Scheduling Work

- Scheduling an increment plan depends on the following input:
  - Requirements
    - What is it you need to develop?
  - Work estimates
    - How much effort is involved?
  - Priorities
    - What’s most important to the customer?

Requirements

- To facilitate incremental scheduling, requirements should ideally demonstrate the following characteristics:
  - Independent
    - Requirements should be defined independent of each other so that their functionality can be demonstrated independent of the availability of other functionality.
    - For functionality that cannot be isolated, dependencies need to be taken into account when scheduling an increment plan.
  - Demonstrate value
    - The development of a requirement should lead to value that can be demonstrated to the customer/client.
  - Appropriately sized
    - Requirements should be no shorter than half a day and no longer than the length of an increment.

Requirements – Practical Viewpoints

- Good ideas for features emerge throughout the project, not just at the beginning when requirements are gathered
- When customers see working software, they identify lots of changes and improvements they'd like to make
- Our customer's business is constantly evolving, and this causes their requirements to change
- There is a lot of miscommunication and misunderstanding – between the team and their customer, or between the customer and their stakeholder – and often this is only spotted when we see working software
- What we design can't always be built, and what we build often doesn't work the first time, so changes must be made
- We are human, and we often forget or overlook things

Estimating Techniques

- Agile estimating is similar to traditional estimating.
- You have the choice to use one or many of existing estimating techniques: ideal days, story points, function points, …
- Ideal days measure estimates in days it would take to complete a requirement if there were no overhead.
  - One person's ideal days may not be the same as another person's ideal days.
  - Ideal days are easy to understand but may set the wrong expectations.
  - You may not want to communicate estimates in days with your client.
- Story points are a measure of relative size.
  - Fibonacci sequence: 1,2,3,5,8,13
  - « Estimating Poker Game »

Story Point Estimates

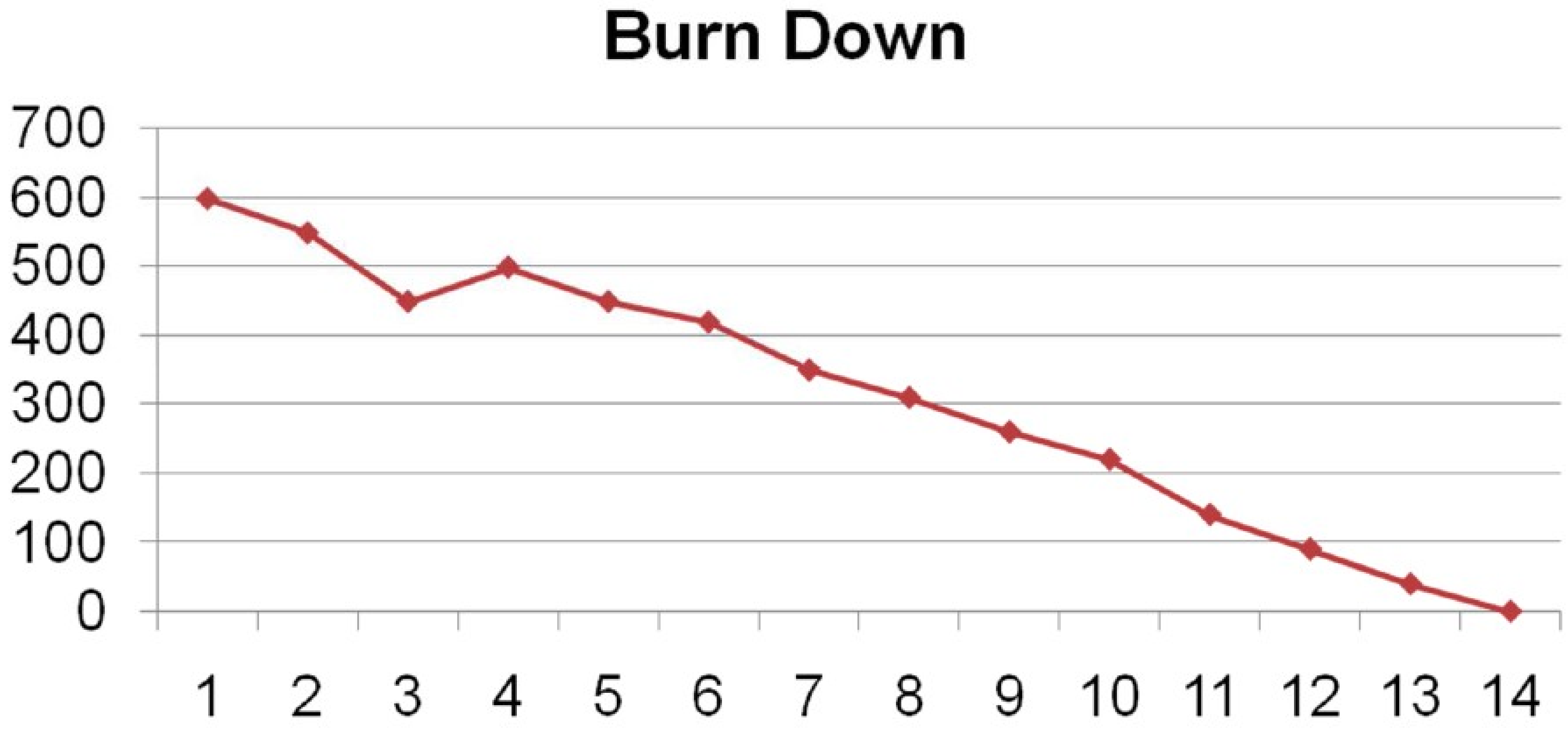
- Story points provide data on relative size.
- Story points are best produced by the development team.
- Story point estimates can be useful complementary estimates to an ADM estimate.
  - They invite the team to be involved with the estimating process.
  - Developers who are close to the details can provide additional input.
  - The team feels more engaged
  - They trigger team discussions around design approach and understanding of requirements
- Given the relative nature of story points, they may be more appropriate for communicating size to the client.
- Using multiple estimating techniques can sometimes lead to better results as they force you to think about discrepancies between estimates.

