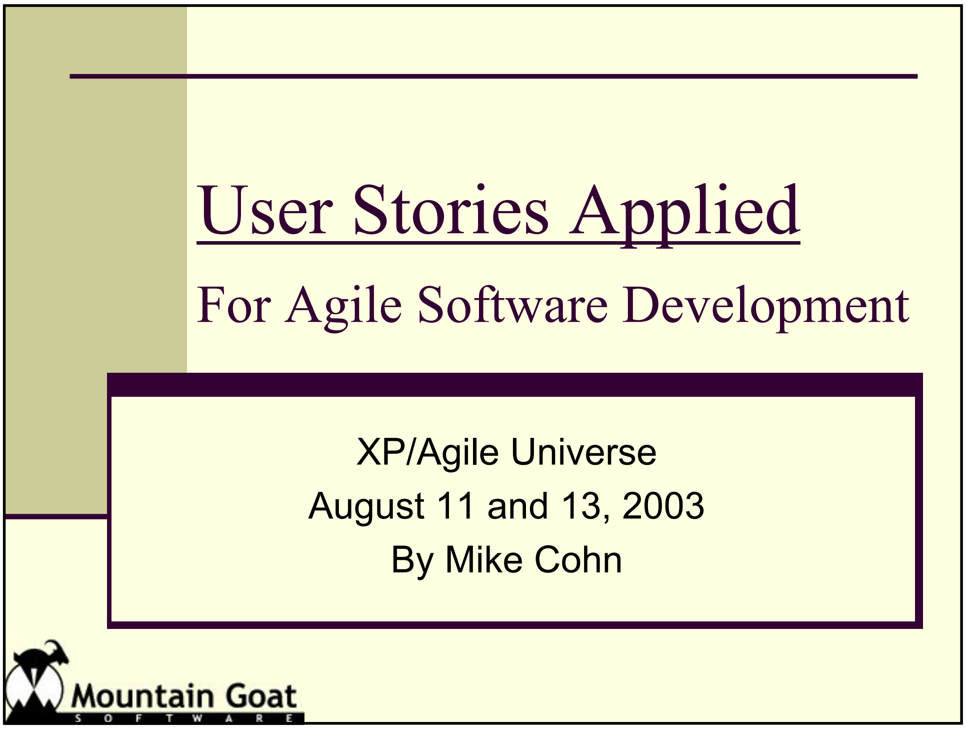
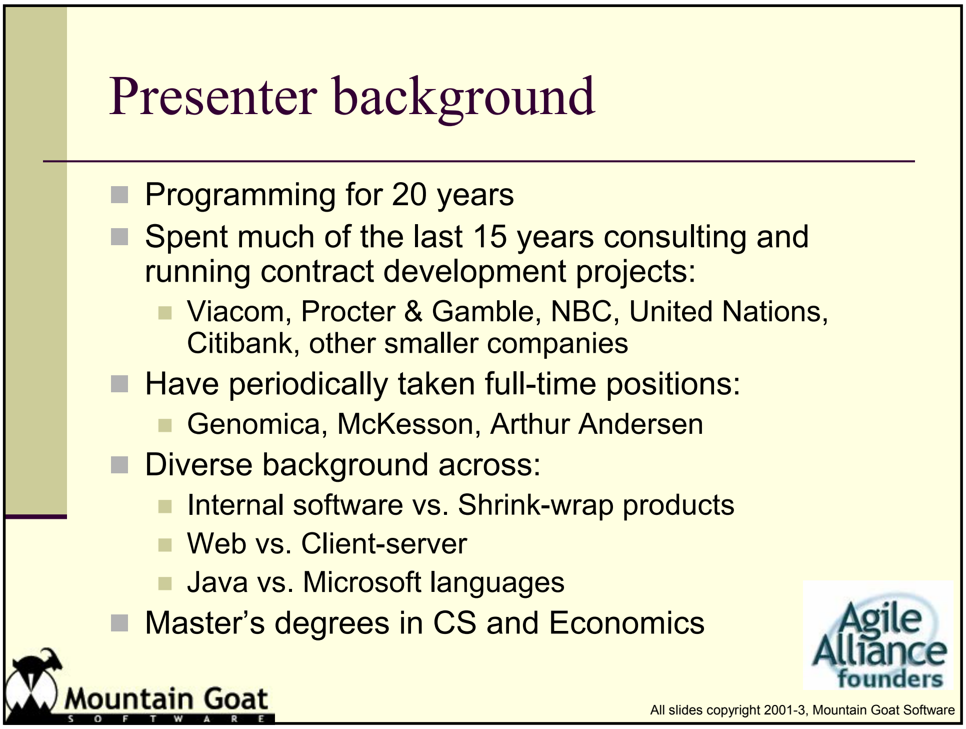


User Stories Applied

For Agile Software Development

XP/Agile Universe
August 11 and 13, 2003
By Mike Cohn

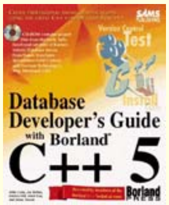


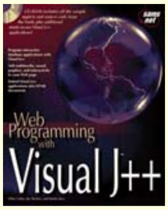
Presenter background

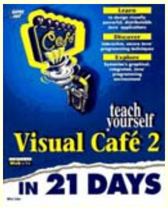
- Programming for 20 years
- Spent much of the last 15 years consulting and running contract development projects:
  - Viacom, Procter & Gamble, NBC, United Nations, Citibank, other smaller companies
- Have periodically taken full-time positions:
  - Genomica, McKesson, Arthur Andersen
- Diverse background across:
  - Internal software vs. Shrink-wrap products
  - Web vs. Client-server
  - Java vs. Microsoft languages
- Master's degrees in CS and Economics

Background, cont.

- Been managing projects since 1987 but remain a programmer at heart
- Books on Java, C++, and database programming.
- Articles in *IEEE Computer, STQE, C++ User's Journal*, etc.
- *User Stories Applied* (Addison-Wesley, XP series) out in early 2004

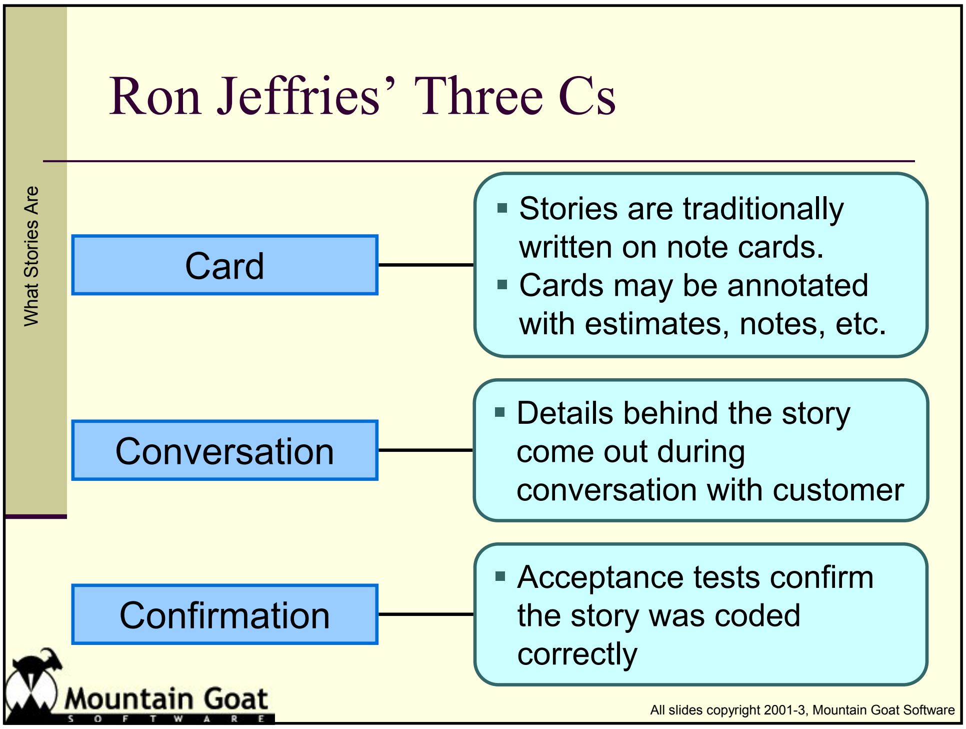






Today's agenda

- Introduction
  - What stories are
  - What stories are not
  - Why stories?
- The User and Customer
  - Who's the user?
  - User roles and personas
- Gathering Stories
- Planning and Estimating
  - Why plans go wrong
  - Estimating user stories
  - Planning with user stories
- Case Study



What Stories Are

Ron Jeffries' Three Cs

Card

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

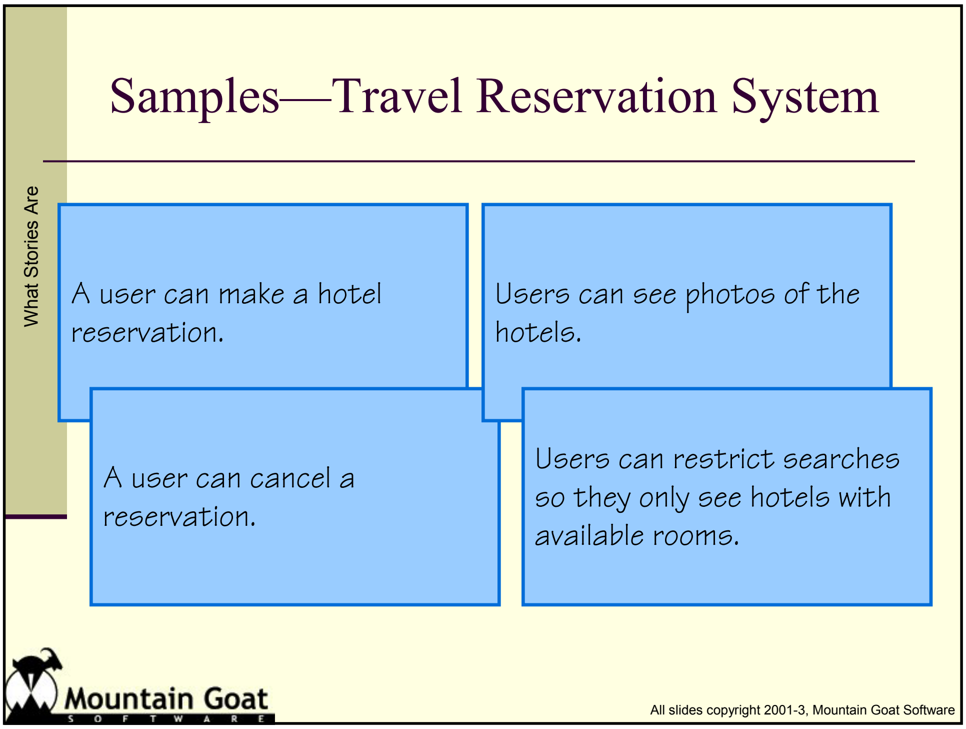
Conversation

- Details behind the story come out during conversation with customer

Confirmation

- Acceptance tests confirm the story was coded correctly

Mountain Goat SOFTWARE



What Stories Are

Samples—Travel Reservation System

A user can make a hotel reservation.

Users can see photos of the hotels.

A user can cancel a reservation.

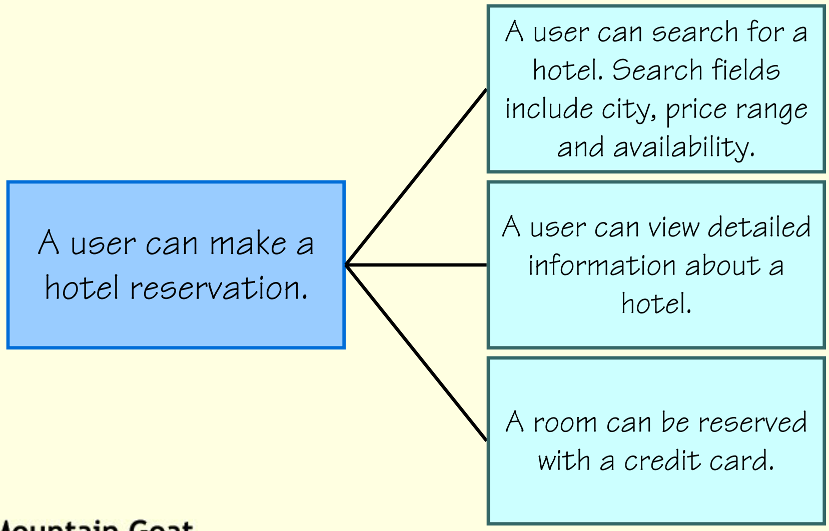
Users can restrict searches so they only see hotels with available rooms.

Mountain Goat SOFTWARE

Where are the details?

- A user can make a hotel reservation.
  - Does she have to enter a credit card?
    - If so, what cards are accepted?
    - Is the charge applied immediately?
  - How can the user search for the hotel?
    - Can she search by city?
    - By quality rating?
    - By price range?
    - By type of room?
  - What information is shown for each room?
  - Can users make special requests, such as for a crib?

Details added with more, smaller stories



A user can make a hotel reservation.

A user can search for a hotel. Search fields include city, price range and availability.

A user can view detailed information about a hotel.

A room can be reserved with a credit card.

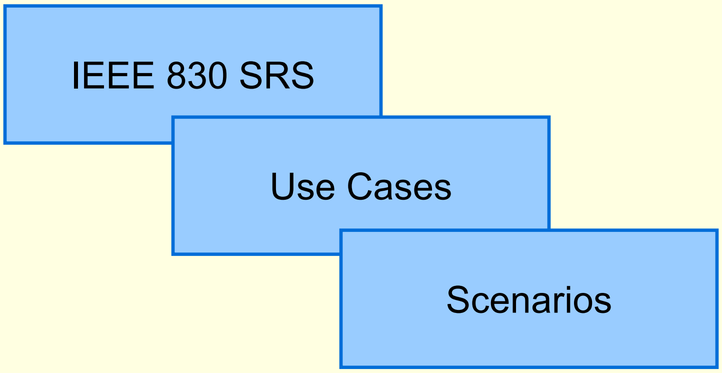
Understanding customer expectations

- "How long does it have to be?"
- Capture expectations as acceptance tests

A user can make a hotel reservation.

- Try it with a valid Visa then a valid MasterCard.
- Enter card numbers that are missing a digit, have an extra digit and have two transposed digits.
- Try it with a card with a valid number but that has been cancelled.
- Try it with a card expiration date in the past.

What stories are not



IEEE 830 SRS

Use Cases

Scenarios

What Stories Are Not

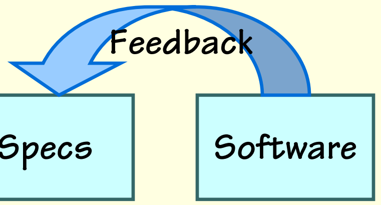
IEEE 830

4. The system shall allow a room to be reserved with a credit card.
    1. The system shall accept Visa, MasterCard and American Express cards.
    2. The system shall charge the credit card the indicated rate for all nights of the stay before the reservation is confirmed.
5. The system shall give the user a unique confirmation number

What Stories Are Not

Problems with IEEE 830

- Time-consuming to write and read
- Tedious to read
    - So readers skim or skip sections
- Assumes everything is knowable in advance



- Are these changes really a “change of scope”?

What Stories Are Not

All requirements are not equal

- Humans want to feel stable
  - Fluidity undermines the stability we feel
- We try to counter the fluidity as early as possible
- “Designers fix a top-level concept based on their initial understanding of a problem.”
  - If they’re right → “Inspiration”
  - If wrong → Design is painted into a corner
- Decomposition reduces overall uncertainty by focusing concrete action on smaller issues
- “May produce a solution for only the first few requirements they encounter.”

Sources: *Making Use* by John M. Carroll (2000) and *Technology and Change* by D.A. Schon (1967).

What Stories Are Not

What are we building?

IEEE Specs

6. The product shall have a gas engine.
7. The product shall have four wheels.
   1. The product shall have a rubber tire mounted to each wheel.
8. The product shall have a steering wheel.
9. The product shall have a steel body.

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

What Stories Are Not

What if we had stories instead?

The product makes it easy and fast for the user to mow her lawn.

The user is comfortable while using the product.

What Stories Are Not

The product

What Stories Are Not

Stories are not use cases

Title: Accept reservation for a room.
Primary Actor: Purchaser
. . .
Main Success Scenario:

1. Purchaser submits credit card number, date, and authentication information.
2. System validates credit card.
3. System charges credit card full amount for all nights of stay.
4. Purchaser is given a unique confirmation number.

