California State University, Sacramento
Computer Science Department

CSC 131

Computer Software Engineering

Fall 2022

Lecture # 5

Agile Processes: Requirements Engineering

Agile Processes: Requirements

User Stories (Scrum, XP)

Very high "pointer to the" requirements from a stakeholder's perspective.

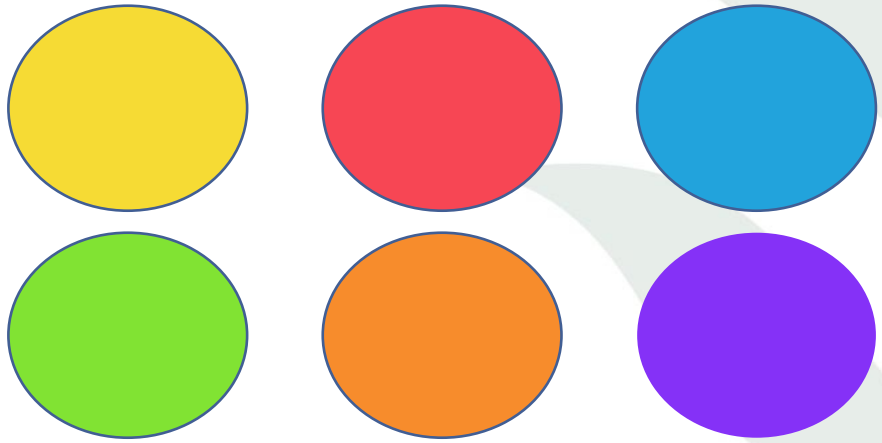
Suggested Wording:

- As a **<type of user>**, I want **<some goal>** so that **<some reason>.**
- As a **student**, I want to **see my current grade** so that I can **decide whether to drop the class**.
- As a **faculty member**, I can **print a class roster** so that I can **take attendance during class**.
- Can also follow a different format (but the one above is most popular) for example**:**
- "A student can view their current grades" or "A faculty member can print a class roster"

User Stories (Scrum, XP)

User Stories Focus on the WHO, WHAT and the WHY - Functionality.

A **user story** should be something that can be accomplished in **one sprint**. If it can't, it needs to be broken down.

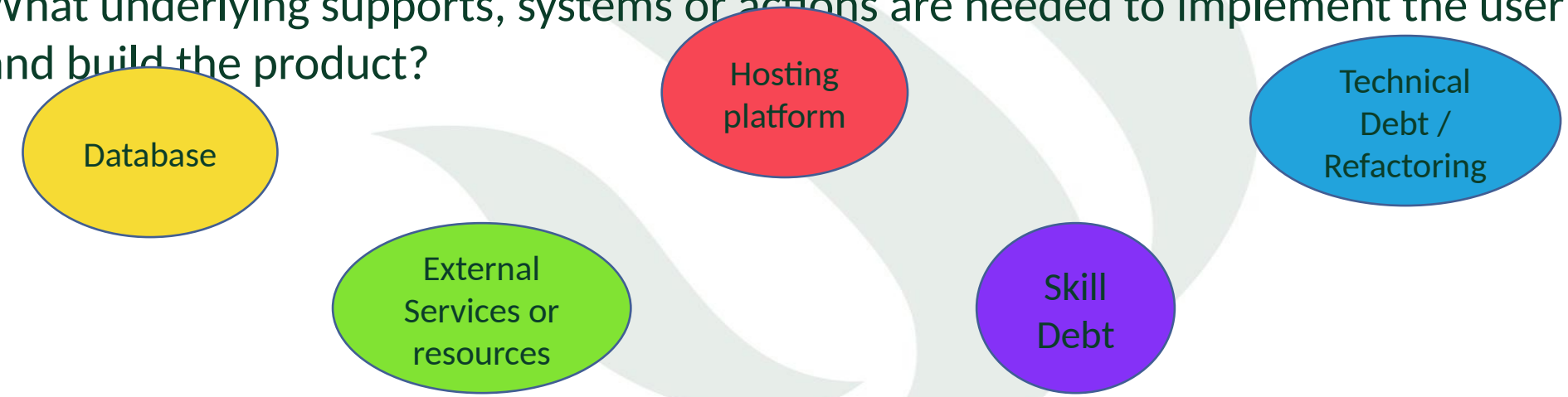


An **epic** is a high-level story spanning months of effort. It needs to be broken down into smaller pieces.

