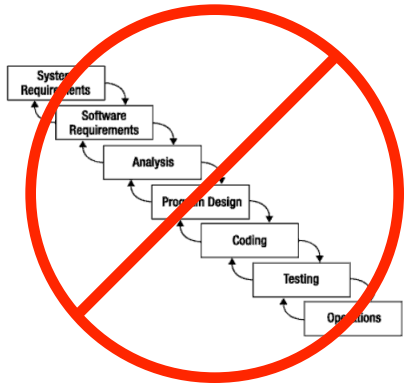
# Reading material!

- Scrum by Michael James ***(required)***
  - **Connect (under Reading Material)**

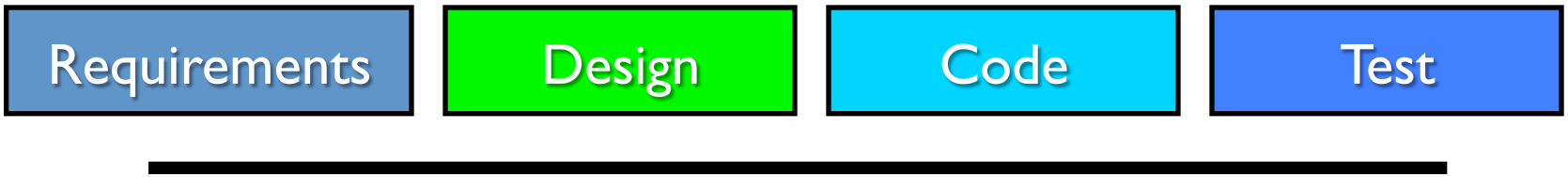
# Scrum

- **Management framework** for incremental product development

- **Self-organizing**, cross-functional teams
- Product progresses in a series of two- to four-week (fixed length) iterations: ***sprints***
- Every iteration produces a potentially shippable (properly tested but not complete) product
- Requirements are captured as items in a list: ***product backlog***
- The business sets the priorities.
- No specific engineering practices prescribed (unlike XP)

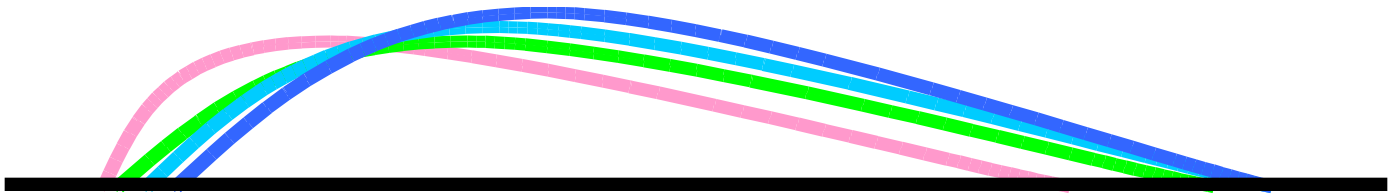


# Sequential vs. Overlap

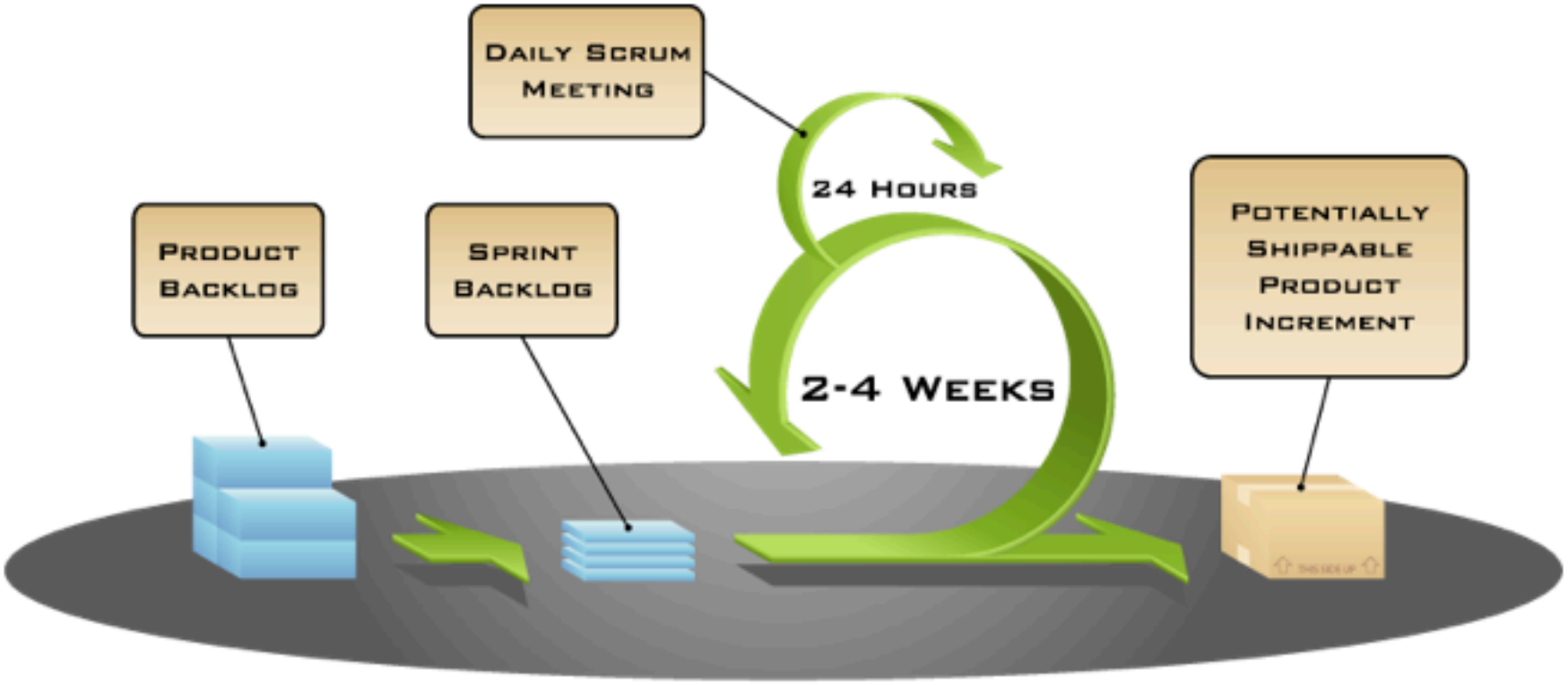



Rather than doing one thing at a time...

