# Split Your User Stories Like You Mean it

Petri Heiramo, CST

Belgrade RSG, Oct 2024

ScrumAlliance®

too generic

# ~~As a user, I would really like to be able to login, so that...~~

How would you improve the story above?

Pick a pair and discuss. 2 minutes.

# Two Things

1. There is just too much text. Lots of words with no added value.
2. User is very often too vague. Seek more descriptive actor names.

For example,

Member can login, so that…

Avoid:

Unnecessary Text

Ambiguous Actors

# System can run automated maintenance scripts at night, so that...

How about this one?

Return to your pair and discuss. 1 minute.

Avoid:

Unnecessary text

Ambiguous Actors

# This is important

Never use system itself as the actor. Always look for an external actor (human or other computer system) who has an interest in the described functionality.

For example,

Admin can schedule maintenance scripts, so that...

Avoid:

Unnecessary text

Ambiguous Actors

System itself as Actor

# Analyst can ~~press a button~~ to ~~open a popup~~ view ~~with~~ financial details, so that...

Third time's the charm, right?

Return to your pair and discuss. 1 minute.

Avoid:

Unnecessary text

Ambiguous Actors

System itself as Actor

# Remember,

Avoid describing the solution in the story. Focus on the problem or need. Any solution descriptions in the story title become hard requirements.

For example,

Analyst can view financial details, so that...

If it's important to write down the solution details, add them into notes associated with the story.

Avoid:

Unnecessary text

Ambiguous Actors

System itself as Actor

Defining the solution

# Backend can push new stock price data to UI every 5s, so that...

In this example, the story describes an interaction between two internal components / layers. This is very similar to the “system itself as Actor”, but using internal boundaries. Please, do not do that.

Instead, use something like this:
Analyst can see live stock data, so that...

Avoid:

- Unnecessary text
- Ambiguous Actors
- System itself as Actor
- Defining the solution
- Internal boundaries

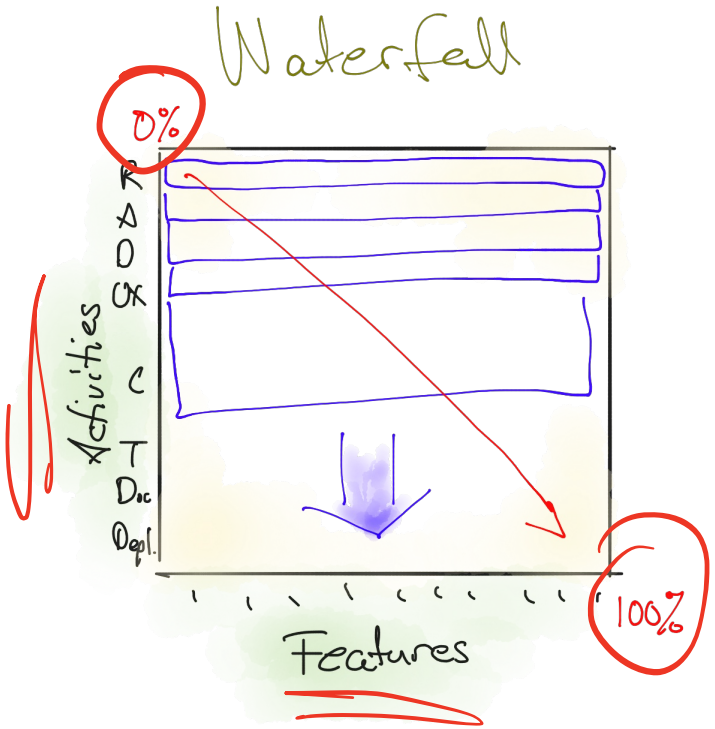
# Why are proper stories important?