Technical Stories

Focus on the system and structures that support functionality.

- Technical stories are not from the user perspective
- Technical stories are from the system or developer perspective.
- What underlying supports, systems or actions are needed to implement the user stories and build the product?



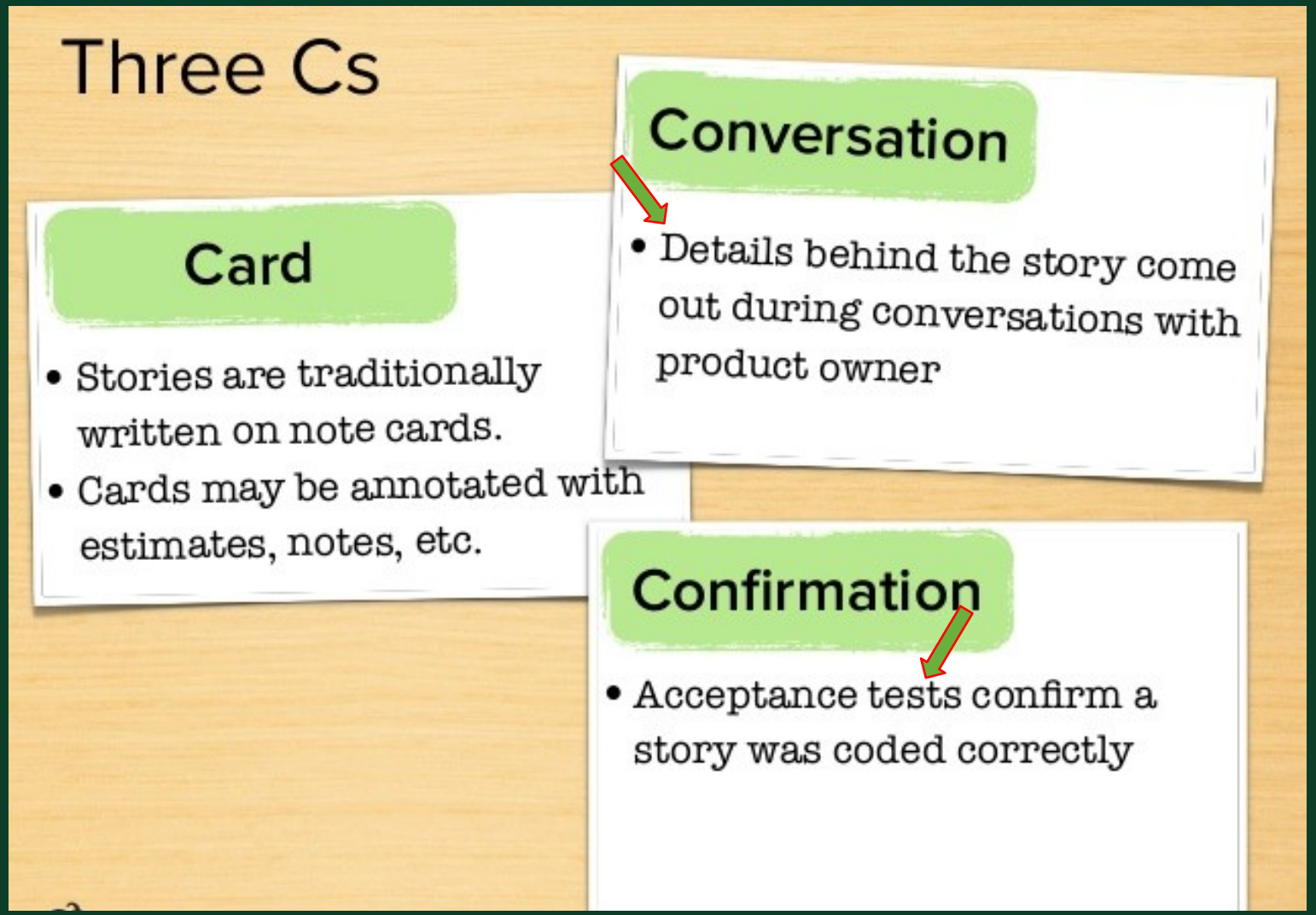
Requirements in Agile Processes: 3Cs

User stories also have **acceptance criteria** or **conditions of satisfaction.**

Three Cs of user stories:

- Card
- Conversation
- Confirmation

Define what it means for the story to be DONE.