Velocity

- Velocity is a measure of how much work is performed in an increment.
- Velocity can be expressed in ideal days or story points.
- An initial velocity is set at the start of the project to schedule work to increments.
- This can be total work estimate divided by the number of increments.
- Less work will be done in the first couple of increments (≈60%)
- Story points can be roughly translated to ideal days by dividing the work estimate by the number of story points.
- Velocity is measured at the end of each increment to track progress and forecast time of completion.
- Velocity forecasts become more accurate as iterations are completed.

Measuring Progress

- Important to define “done” upfront
- Always aim at completing the increment's target velocity



Scheduling Implications

- Early estimates may be highly inaccurate
- Estimates get more accurate as more information is available
- Gather just enough requirement details so that you have data to estimate within an acceptable range of error.
- Spend roughly 20 to 30% of project calendar time on upfront requirements definition to reduce variance within the range of contingency
- Move to design, build, test, deploy iterations for the remainder of the time
- Requirement details that have no significant impact on estimates can be moved to iterations.

User Stories

- What are User Stories?
- Users and user roles
- Why User Stories?

Psychology of User Stories

- Software requirements is a communication problem
- What problem do stories address?
  - Those who want the software must communicate with those who will build it
- We need a way of working together so that resource allocation becomes a shared problem
- Project fails when the problem of resource allocation falls too far to one side
- We cannot perfectly predict a software schedule
  - As users see the software, they come up with new ideas
  - Too many intangibles
  - Developers have a notoriously hard time estimating
  - If we can’t perfectly predict a schedule, we can’t perfectly say what will be delivered

The Paradigm Shift

We make decisions based on the information we have

... but do it often

Rather than making one all-encompassing set Of decisions

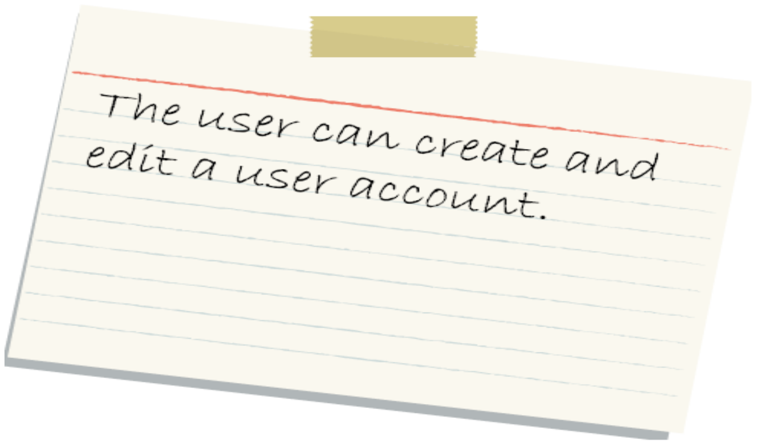
... we spread decision-making across the project