Mountain Goat Software

What Stories Are Not

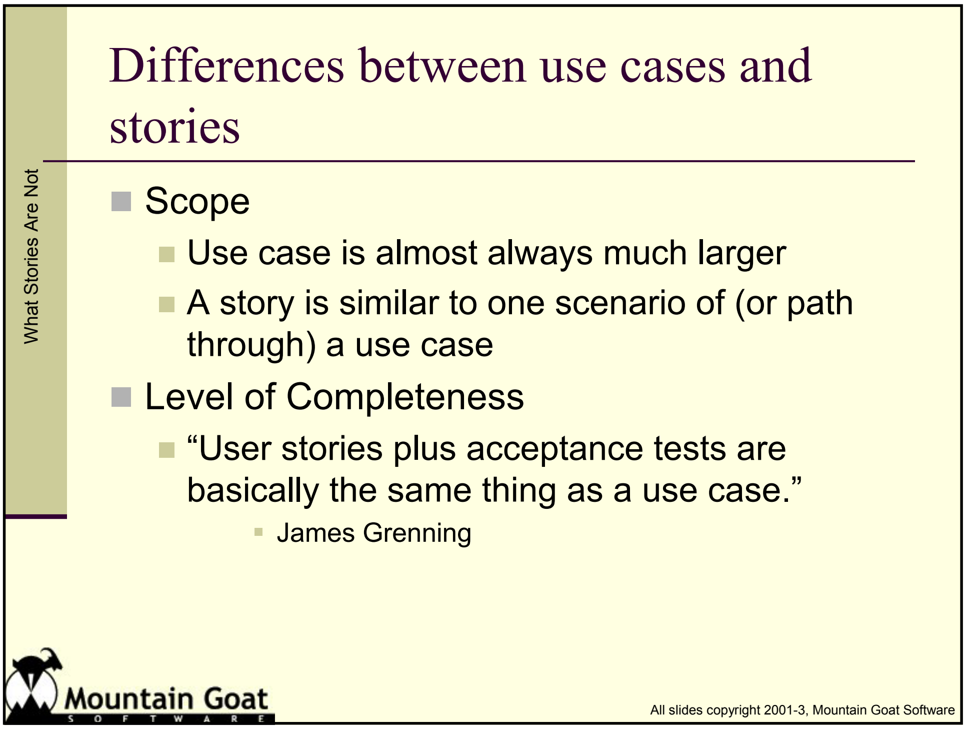
Stories are not use cases

Extensions:

2a The card is not a type accepted by the system.
    2a1 System notifies the user to use a different card.
2b The card is expired.
    2b1 System notifies the user to use a different card.

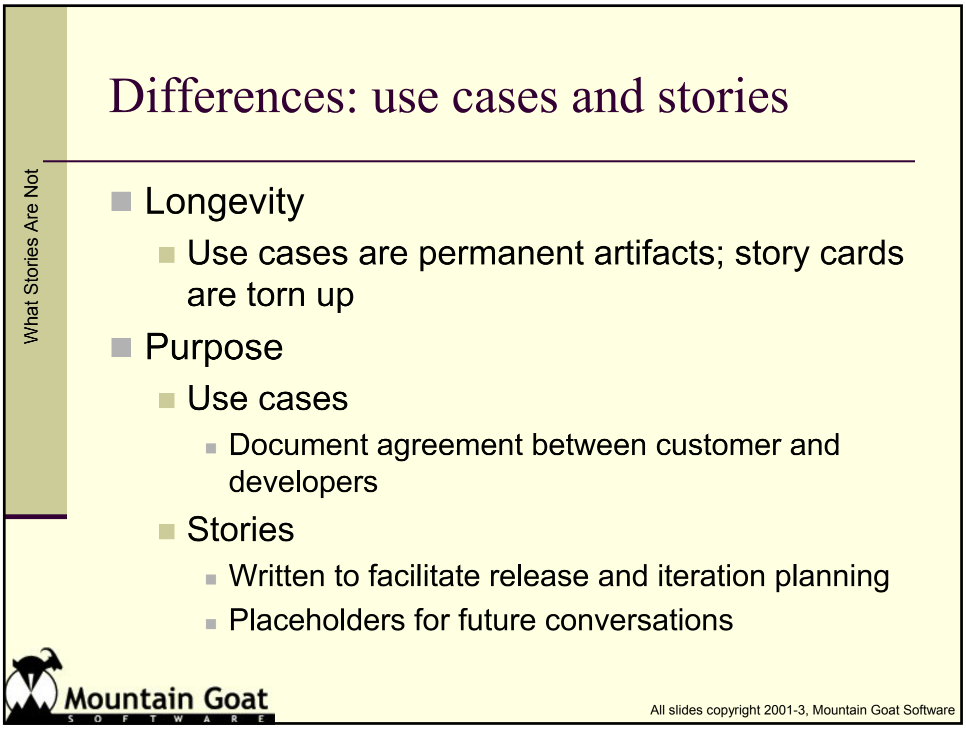
3a The card has insufficient available credit.
    3a1 System charges as much as it can to the current card.
    3b1 User is told about the problem and asked to enter a second card; use case continues at 2

Mountain Goat Software



Differences between use cases and stories

- Scope
  - Use case is almost always much larger
  - A story is similar to one scenario of (or path through) a use case
- Level of Completeness
  - “User stories plus acceptance tests are basically the same thing as a use case.”
    - James Grenning



Differences: use cases and stories

- Longevity
  - Use cases are permanent artifacts; story cards are torn up
- Purpose
  - Use cases
    - Document agreement between customer and developers
  - Stories
    - Written to facilitate release and iteration planning
    - Placeholders for future conversations

What Stories Are Not

Stories aren’t scenarios

- Scenarios are popular for designing human-computer interfaces
  - More prevalent in device design than pure software
- Much more detailed than a user story
- Examples of use back into the early 1970s for strategic planning

Mountain Goat SOFTWARE

An example scenario

Amy is interested in Japanese culture and is planning a trip there. As a child, she lived there for two years. Amy comes to our website and clicks on the Hotel link. She types in Nagoya and the dates November 14 through November 19. She selects the Royal Park Inn. Since she is a dedicated swimmer, rarely missing a day, Amy makes sure the hotel has a lap pool. It doesn’t so she backs up and tries the Sofitel Hotel…

Mountain Goat SOFTWARE

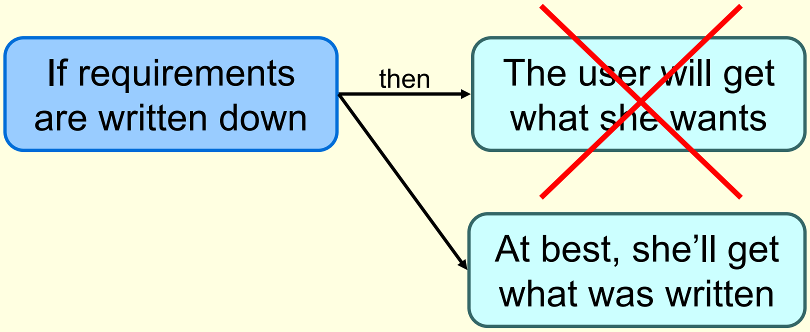
Differences: scenarios and stories

- Scenarios
  - Typically more focused on the user’s goals than user stories are
  - There’s a danger of overspecifying them
  - Much larger than a story
    - Scenario spans multiple stories
    - Not as suitable for planning and tracking
  - Similar focus on conversation rather than writing it all down

Why Stories?

So, why user stories?

- Written words are a shaky foundation



If requirements are written down

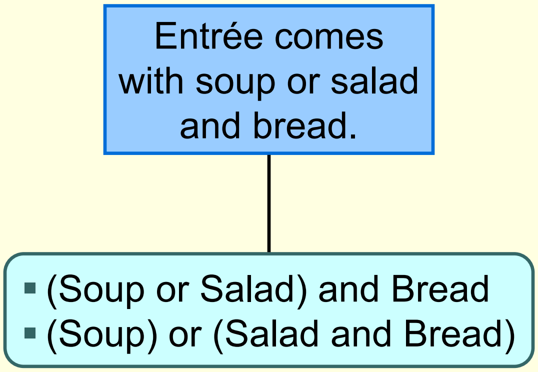
then

~~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.”

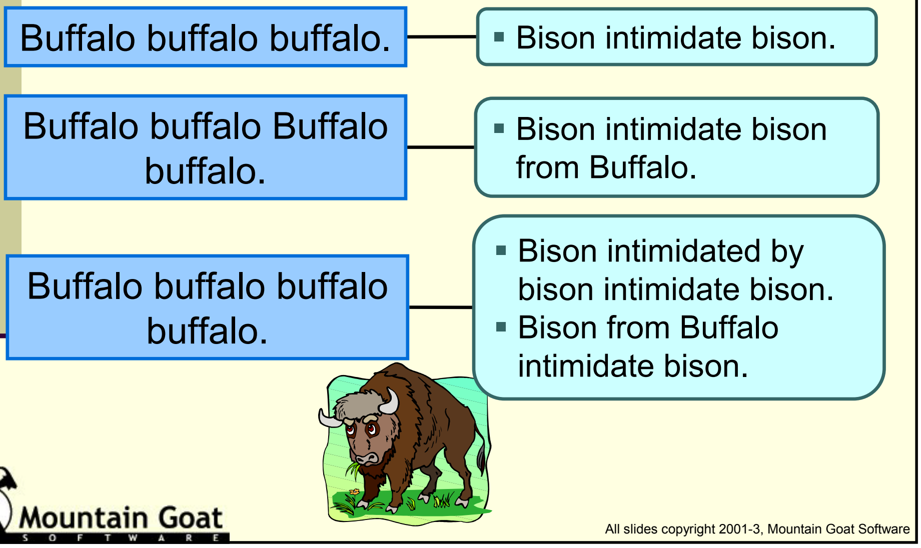
Words are imprecise



Entrée comes with soup or salad and bread.

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

Words have multiple meanings



Buffalo buffalo buffalo.

- Bison intimidate bison.

Buffalo buffalo Buffalo buffalo.

- Bison intimidate bison from Buffalo.

Buffalo buffalo buffalo buffalo.

- Bison intimidated by bison intimidate bison.
- Bison from Buffalo intimidate bison.

Why Stories?

Why user stories?

- Shift focus from written to verbal communication
  - Avoid the need to put everything in writing
  - “***Represent*** customer requirements rather than ***document*** them.”*
    - Rachel Davies, “The Power of Stories,” XP 2001.
- Are relatively small pieces of functionality
  - Better for planning
- Have value to the users
  - Can be understood and prioritized by users

Mountain Goat SOFTWARE

Today’s agenda

☑ Introduction
- ☑ What stories are
- ☑ What stories are not
- ☑ Why stories?

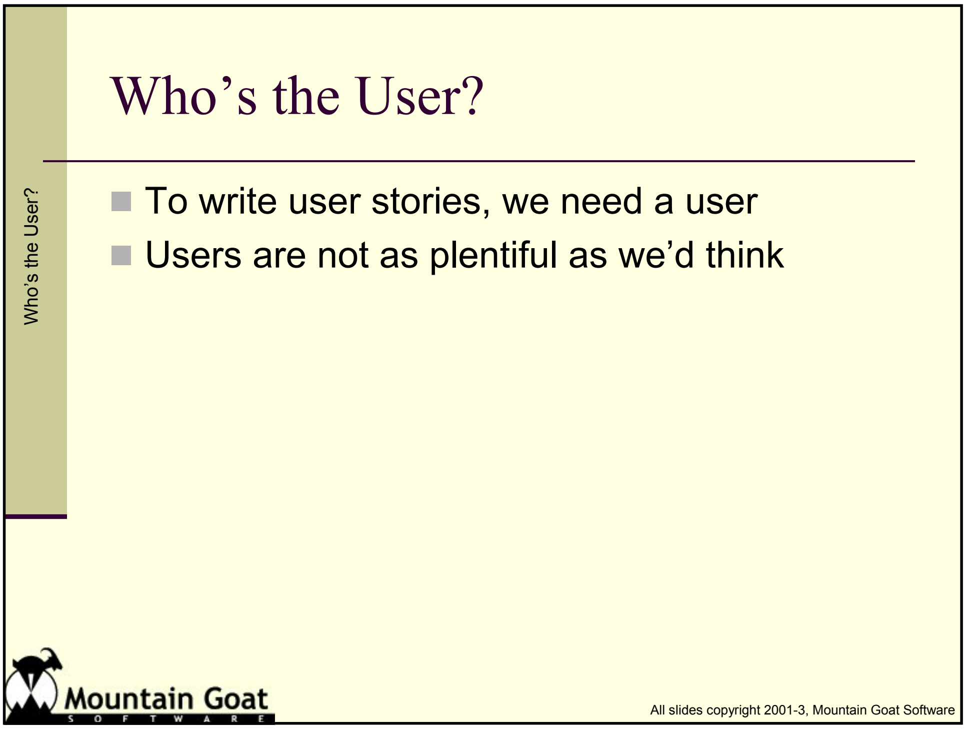
- The User and Customer
  - Who’s the user?
  - User roles and personas

☐ Gathering Stories

☐ Planning and Estimating
- ☐ Why plans go wrong
- ☐ Estimating user stories
- ☐ Planning with user stories

☐ Case Study

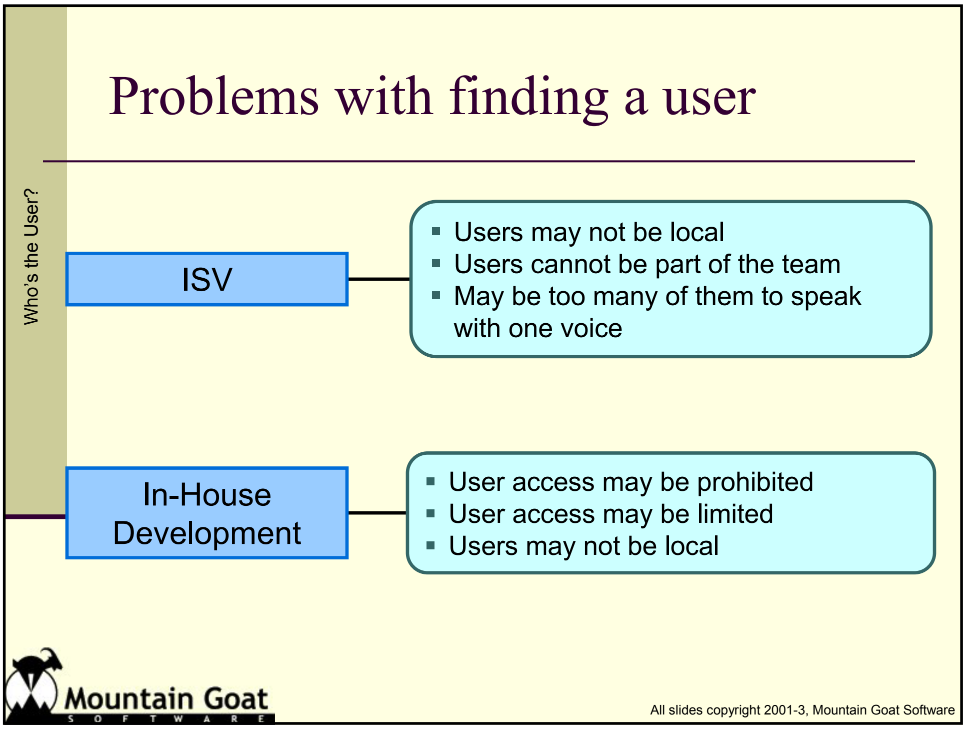
Mountain Goat SOFTWARE



Who’s the User?

Who’s the User?

- To write user stories, we need a user
- Users are not as plentiful as we’d think



Problems with finding a user

Who’s the User?

ISV

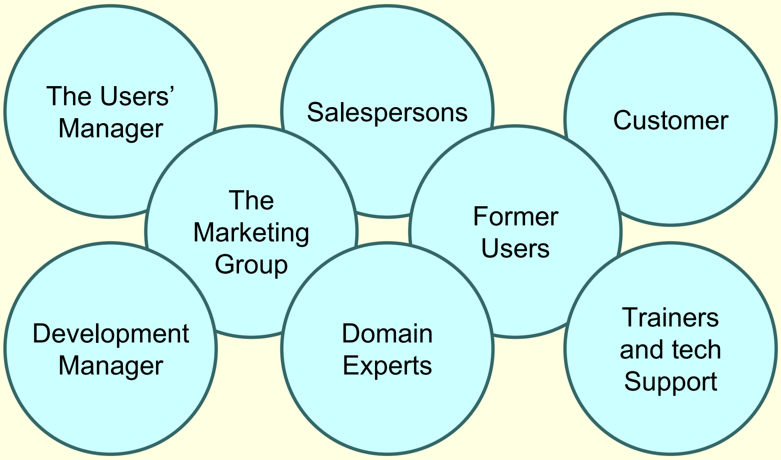
- Users may not be local
- Users cannot be part of the team
- May be too many of them to speak with one voice

In-House Development

- User access may be prohibited
- User access may be limited
- Users may not be local

Who's the User?

User proxies



The Users' Manager

Salespersons

Customer

The Marketing Group

Former Users

Development Manager

Domain Experts

Trainers and tech Support

Who's the User?

The users' manager

- Common when an organization wishes to restrict access to real users
- A "bait-and-switch" unless manager is also (still?) a user
- Managers have different usage patterns
- Manager may opt into this role from ego
  - "I know everything they need."
- Stories change as they are retold from user to manager
  - "Searches take five minutes."

Who's the User?