“Where do we start”

1. The conversation **starts** when the User Story is first introduced to the team.
   - Read story and seek clarification from our Product Owner.
   - Discuss acceptance criteria.
   - Team provides an estimate, (seek more information if needed).
2. The conversation will **carry on throughout the implementation**.
   - Continual story refinement
     - Sprint planning tasks breakdown of user story.
     - Sprint coding & testing.
3. Conversations **happen up until** the point where the story is delivered to the customer.
4. The **final conversations** about a User Story happen in the Sprint Review.
   - Discuss what was implemented with stakeholders.

Agile requirements are understood through conversations.

Definition of Done (DOD) / Acceptance Criteria

The DOD defines the completeness criteria of a story. It includes all items that must be completed to claim that the story is done by the development team.

An example:

- Each criterion is verified during testing
- Coding tasks completed.
- Unit testing – written and passed.
- Regression test reviewed and passed.
- Integration test reviewed and passed.
- Code checked in to repository

Agile
Verifying and Validating Requirements

- The confirmation card is used to verify that the requirements were met.
- The sprint review includes stakeholders who will approve or disapprove the latest additions.
- The most careful check of requirements is during the reviews at the **conclusion** of each sprint.

- Major threat to this approach
  - Usually, **not all** stakeholders can attend every sprint review.

What if a user story is too big to fit into a Sprint?

Techniques for breaking up user stories

1. Look for a literal or implied **conjunction** “and” or “or” in the story and create two or more new stories from the parts around the conjunction.
2. Split by **process step** – every step is a new user story
3. Split by **I/O channel** – each I/O channel becomes a separate story
4. Split by user **options** – the options become the user stories
5. Split by **role/persona** – each role/persona becomes a separate story
6. Split by **data range** – every significant range is a new user story (e.g. 0, 1-9, 10-99 or last year, this year, next year).
7. Split by **CRUD action** – create, read, update, delete (only applicable if each action has business logic and they are asymmetric).

User Stories that are too big.

A user story should be a piece of work that can be accomplished in ONE sprint.

If the user story is too large to be completed in one sprint, **break it down** into multiple stories.

User Stories

Created during upfront and ongoing Discovery

As a user I want to create an account so that I can get discounts when I shop online.

Priority – 1 (High) Estimate – 5 Points

- As a user I want to create a basic account so I can shop online.
  As a user I want to create a full account so that I can get discounts.
  (**option** split)

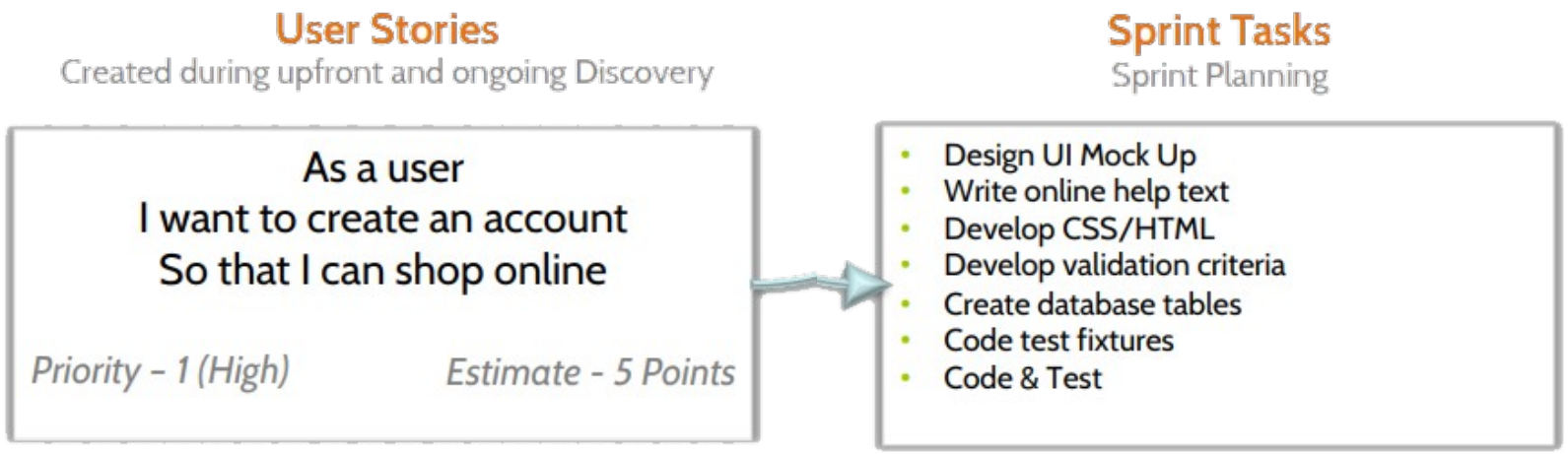
- As a user I want view and add items to my cart so I can purchase them. As a user I want to create an account so that I can get discounts.
  (**process step** split purchase vs. discount)