This is where User Stories come in

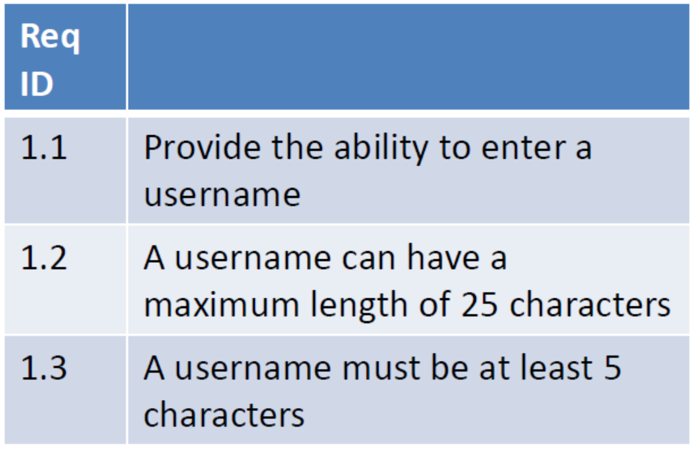
User Stories

- Agile methodologies prefer user stories to gather requirements.
- User stories are short written descriptions of desired functionality based on client conversations.
- Lower-level details are left out to ease communication between customer and development and are documented elsewhere (notes, annex, test script, ...) but need to provide sufficient detail to allow accurate estimating.

User stories are often written on cards


The user can create and edit a user account.

We want to focus on building functionality

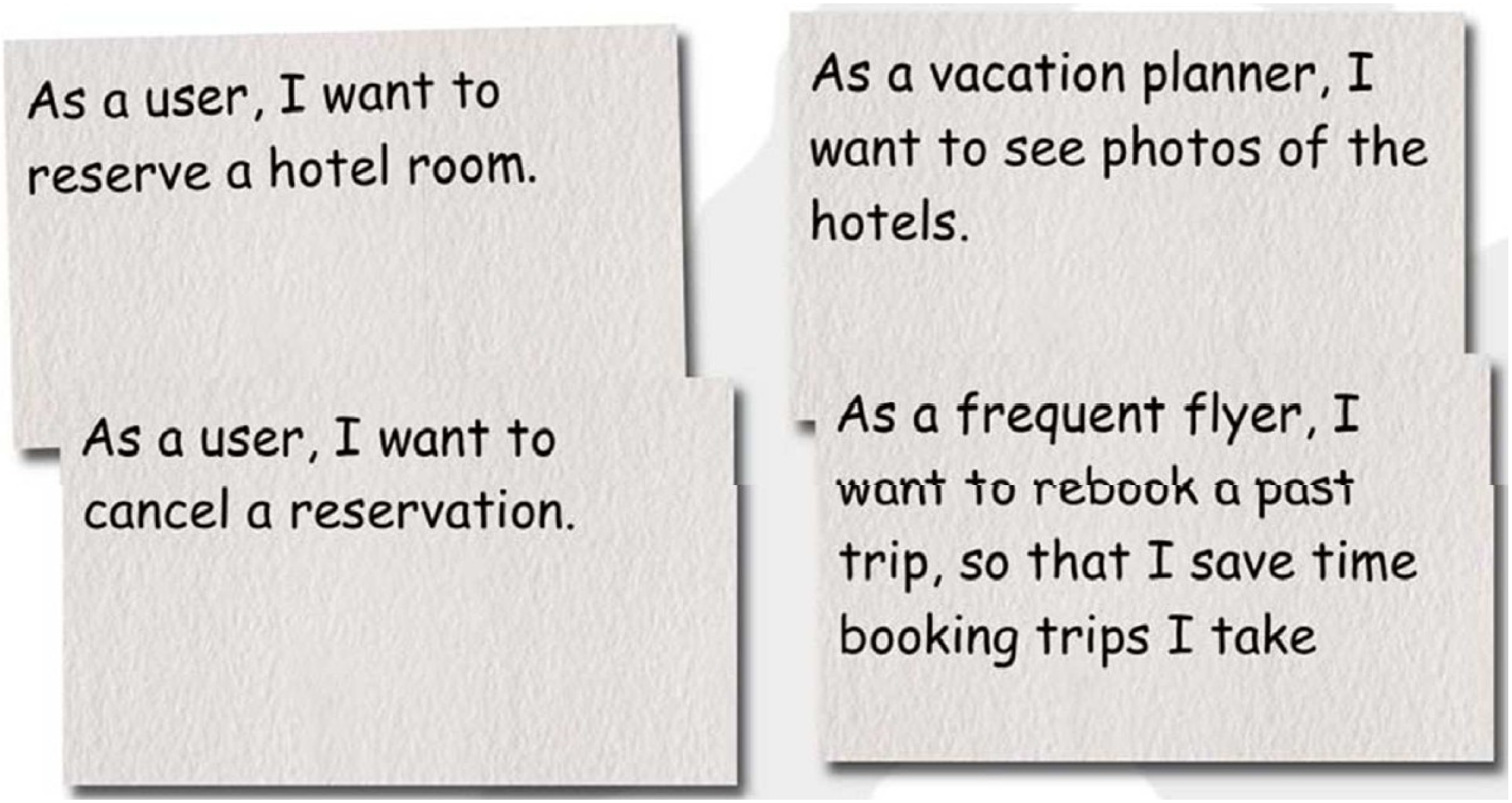


Req ID	
1.1	Provide the ability to enter a username
1.2	A username can have a maximum length of 25 characters
1.3	A username must be at least 5 characters

What are User Stories?