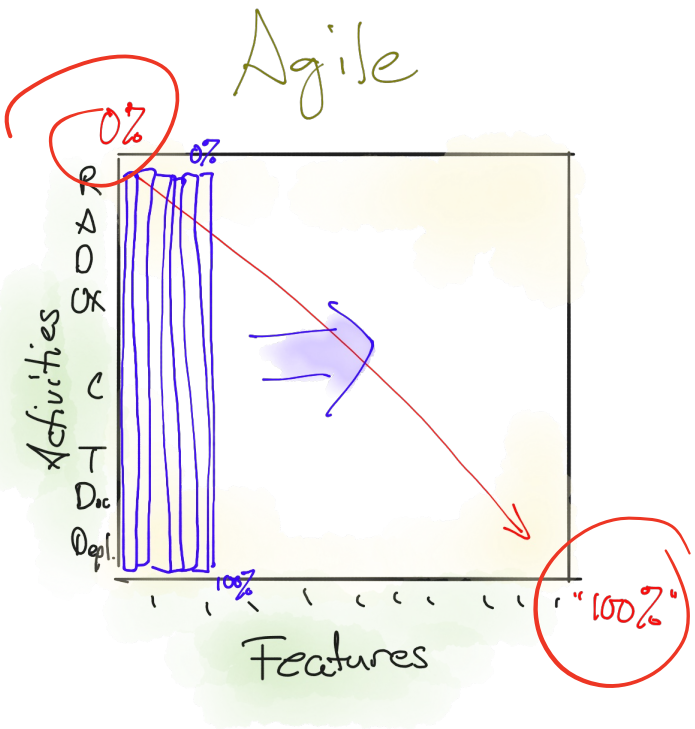
ScrumAlliance®

**Well, they aren’t, really.**

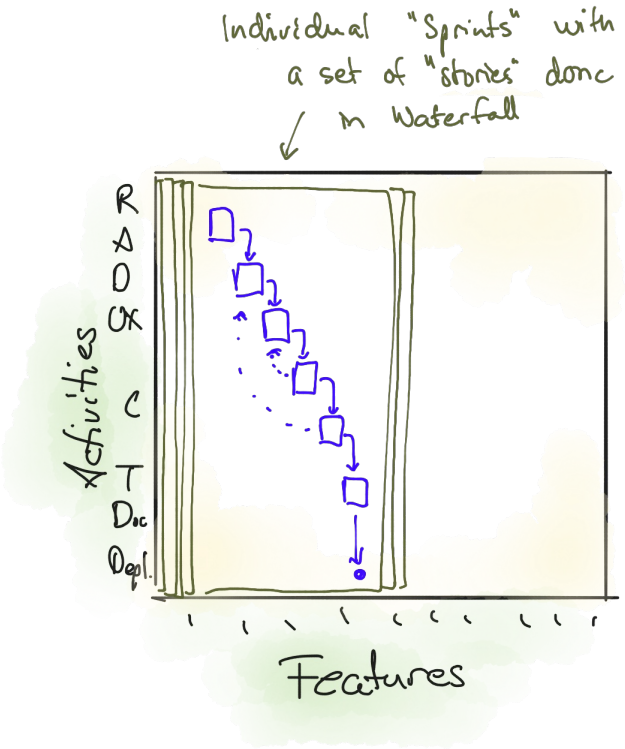
**What *is* important, is vertical slicing.**