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



# Scrum







Mountain Goat Software, LLC

taken from www.mountaingoatsoftware.com (Mike Cohn)

# Scrum Framework

## Roles

- Product owner
- Scrum Master
- Team

## Ceremonies

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

## Artifacts

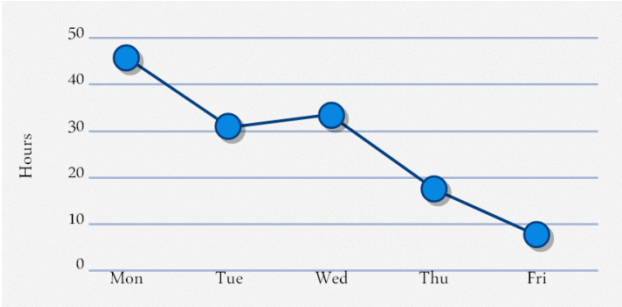
- Product backlog
- Sprint backlog
- Burndown charts

# Scrum - Roles

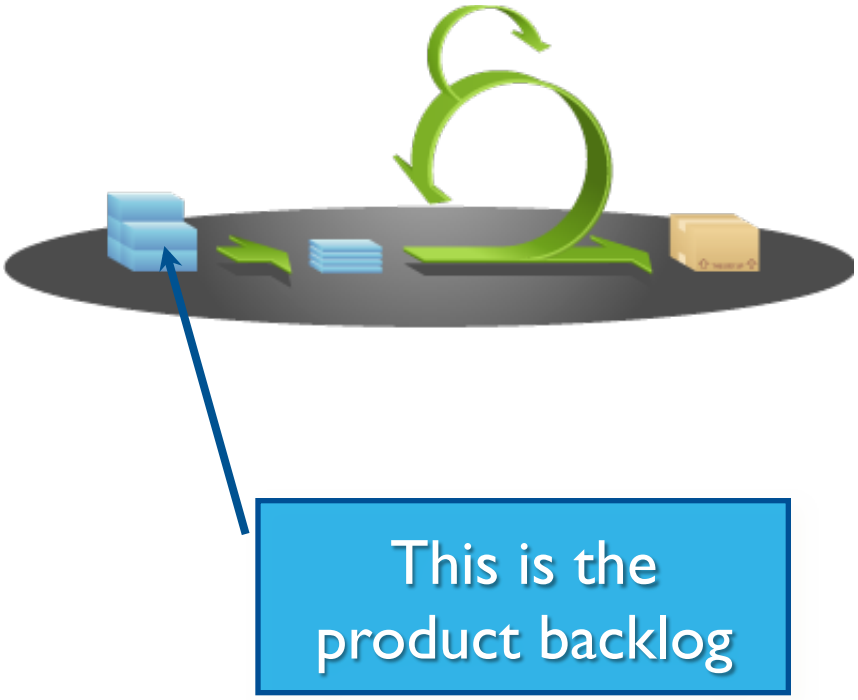
- Product Owner
  - Defines features of the product
  - Prioritizes features according to market value
  - Adjusts features and priorities every iteration, as needed
- ScrumMaster
  - Facilitates Scrum process
  - Helps resolve obstacles
  - Shields team from external interferences
  - NOT the manager (not the decision maker)
- Team
  - Self-organizing, self-managing, cross-functional
  - 7 (+/- 2) people

# Scrum: Artifacts

- Product Backlog
  - prioritized list of requirements (or wishlist) that describes all desired functionality
- Product Backlog Item
  - specifies a customer-centric feature (User Story form) – effort estimated by team, business value estimated by Product Owner
- Sprint Backlog
  - contains list of tasks that are negotiated by team and product owner from the Product Backlog for the sprint (negotiated PBIs broken down into specific tasks)
- Burndown Chart
  - Total remaining items
  - Motivation for team



# Product Backlog



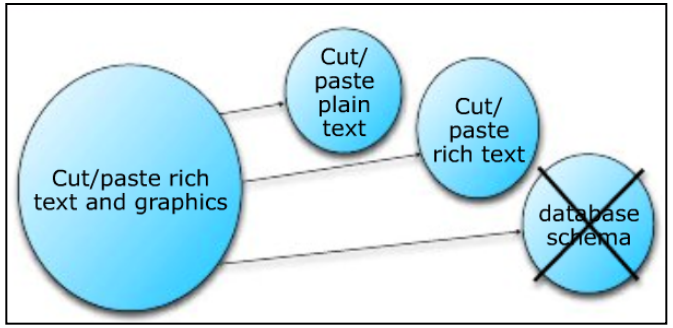

- The requirements
- A list of all desired work on project
- Ideally expressed as a list of user stories along with "story points", such that each item has value to users or customers of the product
- Prioritized by the product owner
- Reprioritized at start of each sprint

# Product Backlog Item: User Stories

- Specifies the WHAT not the HOW of a feature
- In **User Story** form
  - Show what the user gets from the product
  - Show what value the system adds to the user
  - generally follows (but not restricted to) the format:

- **As a [type of user], I want/can/need/etc. [goal or need], so that I can [reason].**

- Large PBIs (called 'Epics') are split into thin vertical slices, not horizontal implementation slices



# User Stories

- Example of User Stories:
  - As a user, I want to search for contacts so I can message them.
  - As a customer, I want to search for product items so I can buy them.
  - As an employer, I want to post a job on the website so people can apply for it.

- NOT User Stories:
  - Implement contact list view ContactListView.java
  - Define the product table database schema
  - Automate the job posting algorithm

# User Stories

User Story also

- has a description of the story providing additional detail
- specifies **acceptance criteria** that defines what is meant for this feature to be DONE
- provides (relative) **estimate** of the required effort (e.g. story points)

# INVEST (for a good user story)

- **Independent**: The user story should be self-contained, in a way that there is no inherent dependency on another user story.
- **Negotiable**: User stories, up until they are part of an iteration, can always be changed and rewritten.
- **Valuable**: A user story must deliver value to the end user.
- **Estimable**: You must always be able to estimate the size of a user story.
- **Sized**: User stories should not be so big as to become impossible to plan/task/prioritize with a certain level of certainty.
- **Testable**: The user story or its related description must provide the necessary information to make test development possible.

# Example – User Story

- As a shopper, I can use my credit card so that I can purchase items.

- **Note:** Accept Visa, MasterCard, American Express.

- **Test:** (on the back of the story card)
  - Test with Visa, MasterCard and American Express (pass)
  - Test with good, bad and missing card ID numbers.
  - Test with expired cards (fail).
  - Test with over $100 and under $100.

# Example – User Story & Test

- As a Creator, I want to upload a video from my local machine so that all users can view it.
- Note: …
- Test:
- Click the "Upload" button.
- Specify a video file to upload.
  - Check that .flv, .mov, .mp4, .avi, and .mpg extensions are supported.
  - Check that other filetypes aren't able to be uploaded.
  - Check that file size larger than 100MB results in an error.
  - Check that movies longer than 10 mins result in an error.
- Click "Upload Video".
- Check that progress is displayed in real time.

# User Stories

"As a student, I want to submit my assignment and view my overall GPA."

Is this a good user story? Why?

# User Stories

"As a student, I want to submit my assignment and view my overall GPA."