- 3 Cs
  - Card
    - User stories are converntionally written on cards
      - Although, increasingly software tools are being used.
    - Cards may include estimates, notes, etc.
  - Conversation
    - Details emerge from conversations with product owners
  - Confirmation
    - Acceptable tests confirm correctly coded stories

What are User Stories?



As a user, I want to reserve a hotel room.

As a vacation planner, I want to see photos of the hotels.

As a user, I want to cancel a reservation.

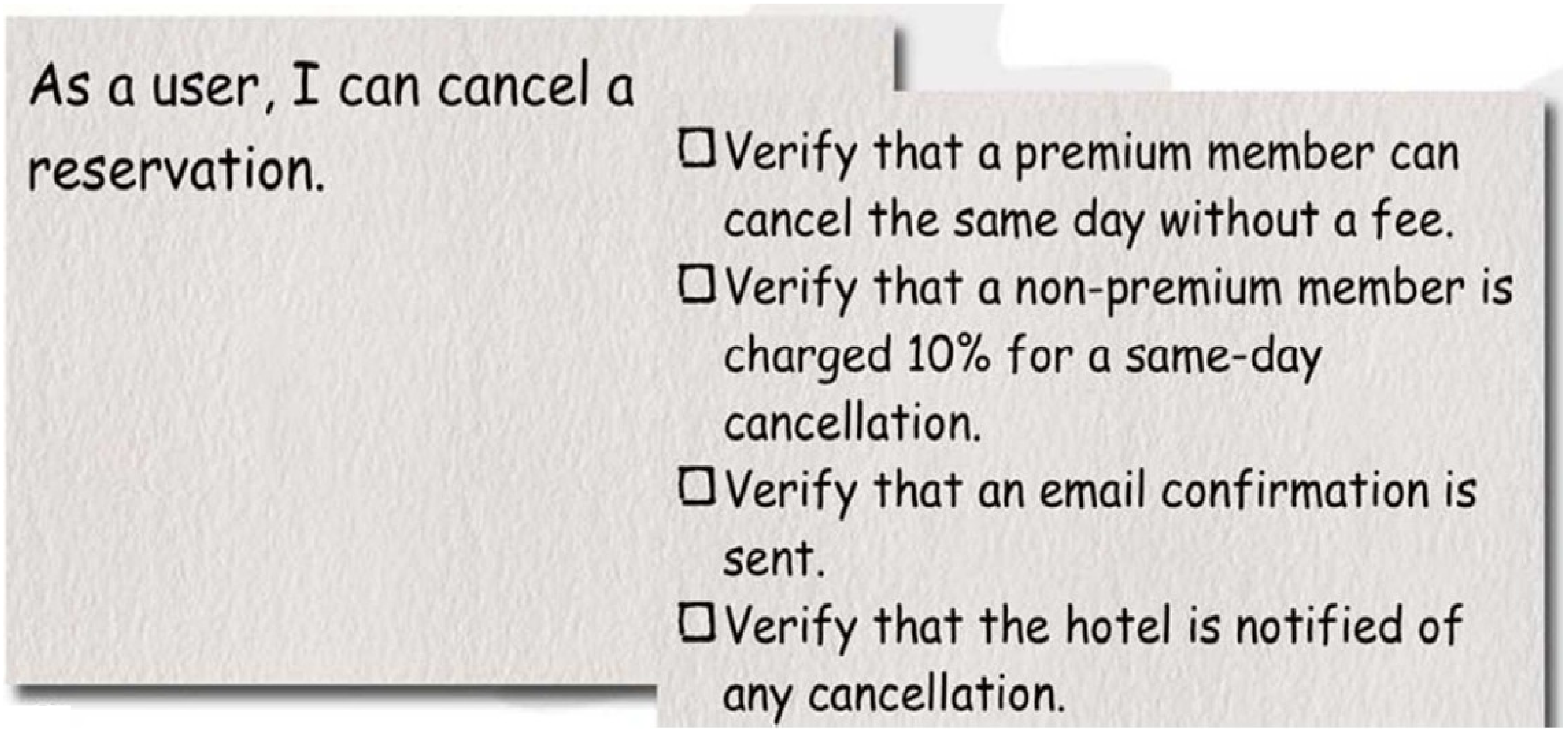
As a frequent flyer, I want to rebook a past trip, so that I save time booking trips I take

Where are the details?

- As a user, I can cancel a reservation.
  - Does the user get a full or partial refund?
    - Is the refund to her credit card or is it site credit?
  - How far ahead must the reservation be cancelled?
    - Is that the same for all hotels?
    - For all site visitors? Can frequent travellers cancel later?
  - Is a confirmation provided to the user?
    - How?

