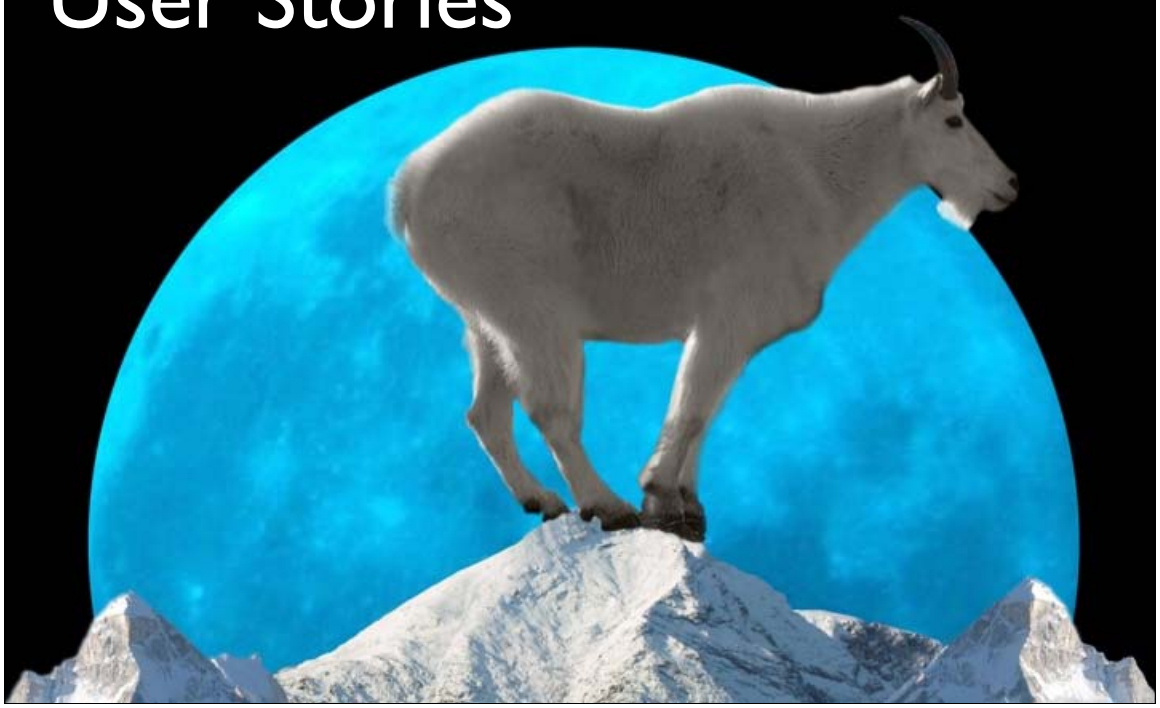
# An Introduction to User Stories



1

# Mike Cohn - background


## Agile coach and trainer

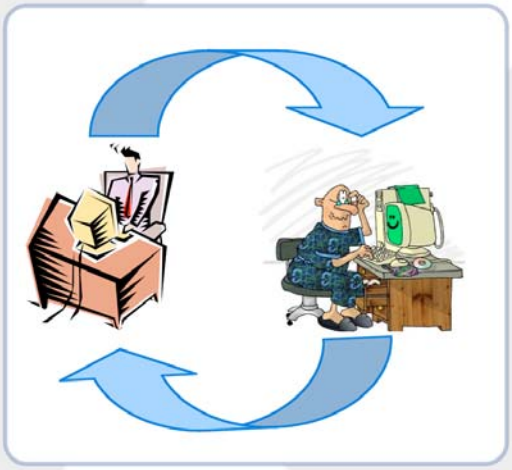
- Founding member and director of Agile Alliance and Scrum Alliance
- Founder of Mountain Goat Software
- Ran my first Scrum project back in 1995
- Typical programmer to manager etc. progression



2

# What problem do stories address?

- Software requirements is a communication problem
- Those who want the software must communicate with those who will build it







# Balance is critical

- If either side dominates, the business loses
- If the business side dominates…
  - …functionality and dates are mandated with little regard for reality or whether the developers understand the requirements
- If the developers dominate…
  - …technical jargon replaces the language of the business and developers lose the opportunity to learn from listening

