Effective User Stories for Agile Requirements

Mike Cohn
August 14, 2007

1

Mike Cohn - background

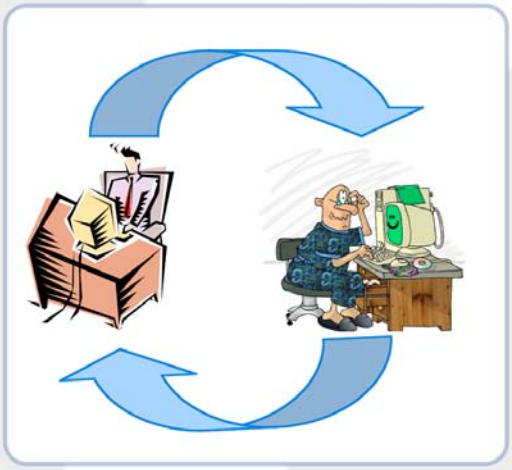
Agile coach and trainer

- 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

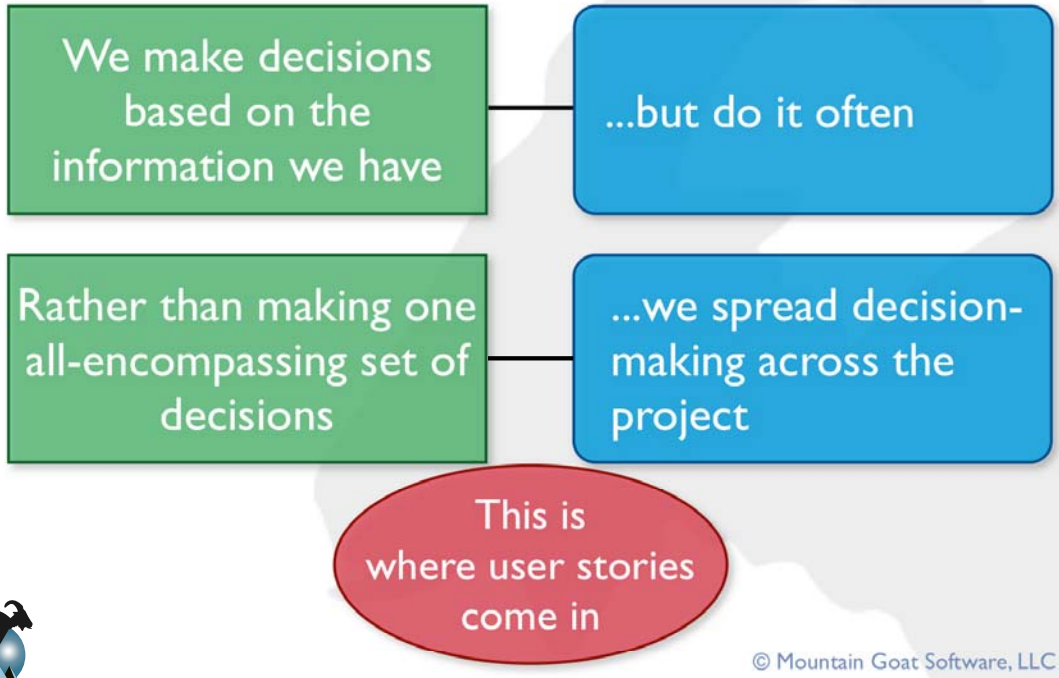
If the business shoulders the responsibility...

- 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?



Today's agenda

- What stories are
- Users and user roles
- Gathering stories
- INVEST in good stories
- Why user stories

Poor requirements

Poor requirements are often listed as one of the chief causes of project failure.

1. What are some problems you can attribute to a poor requirements process?
2. What constitutes a poor requirements process?