A) Details as Conditions

- Conditions of satistaction can be added to a story
  - These are essentially tests.

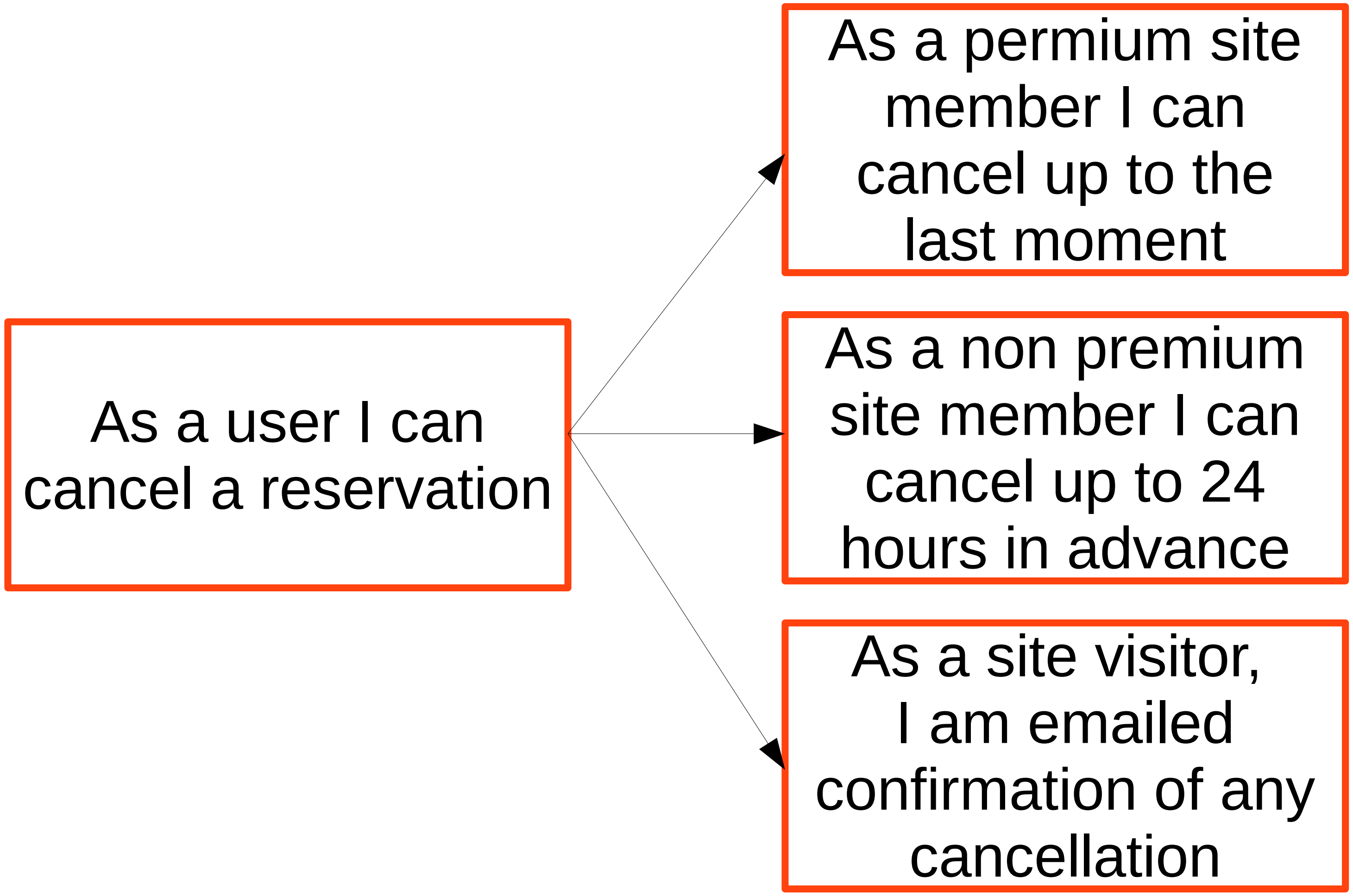


As a user, I can cancel a reservation.

- ☐ Verify that a premium member can cancel the same day without a fee.
- ☐ Verify that a non-premium member is charged 10% for a same-day cancellation.
- ☐ Verify that an email confirmation is sent.
- ☐ Verify that the hotel is notified of any cancellation.