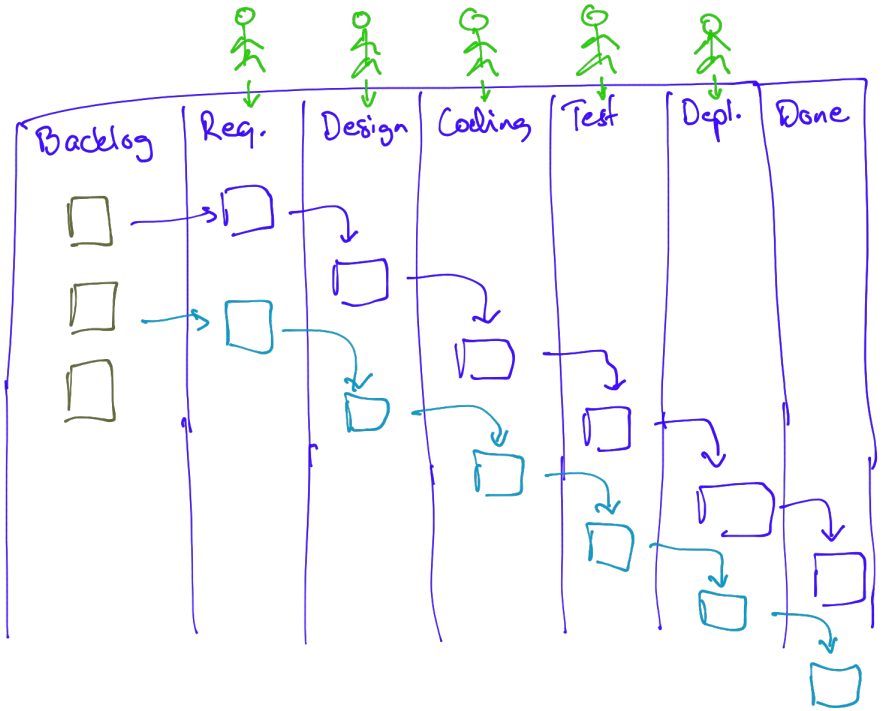
# This here is obvious:



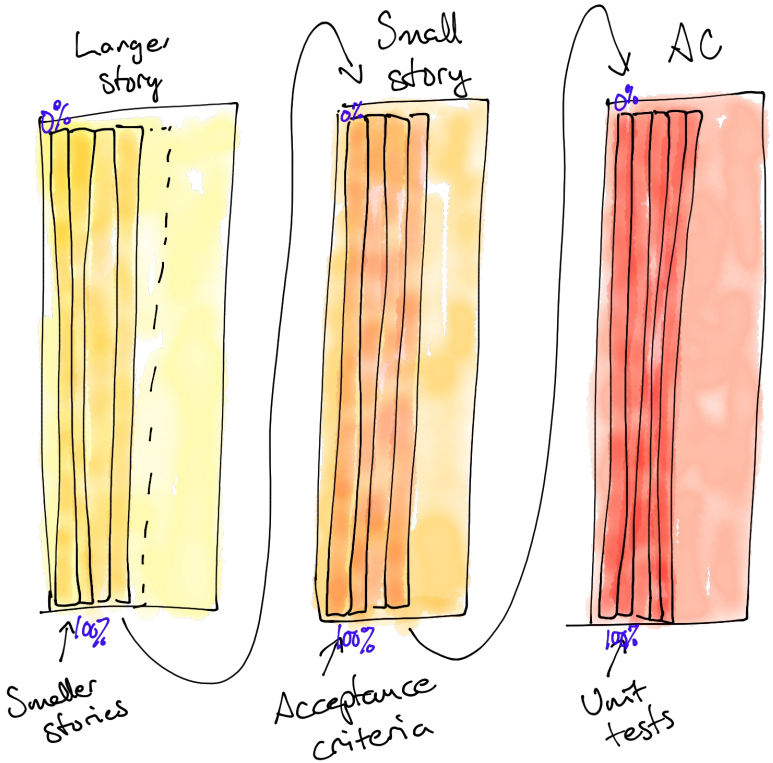



# But this often happens – mini-waterfalls



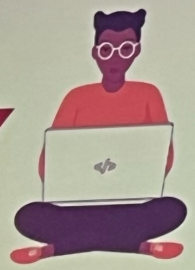
# We need to maintain the vertical slicing



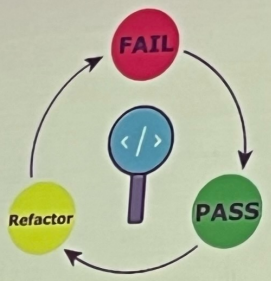
# A Real Pull System for Software

This is PULL - Demand flows this way.