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

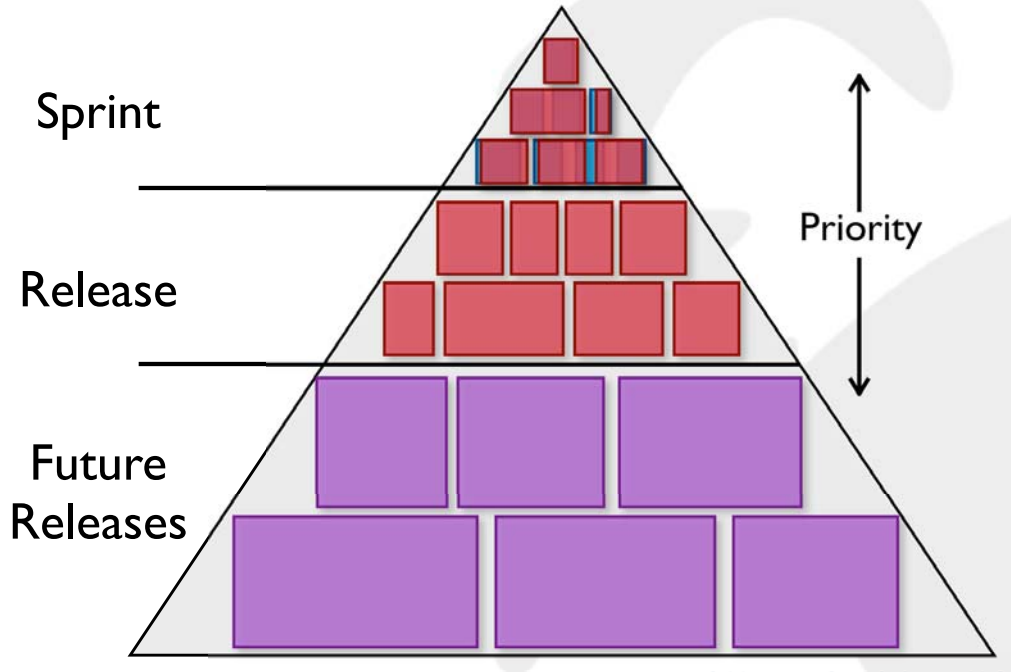
As a non-premium member, I can cancel up to 24 hours in advance.

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

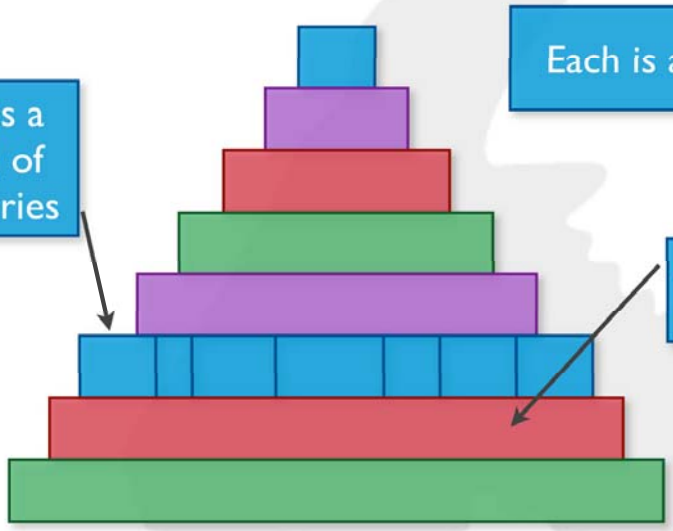
Techniques can be combined

- 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



User stories on the product backlog



A theme is a collection of related stories

Each is a story

An epic is a large 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.

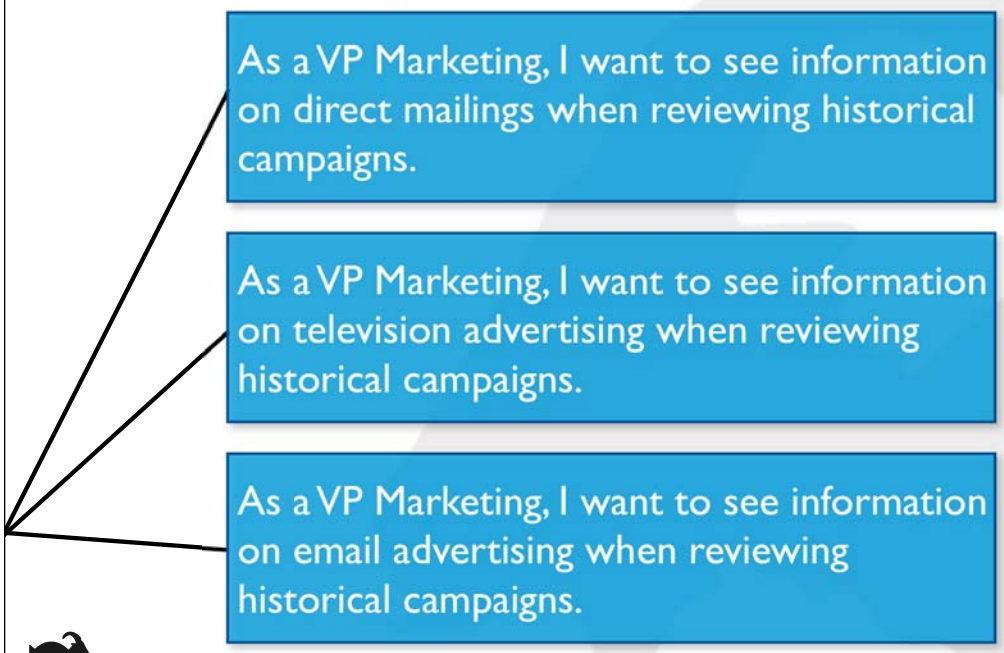
An epic;
weeks to implement

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.