No, because:

(1) It does not follow the format:

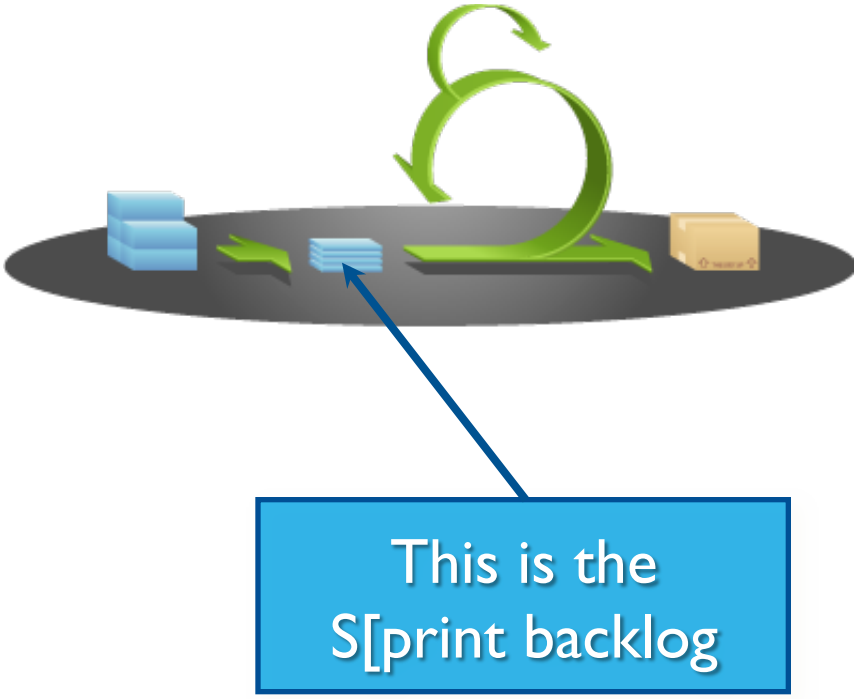
"As a role I want goal so that benefit"

(2) It is composed of two requirements.

# Class Activity

- In groups of 2 or 3, **on paper**:
  - Write down 3 user stories for developing an online music store (such as the music part of iTunes)
  - Include:
    - Effort estimation
    - Acceptance tests

# Sprint Backlog




- List of tasks Scrum team **commits to** for sprint (negotiated PBIs for sprint broken down into specific tasks)
- Based on priorities and team's perception of required time (each normally between 4 and 16 hours)

# Sprint Backlog

SMART:

- Specific, Measurable, Achievable, Relevant, Time-Boxed

- Tasks in Sprint Backlog represent **developer's (technical)** perspective, not customer perspective

# Example – Sprint Backlog Items

- **Real Weather App:**
- PBI: As a subscriber, I want to see a 10-day forecast of conditions so that I can plan at least a week ahead
  - Sprint Task 1: Parse the weather data in day packs
  - Sprint Task 2: Push several days of data to the client

- PBI: As a subscriber, I want to see precipitation accumulations so that I can plan my activities.
  - ST1: Parse snow/rain data from the provider's data
  - ST2: Push the snow/rain data to the client
  - ST3: Redesign client screen a bit
  - ST4: Refactor the server code

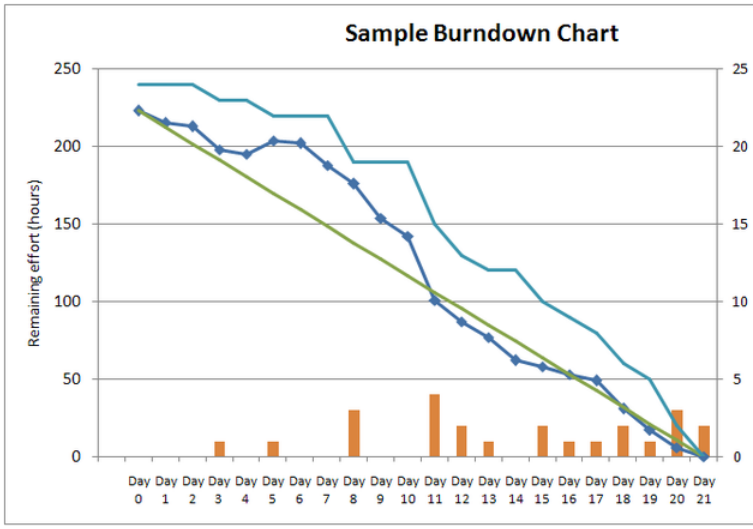
# Tools to manage Backlogs

- ScrumWorks Pro (commercial)
- trello.com (free)
- Excel

# Sprint Burndown Chart

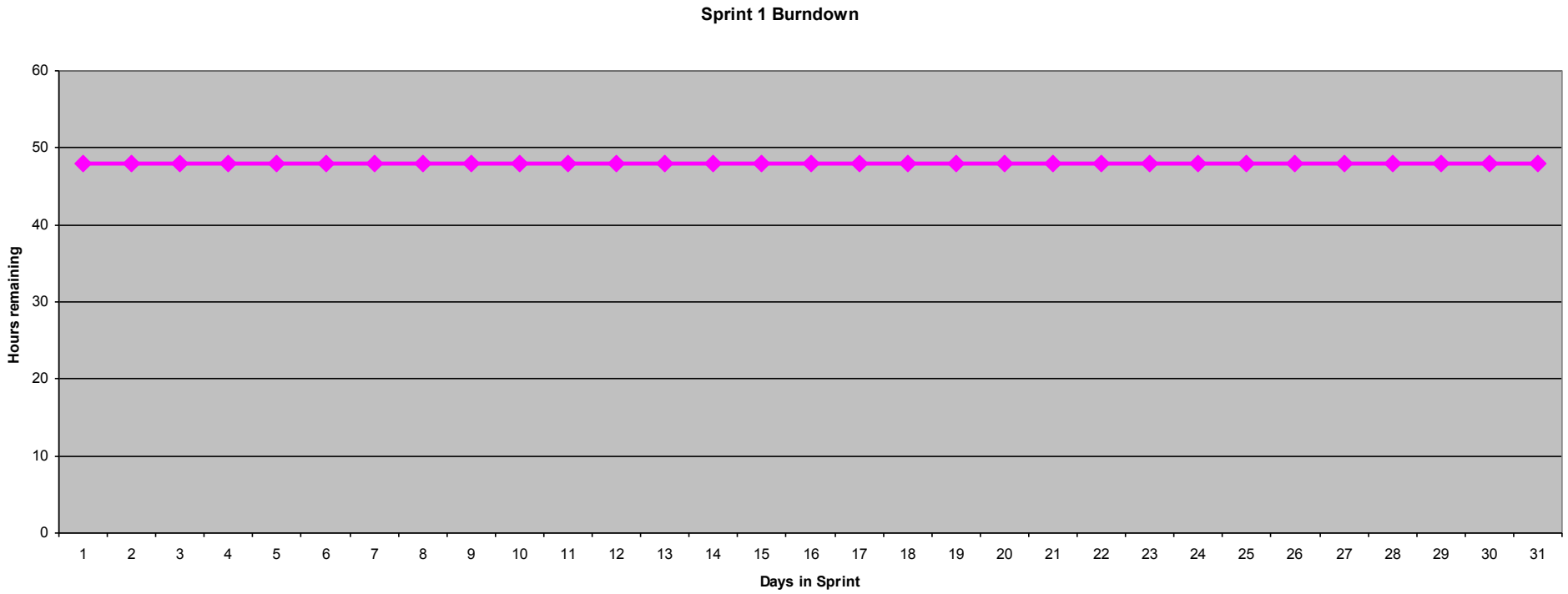
- A display of what work has been completed and what is left to complete
  - one for each developer or work item
  - updated every day
  - (make best guess about hours/points completed each day)

- *variation:* Release burndown chart
  - shows overall progress
  - updated at end of each sprint
- Velocity