- Does a **user need** an account just to shop?
  As a user I want to complete my purchase so I can receive ordered items.
  As a marketer, I want users to create an account so I can target them.
  (**role/persona** split)

User Story Split - Example

A user story should be a piece of work that can be accomplished in ONE sprint.

How could you split this user story if the tasks were too much to accomplish in one sprint?

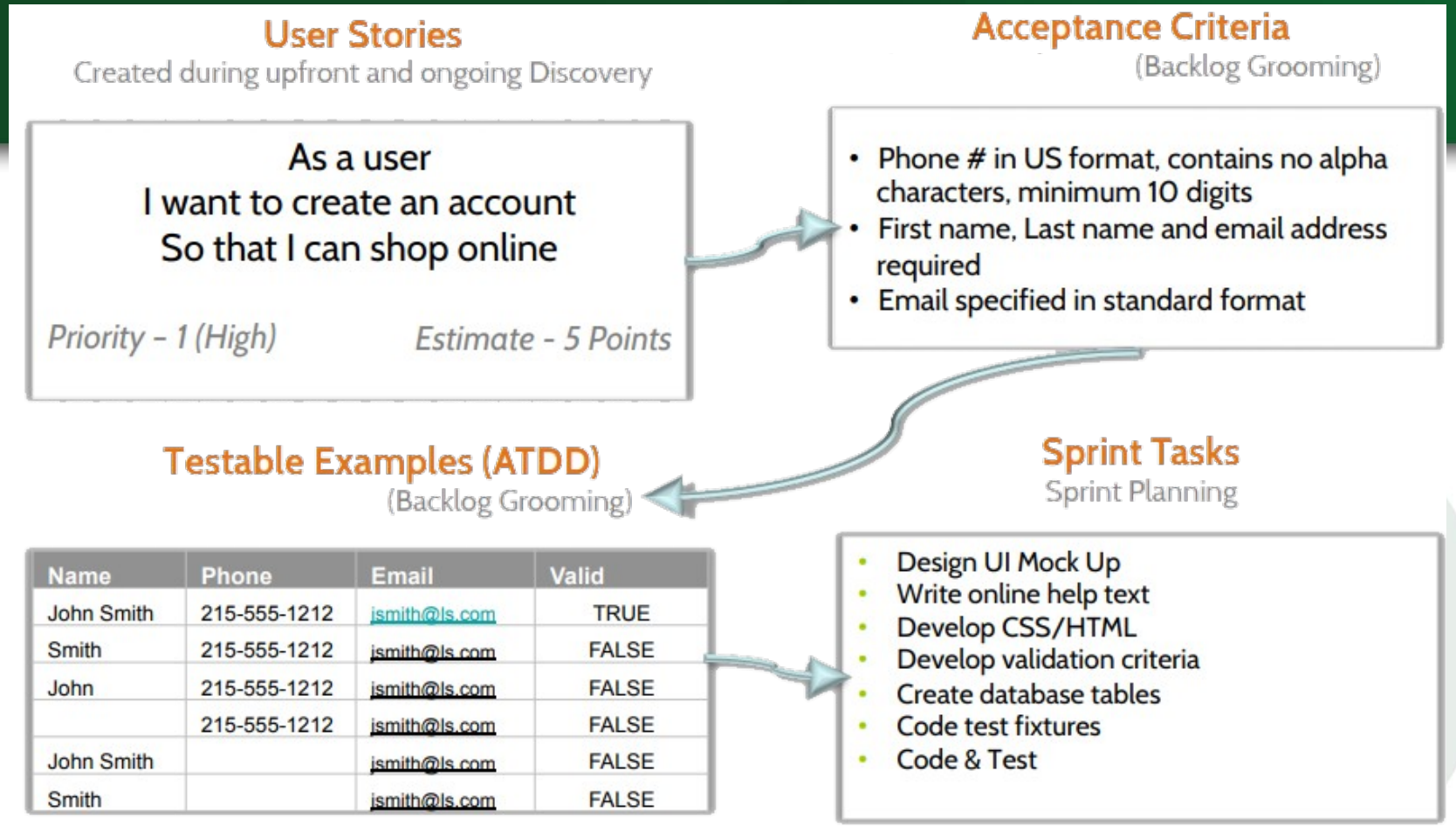


- If tasks are clearly too many for one sprint, consider splitting the story.
- If tasks failed to complete, they'll have to be moved to the next Sprint (imperfect world).

Splitting a User Story Due to Sprint Tasks