Implementation-size stories;
days to implement

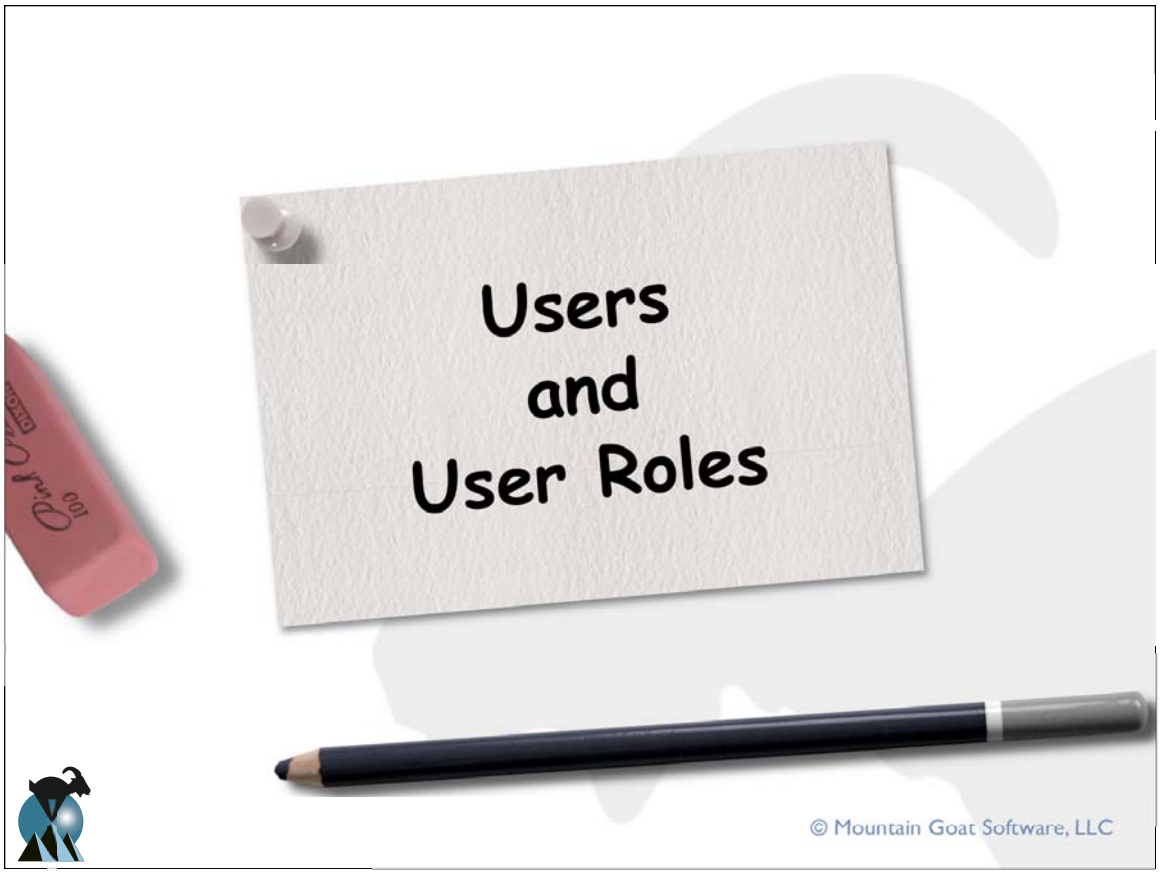
As a VP Marketing, I can select which type of campaigns (direct mail, TV, email, radio, etc.) to include when reviewing the performance of historical promotional campaigns.