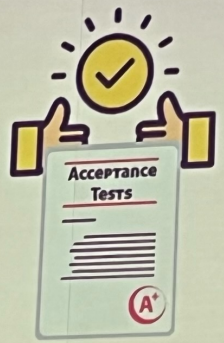
End Here



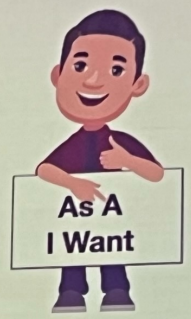
Developers write code to pass tests.



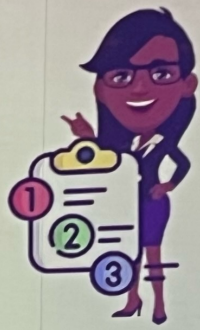

Write Failing Tests
TDD/Clean Code




Acceptance Criteria
ATDD




Business Analysis




Customer Needs

Start Here

This is PULL - Fulfillment flows this way.

**Demand flows upstream. Fulfillment flows downstream. A kanban pull system is one where only downstream teams or processes can add work to an upstream queue. Adding work further upstream is a push system.**

# Let’s start drilling down

# Member can login, so that...

This is an “epic”, i.e. “too large and unclear”.
What features could we split out as smaller stories?

Pick a pair and discuss. 2 minutes.

Avoid:

Unnecessary Text
Ambiguous Actors
System itself as Actor
Defining the solution
Internal boundaries
Activity breakdown

# For example,

Visitor can create an account, ...

Member can login with social media accounts, ...

First-time member can see a helpful welcome page, ...

Admin can set up Single Sign-On, ...

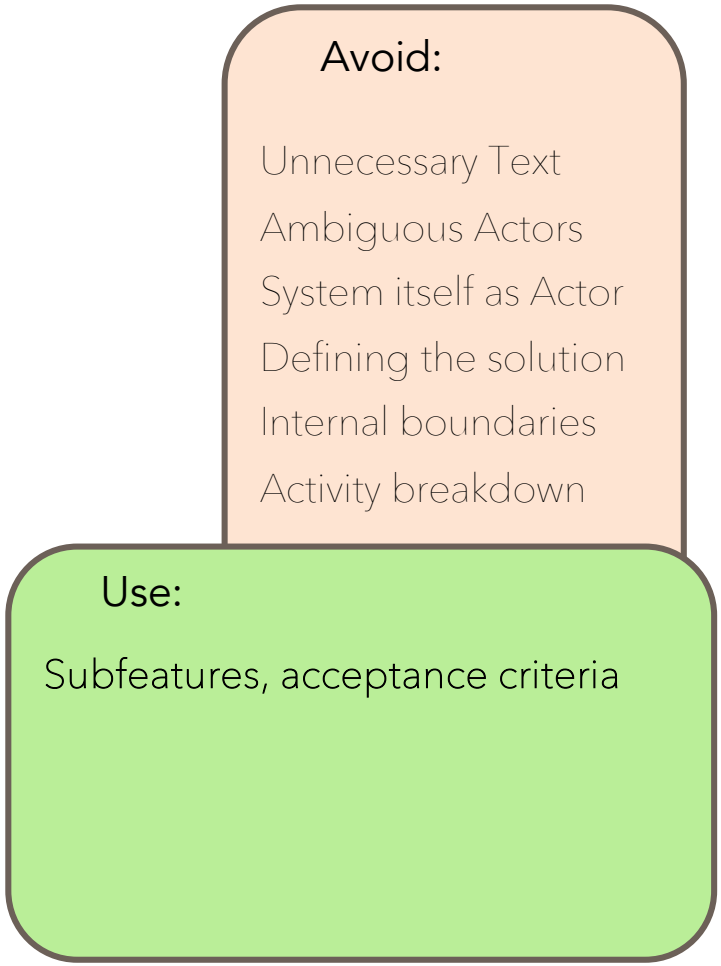
Member can logout, ...

Member will be automatically logged out after 15 minutes of inactivity, ...

Member can reset a forgotten password, ...

Admin can manage user accounts, ...