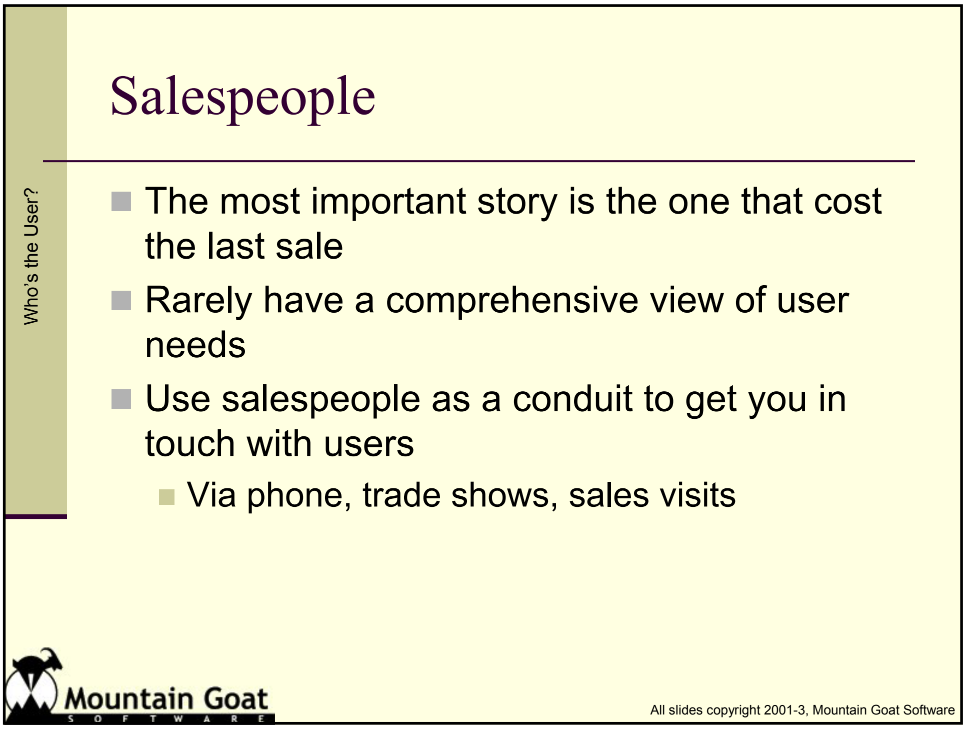
A development manager

- One of the worst possible choices unless…
  - …writing software for development managers
- Will almost certainly have conflicting goals
  - Make the users happy, but beat the deadline and get a bonus
  - Deliver the most important functionality first, but do the cool technology pieces "just a bit earlier"
- Rarely has hands-on experience as a user

Who's the User?

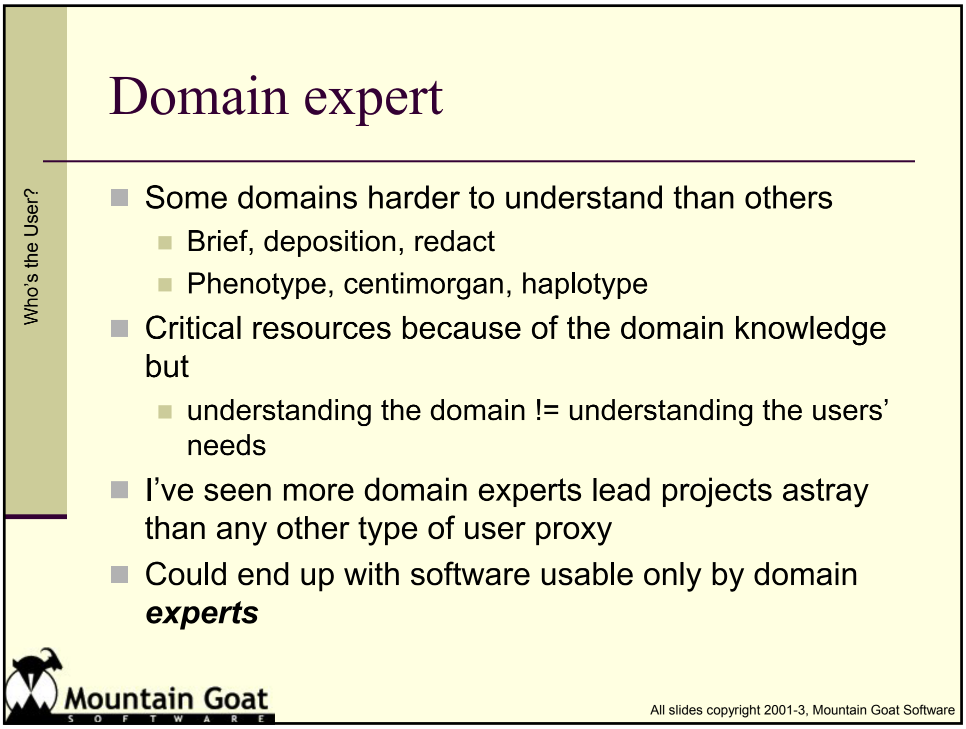
The marketing group

- Understands markets not users
  - Leads to focus on feature quantity not quality
  - Market consists of buyers, not necessarily users
- May think they know so much about the market they can infer what users will want
- Example: an automated book



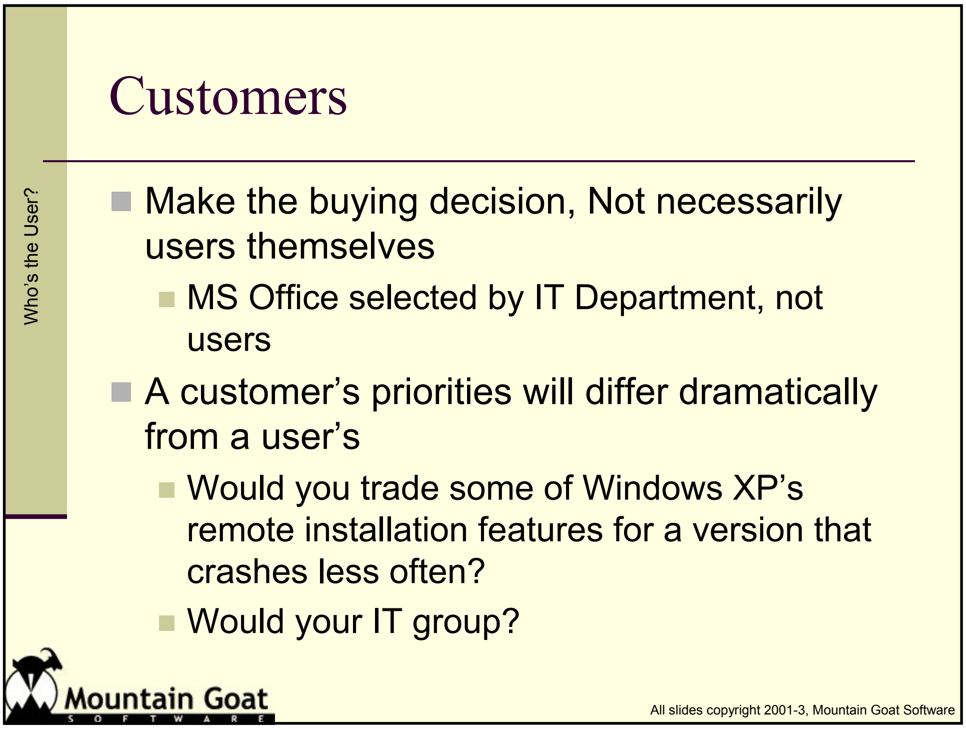
Salespeople

- The most important story is the one that cost the last sale
- Rarely have a comprehensive view of user needs
- Use salespeople as a conduit to get you in touch with users
  - Via phone, trade shows, sales visits



Domain expert

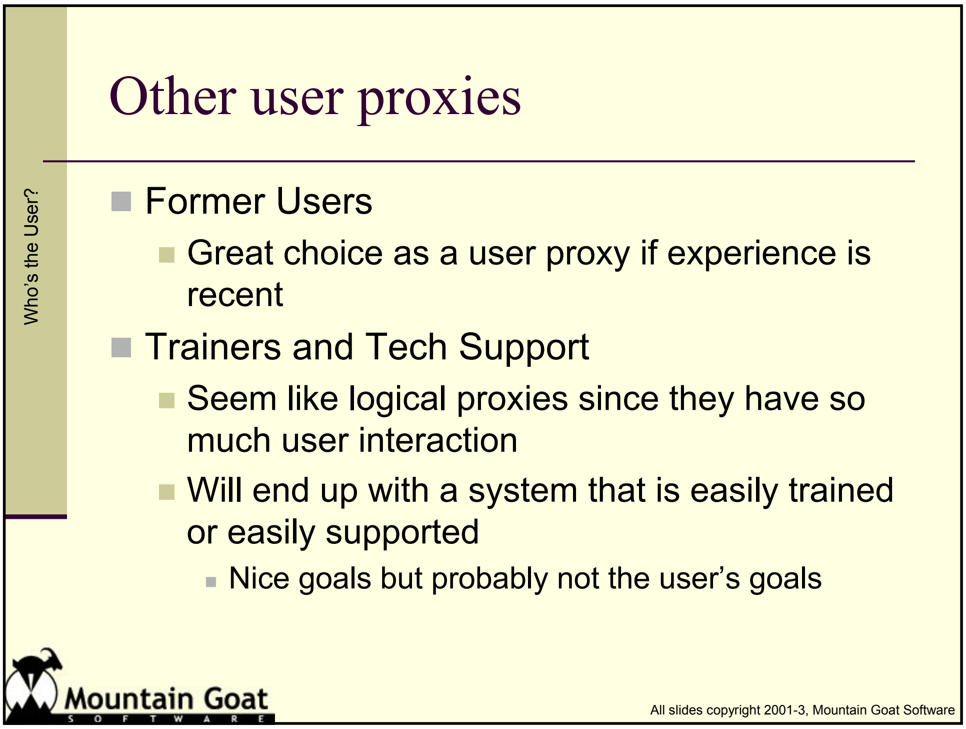
- Some domains harder to understand than others
  - Brief, deposition, redact
  - Phenotype, centimorgan, haplotype
- Critical resources because of the domain knowledge but
  - understanding the domain != understanding the users' needs
- I've seen more domain experts lead projects astray than any other type of user proxy
- Could end up with software usable only by domain ***experts***



Who's the User?

Customers

- Make the buying decision, Not necessarily users themselves
  - MS Office selected by IT Department, not users
- A customer's priorities will differ dramatically from a user's
  - Would you trade some of Windows XP's remote installation features for a version that crashes less often?
  - Would your IT group?



Who's the User?

Other user proxies

- Former Users
  - Great choice as a user proxy if experience is recent
- Trainers and Tech Support
  - Seem like logical proxies since they have so much user interaction
  - Will end up with a system that is easily trained or easily supported
    - Nice goals but probably not the user's goals

Who's the User?

What to do when access to users is restricted

- User Task Force
  - From 3 – 12 users
  - Tell the proxy that the task force is just for bouncing ideas off
  - Proxy is final decision-maker
    - Will rarely go against the opinion of the task force
  - Demo the software to this group as often as possible
    - Incorporate feedback into next iteration

Mountain Goat SOFTWARE

Who's the User?

What to do when there are no users

- Use more than one proxy
  - Use different types (e.g., Marketing + Domain Expert)
- Look at competing products
  - Product reviews, online newsgroups, user's guide
- Release early to real users

Mountain Goat SOFTWARE

Can you do it yourself?

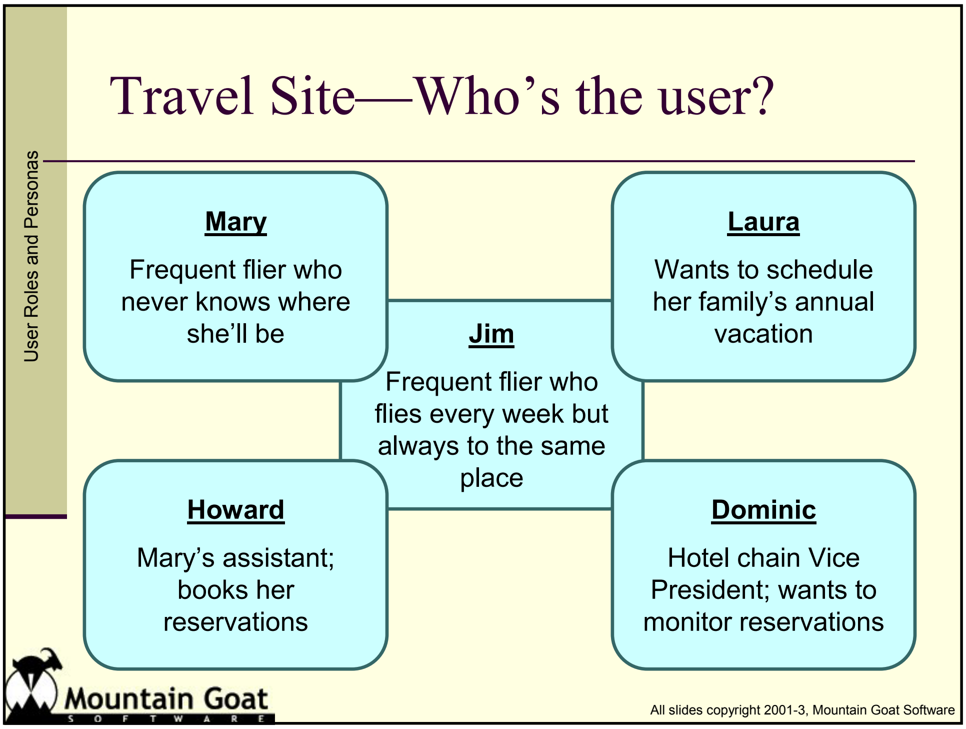
- Working with a user proxy has disadvantages

But

- A user proxy is still better than a development team taking guesses

"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 and Personas

Travel Site—Who's the user?

Mary

Frequent flier who never knows where she'll be

Laura

Wants to schedule her family's annual vacation

Jim

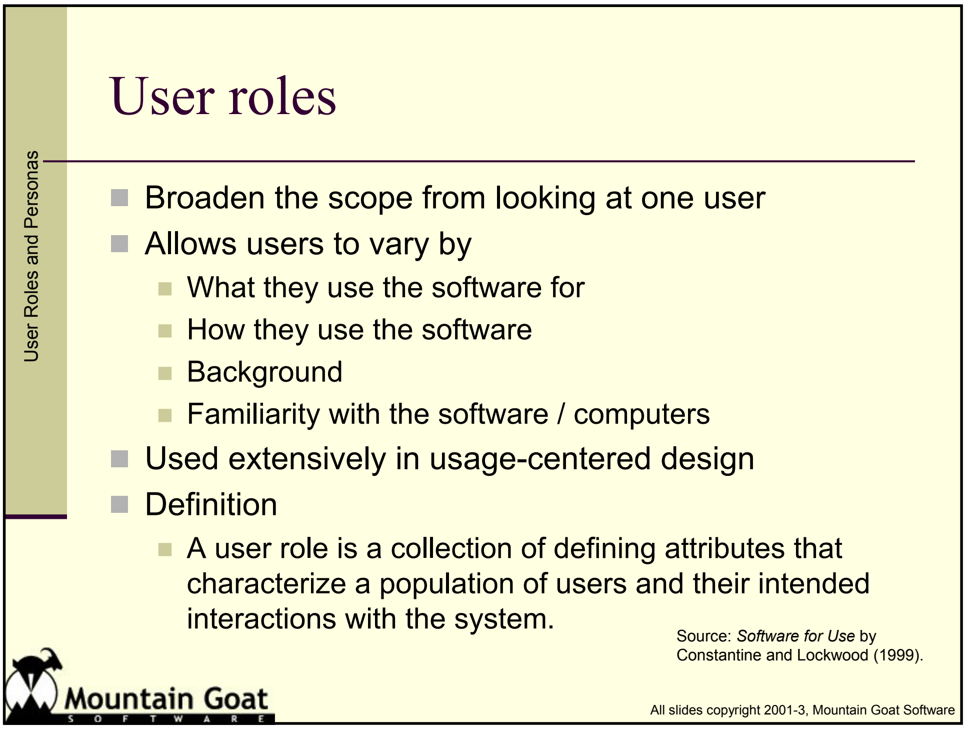
Frequent flier who flies every week but always to the same place

Howard

Mary's assistant; books her reservations

Dominic

Hotel chain Vice President; wants to monitor reservations



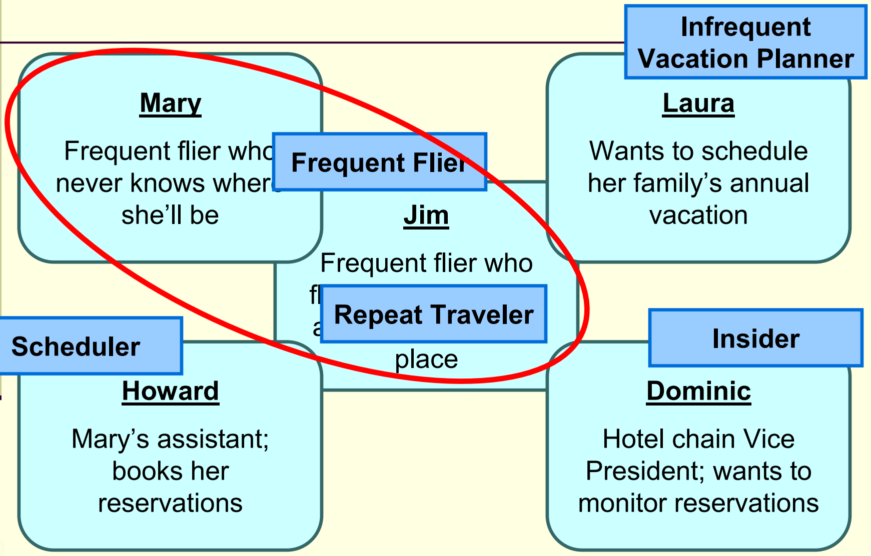
User Roles and Personas

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
- Definition
  - A user role is a collection of defining attributes that characterize a population of users and their intended interactions with the system.