An example



Augment as necessary

- User stories don't have to be the end-all, be-all of requirements
- Augment them with written documentation as appropriate
  - Business rules
  - Data dictionaries
  - Use cases
  - Examples of inputs and expected result



"The User"

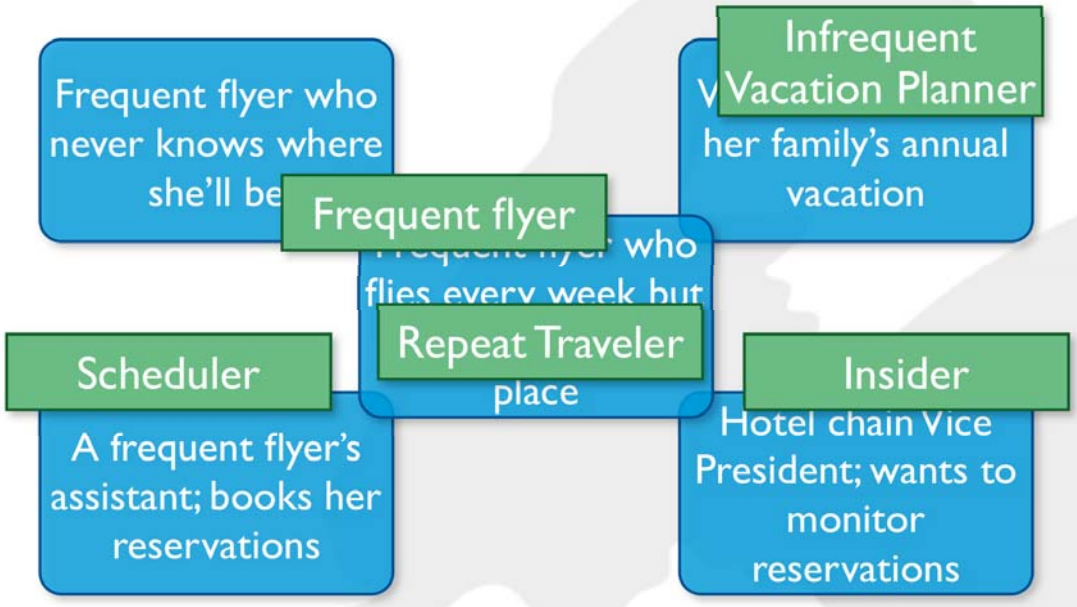
- Many projects mistakenly assume there's only one user:
  - "The user"
- Write all stories from one user's perspective
- Assume all users have the same goals
- Leads to missing stories