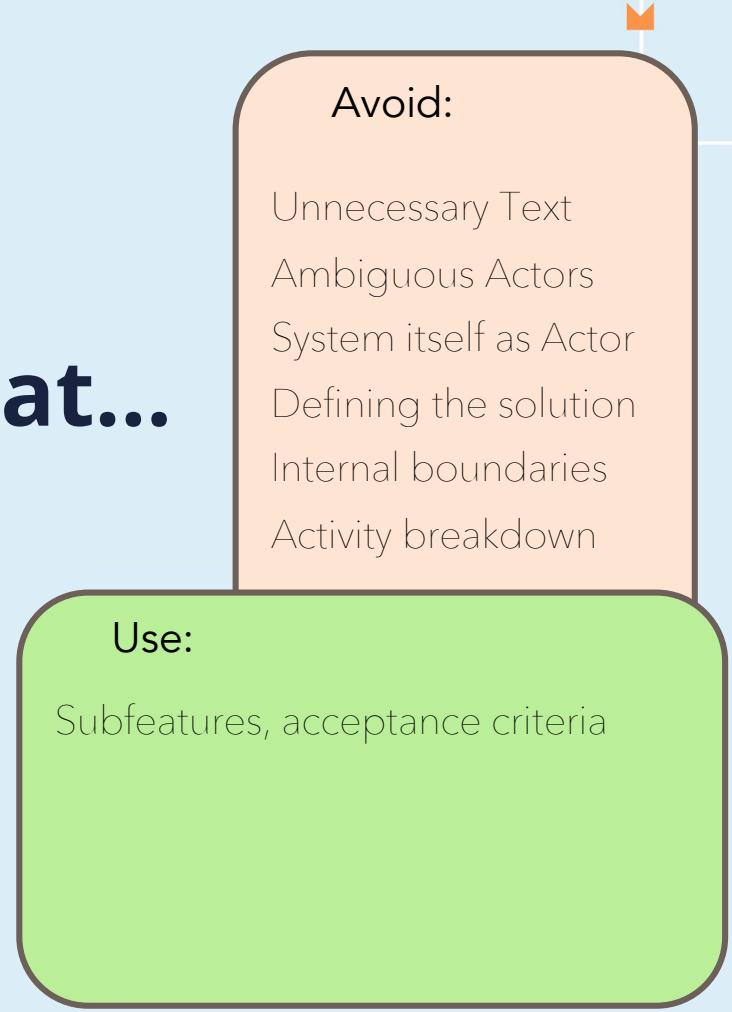
Member can stay logged in between sessions, ...



# Member can login with social media accounts, so that...

Let's keep going deeper. What smaller stories could we split out of this one?

Pick a pair and discuss. 2 minutes.