A task is a piece of activity that is required to get a story done.

Detail Tasks needed during the sprint to complete the user story.



ATDD = Acceptance Test Driven Development, optional step

Pmi.org

Sample Flow From User Story to Sprint Tasks

- Refine requirements
- Begin solution design.
- Identify tasks needed to complete user story.
- Use Acceptance Criteria to identify tests needed to verify the requirement.
- Detail Tasks needed during the sprint to complete the user story.

Product Backlog - Holds User Stories

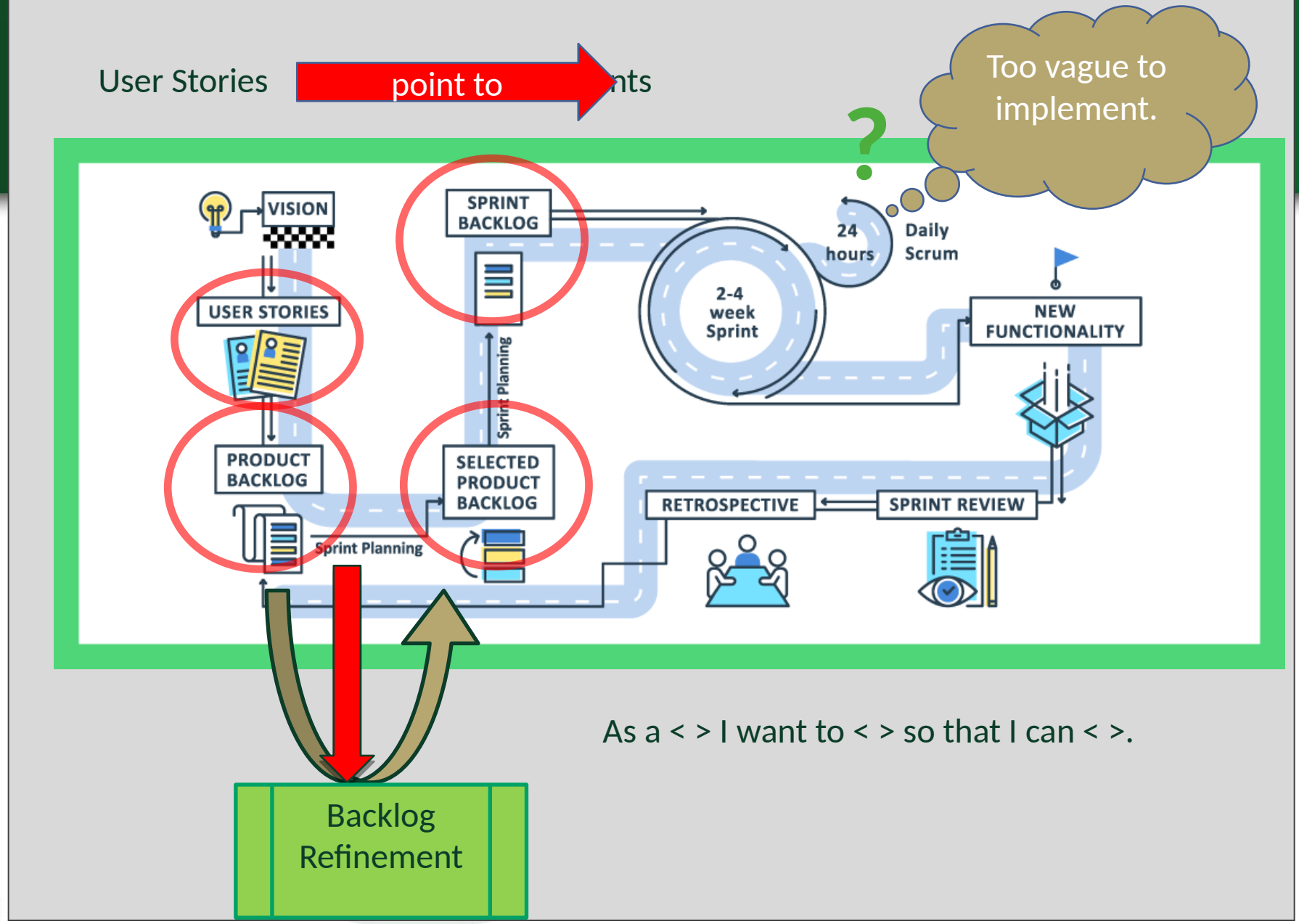
Ordered list of **everything** that will be needed in the product & single source of “requirements”