# Resource allocation

- 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





# Responsibility for resource allocation

**If developers shoulder the responsibility...**

- May trade quality for additional features
- May only partially implement a feature
- May solely make decisions that should involve the business side

**If the business shoulders the responsibility...**

- Lengthy upfront requirements negotiation and signoff
- Features are progressively dropped as the deadline nears

# Imperfect schedules

- 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





# So what do we do?







# Ron Jeffries' Three Cs

**Card**
- Stories are traditionally written on note cards.
- Cards may be annotated with estimates, notes, etc.

**Conversation**
- Details behind the story come out during conversations with product owner

**Confirmation**
- Acceptance tests confirm the story was coded correctly

Source: XP Magazine 8/30/01, Ron Jeffries.

# Samples from a travel website

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

As a user, I want to cancel a reservation.

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

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





# 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 travelers cancel later?
  - Is a confirmation provided to the user?
    - How?

# Details as conditions of satisfaction

- The product owner's conditions of satisfaction 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.





# Details added in smaller sub-stories

As a user, I can cancel a reservation.

As a premium site member, I can cancel a reservation up to the last minute

As a non-premium member, I can cancel up to 24 hours in advance.

As a site visitor, I am emailed a confirmation of any cancelled

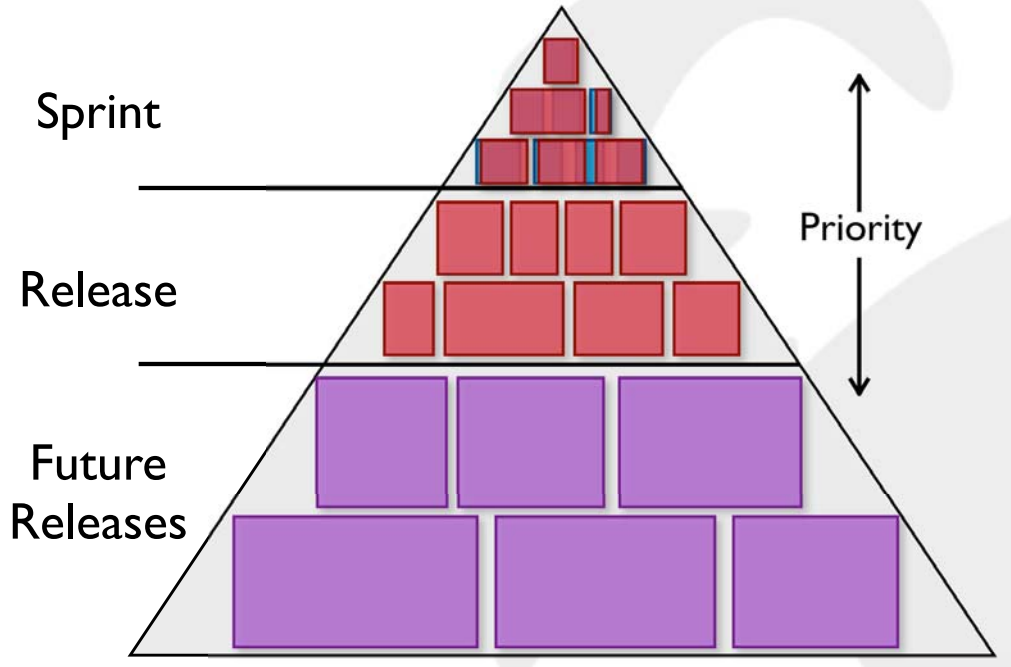
# Techniques can be combined

- These approaches are not mutually exclusive
- Write stories at an appropriate level
- By the time it's implemented, each story will have conditions of satisfaction associated with it