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

Common attributes

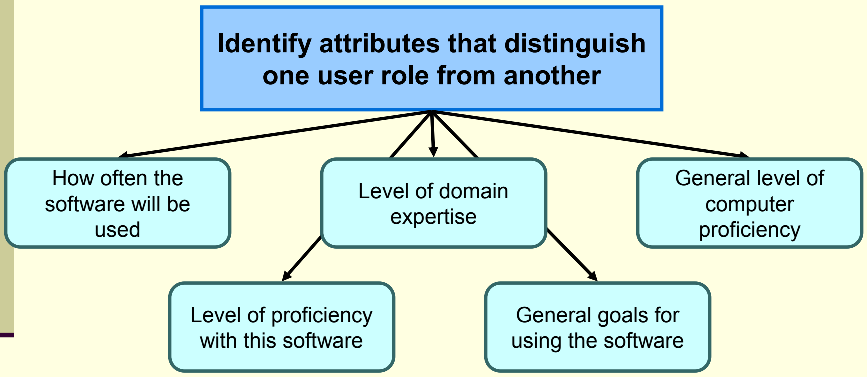

Infrequent Vacation Planner
Laura
Wants to schedule her family's annual vacation
Mary
Frequent flier who never knows where she'll be
Frequent Flier
Jim
Frequent flier who fl
a
Repeat Traveler
place
Scheduler
Howard
Mary's assistant; books her reservations
Insider
Dominic
Hotel chain Vice President; wants to monitor reservations

User story / role matrix

	Book Simple Trips	Save and reuse trips	Booking reports	Research
Frequent Flier	√			√
Repeat Traveler	√	√		
Scheduler	√	√		√
Insider			√	
Vacation Planner	√			√

User Roles and Personas

User role modeling


Identify attributes that distinguish one user role from another
How often the software will be used
Level of domain expertise
General level of computer proficiency
Level of proficiency with this software
General goals for using the software

Mountain Goat Software

User Roles and Personas

Document the user role

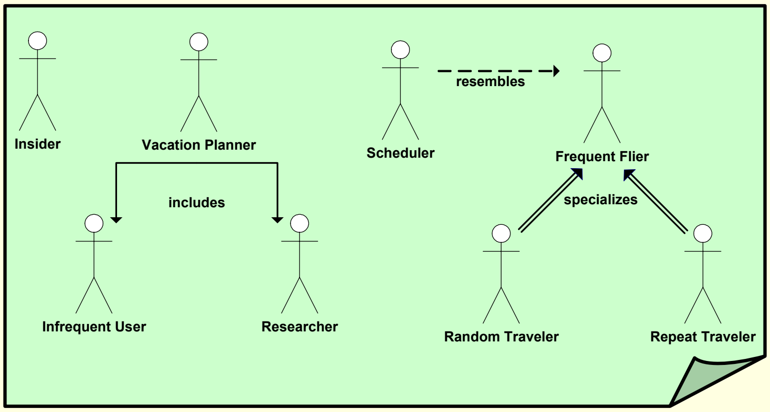
<u>User Role: Infrequent Vacation Planner</u>

Not particularly computer-savvy but quite adept at using the web. Will use the software infrequently but intensely (perhaps 5 hours to research and plan a trip). Values richness of experience (lots of content) over speed. But, software must be easy to learn and also easily recalled months later.

Mountain Goat Software

User Role Map

- Hard to “see” the user role inter-relationships using only text



Personas

- A central element of Alan Cooper’s interaction design
- A persona is an imaginary representation of a user role
- A natural extension to user roles
- Generally, avoid picking personas who are real users

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

User Roles and Personas

Add details to each persona

- Likes, dislikes
- When, where, why
- Model and make of car
- Job
  - Not “is a florist” but “works as a florist at Lake Park Florist”)
- Goals
  - Not “planning a vacation but “planning the family vacation to Yellowstone”

User Roles and Personas

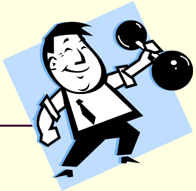
A sample persona

Jim lives in four bedroom house in a nice suburb north of Chicago. However, he works as a vice president of marketing in Sacramento, California. Three weeks out of every four he flies from Chicago to Sacramento on Monday morning and then flies home on Friday. The company lets him work every fourth week out of his home. Jim schedules his own flights, usually a month or more in advance. He’s partial to United Airlines but is always on the lookout for bargain fares so that the company will allow him to continue to live in Chicago. Jim quickly learns most software but becomes very impatient when he finds a bug or when a website is slow.

Using roles and personas

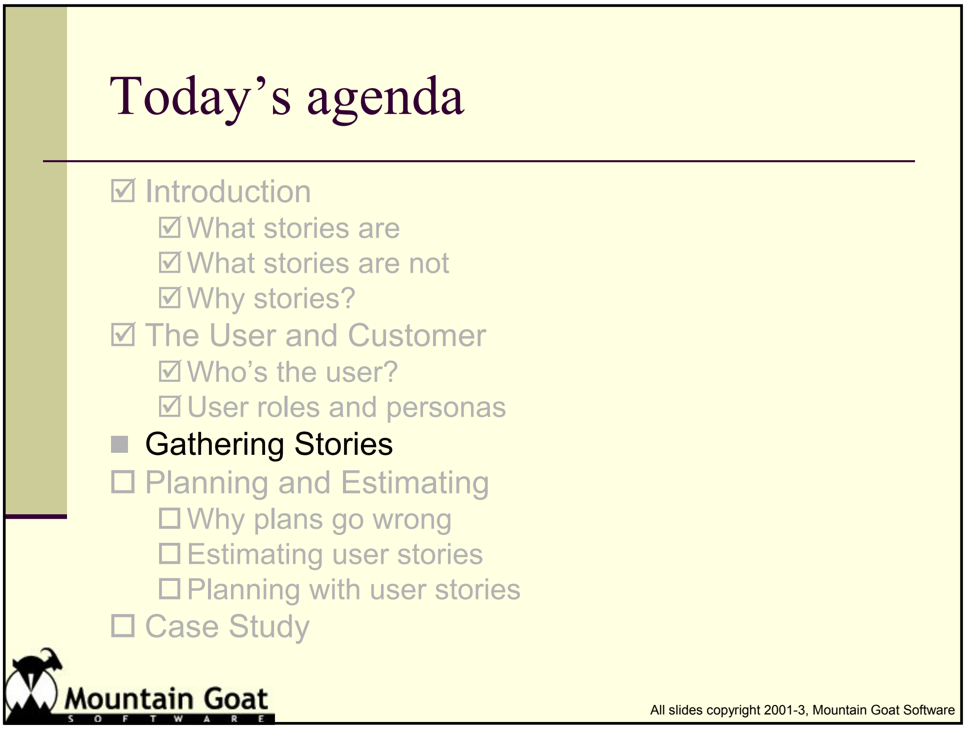
- Start thinking of the software as solving the needs of real people
- Avoid saying “the user” and instead say
  - “A Frequent Flier…”
  - “A Repeat Traveler…”
  - “Jim…”

Exercise



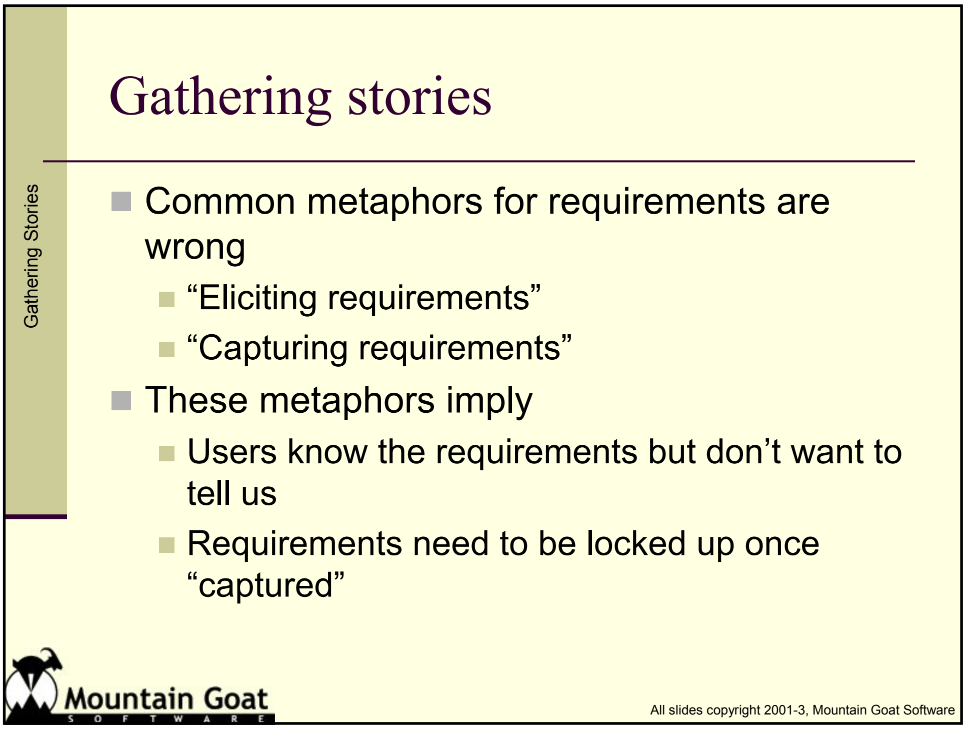
We have been asked to develop a new job posting and search site.

1) What roles are there?
2) Which roles are the most important to satisfy?
3) Which would you extend into personas?



Today’s agenda

- ☑ Introduction
  - ☑ What stories are
  - ☑ What stories are not
  - ☑ Why stories?
- ☑ The User and Customer
  - ☑ Who’s the user?
  - ☑ User roles and personas
- Gathering Stories
- ☐ Planning and Estimating
  - ☐ Why plans go wrong
  - ☐ Estimating user stories
  - ☐ Planning with user stories
- ☐ Case Study



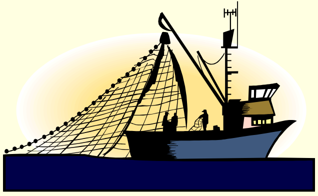
Gathering Stories

Gathering stories

- Common metaphors for requirements are wrong
  - “Eliciting requirements”
  - “Capturing requirements”
- These metaphors imply
  - Users know the requirements but don’t want to tell us
  - Requirements need to be locked up once “captured”

The proper metaphor

- Trawling* for requirements
  - Trawl: “sift through as part of a search” (OAD)
- Metaphor captures these aspects:
  - Requirements can be captured with different sized nets
  - Requirements change, mature, possibly die
  - Skill is a factor

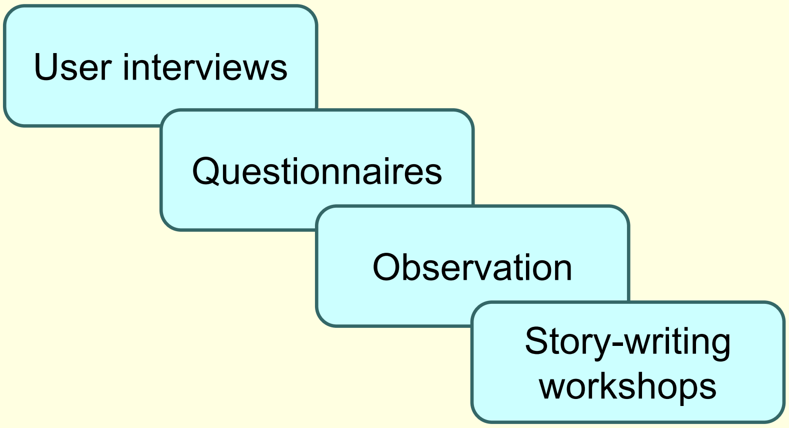


Source: *Mastering the Requirements Process* by Suzanne and James Robertson, 1999.

A little is enough, or is it?

- Agile processes acknowledge that we cannot trawl with such a fine net that we can write all the user stories upfront
- However,
  - This doesn’t mean we shouldn’t write as many as we can

Techniques for trawling for user stories



User interviews

Questionnaires

Observation

Story-writing workshops

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
  - Having a problem does not uniquely qualify you for knowing how to solve it

Open-ended and context-free questions

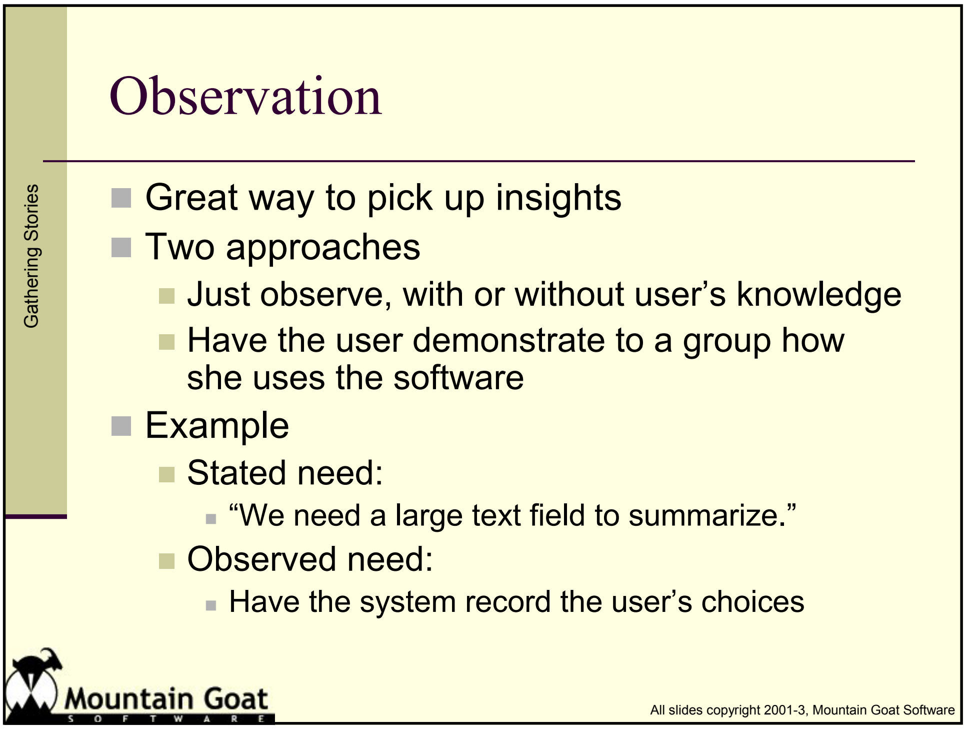
Gathering Stories

- “Would you like it in a browser?”
- Two problems:
  - A closed-ended question
  - Has no context
- Instead ask:
  - “Would you like it in a browser rather than as a native Windows application even if it means reduced performance, a poorer overall user experience, and less interactivity?”
- Still, that question can be improved
  - “What would you be willing to give up in order to have it in a browser?”