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 usage-centered design

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

Common attributes



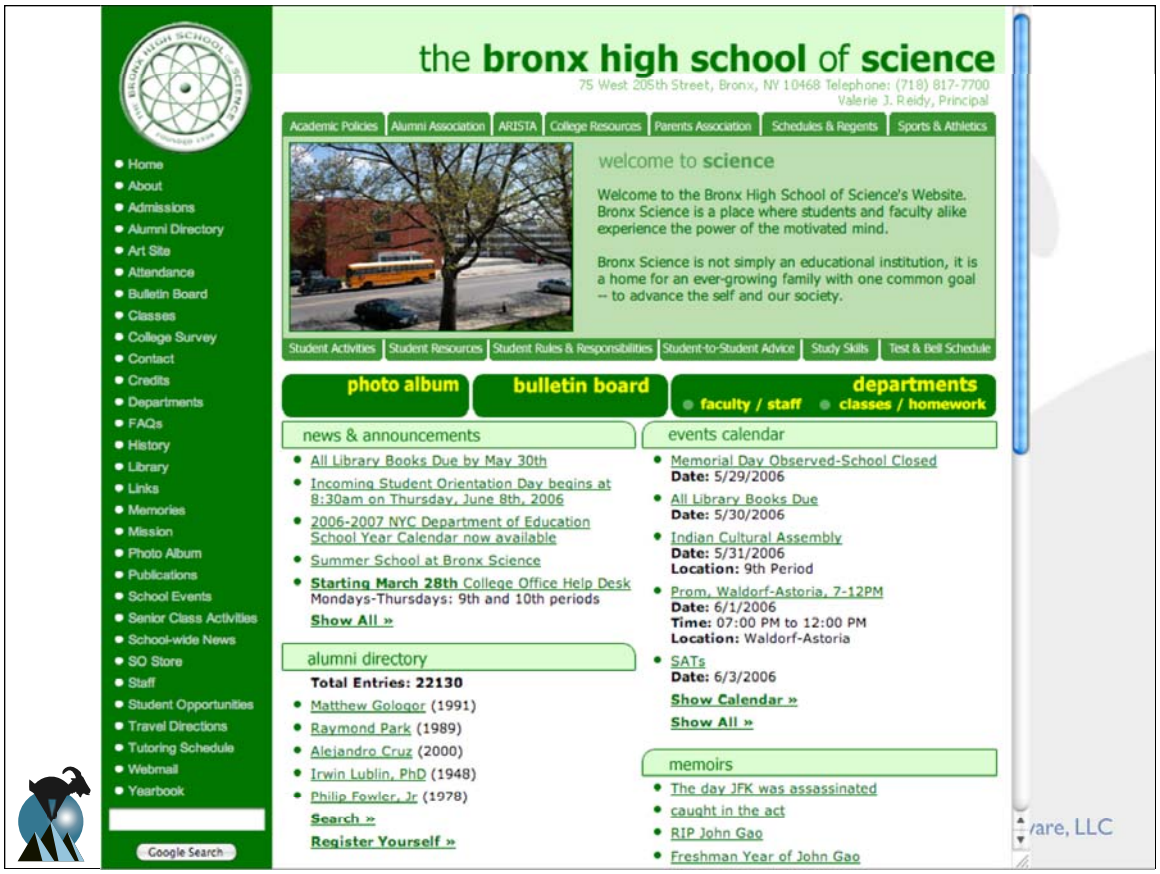
User role brainstorming

- Brainstorming meeting
  - Customer, developers, anyone who understands a product's intended users
- Everyone grabs a stack of cards
- Write role names on cards
  - As fast as possible and with no judgment
    - No turns
  - Place card on table
  - Call out role name as you place it

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.

1. Brainstorm the user roles who will interact with this site.



29

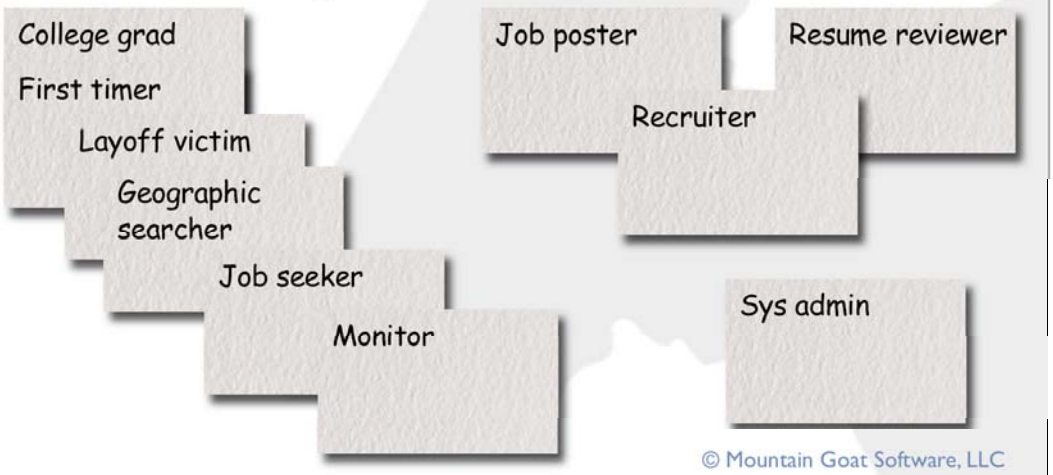
User role modeling steps

- [x] Brainstorm an initial set of user roles
- [ ] Organize the initial set
- [ ] Consolidate roles
- [ ] Refine roles