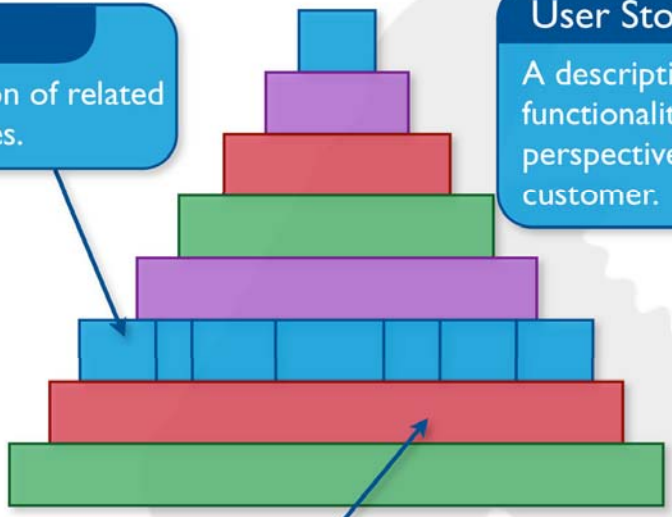
# The product backlog iceberg




# Stories, themes and epics

**Theme**
A collection of related user stories.

**User Story**
A description of desired functionality told from the perspective of the user or customer.



**Epic**
A large user story.





# An example

As a VP Marketing, I want to review the performance of historical promotional campaigns so that I can identify and repeat profitable ones.

Clearly an epic

As a VP Marketing, I want to select the timeframe to use when reviewing the performance of past promotional campaigns, so that I can identify and repeat profitable ones.

Epics??

As a VP Marketing, I can select which type of campaigns (direct mail, TV, email, radio, etc.) to include when reviewing the performance of historical promotional campaigns.

# An example

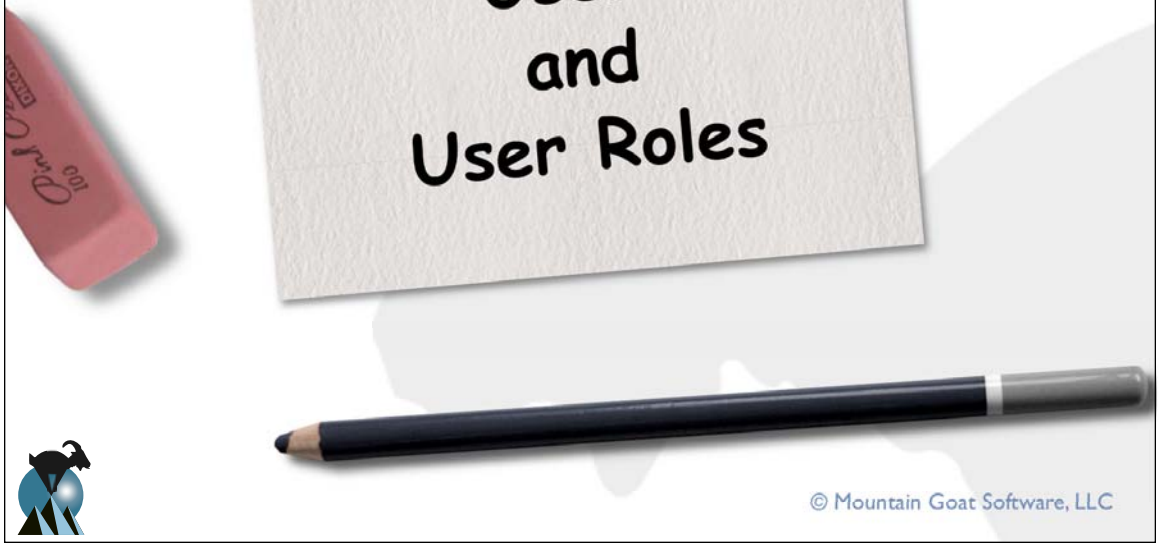
As a VP Marketing, I want to see information on direct mailings when reviewing historical campaigns.

As a VP Marketing, I want to see information on television advertising when reviewing historical campaigns.

As a VP Marketing, I want to see information on email advertising when reviewing historical campaigns.