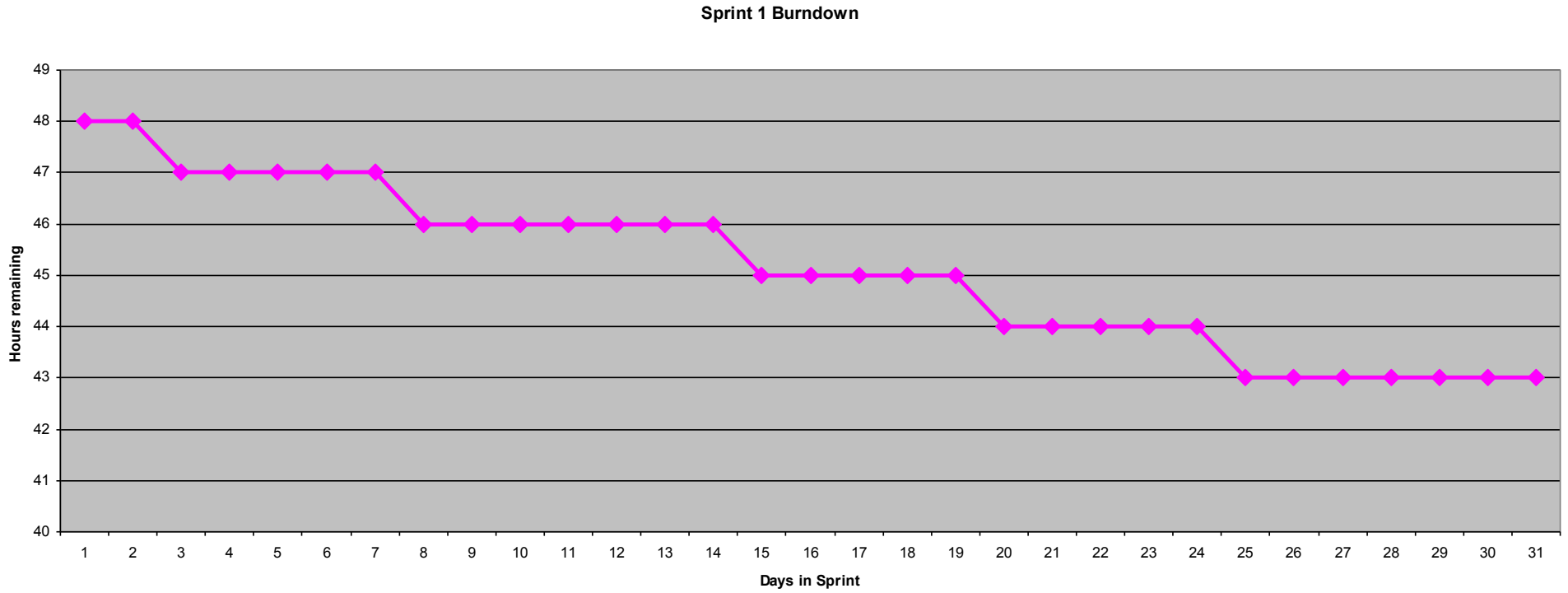
# Burndown Example 1

No work being performed



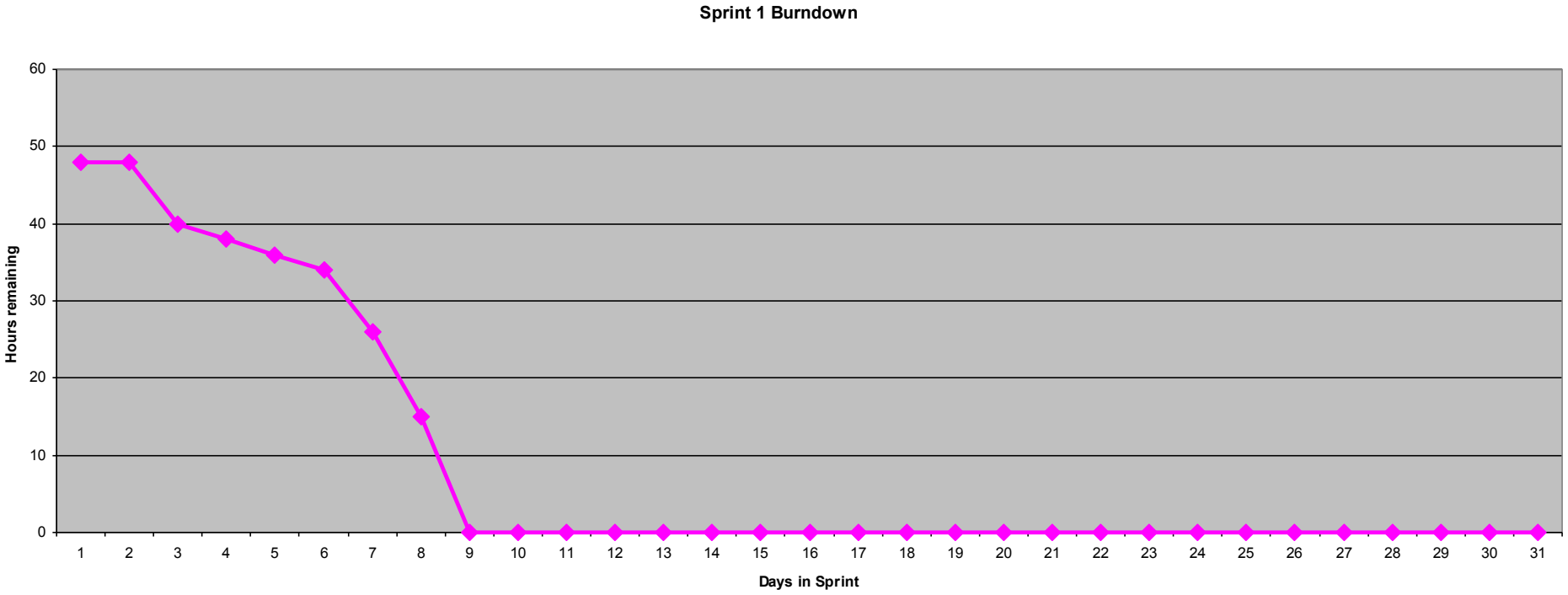
# Burndown Example 2

Work being performed, but not fast enough

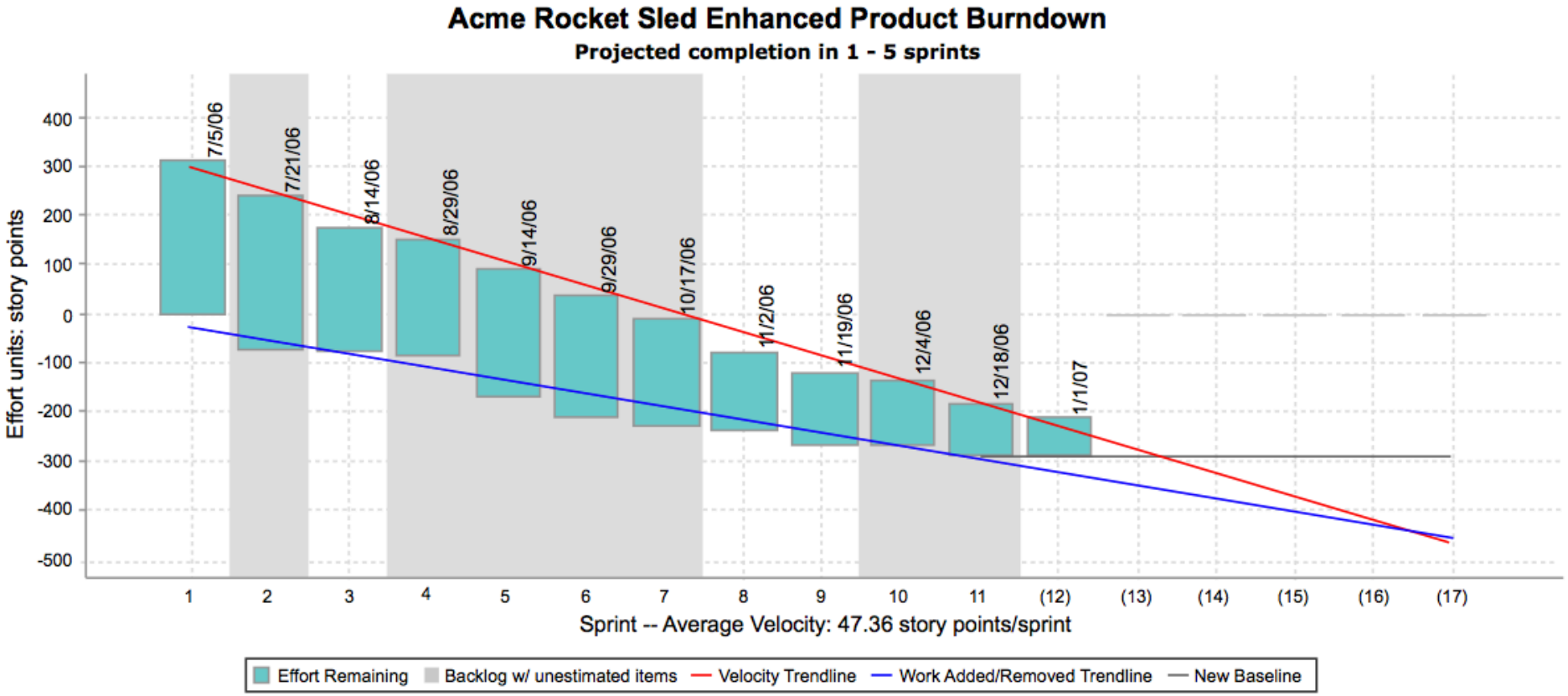


# Burndown Example 3

Work being performed, but too fast!