Questionnaires

Gathering Stories

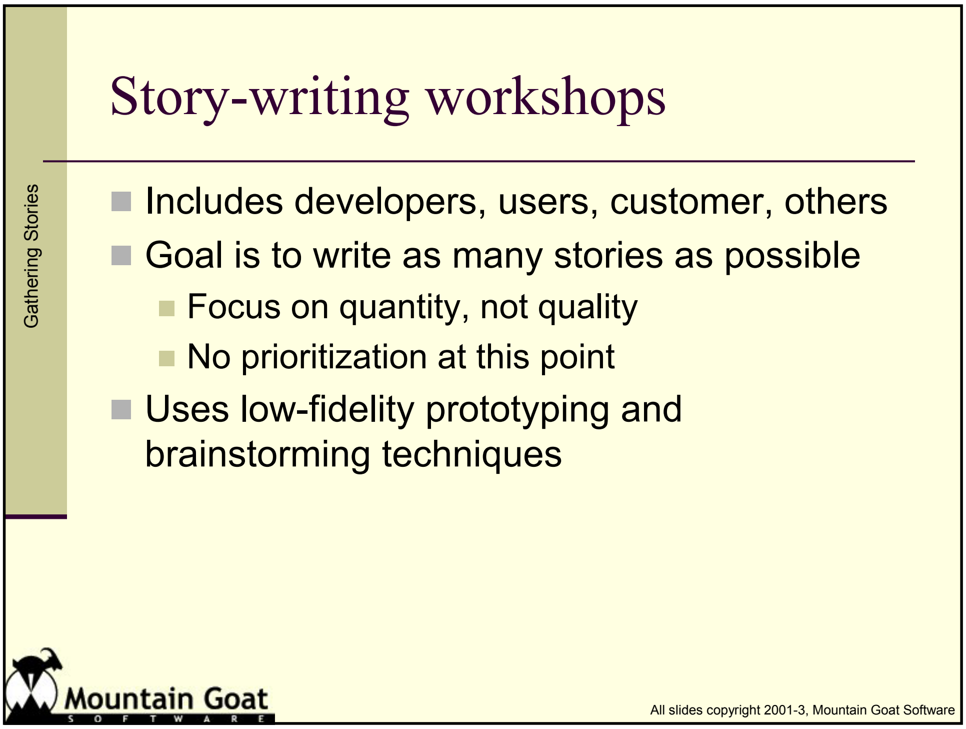
- Good technique for learning more about stories you already have
- If you have a large user base, great way to get information to help prioritize stories
- Not effective as a primary means of trawling for new stories



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

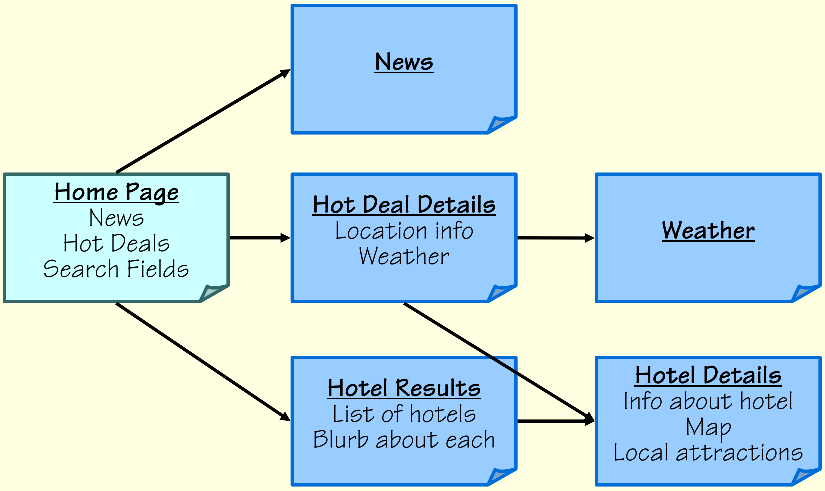


Gathering Stories

Story-writing workshops

- Includes developers, users, customer, others
- Goal is to write as many stories as possible
  - Focus on quantity, not quality
  - No prioritization at this point
- Uses low-fidelity prototyping and brainstorming techniques

A low-fidelity prototype



Home Page
News
Hot Deals
Search Fields

News

Hot Deal Details
Location info
Weather

Weather

Hotel Results
List of hotels
Blurb about each

Hotel Details
Info about hotel
Map
Local attractions

Low-fidelity prototyping

- Use paper, note cards, white board, big Post-its
- Prototype is of components or areas within the application, *not* of actual screens
  - Hotel Results could be on Home Page or be a separate page
- Doesn’t require knowledge of how screens will look
- Throw it away a day or two later
- Works better to go depth-first

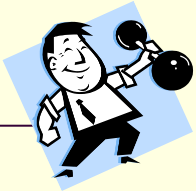
Gathering Stories

Creating the low-fidelity prototype

- Start with an empty box:
  - "Here's the main screen in the system"
- Ask open-ended, context-free questions as you go:
  - What will the users most likely want to do next?
  - What mistakes could the user make here?
  - What could confuse the user at this point?
  - What additional information could the user need?
- Consider these questions for each user role

User Roles and Personas

Exercise



1) Write some stories, based on the user roles for our job posting and search site.

Today's agenda

- ☑ Introduction
  - ☑ What stories are
  - ☑ What stories are not
  - ☑ Why stories?
- ☑ The User and Customer
  - ☑ Who's the user?
  - ☑ User roles and personas
- ☑ Gathering Stories
- Planning and Estimating
  - Why plans go wrong
  - Estimating user stories
  - Planning with user stories
- □ Case Study

Mountain Goat SOFTWARE

Why plans go wrong

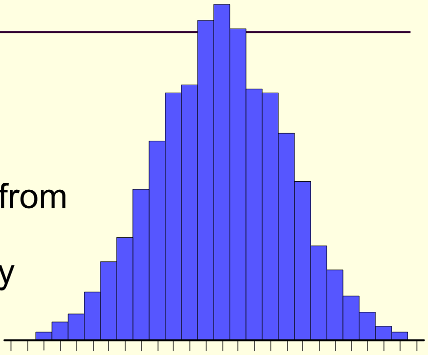
Why Plans Go Wrong

- Before we can build a successful release plan, need to know why most plans fail
- Three reasons
  1. Tasks are assumed to be independent
  2. Lateness is passed down the schedule; earliness is not
  3. Student Syndrome

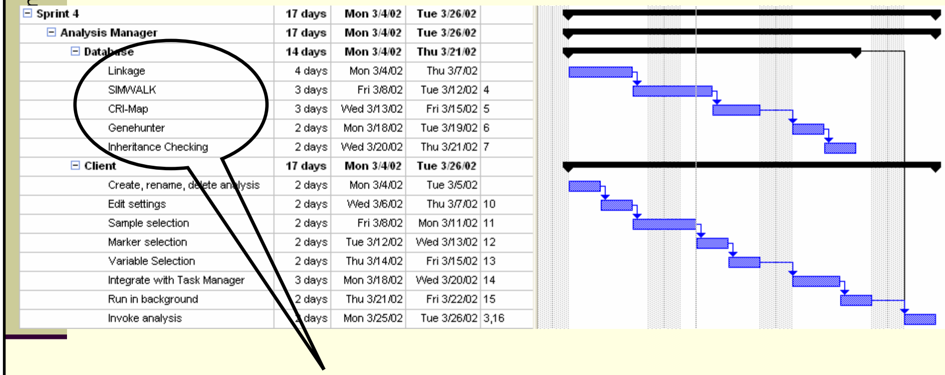
Mountain Goat SOFTWARE

Task Independence

- Sum of five dice
- Central Limit Theorem
  - Sum of a number of independent samples from any distribution is approximately normally distributed
- This means that
  - some are bigger
  - some are small
  - but overall things average out



Does CLT apply to software?

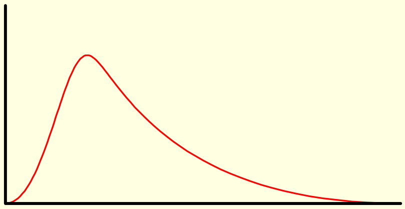


⊟ **Sprint 4**	**17 days**	**Mon 3/4/02**	**Tue 3/26/02**	
⊟ **Analysis Manager**	**17 days**	**Mon 3/4/02**	**Tue 3/26/02**	
⊟ **Database**	**14 days**	**Mon 3/4/02**	**Thu 3/21/02**	
Linkage	4 days	Mon 3/4/02	Thu 3/7/02	
SIMWALK	3 days	Fri 3/8/02	Tue 3/12/02	4
CRI-Map	3 days	Wed 3/13/02	Fri 3/15/02	5
Genehunter	2 days	Mon 3/18/02	Tue 3/19/02	6
Inheritance Checking	2 days	Wed 3/20/02	Thu 3/21/02	7
⊟ **Client**	**17 days**	**Mon 3/4/02**	**Tue 3/26/02**	
Create, rename, delete analysis	2 days	Mon 3/4/02	Tue 3/5/02	
Edit settings	2 days	Wed 3/6/02	Thu 3/7/02	10
Sample selection	2 days	Fri 3/8/02	Mon 3/11/02	11
Marker selection	2 days	Tue 3/12/02	Wed 3/13/02	12
Variable Selection	2 days	Thu 3/14/02	Fri 3/15/02	13
Integrate with Task Manager	3 days	Mon 3/18/02	Wed 3/20/02	14
Run in background	2 days	Thu 3/21/02	Fri 3/22/02	15
Invoke analysis	2 days	Mon 3/25/02	Tue 3/26/02	3,16

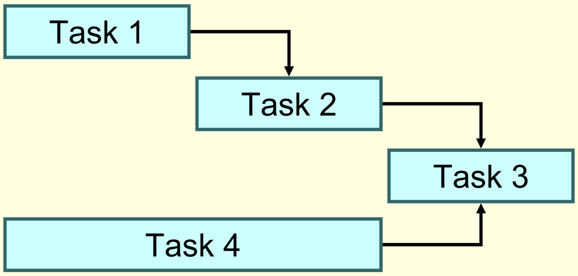
Highly correlated tasks

CLT and software

- The tasks on a software Gantt chart are not independent
  - Many tasks involve similar work; if one estimate is wrong the others tend to be wrong
  - There may be systematic error in the estimates
    - “Jay Days”
- Software estimates tend not to be normally distributed
  - When asked for a point estimate programmers respond with the mode



Lateness is passed along the schedule



- Task 3 starts:
  - **LATE** if 1, 2 *or* 4 is late
  - **EARLY** only if 2 *and* 4 are early, and resource is available

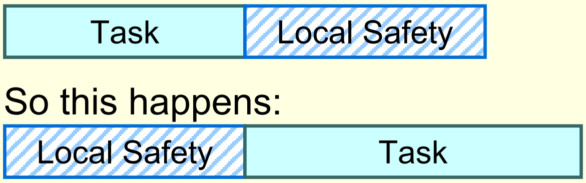
Why Plans Go Wrong

Student syndrome

- Refers to starting tasks at the last possible moment that doesn't preclude an on-time completion
  - e.g., starting a term paper the night before it's due

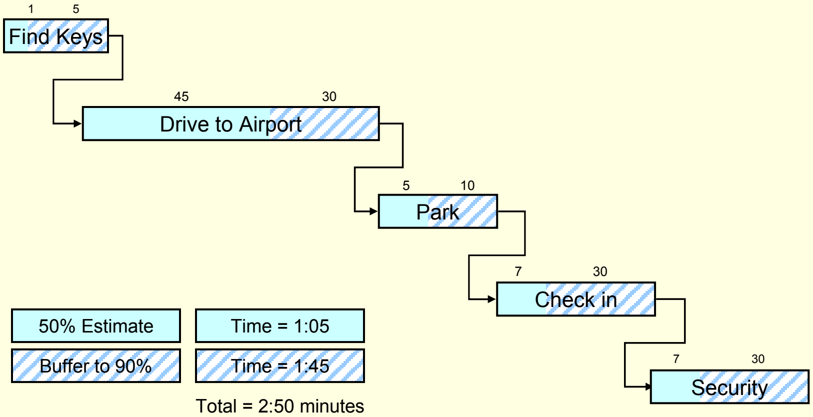
Estimate shows this:

So this happens:



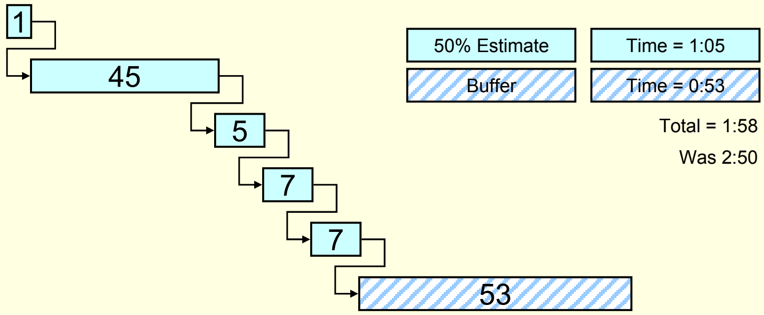
Why Plans Go Wrong

My trip to the airport



Total = 2:50 minutes

Trip to the airport with a project buffer



1

45

5

7

7

53

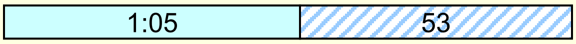
50% Estimate	Time = 1:05
Buffer	Time = 0:53

Total = 1:58

Was 2:50

A project buffer isn’t padding

- Padding is extra time you don’t think you’ll need but add to be safe
- You will need the project buffer
  - Even with the project buffer you’re not guaranteed to be done on time
- I have a 3% chance of making it to my flight in 65 minutes
  - $50\% \times 50\% \times 50\% \times 50\% \times 50\% = 3.125\%$



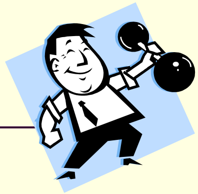
1:05

53

- Would you call something that increases your odds of success from 3% “padding”?

Why Plans Go Wrong

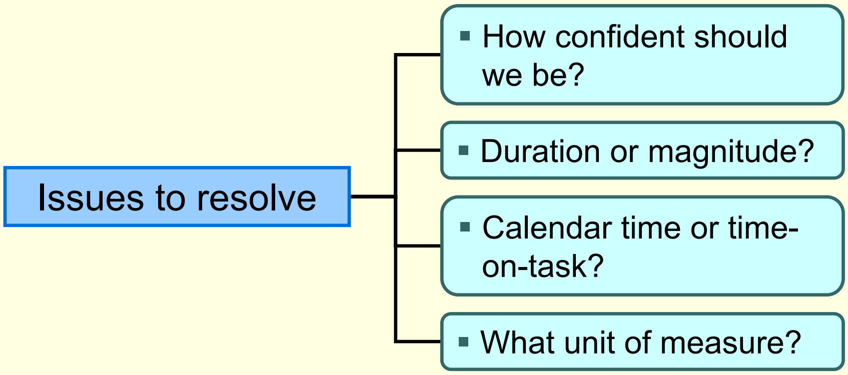
Exercise



What are the tasks involved in selecting an install tool (e.g., InstallShield, InstallAnywhere, etc.) for corporate-wide use?

Estimating Stories

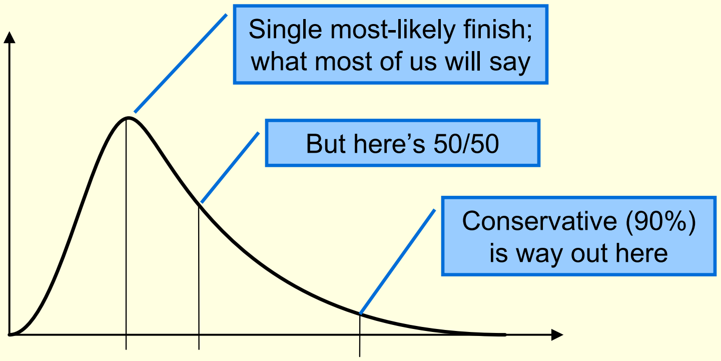
Estimating stories



Issues to resolve

- How confident should we be?
- Duration or magnitude?
- Calendar time or time-on-task?
- What unit of measure?

How confident should we be?



Single most-likely finish; what most of us will say

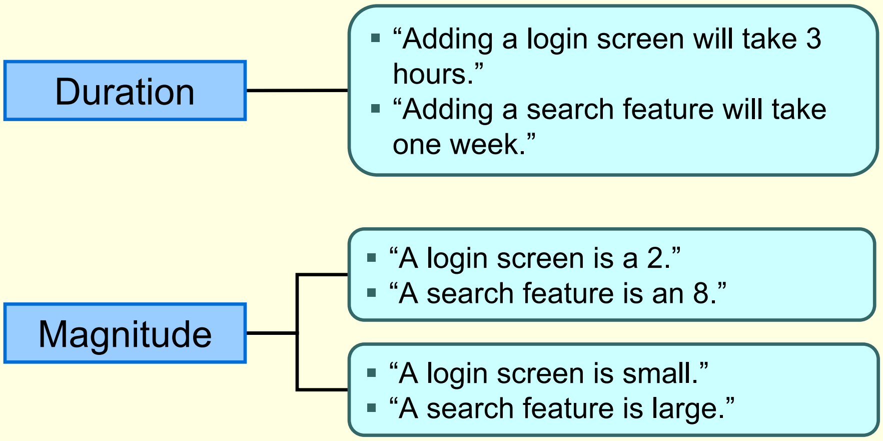
But here’s 50/50

Conservative (90%) is way out here

Give both 50% and 90% estimates

- 50% estimates
  - Remove all *local safety*: no “padding”
  - An estimate you should / will miss half the time
- 90% estimates
  - Not really a worst case
    - No lightning strikes or busses running over people
  - Keep in mind that you’ll even exceed this estimate occasionally

Duration vs. magnitude



Duration

- “Adding a login screen will take 3 hours.”
- “Adding a search feature will take one week.”

Magnitude

- “A login screen is a 2.”
- “A search feature is an 8.”

- “A login screen is small.”
- “A search feature is large.”

Problems with magnitude

- Values must be meaningful and distinguishable
  - How do you tell a “67” from a “68”?
- Eventually you need to convert an estimate of magnitude into an estimate of duration
  - “We’ll be done in 8 mediums, 3 smalls and 4 larges.”
  - “We’ll be done in 43 Gummi Bears.”
- Developers may make an implicit conversion

Estimating Stories

Advantages to magnitude

- Some developers find it much easier to say “this is like that”
  - But, that can still be done with duration:
    - “This is like that and that took two days.”
- The abstractness can prevent forcing estimates to meeting a date
  - “My boss wants this in two weeks, I guess I’ll say ‘two weeks.’”