B) Details Added to Sub-Stories

- Details can be added to sub-stories

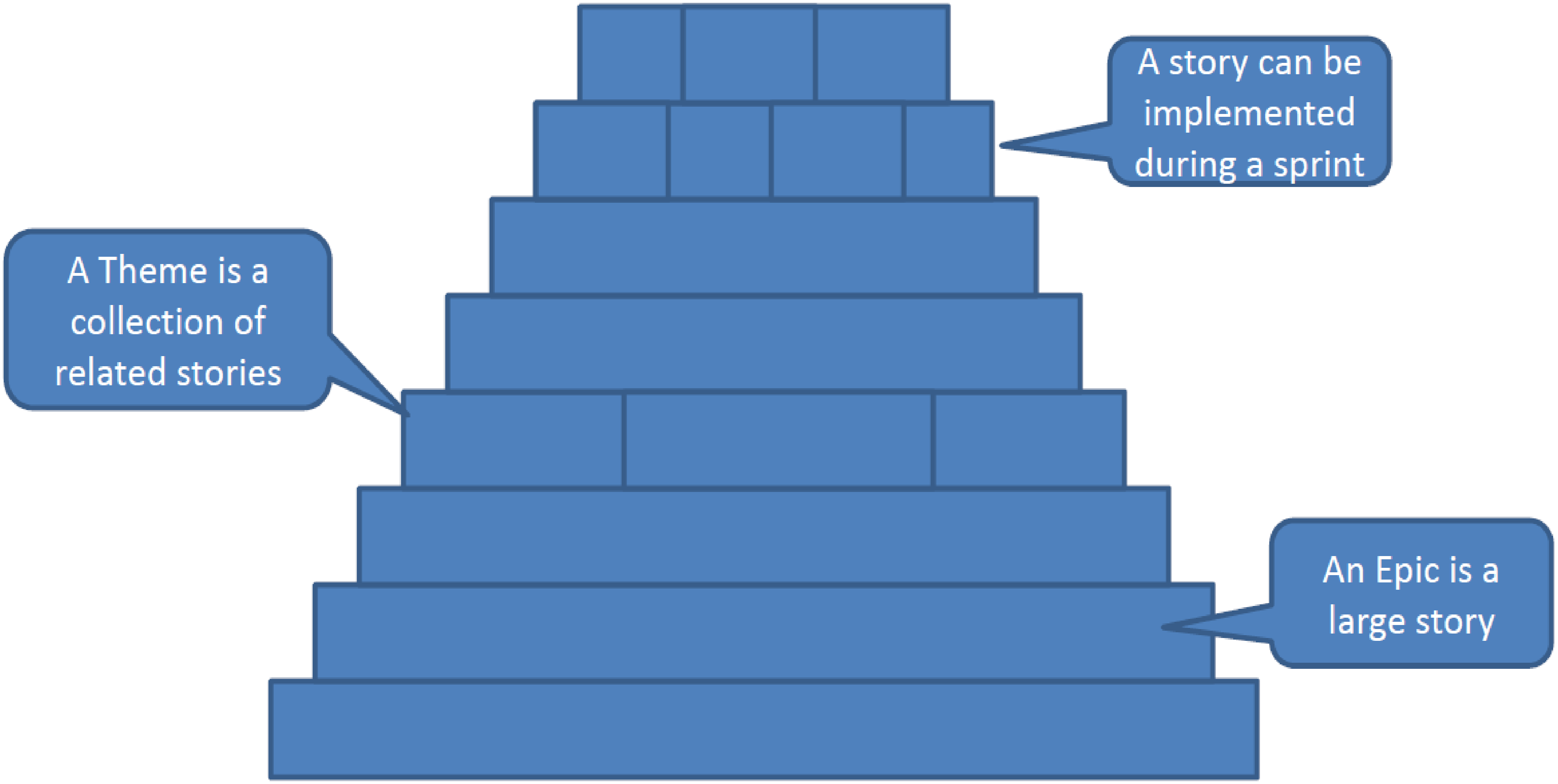

As a user I can cancel a reservation
As a permium site member I can cancel up to the last moment
As a non premium site member I can cancel up to 24 hours in advance
As a site visitor, I am emailed confirmation of any cancellation

Handling Details

- These techniques can be combined
- The approaches are not mutually exclusive
- At implementation time
  - All stories must have satisfaction conditions

Product Backlog

- Product backlog is made up of stories


A story can be implemented during a sprint
A Theme is a collection of related stories
An Epic is a large story

Augment as Necessary

- User stories are not dogma
- Augment them with written documentation as appropriate
  - Business rules
  - Data dictionaries
  - Use cases
  - Examples of inputs and expected result

“The User”

- Many projects mistakenly assume there’s only one user:
  - “The user”
- Write all stories from one user’s perspective
- Assume all users have the same goals
- Leads to missing stories

User Roles

- Broaden the scope from looking at one user
- Allows users to vary by
  - What they use the software for
  - How they use the software
  - Background
  - Familiarity with the software / computers
- Used extensively in user-centred design