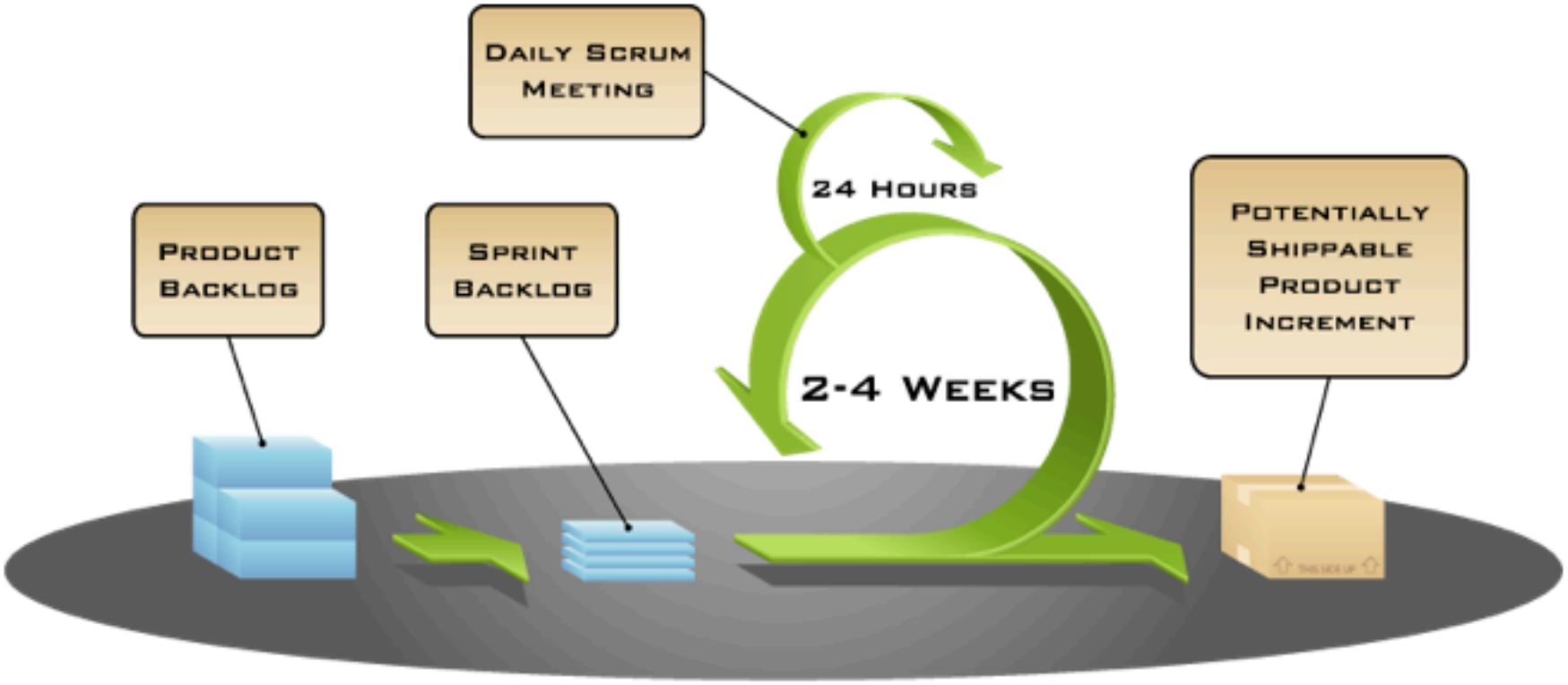
# Velocity vs Discovery



# Presentation Topics

- See Connect for assigned topics and schedule

# Scrum







Mountain Goat Software, LLC

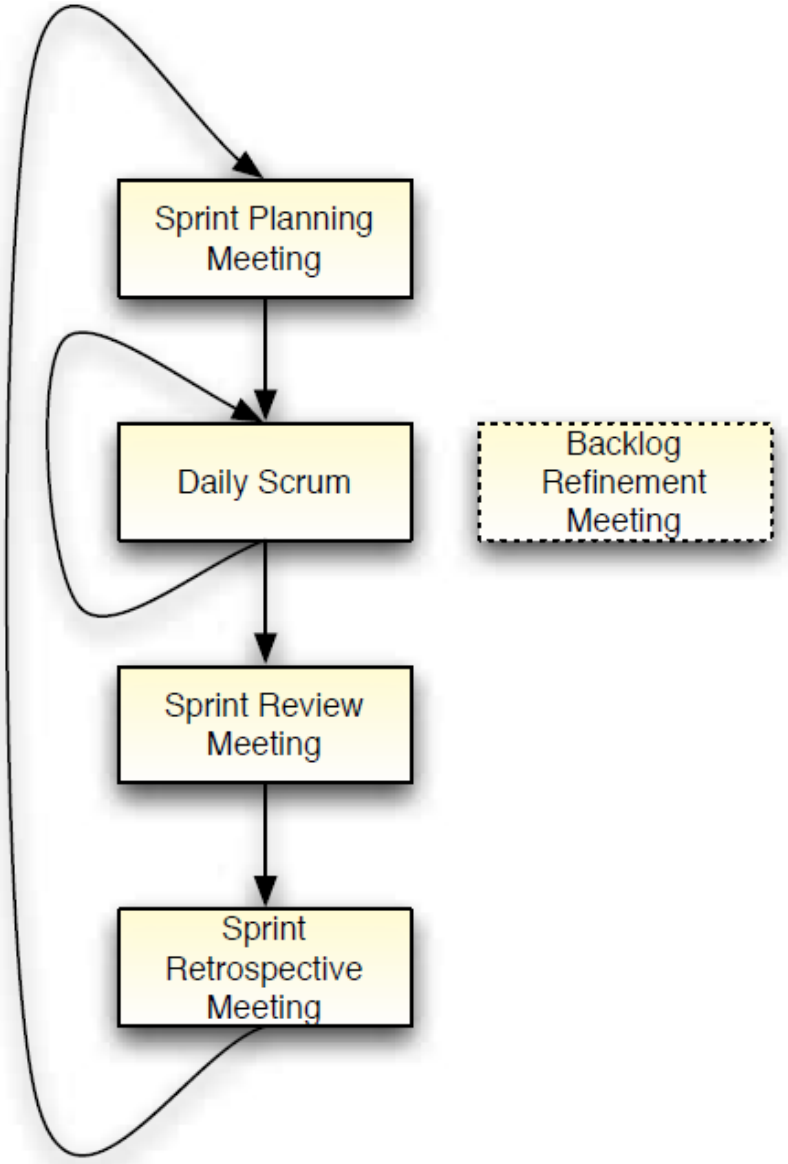
taken from www.mountaingoatsoftware.com (Mike Cohn)

# In your own group

Describe the differences between

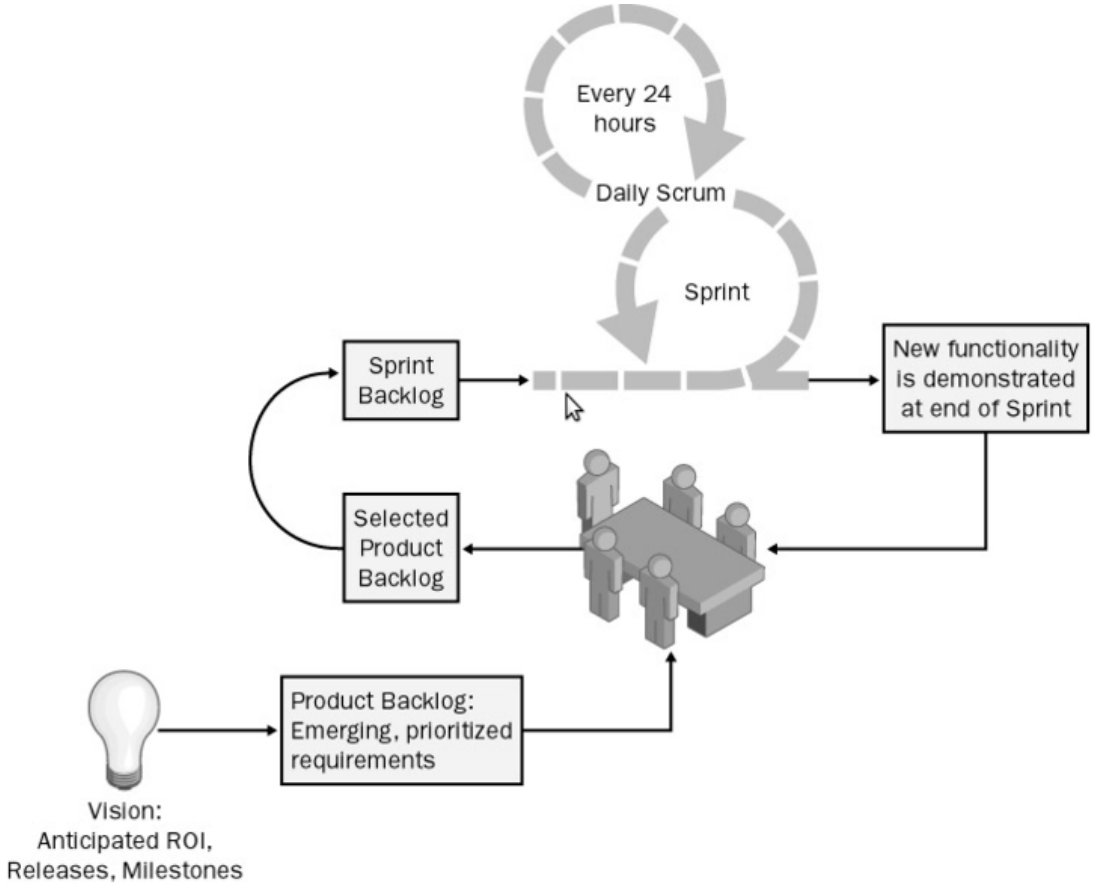
1. **Scrum** and **XP**
2. **Product Owner** and **Scrum Master**
3. **Product Backlog Item** and **Spring Tasks**