Mountain Goat SOFTWARE

Estimating Stories

Duration / magnitude recommendation

- I prefer estimating in an abstract form duration
  - (As we’ll see in a few minutes)
- If you prefer magnitude, that will work, too
  - May not be that different from what I’ll describe

Mountain Goat SOFTWARE

Estimating Stories

Calendar time vs. time-on-task

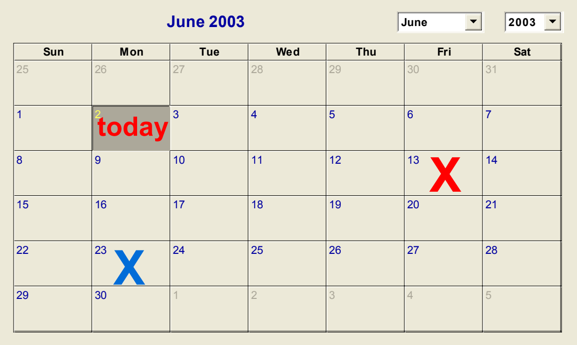
- Calendar time
  - Monday has 8 hours
- Time-on-Task
  - Called “Ideal Time” in XP
  - Monday has
    - 3 hours of meeting
    - 1 hour of email
    - 4 hours of programming (time-on-task)

Mountain Goat SOFTWARE

Estimating Stories

“How long will this take?”

- “Two weeks.”
- Two *calendar* weeks or two weeks worth of *time on task*?



June 2003

June ▾ 2003 ▾

Sun	Mon	Tue	Wed	Thu	Fri	Sat
25	26	27	28	29	30	31
1	2 today	3	4	5	6	7
8	9	10	11	12	13 X	14
15	16	17	18	19	20	21
22	23 X	24	25	26	27	28
29	30	1	2	3	4	5

Mountain Goat SOFTWARE

So, Ideal Time or Calendar Time?

- Estimate in Ideal Time
- Too hard to consider all the factors that affect calendar time at the same time you're estimating

Factors affecting elapsed time

- Vacations
- Sick time
- All-company meetings
- Department meetings
- Personnel issues
- Training
- Email
- Phone Calls
- Demos
- Special projects
- Reviews & walk-throughs
- Interviewing candidates
- Spikes
- Leaves of absence
- Sabbaticals

But, there’s a problem

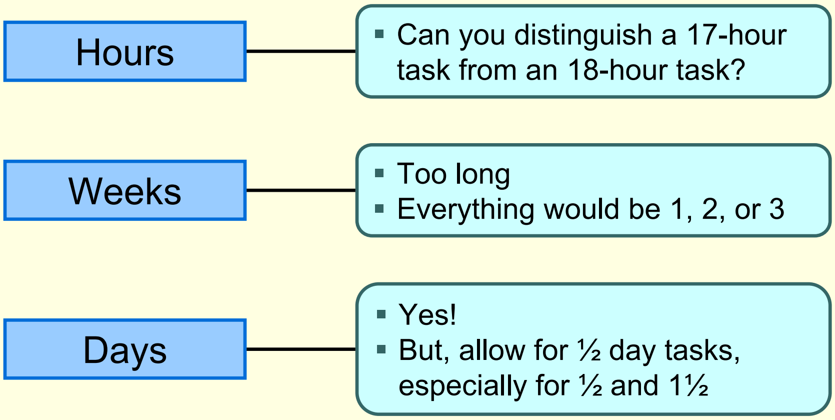
- Whose ideal time? Yours? Mine?

How do we add
Your Ideal Time
to
My Ideal Time?

Experienced Senior Programmer Days

- How?
  - Define an archetypal programmer and estimate how long it will take her
  - I like an “Experienced Senior Programmer”
    - But it can vary and depends on the team
- Why?
  - Estimates can be more honest
    - If questioned, “Oh, it wouldn’t take *me* that long.”
  - Bias toward insufficient estimates goes away
  - Estimates can be added and compared

Unit of measure



Hours

- Can you distinguish a 17-hour task from an 18-hour task?

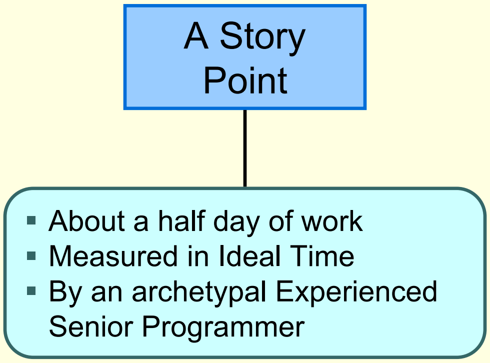
Weeks

- Too long
- Everything would be 1, 2, or 3

Days

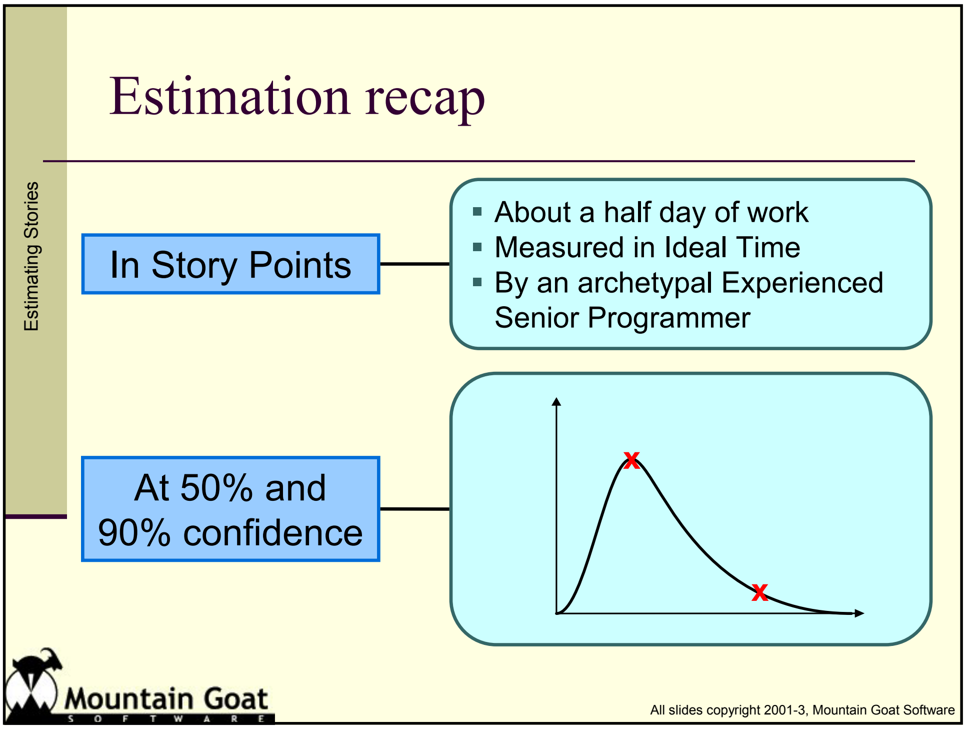
- Yes!
- But, allow for ½ day tasks, especially for ½ and 1½

The Story Point



A Story Point

- About a half day of work
- Measured in Ideal Time
- By an archetypal Experienced Senior Programmer



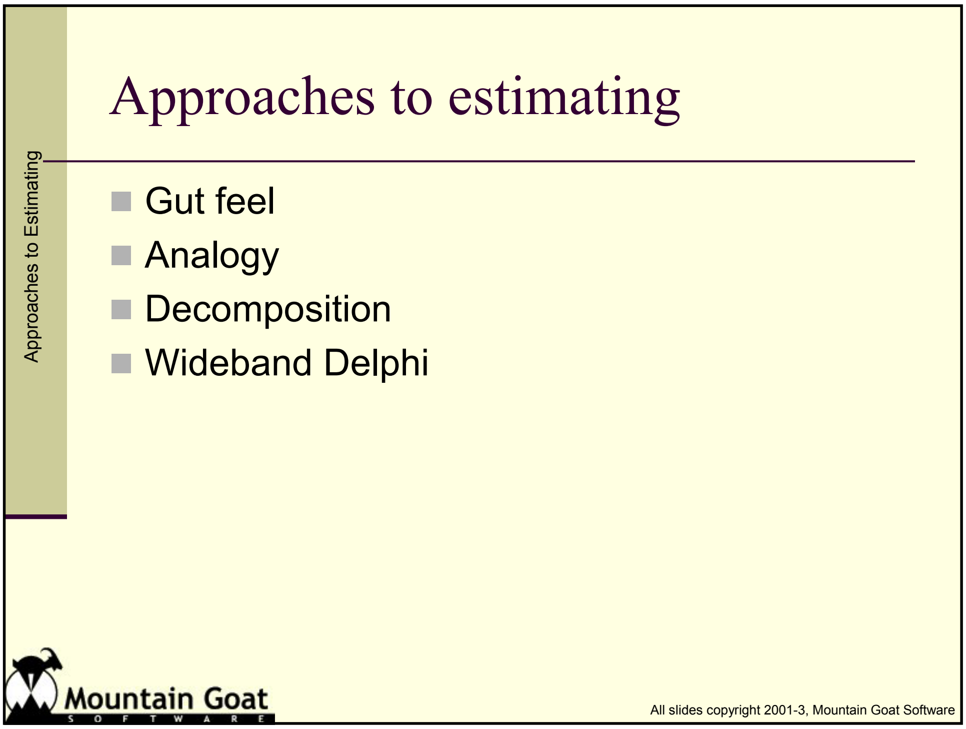
Estimating Stories

Estimation recap

In Story Points

- About a half day of work
- Measured in Ideal Time
- By an archetypal Experienced Senior Programmer

At 50% and 90% confidence



Approaches to Estimating

Approaches to estimating

- Gut feel
- Analogy
- Decomposition
- Wideband Delphi

Approaches to Estimating

Gut feel

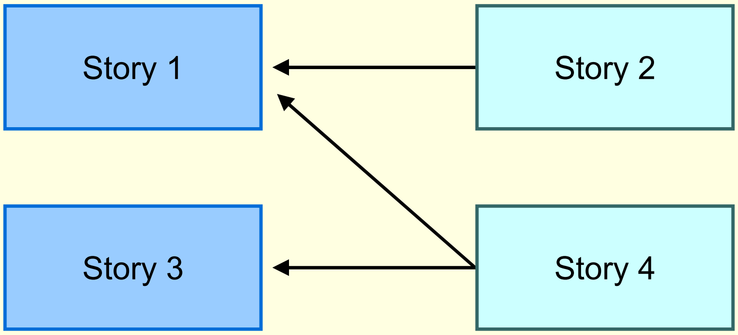
- Good as a reasonableness check

Approaches to Estimating

Analogy

- Analogy
  - “This story is like that story, so it’s estimate is what that story’s estimate was.”
  - Works if baseline story has been coded estimate turns out right
  - Prone to systematic error
    - The baseline story was wrong so all estimates done by analogy to it are wrong
  - Triangulate
    - Estimate by analogy to two different stories

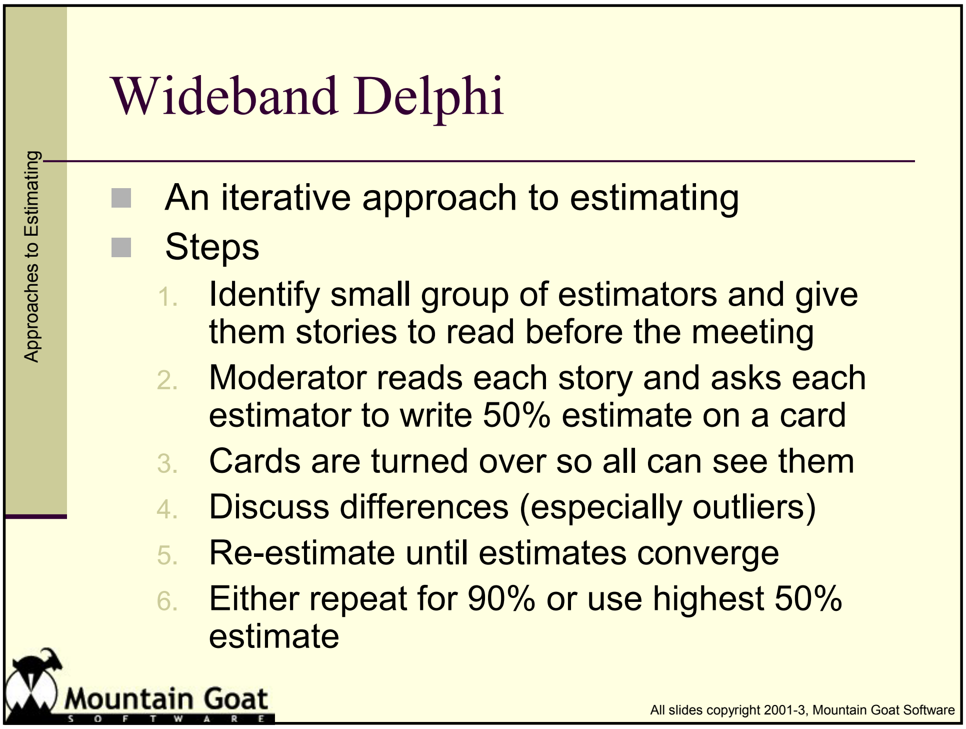
Triangulation



- Don’t triangulate Story 4 by estimating by analogy to both Story 1 and Story 2.

Decomposition

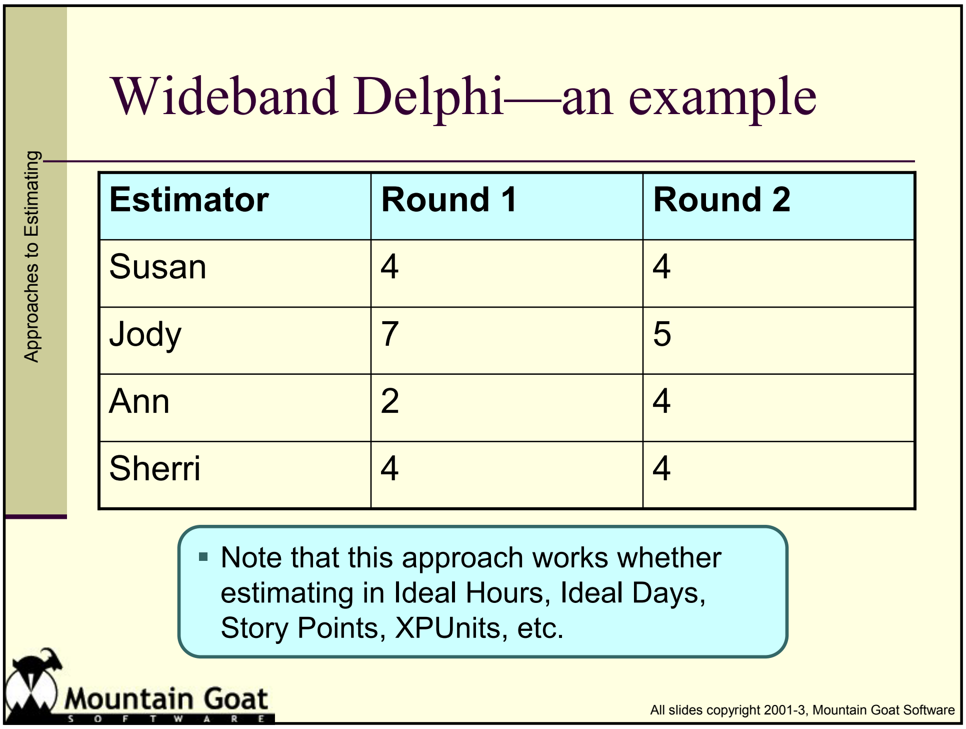
- Breaking a big story into littler stories or tasks
- Idea is, you know how long the smaller tasks take
- Sometimes very useful
- But decomposing too far causes problems
  - Forgotten tasks
  - Summing lots of small errors can be big number



Approaches to Estimating

Wideband Delphi

- An iterative approach to estimating
- Steps
  1. Identify small group of estimators and give them stories to read before the meeting
  2. Moderator reads each story and asks each estimator to write 50% estimate on a card
  3. Cards are turned over so all can see them
  4. Discuss differences (especially outliers)
  5. Re-estimate until estimates converge
  6. Either repeat for 90% or use highest 50% estimate



Approaches to Estimating

Wideband Delphi—an example

Estimator	Round 1	Round 2
Susan	4	4
Jody	7	5
Ann	2	4
Sherri	4	4

- Note that this approach works whether estimating in Ideal Hours, Ideal Days, Story Points, XPUnits, etc.

Approaches to Estimating

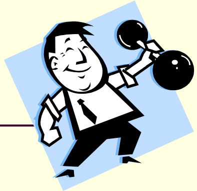
Anonymity of estimates

- Boehm and others recommend doing estimates anonymously
  - There are advantages
    - People speak more freely
    - Discussion won’t be dominated by a strong personality or the most senior developer
  - But there are also disadvantages
    - Tends to take longer
    - Less communication
      - Notes provided on written estimates are usually less than the give-and-take during a meeting
  - Goes against XP’s value of courage
- My recommendation
  - No anonymity needed, but use your judgment

Mountain Goat SOFTWARE

Approaches to Estimating

Exercise



Do a Wideband Delphi estimate of the tasks involved in selecting the install tool.