- The **Product Owner** is responsible for the **Product Backlog**, keeping up to date and prioritized.
- It lays out the initially known and best-understood requirements (user and technical stories).
- The requirements never stop evolving in reaction to changing stakeholder and product demands
- Lists all features, functions, requirements, enhancements, and fixes that must be made in the future.
- Product Backlog items have the attributes of a and **description, priority, estimate** and **value**.

Requirements in Scrum

Make specification and change as easy and cheap as possible by:

- *Delay **choosing requirements** as long as possible*
- ***Delay refinement as long as possible***
- ***Determine requirements in light of current product features***



SACRAMENTO STATE
Redefine the Possible

Backlog Refinement

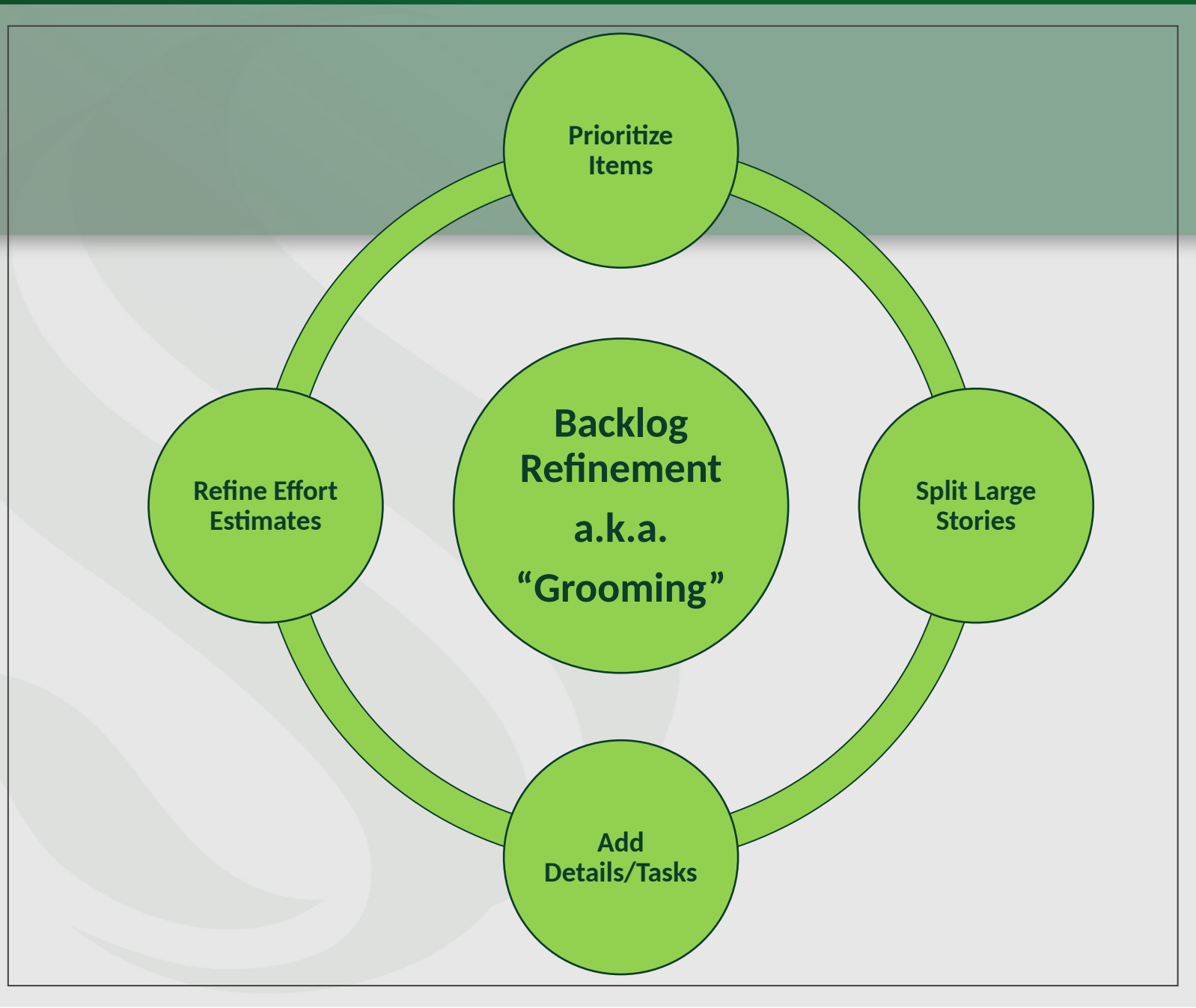
Product Backlog items are reviewed and revised.