# Scrum: Process

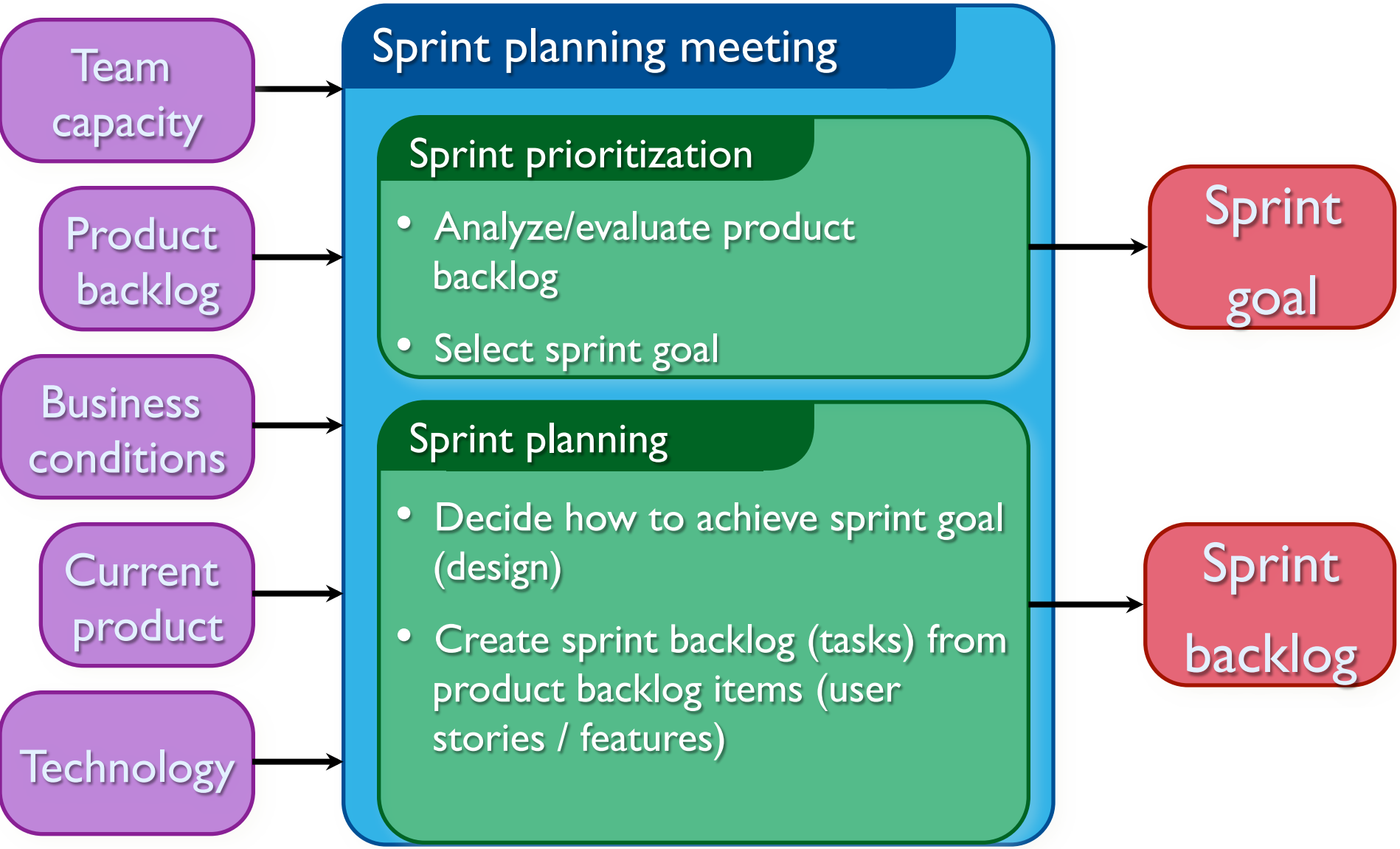


# Spring Planning Mtg

- Before each sprint
- No more than 8 hours (a day)
- Product Owner + Scrum Team



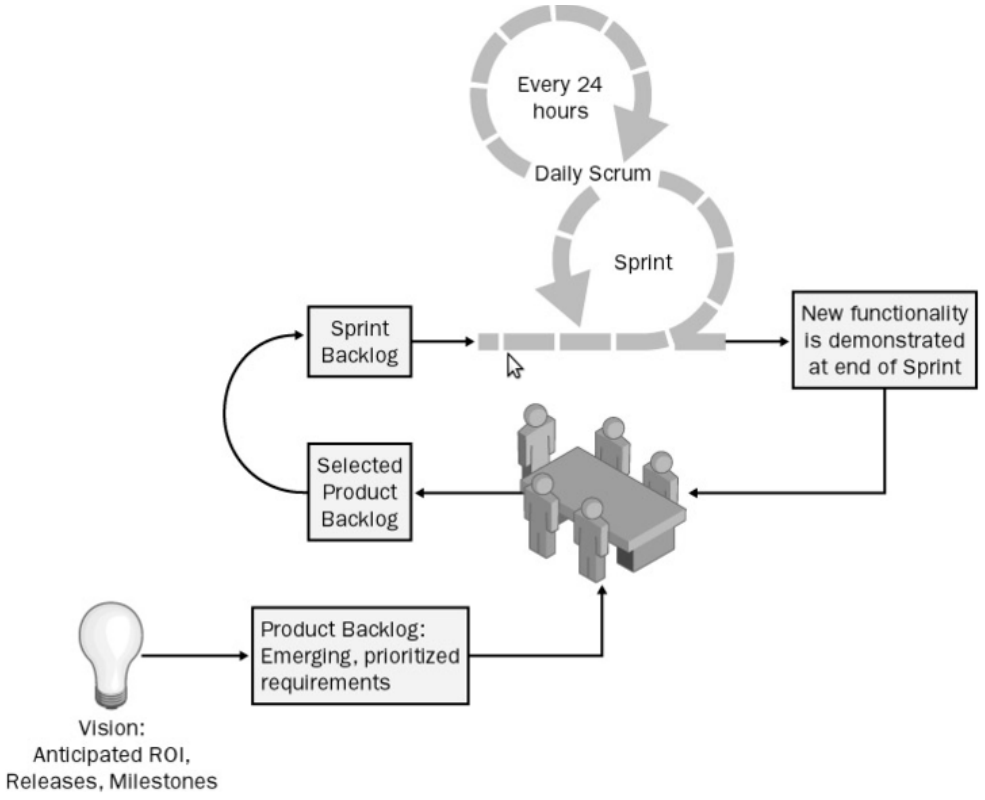
# Sprint Planning Mtg.



# The Sprint Starts

Who is responsible for prioritizing work and for selecting the work the team will perform during a sprint?

- Product Owner