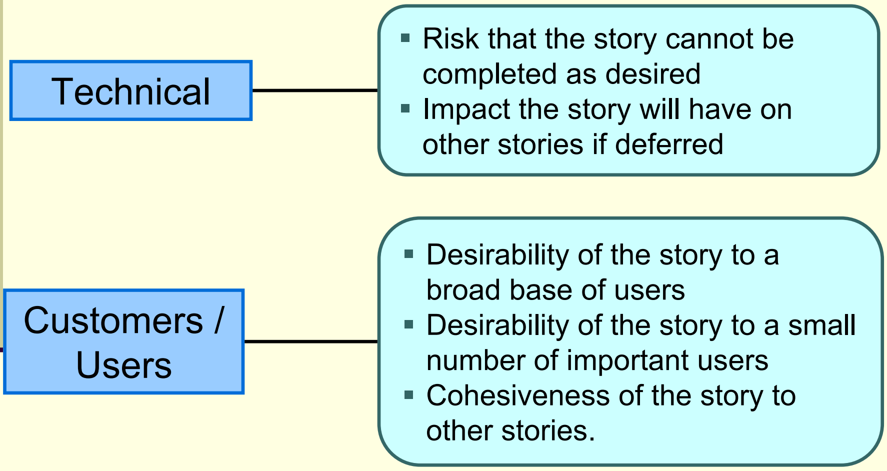
Mountain Goat SOFTWARE

Today’s agenda

- ☑ Introduction
  - ☑ What stories are
  - ☑ What stories are not
  - ☑ Why stories?
- ☑ The User and Customer
  - ☑ Who’s the user?
  - ☑ User roles and personas
- ☑ Gathering Stories
- ☑ Planning and Estimating
  - ☑ Why plans go wrong
  - ☑ Estimating user stories
  - Planning with user stories
- □ Case Study

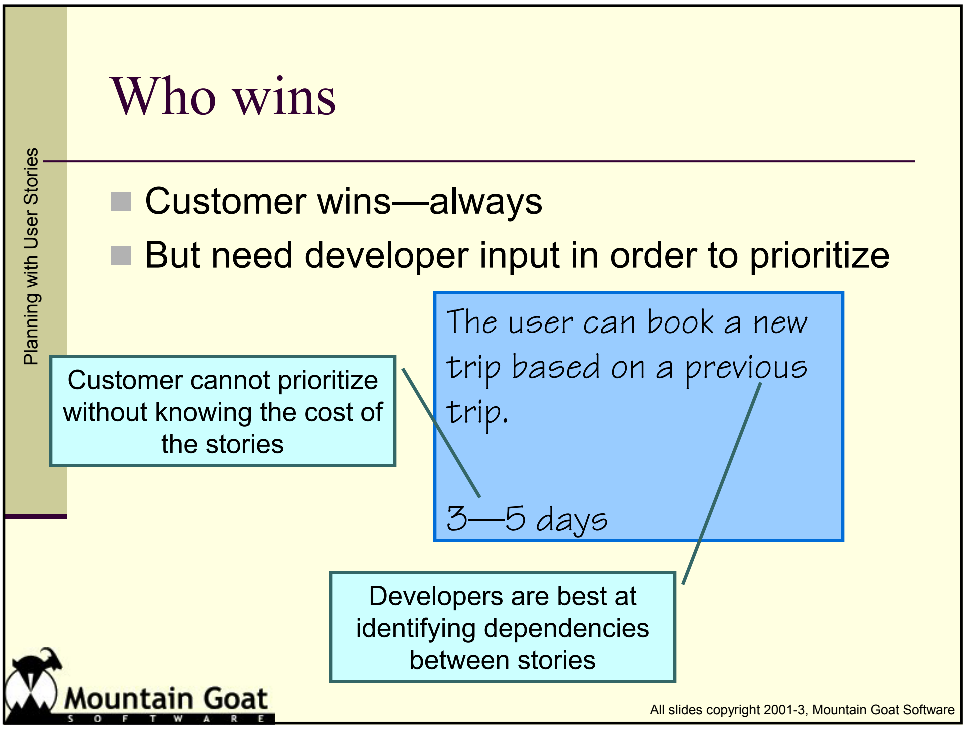
Planning with User Stories

Different dimensions to prioritization



Technical
- Risk that the story cannot be completed as desired
- Impact the story will have on other stories if deferred

Customers / Users
- Desirability of the story to a broad base of users
- Desirability of the story to a small number of important users
- Cohesiveness of the story to other stories.



Who wins

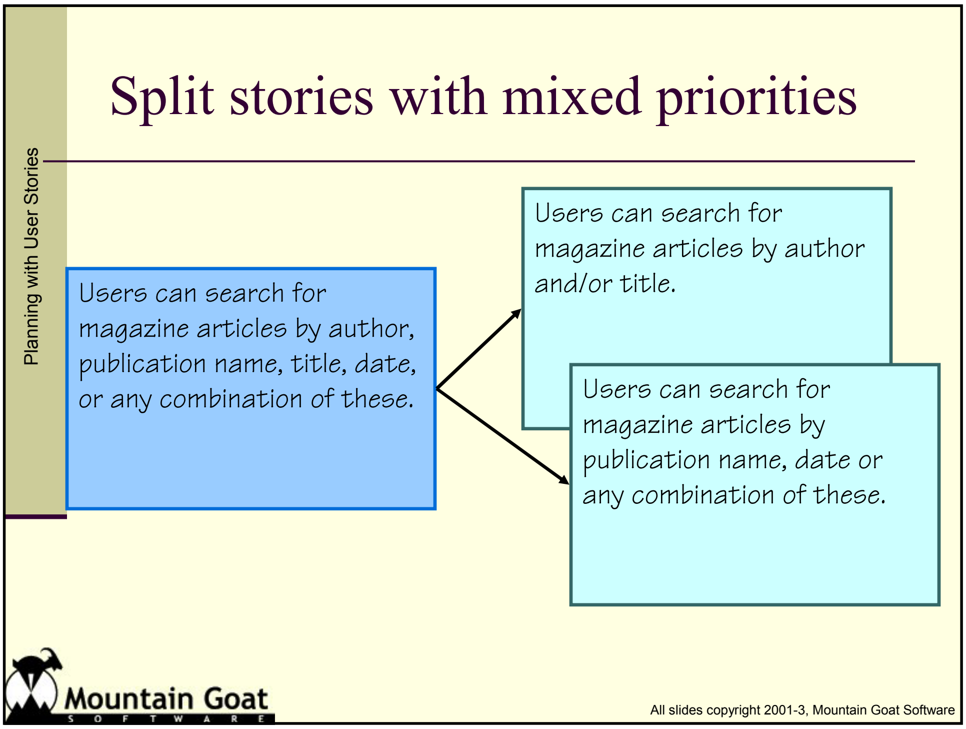
- Customer wins—always
- But need developer input in order to prioritize

The user can book a new trip based on a previous trip.

3—5 days

Customer cannot prioritize without knowing the cost of the stories

Developers are best at identifying dependencies between stories



Split stories with mixed priorities

Users can search for magazine articles by author, publication name, title, date, or any combination of these.

Users can search for magazine articles by author and/or title.

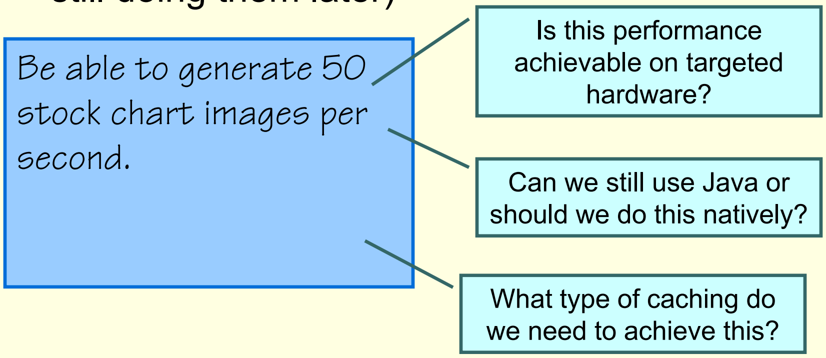
Users can search for magazine articles by publication name, date or any combination of these.

Risky stories vs. Juicy stories

- Agile is firmly in the camp of doing the “juicy bits” first
- But cannot totally ignore risk
  - If some stories are very risky, the developers need to tell the customer
- Example: Expectation Maximization

Infrastructural stories

- Infrastructural stories are usually best assessed by the risk of deferring them (but still doing them later)



Be able to generate 50 stock chart images per second.

Is this performance achievable on targeted hardware?

Can we still use Java or should we do this natively?

What type of caching do we need to achieve this?

From Story Points to Calendar Days

- Two steps:

1. Determine the size of the buffer
2. Use *velocity* to convert from Story Points to calendar days

Velocity represents a team’s rate of progress over a period of time.

How long should the buffer be?

- Simple rule
  - Use 50% of the unbuffered (50%) schedule
- More sophisticated, usually better

$$\sqrt{(w_1-a_1)^2+(w_2-a_2)^2+\cdots+(w_n-a_n)^2}$$

  - w = worst case
  - a = average case

Planning with User Stories

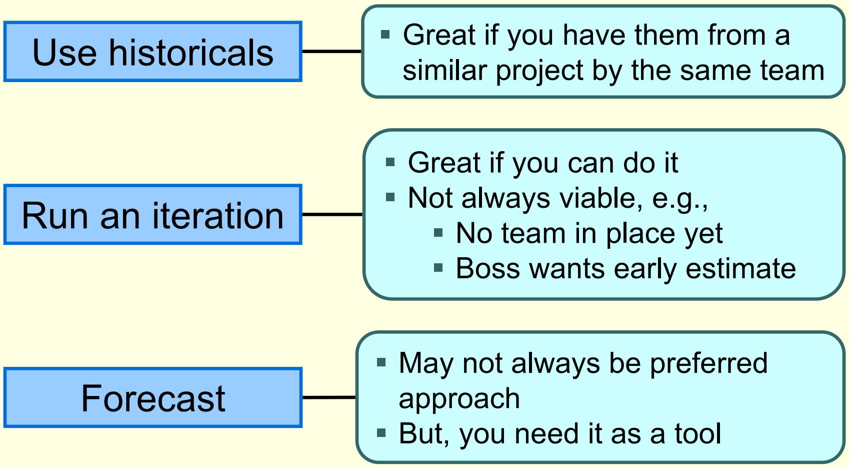
Sample buffer calculation

Story	50%	90%	(90%—50%)²
Story 1	2	5	9
Story 2	3	5	4
Story 3	1	1	0
Story 4	1	3	4
Story 5	5	8	9
Story 6	5	6	1
Total	17	28	27

$$Schedule = 17 + \sqrt{27} = 17 + 5.2 = 22$$

Planning with User Stories

Getting an initial velocity



Use historicals

- Great if you have them from a similar project by the same team

Run an iteration

- Great if you can do it
- Not always viable, e.g.,
  - No team in place yet
  - Boss wants early estimate

Forecast

- May not always be preferred approach
- But, you need it as a tool

Forecasting initial velocity

- Estimate each developer’s productivity relative to the archetypal Experienced Senior Programmer used in the estimates
- Considerations
  - Programming skill
  - Domain knowledge
  - Availability to actual code
  - Vacation

Example: forecasting initial velocity

Developer	Iteration 1	Iteration 2	Iteration 3	Thereafter
Susan	.5	.6	.7	.7
Ann	.5	.5	.5	.5
Randy	.2	.3	.4	.4
Clark		.2	.3	.4
Vlade	.5	.6	.7	.7
Chris	.8	.9	1.0	1.0
Total	**2.5**	**3.1**	**3.6**	**3.7**

Example: An unknown team

- If you don't know the team, make generic guesses
- Useful when you're planning a project the company might do down the road

Developer	Iteration 1	Iteration 2	Iteration 3	Thereafter
Programmer 1	.5	.6	.7	.7
Programmer 2	.5	.6	.7	.7
Programmer 3	.3	.4	.5	.5
Programmer 4	.3	.4	.5	.5
Total	**2.5**	**3.1**	**3.6**	**3.7**

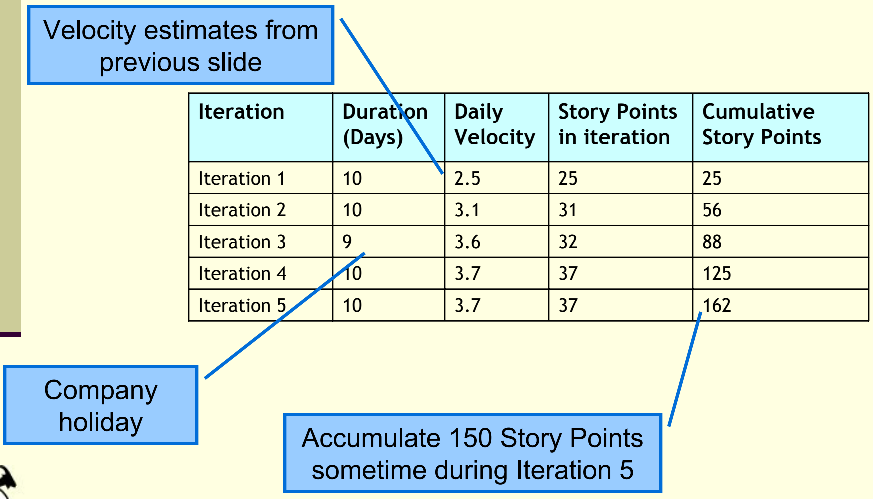
Full example of planning a release

Story	50%	90%	(90%—50%)²
Story 1	2	5	9
Story 2	3	5	4
…	…	…	0
Total	**117**	**200**	**1089**

$$117+\sqrt{1089}=117+33=150$$

Developer	Iteration 1	Iteration 2	Iteration 3	Thereafter
Susan	.5	.6	.7	.7
Ann	.5	.5	.5	.5
Randy	.2	.3	.4	.4
Clark		.2	.3	.4
Vlade	.5	.6	.7	.7
Chris	.8	.9	1.0	1.0
Total	**2.5**	**3.1**	**3.6**	**3.7**

Example, continued



Velocity estimates from previous slide

Iteration	Duration (Days)	Daily Velocity	Story Points in iteration	Cumulative Story Points
Iteration 1	10	2.5	25	25
Iteration 2	10	3.1	31	56
Iteration 3	9	3.6	32	88
Iteration 4	10	3.7	37	125
Iteration 5	10	3.7	37	162

Company holiday

Accumulate 150 Story Points sometime during Iteration 5

Communicating the estimate

- When communicating the estimate to management
  - Don’t talk about the project buffer
    - Don’t necessarily hide it
    - I include it in a document on the estimation approach, rather than in the estimate itself
  - Clearly state your assumptions
  - Stress that it will be refined
    - Then refine it!
    - Not fair to “refine” it only with a big slip at the end

Refining an estimate

What if only 22 story points got done?

Iteration	Duration (Days)	Daily Velocity	Story Points in iteration	Cumulative Story Points
Iteration 1	10	2.5	25	~~25~~ 22
Iteration 2	10	~~3.1~~	~~31~~ 23	~~56~~ 45
Iteration 3	9	~~3.6~~	~~32~~ 22	~~88~~ 66
Iteration 4	10	~~3.7~~	~~37~~ 25	~~125~~ 91
Iteration 5	10	~~3.7~~	~~37~~ 25	~~162~~ 116
Iteration 6	10		25	141
Iteration 7	10		25	166

- It’s OK to assume improvements over the *initial* iterations

Why agile planning works

- Agile planning is built on user stories
  - Traditional planning is built on tasks
- Stories are larger than the tasks
  - Because of this stories are less prone to systematic error
- Stories are more independent than tasks
  - Part of a story may be interdependent with another story; but not the entire story
- Continuous re-estimation and recalibration

Planning with User Stories

Why agile planning works, cont.