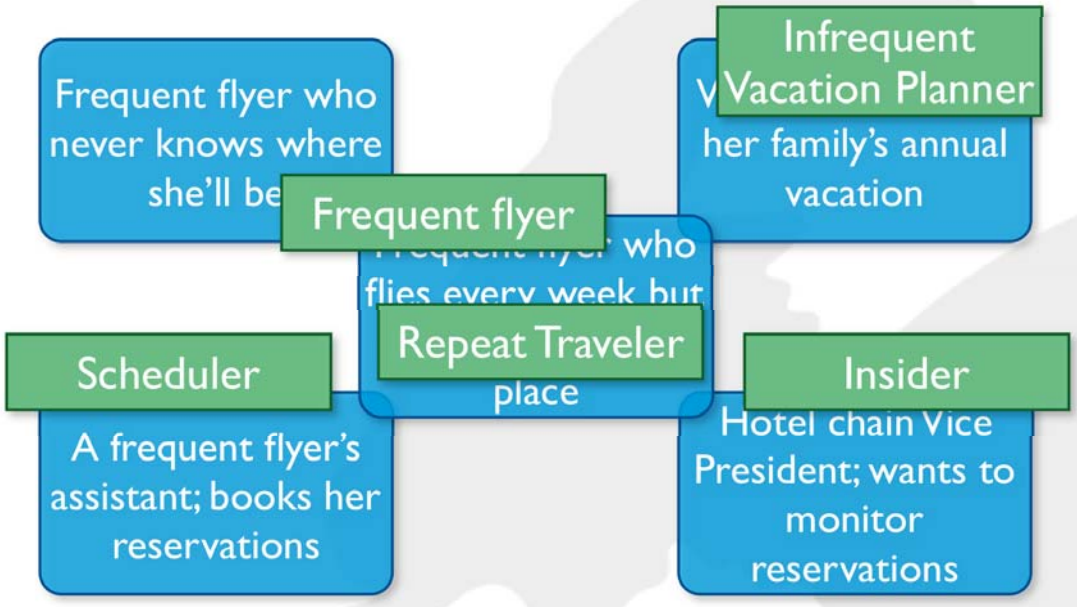
# "The User"

- 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





# Common attributes




# 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 usage-centered design

Source: *Software for Use* by Constantine and Lockwood (1999).





# System and programmer users

As the payment verification system, I want all transactions to be well-formed XML.

As a programmer, I want an API for deleting widgets from the database.

# Advantages of using roles

| | |
|---|---|
| Users become tangible | Start thinking of software as solving needs of real people. |
| Avoid saying "the user" | Instead we talk about "a frequent flyer" or "a repeat traveler" |
| Incorporate roles into stories | "As a <user role>, I want to <goal> so that <benefit>." |





# A horrible question

"Would you like it in a browser?"

"Of course, now that you mention it!"

- A problem:
  - The question is closed
    - {Yes | No}





# We can do better

"What would you think of having this app in a browser rather than as a native Windows application, even if it means reduced performance, a poorer overall user experience, and less interactivity?"

- It's open
  - Full range of answers
- But it has too much context

# A better way to ask

"What would you be willing to give up in order to have it in a browser?"

- We want to ask questions that are
  - Open-ended
  - Context-free





# My context isn't your context


| | |
|---|---|
| You hear | "Increase the temperature." |
| He meant | "Move the temperature closer to what we call warm." |

# It's my problem, I know the solution

- Having a problem does not uniquely qualify you to solve it
- "It hurts when I go like this…"





# We need to stop asking users

- Since users don't know how to solve their problems, we need to stop *asking*
- We need to *involve* them instead

| | |
|---|---|
| Empirical design | • Designers of the new system make decisions by studying prospective users in typical situations |
| Participatory design | • The users of the system become part of the team designing the behavior of the system |