- Scrum Master
- Scrum Team



# Daily Scrum Meeting

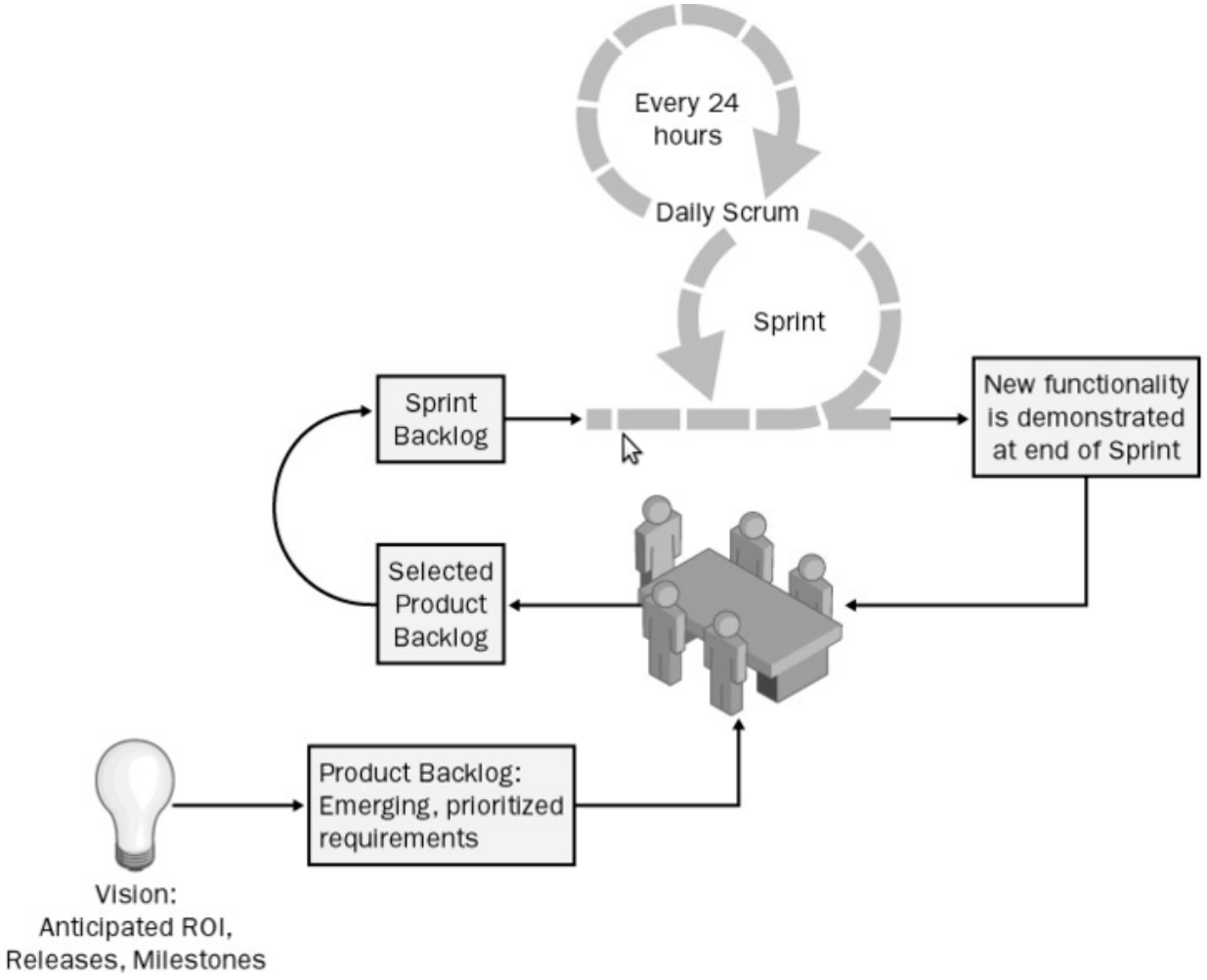
- Parameters
  - Daily, ~15 minutes, Stand-up
  - Same place, same time
  - Anyone late pays a $1 fee
- Not for problem solving
  - Whole world is invited
  - Only **team members**, **Scrum Master**, ***[product owner]***, can talk
  - Helps avoid other unnecessary meetings