Exercise 1

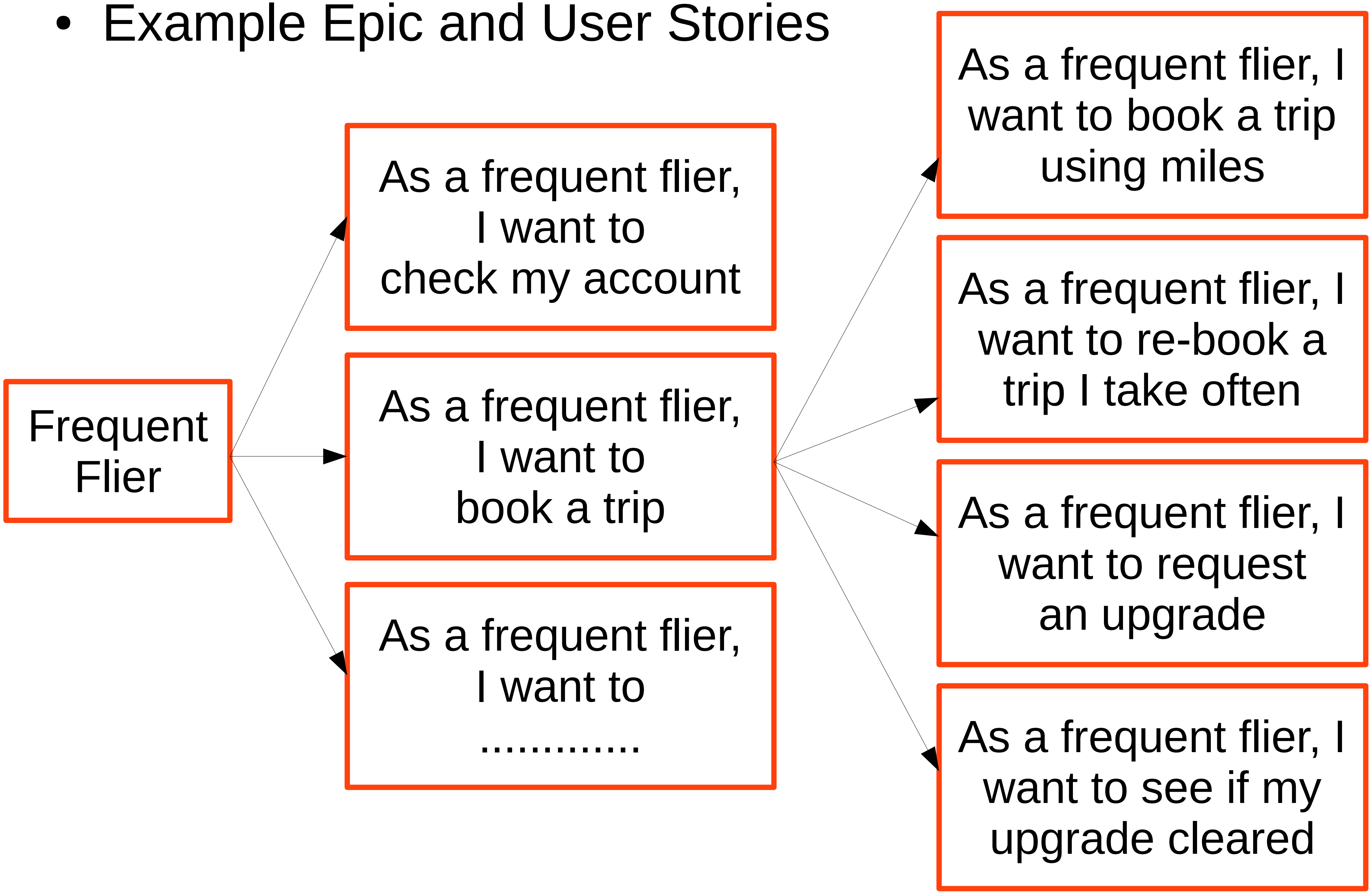
- User Role Brainstorming

We've been hired by to develop a website for the local high school. To get us started they've said they like the school website on the next page.

Brainstorm the user roles who will interact with this site.

Exercise 2

- Example Epic and User Stories


Frequent Flier
As a frequent flier, I want to check my account
As a frequent flier, I want to book a trip
As a frequent flier, I want to ....
As a frequent flier, I want to book a trip using miles
As a frequent flier, I want to re-book a trip I take often
As a frequent flier, I want to request an upgrade
As a frequent flier, I want to see if my upgrade cleared

Exercise 2

Start with the roles you've identified. For two or three, Think of their top-level goals and write some epics. Then convert a couple of epics into more usable stories.

Tip:
Try this temple:
“As a <user role>, I want
<goal> so that <reason>”.

Why User Stories?

Stories shift the focus from writing to talking

Stories emphasize the user's goals not the system's attributes.