Details, **Estimates**, and **Priority** are added to Backlog items.

Product Owner & Development Team **collaborate on the details** of Backlog items.

Scrum Team decides how and when refinement is done.

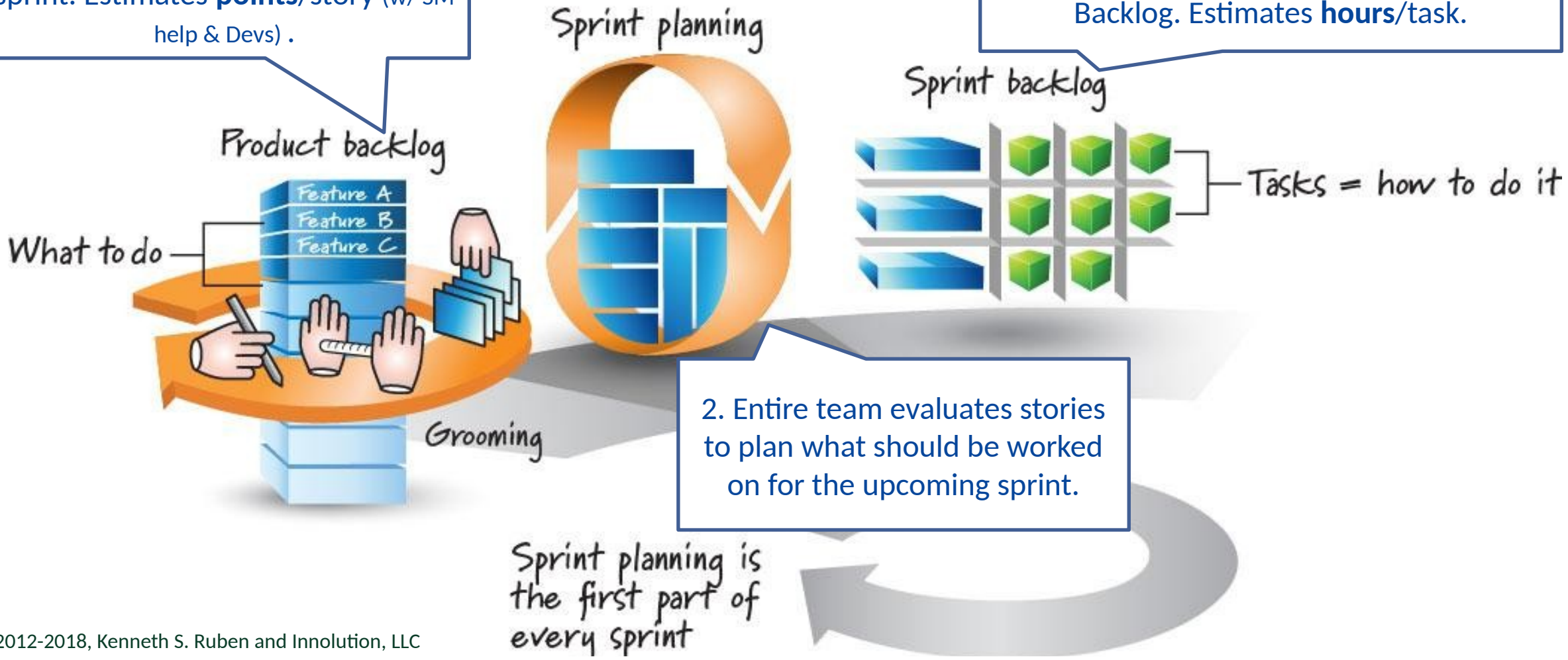
Backlog items can be updated at any time by the Product Owner.