30

Organize the initial set

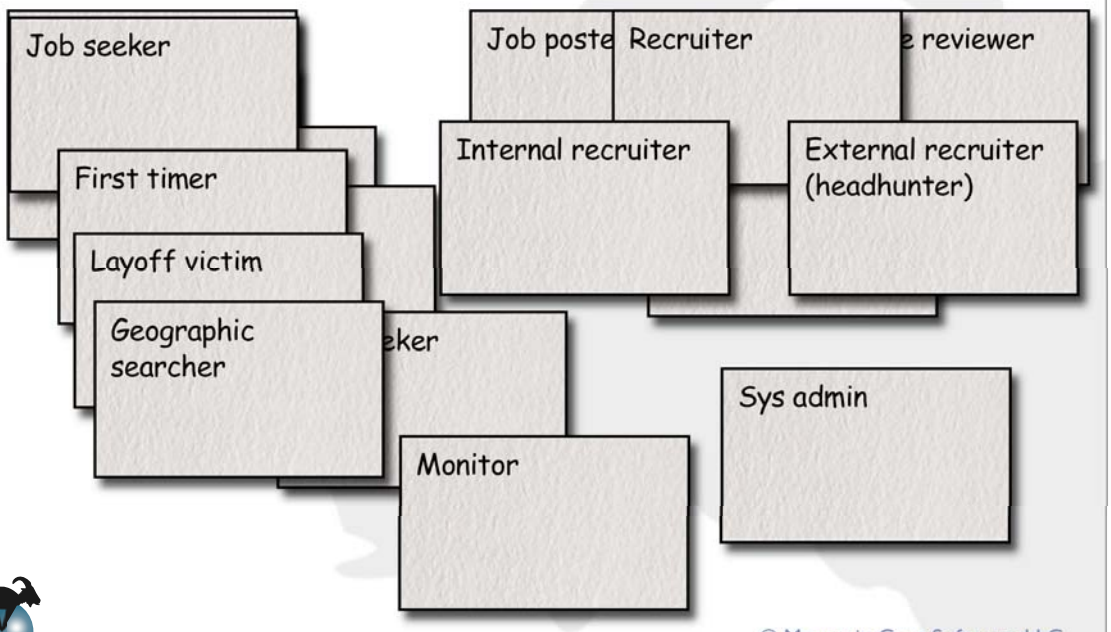
- Arrange cards spatially to indicate overlapping and similar roles
  - Use any arrangement rules you want



Consolidate roles

- Discuss what is meant by each card
- Arrange cards spatially to indicate overlapping and similar roles
  - Use any arrangement rules you want
- Look for cards to
  - Combine
  - Replace with a more generic/different card
- Eliminate cards that are unimportant to the success of the product

Consolidating–an example



Organize and consolidate

1. Organize your initial set of user roles.
2. Consolidate the user roles.

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>."

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.

Three variations

Abusers

- Think of who might abuse the system
- Write stories to prevent these abuses

Extreme characters

- Think of someone unlikely to use your product
- What might they want?

Personas

- Make a role real with a name, photo and so on

37

38

Techniques for gathering stories

- Questionnaires
- Observation
- User Interviews
- Story-writing workshops

Questionnaires

- Good technique for learning more about stories you already have
- If you have a large user base, great way to get information to help prioritize stories
- Not effective as a primary means of trawling for new stories

Observation

- Great way to pick up insights
- Two approaches
  - Just observe, with or without user's knowledge
  - Have the user demonstrate to a group how she uses the software

Observation example

- Stated need:
  - "We need a large text field to summarize."
- Observed need:
  - Have the system record the user's choices

Interviews

- Default approach taken by many teams
- Selection of interviewees is critical
  - Try to interview as many user roles as possible
- Cannot just ask "So whaddaya want?"
  - Most users are not adept at understanding their true needs

My context isn't your context

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

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

The best 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

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

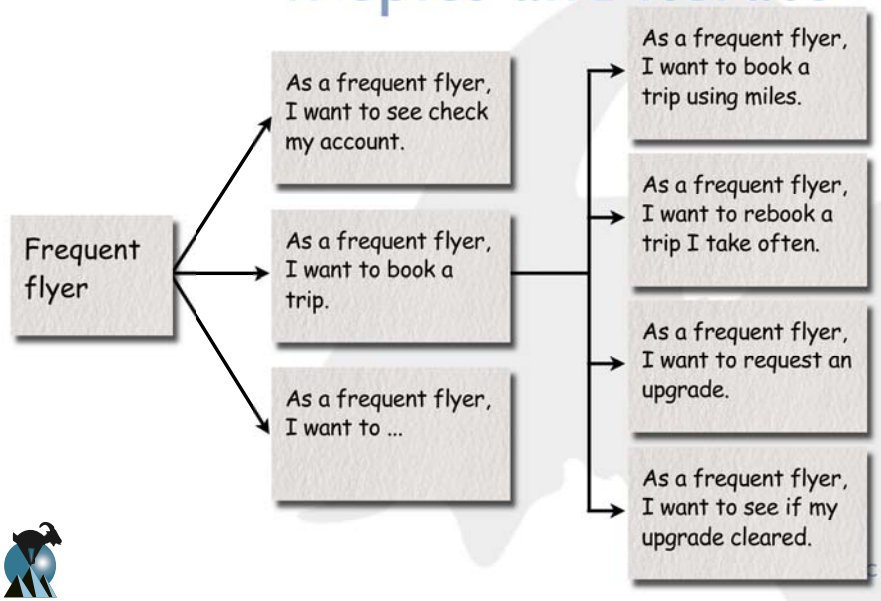
- Since users don't know how to solve their problems, we need to stop *asking*
- We need to *involve* them instead