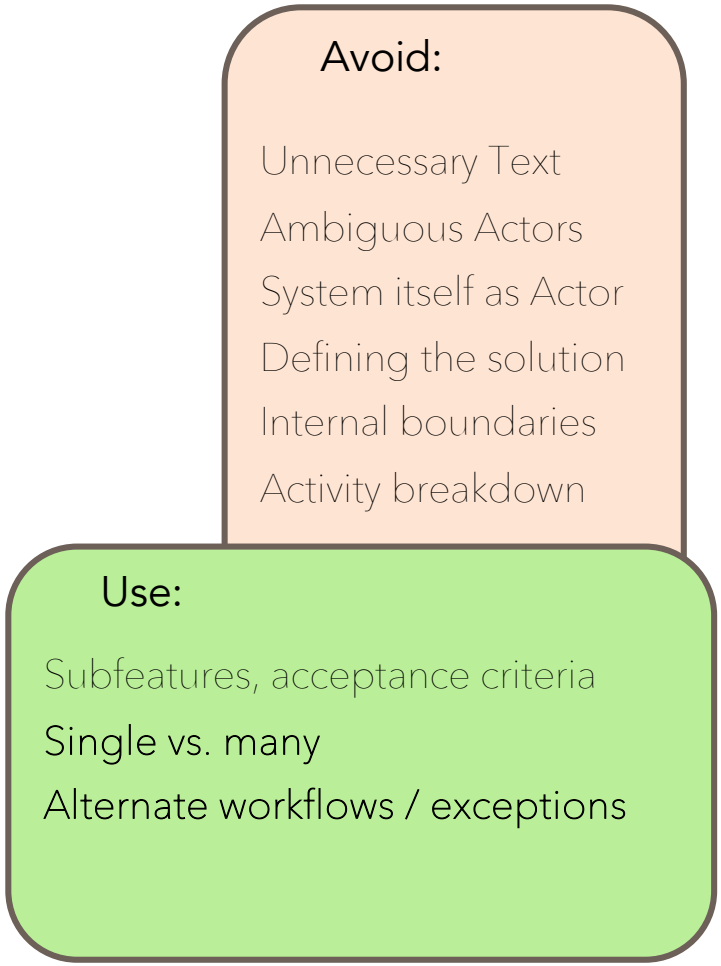
# For example,

Member can login with Facebook, …
(+ stories for the others)

Member can see a longer error notification when the Facebook login fails, …

Admin can manage allowed login services, …

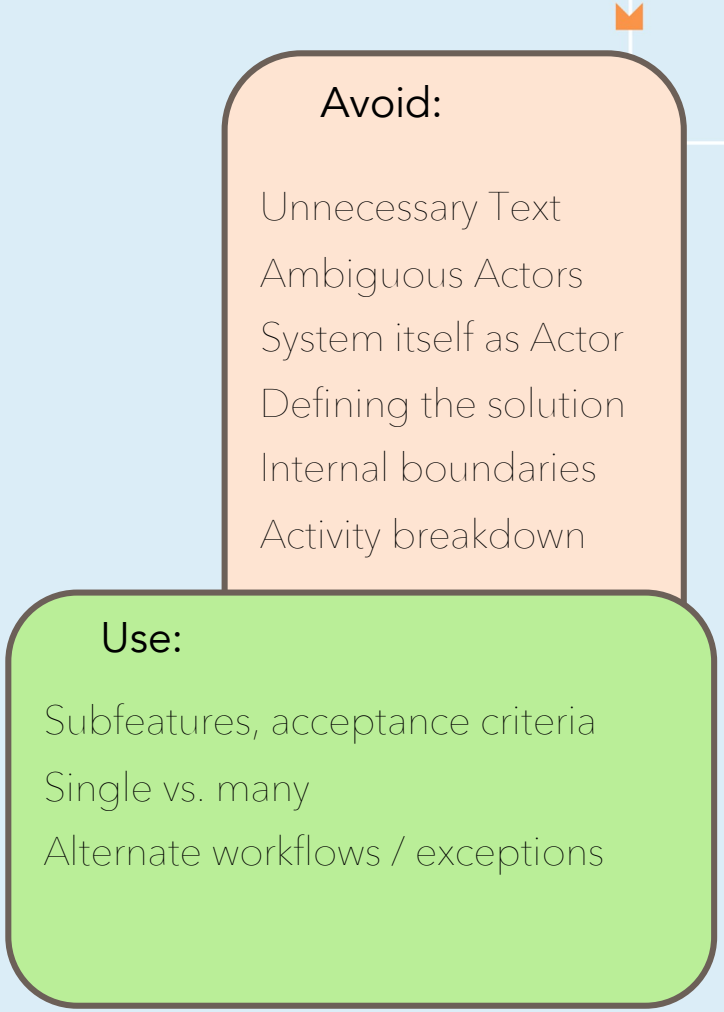
Admin can force re-logins with social media accounts, …



# Member can login with Facebook, so that...

This might already be small enough for development in one Sprint. If we still wanted to make it smaller, how could we possibly do it? Do NOT consider smaller subfeatures, but use some other approach.

Pick a pair and discuss. 2 minutes.