SACRAMENTO STATE
Redefine the Possible

1. Product Owner ensures the backlog has stories ready for the sprint. Estimates **points**/story (w/ SM help & Devs) .

3. Dev team (w/ SM help) breaks down stories into the tasks needed to complete them – adds stories and tasks to Sprint Backlog. Estimates **hours**/task.



2. Entire team evaluates stories to plan what should be worked on for the upcoming sprint.

USER STORIES: FROM PRODUCT BACKLOG TO SPRINT BACKLOG

Scrum - Sprint Planning Meetings

The team should strive to have a “minimum viable” product at the end of every sprint

Team members report on their availability.

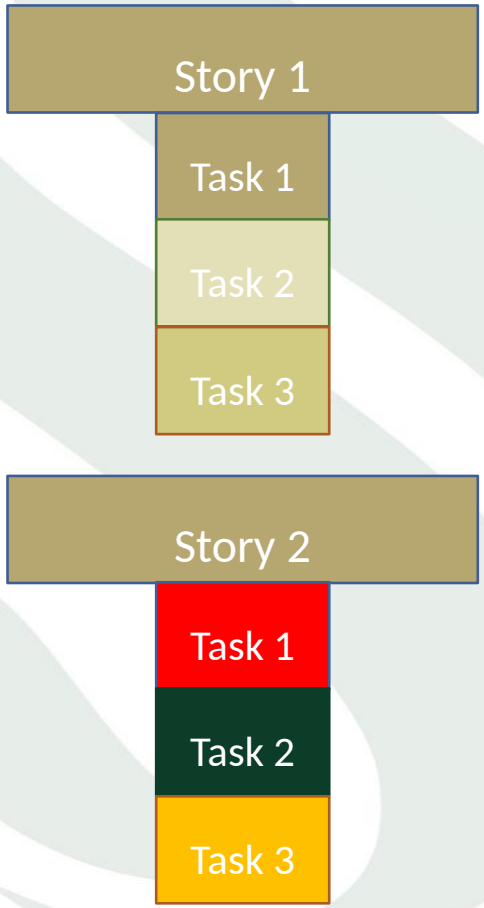
- The top item in the product backlog is broken down into tasks that are recorded in the Sprint Backlog (if not already done via grooming).
- Team members volunteer for the tasks (i.e., take ownership of them) and provide time estimates, in hours.
- The process continues until all available hours are accounted for.
- It takes at least 1 sprint to be able to understand how story points translate to hours.

Scrum - Sprints

Product Backlog

Story 1
Story 2
Story 3
Story 4
Story 5
Story 6
Story 7

Sprint Backlog



Sprint Tasks

SACRAMENTO STATE
Redefine the Possible

Agile: Requirements Management

- Agile projects begin with some project mission statement in the form of epics or feature.
- The product owner refines these high-level features into **user stories**.
- During each sprint, user stories are **refined** into more detailed requirements and then implemented.
- At the conclusion of each sprint, stakeholders and the product owner validate requirements during **sprint review**.
  - Requirements management is a **core activity** of agile methods, but it is incorporated into the broader processes of **backlog grooming** and sprinting, and not distinguished as a separate process.
- It is not necessary to keep historical record of implemented user stories or backlog tasks.

User Stories (Scrum, XP)

User Stories Focus on the WHO, WHAT and the WHY.

A user story should be something that can be accomplished in **one sprint**. If it can't, it needs to be broken down.

Epics – High level stories spanning months of effort, needs to be broken down into smaller pieces.

Features – Descriptions differ:

- Larger piece of functionality than a story but smaller than an epic.
- Requirements from the system's perspective/ backend items (not a user's story)

Themes – A collection of user stories.