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



A story-writing workshop

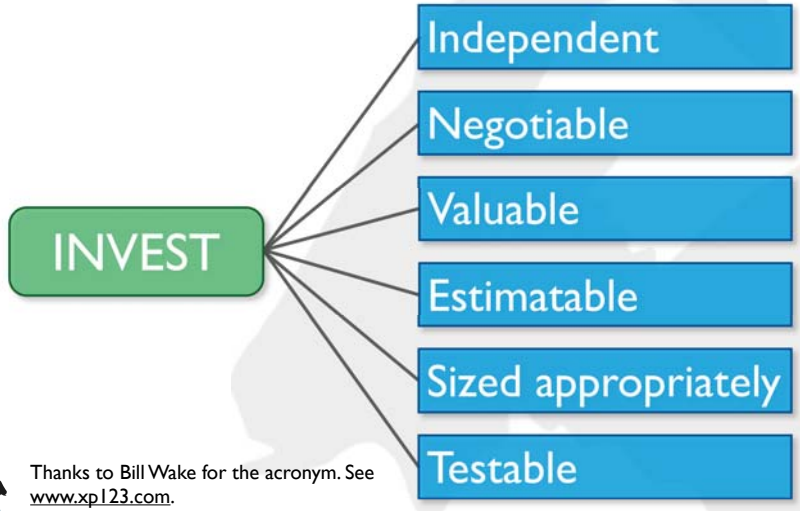
Start with the roles you've identified. For two or three, think of their top-level goals and write some epics. Then convert a couple of epics into more usable stories.

A tip:

Try this template:

"As a <user role>, I want <goal> so that <reason>."

What makes a good story?



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

Independent

- Avoid introducing dependencies
  - Leads to difficulty prioritizing and planning

As a customer, I can pay for the items in my cart with a Visa card.

As a customer, I can pay for the items in my cart with a MasterCard.

As a customer, I can pay for the items in my cart with an American Express card.

- First story will take 3 days to develop
  - It doesn't matter which is first
- Others will each take 1 day

Making stories independent

Combine the stories	• As a customer, I can pay with a credit card.
Split across a different dimension	• As a customer, I can pay with a first type of credit card. • As a customer, I can pay two additional types of credit card.
Write two estimates and move on	• 3 days if done first; 1 otherwise

What about this approach?

Extract technical commonalities

- As a programmer, I need to code the infrastructure for processing credit cards.
- As a customer, I can pay with a Visa.
- As a customer, I can pay with a MasterCard,
- As a customer, I can pay with an American Express.

- Sometimes necessary but not ideal
- Why?

Another example

As a user, I can search for a hotel on fields such as hotel brand, quality rating, availability on specific dates, proximity to an attraction (airport, amusement park, etc.), and more.

As a user, I can do an advanced search for a hotel on 2-3 of these fields.
Possible fields: hotel brand, quality rating, availability on specific dates, proximity to an attraction, etc.

As a user, I can search for a hotel on additional fields.
Possible fields: hotel brand, quality rating, availability on specific dates, proximity to an attraction, etc.

Negotiable

- Stories are not contracts
- Do not need to include all details
  - Too many details give the impressions of
    - false precision or completeness
    - that there's no need to talk further
- Need some flexibility so that we can adjust how much of the story gets implemented
  - If the card is contract then it needs to be estimated like a contract
- Not all stories need to be negotiable, but some do

Which is more negotiable?

1. Print dialog allows the user to edit the printer list. The user can add or remove printers from the printer list. The user can add printers either by auto-search or manually specifying the printer DNS name or IP address. An advanced search option also allows the user to restrict his search within specified IP addresses and subnet range.

2

As a user, I can add printers to the printer list.

- ☐ Auto-search
- ☐ Manually specify DNS name
- ☐ Manually specify IP address

Note: I've got some "advanced" ways to add printers, too. See me if you have time.

Valuable

- Stories must be valuable to either:

Users

- As a user, I can search for a job by title and salary range.

Customers

- As the sponsor of this project, I want it to pass an ISO 9001 audit.
- As a sponsor of this project, I want it to produce documentation in compliance with CMMI level 3.
- As a system administrator, I want all configuration information for all users stored in a central location.

Stories valued by developers

- Should be rewritten to show the benefit

All connections to the database are through a connection pool.	As a purchaser of this system, I want it usable by 50 users with a five-user database license.
All error handling and logging is done through a set of common classes.	As a user, I want all errors presented and logged in a consistent manner.