Stories are equally understandable by developers and customers

Stories support and encourage iterative development

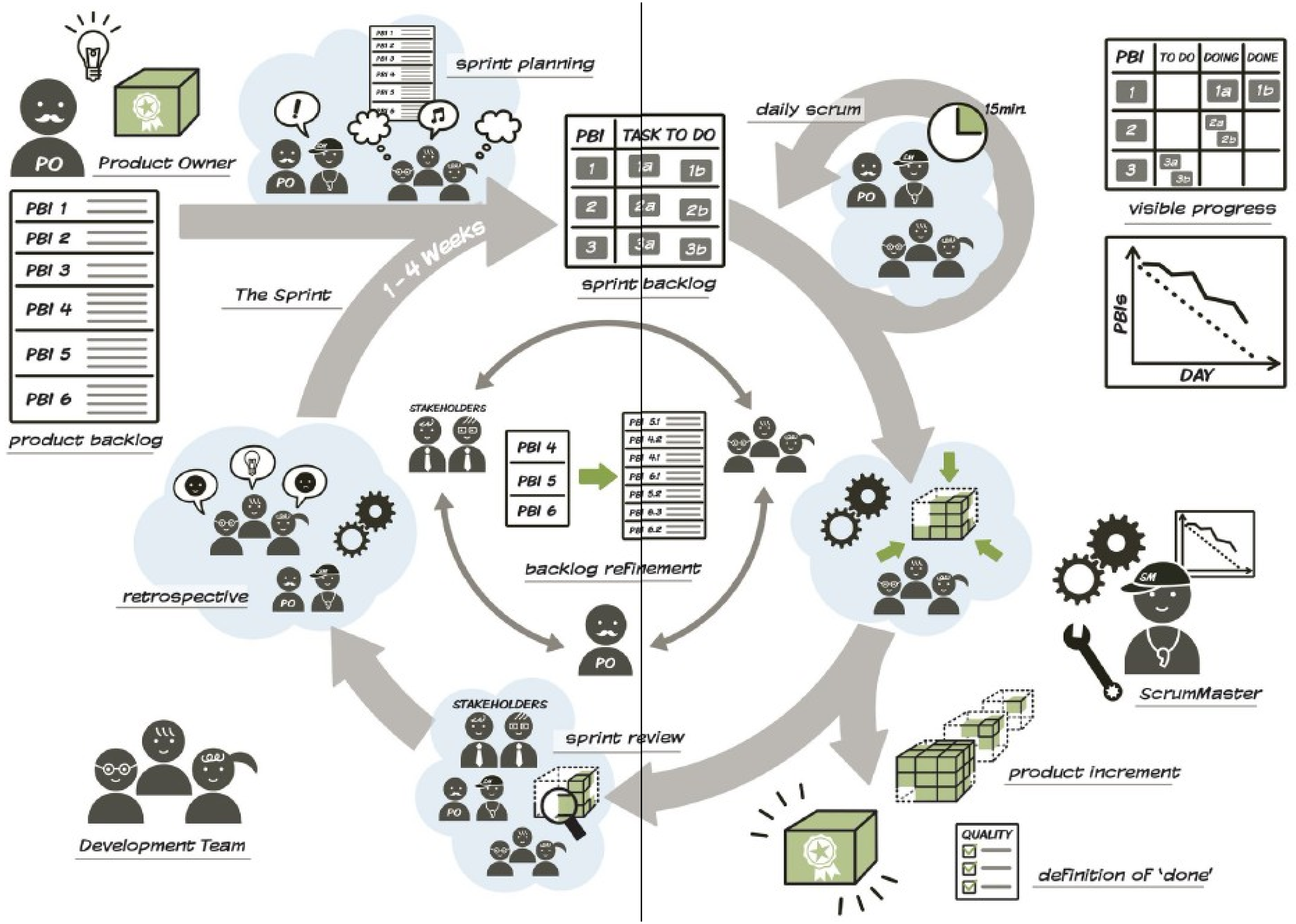
Stories are the right size for planning

Stories support participatory design

Summary

- Focus on software deliverables
  - Can be reviewed and tested
  - Help improve quality of estimates
- Understanding Scrum
  - Scrum Roles, Scrum Process, Requirements
  - Prioritising work that brings value and Measuring Work progress
- What are User Stories?
  - Users and user roles
  - Exercises
  - Why User Stories?

Scrum Flow


Product Owner
PO
PBI 1
PBI 2
PBI 3
PBI 4
PBI 5
PBI 6
product backlog
sprint planning
The Sprint
1 - 4 Weeks
PBI
TASK TO DO
1
1a
1b
2
2a
2b
3
3a
3b
sprint backlog
daily scrum
15min.
PBI
TO DO
DOING
DONE
visible progress
PBIs
DAY
STAKEHOLDERS
backlog refinement
retrospective
sprint review
product increment
QUALITY
definition of 'done'
ScrumMaster
SM
Development Team