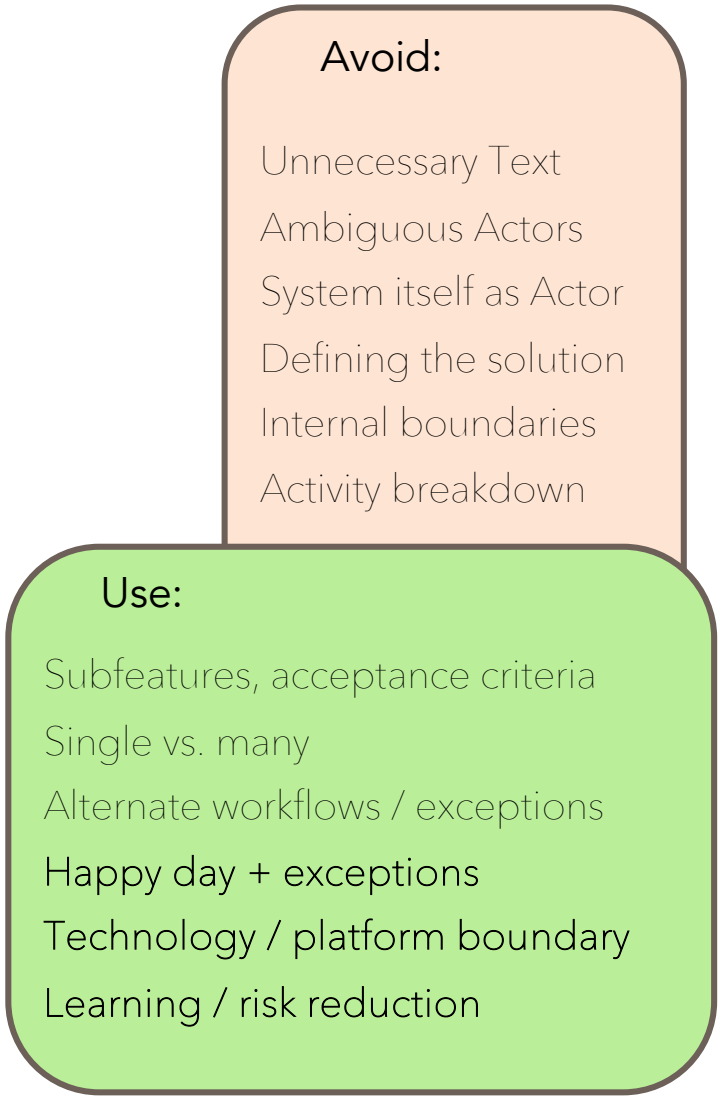


# For example,

Member can login with Facebook in happy day scenario, …
(+ stories for exception scenarios)

Member can login with Facebook on a mobile phone, …

Spike: test how to use Facebook API for login



# And more...

Admin can run a maintenance script /

Admin can schedule script runs

Member can login without password restriction

Member can login with ugly UI

Member can login with a system component

Member in Serbia can login

Member can request GDPR info delivered overnight

Use:

Subfeatures, acceptance criteria

Single vs. many

Alternate workflows / exceptions

Happy day + exceptions

Technology / platform boundary

Learning / risk reduction

Manual vs. automated

Simplified algorithm first

Make it work first, then pretty

Generic tech vs. custom

Client boundary

Batch processing vs. online

## Use (not a full list):

- Subfeatures, acceptance criteria
- Single vs. many
- Alternate workflows / exceptions
- Happy day + exceptions
- Technology / platform boundary
- Learning / risk reduction
- Manual vs. automated
- Simplified algorithm first
- Make it work first, then pretty
- Generic tech vs. custom
- Client boundary
- Batch processing vs. online

## Avoid:

- Unnecessary Text
- Ambiguous Actors
- System itself as Actor
- Defining the solution
- Internal boundaries
- Activity breakdown

# Thank you!

Petri.heiramo@gmail.com
Twitter: @pheiramo
LinkedIn: https://www.linkedin.com/in/pheiramo/

ScrumAlliance®