Rewrite this as a story:

Refactor the payroll processing code.

As a
I want

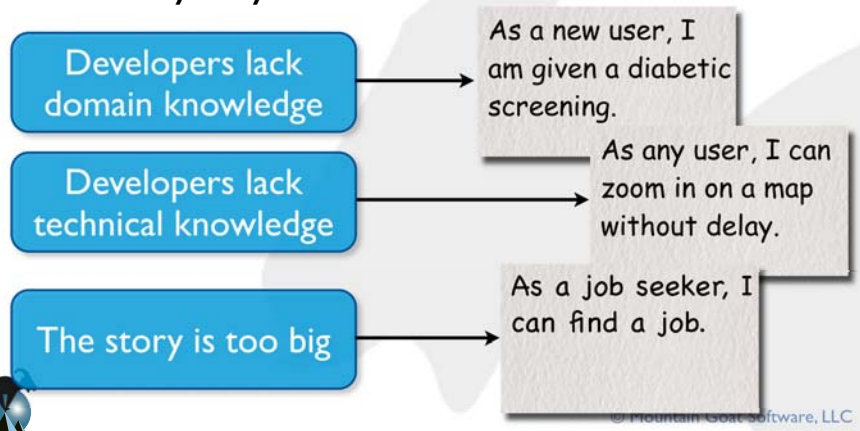
so that

Refactor

To change the structure but not the behavior of code.

Estimatable

- Because stories are used in planning
- A story may not be estimatable if



Sized appropriately

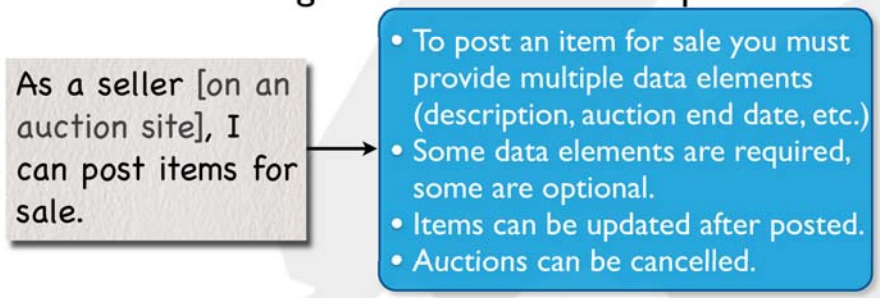
- Small stories for the near future
- Epics for further out
- Large stories (epics) are
  - Hard to estimate
  - Hard to plan
    - Won't fit in a single iteration
- Two types of large story
  - Complex story
  - Compound story

Complex stories

- A story that is inherently large and cannot easily be disaggregated into constituent stories
- Very rare
- Some stories look complex because we don't know enough
  - Use a *spike* in those situations
    - First iteration: acquire knowledge
    - Second iteration: do the work

Compound stories

- An epic that comprises multiple shorter stories
- Often hide a great number of assumptions



Splitting a compound story

Split along operational boundaries (CRUD)

- As a seller, I can create a new auction listing.
- As a seller, I can update an existing auction listing.
- As a seller, I can delete an auction listing.

Splitting a compound story

Split along data boundaries

- As a seller, I can create and update the description of an auction item.
- As a seller, I can add, update or remove a photo from an auction listing.

More advice on splitting stories

Remove cross-cutting concerns

Don't meet performance targets

Avoid splitting stories into tasks before iteration planning

Testable

- Tests demonstrate that a story meets the customer's expectations
- Automate, automate, automate

A user must find the software easy to use.	As a novice user, I am able to complete common workflows without training.
A user must never have to wait long for a screen to appear.	As a user, I want to see new screens within 2 seconds 95% of the time.

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."

Words are imprecise

Entrée 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?

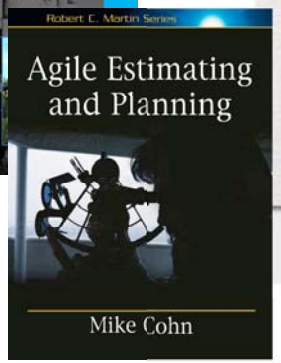
2. Stories are equally understandable by developers and customers.
3. Stories support and encourage iterative development.
4. Stories are the right size for planning.
5. Stories support participatory design.

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.

Mike Cohn contact info



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

MOUNTAIN GOAT SOFTWARE