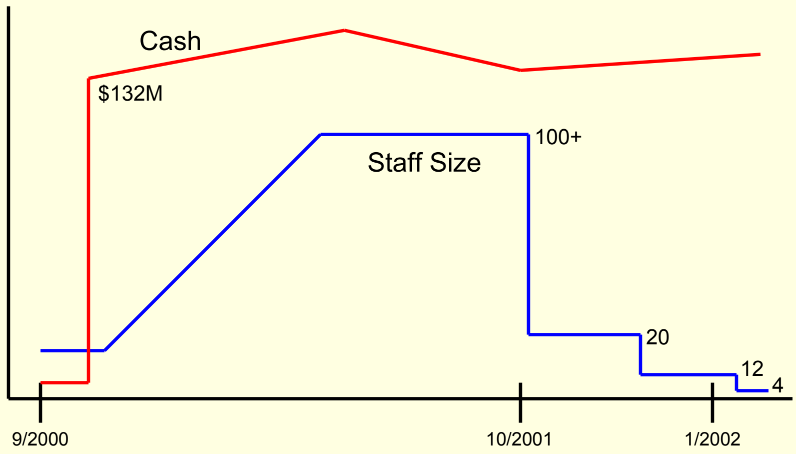
- No overall Gantt or PERT chart
  - Each day, each person picks what she'll do
    - Lateness doesn't pass down an agile schedule
    - Earliness does pass down
  - Naturally, there are some dependencies which cause exceptions to this
- No Student Syndrome
  - No Gantt chart saying what to do today and how long I have
  - Increased visibility through daily standup meetings and pair programming

Today's agenda

- ☑ Introduction
  - ☑ What stories are
  - ☑ What stories are not
  - ☑ Why stories?
- ☑ The User and Customer
  - ☑ Who's the user?
  - ☑ User roles and personas
- ☑ Gathering Stories
- ☑ Planning and Estimating
  - ☑ Why plans go wrong
  - ☑ Estimating user stories
  - ☑ Planning with user stories
- Case Study

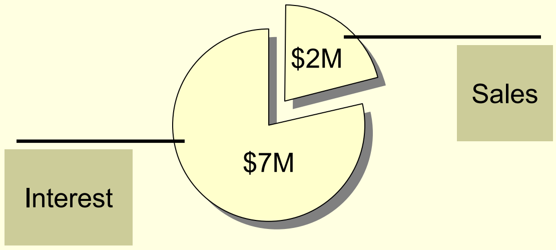
Case Study

Case Study—Cosmodemonic Biotech



Case Study

Revenue



Case Study

How we scaled down

- Took engineering group from 100 employees down to 25
- Then down to 12
  - 6 programmers
  - 4 testers
  - 1 manager
  - Me
- I had two weeks notice
- No one on the team was told what was coming

Case Study

Rewriting the client

- Client was most complex so we decided to rewrite it
- How could we verify we were rebuilding what our customers wanted?
  - Couldn’t use existing requirements
    - 2,000 pages of use cases
    - Too detailed, inconsistent, out of date
  - All analysts were gone
  - What about the test cases?

Case Study

Considering the test cases

- We had a really good test team
- They’d been writing tests all through development
- If the new client passed the old tests, all the requirements were fulfilled

Mountain Goat Software

Case Study

Two problems

- Tests were tied to the old user interface
- Like the use cases the tests were too detailed for everyday use
  - Had to read too many steps to understand them

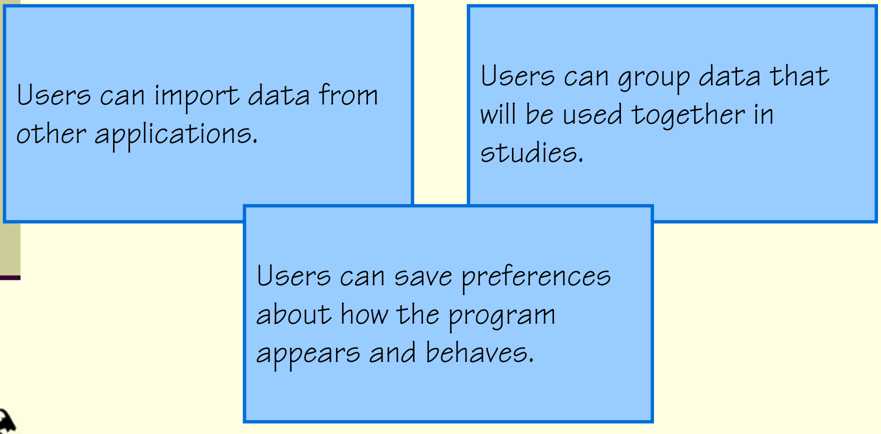
Step	Result
Open the formula editor	
Type floor(2 * age)	
Press “check formula”	Message “Formula is OK” is displayed
Type floor(2 * age	
Press “check formula”	Message “Invalid formula” is displayed
Type floor 2 * age)	
Press “check formula”	Message “Invalid formula” is displayed
Type floor ((2 * age))	
Press “check formula”	Message “Formula is OK” is displayed

Mountain Goat Software

Case Study

The solution—User Stories

- Some example stories:



Users can import data from other applications.

Users can group data that will be used together in studies.

Users can save preferences about how the program appears and behaves.

Case Study

How we got there

- Remaining testers went through existing test cases
  - Distilled each test case into 0 or more stories
- Programmers started coding without any stories
  - Then would grab a collection of stories for a functional area when the tester was done
- Testers tested each story when programmers said an area was done
- Ended up with 1,400 stories
  - Contrast that with over 2,000 pages of use cases
- We used Excel because we weren’t collocated

Case Study

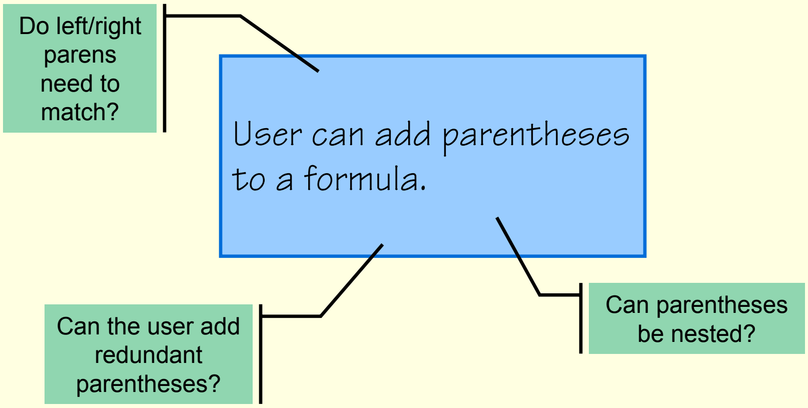
An example

Step	Result
Open the formula editor	
Type floor(2 * age)	
Press “check formula”	Message “Formula is OK” is displayed
Type floor(2 * age	
Press “check formula”	Message “Invalid formula” is displayed
Type floor 2 * age)	
Press “check formula”	Message “Invalid formula” is displayed
Type floor ((2 * age))	
Press “check formula”	Message “Formula is OK” is displayed

User can add parentheses to a formula.

Case Study

A huge amount of detail was left out



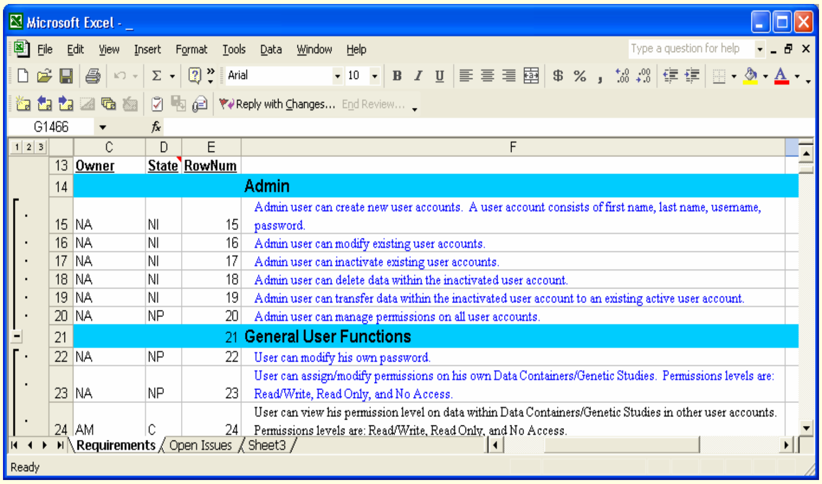
User can add parentheses to a formula.

Do left/right parens need to match?

Can the user add redundant parentheses?

Can parentheses be nested?

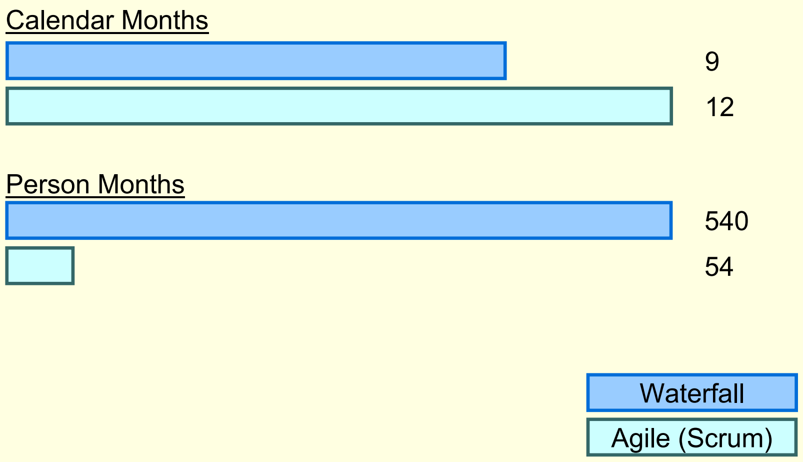
Our only user story documentation



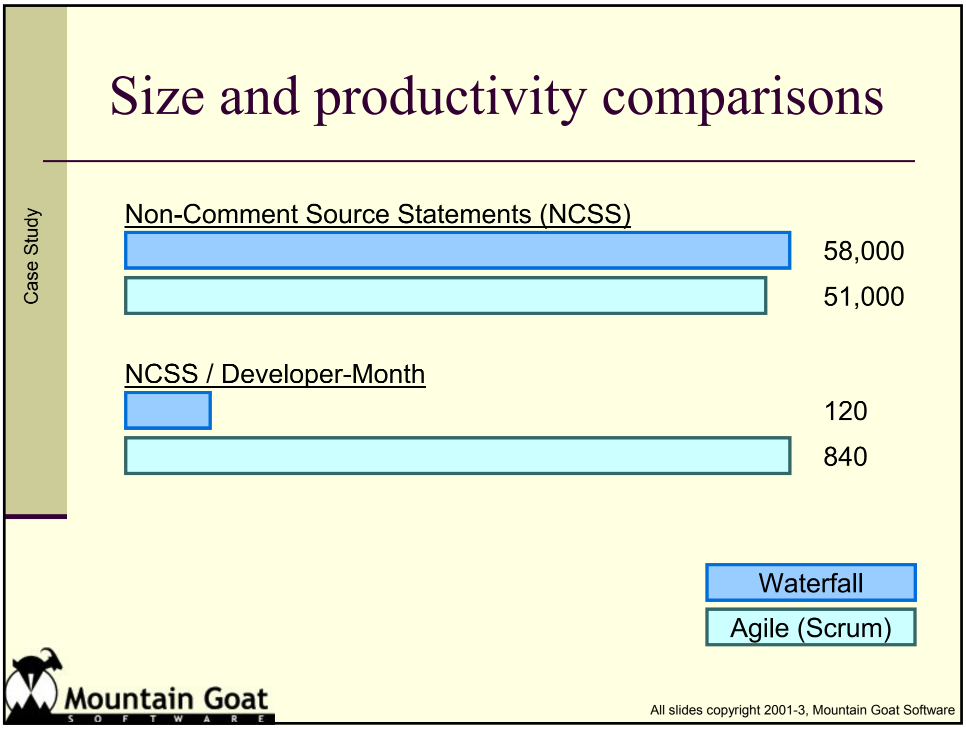
	C	D	E	F
13	Owner	State	RowNum	
14				Admin
15	NA	NI	15	Admin user can create new user accounts. A user account consists of first name, last name, username, password.
16	NA	NI	16	Admin user can modify existing user accounts.
17	NA	NI	17	Admin user can inactivate existing user accounts.
18	NA	NI	18	Admin user can delete data within the inactivated user account.
19	NA	NI	19	Admin user can transfer data within the inactivated user account to an existing active user account.
20	NA	NP	20	Admin user can manage permissions on all user accounts.
21			21	General User Functions
22	NA	NP	22	User can modify his own password.
23	NA	NP	23	User can assign/modify permissions on his own Data Containers/Genetic Studies. Permissions levels are: Read/Write, Read Only, and No Access.
24	AM	C	24	User can view his permission level on data within Data Containers/Genetic Studies in other user accounts. Permissions levels are: Read/Write, Read Only, and No Access.

Mountain Goat SOFTWARE

Effort comparisons



Mountain Goat SOFTWARE



Case Study

Size and productivity comparisons

Non-Comment Source Statements (NCSS)

58,000

51,000

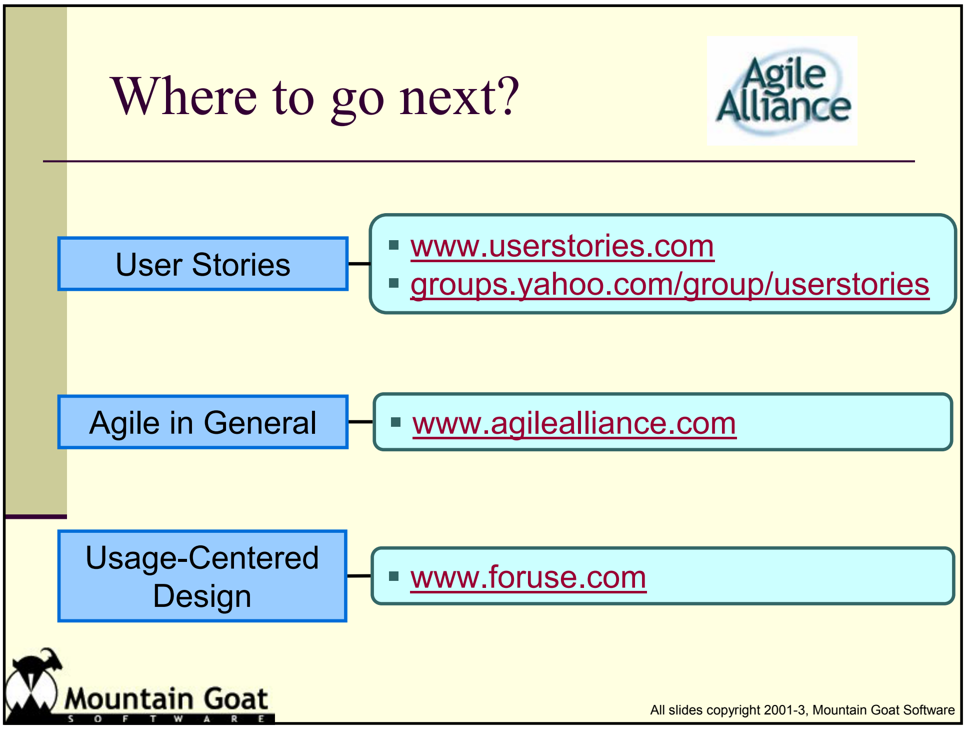
NCSS / Developer-Month

120

840

Waterfall

Agile (Scrum)

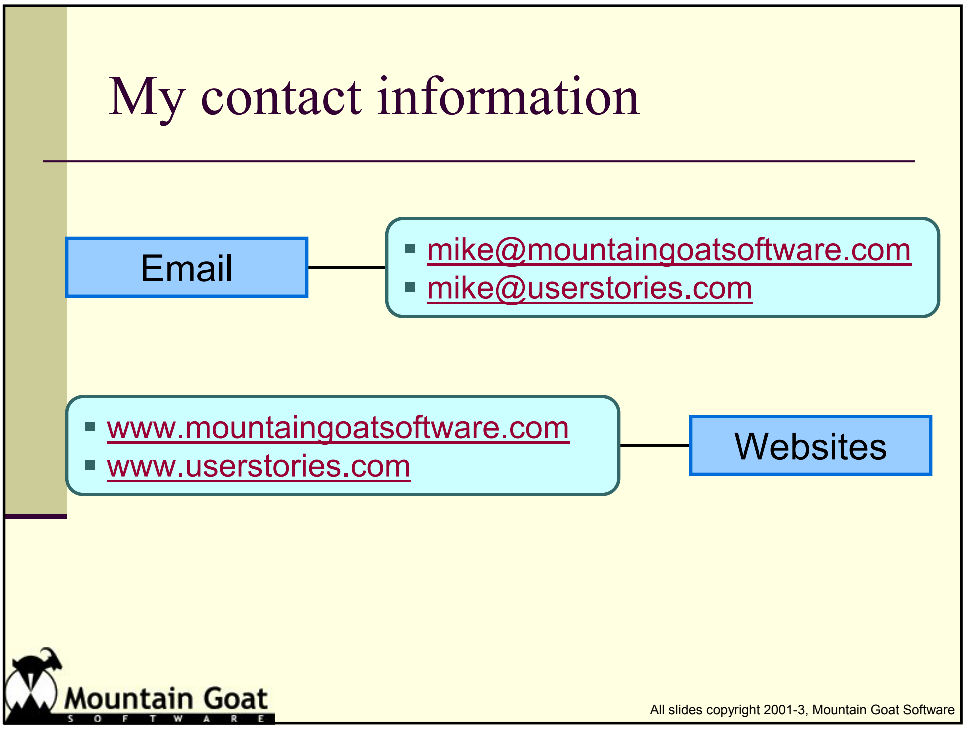


Where to go next?

User Stories
- www.userstories.com
- groups.yahoo.com/group/userstories

Agile in General
- www.agilealliance.com

Usage-Centered Design
- www.foruse.com



My contact information

Email

- mike@mountaingoatsoftware.com
- mike@userstories.com

- www.mountaingoatsoftware.com
- www.userstories.com

Websites

Mountain Goat SOFTWARE