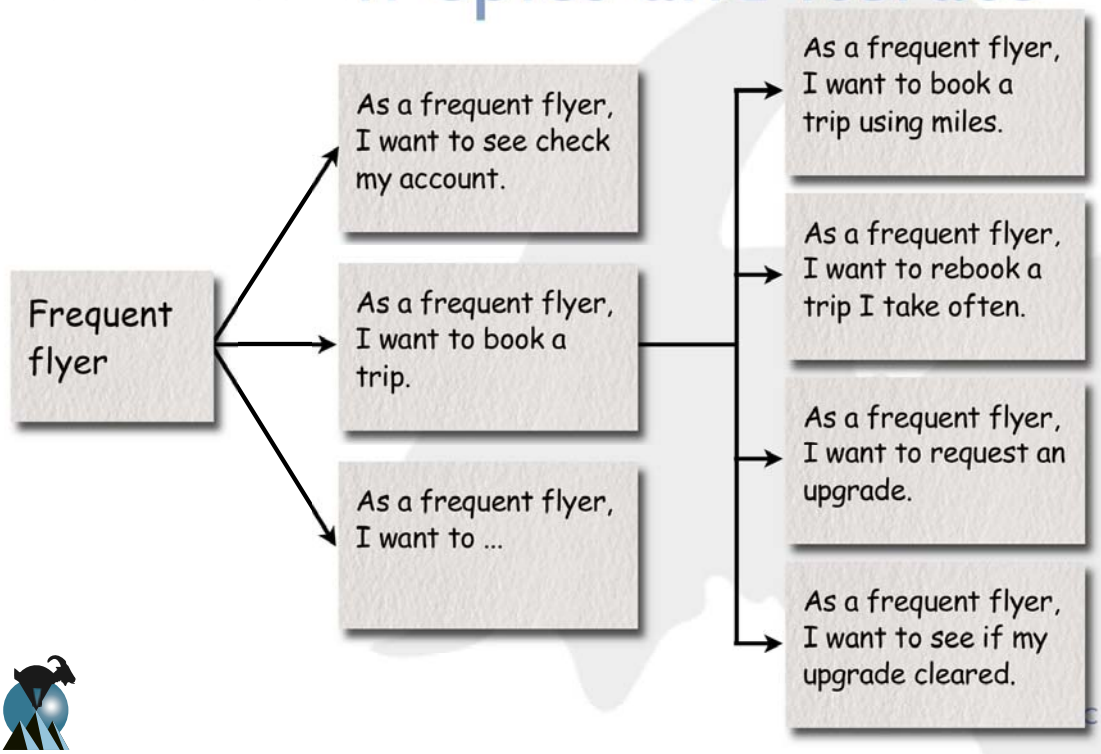
# Story-writing workshops

- Includes developers, users, customer, others
- Brainstorm to generate stories
- Goal is to write as many stories as possible
  - Some will be "implementation ready"
  - Others will be "epics"
- No prioritization at this point





# Start with epics and iterate









# What makes a good story?

- INVEST
  - Independent
  - Negotiable
  - Valuable
  - Estimatable
  - Sized appropriately
  - Testable

Thanks to Bill Wake for the acronym. See www.xp123.com.

# INVESTing in good stories

- Independent
  - Dependenices lead to problems estimating and prioritizing
  - Can ideally select a story to work on without pulling in 18 other stories
- Negotiable
  - Stories are not contracts
  - Leave or imply some flexibility
- Valuable
  - To users or customers, not developers
  - Rewrite (most) developer stories to reflect value to users or customers





# INVESTing in good stories

- Estimatable
  - Because plans are based on user stories, we need to be able to estimate them
- Sized Appropriately
  - Complex stories are intrinsically large
  - Compound stories are multiple stories in one
- Testable
  - Stories need to be testable


39

1. Stories shift the focus from writing to talking.

If requirements are written down → then

~~The user will get what she wants~~

At best she'll get what was written

"You built what I asked for, but it's not what I need."



40

# Words are imprecise

Main dish comes with soup or salad and bread.

- (Soup or Salad) and Bread
- (Soup) or (Salad and Bread)





# Examples

The user can enter a name. It can be 127 characters.

- Must the user enter a name?
- Can it be other than 127 chars?

The system should prominently display a warning message whenever the user enters invalid data.

- What does should mean?
- What does prominently display mean?
- Is invalid data defined elsewhere?

2. Stories are equally understandable by developers and customers.
3. Stories support and encourage iterative development.
4. Stories are the right size for planning.
5. Stories support participatory design.





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

**What are we building?**

1. The product shall have a gas engine.
2. The product shall have four wheels.
   2.1. The product shall have a rubber tire mounted to each wheel.
3. The product shall have a steering wheel.
4. The product shall have a steel body.

Source: Adapted from *The Inmates are Running the Asylum* by Alan Cooper (1999).

# What if we had stories instead?

As a user, I want to mow my lawn quickly and easily.

As a user, I want to be comfortable while mowing my lawn.





# The product

# Most importantly...

**Don't forget the purpose**

The story text we write on cards is less important than the conversations we have.





# Mike Cohn contact info


mike@mountaingoatsoftware.com
www.mountaingoatsoftware.com
(720) 890-6110 (office)
(303) 810-2190 (mobile)

MOUNTAIN GOAT SOFTWARE