Three questions answered by each team member:

1. **What did you do yesterday?**
2. **What will you do today?**
3. **What obstacles are in your way?**

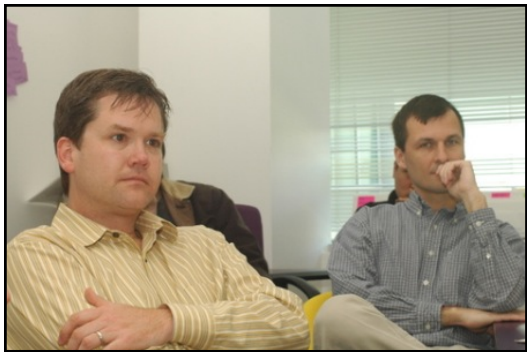
# The Spring Review

After each sprint



# The Sprint Review

- Team presents what it accomplished during the sprint
- A demo of new features or underlying design/ architecture
- To Product Owner and stakeholders
- Informal
  - 2-hour prep time rule
  - No slides
- Invite the world



# Sprint Retrospective

- After each Sprint, the Sprint team meets to reflect on its own process
  - What went well? What went wrong?
- Do **NOT**
  - Blame!
  - Get stuck in the past!
- Focus on the future: How can we improve?

# How not to be a Scrum Master

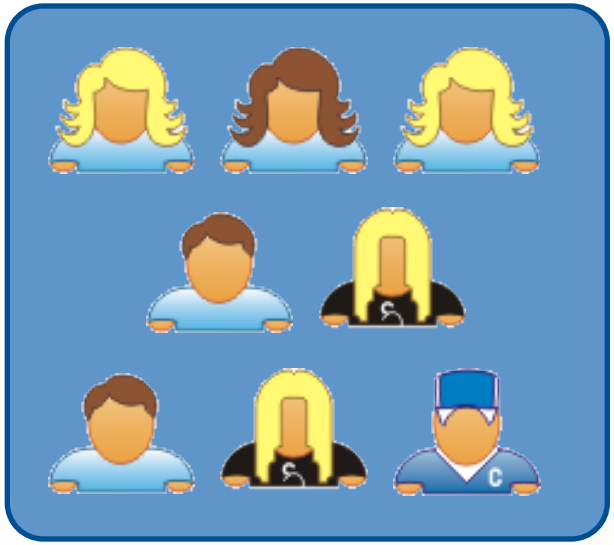
- https://www.youtube.com/watch?v=GGbsgs611MM

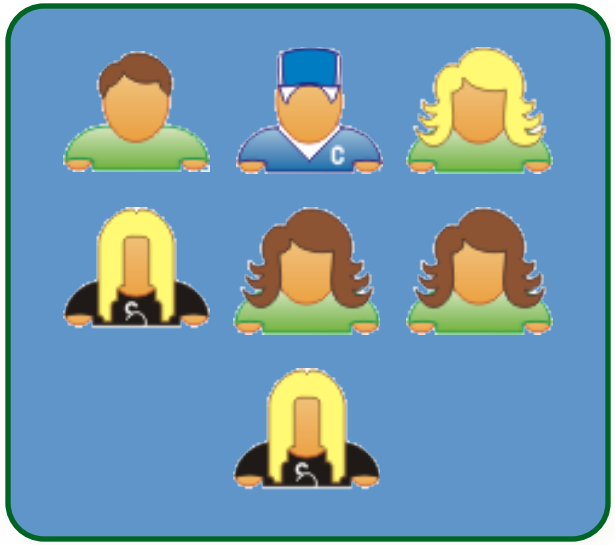
# Scalability

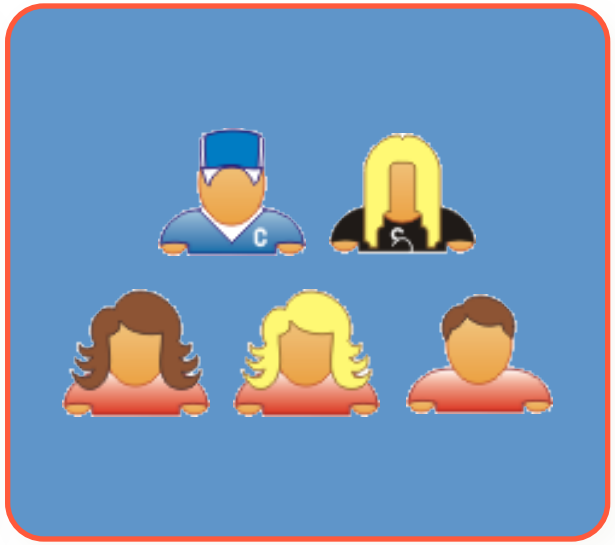
- Typical individual team is 7 ± 2 people
  - Scalability comes from teams of teams

- Scrum has been used on multiple 500+ person projects

# Scaling: Scrum of Scrums







# Scrum at Microsoft

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

# Class Activity

In your own group:

1) Discuss how your team will use Scrum for the lab assignments and project
2) Appoint a Scrum Master for your group
3) Write down 3 user stories for adding **UNDO**, **Save**, and **Difficulty** functionalities to JPacman.
   - Include:
     - **Effort estimation (story points or time)**
     - **Acceptance tests**

# Example – User Story

- As a shopper, I should be able to use my credit card, so that I can purchase items without cash.

- **Note:** Accept Visa, MasterCard, American Express.

- **Test:** (on the back of the story card)
  - Test with Visa, MasterCard and American Express (pass)
  - Test with good, bad and missing card ID numbers.
  - Test with expired cards (fail).
  - Test with over $100 and under $100.

# Summary: Software Process

- Why process
- Traditional, ‘waterfall’ models
- Agile models
  - XP/TDD
  - Scrum
    - User stories