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
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

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.
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 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
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

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”?
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 are we building?

<table>
<thead>
<tr>
<th>IEEE Specs</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. The product shall have a gas engine.</td>
</tr>
<tr>
<td>7. The product shall have four wheels.</td>
</tr>
<tr>
<td>1. The product shall have a rubber tire mounted to each wheel.</td>
</tr>
<tr>
<td>8. The product shall have a steering wheel.</td>
</tr>
<tr>
<td>9. The product shall have a steel body.</td>
</tr>
</tbody>
</table>

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

The product
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.

**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.
### 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
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

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...
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

So, why user stories?

- Written words are a shaky foundation
  - If requirements are written down
    - 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 user stories?

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

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
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

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

ISV

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

In-House Development
Who’s the User?

User proxies

- The Users’ Manager
- Salespersons
- Customer
- The Marketing Group
- Former Users
- Trainers and tech Support
- Development Manager
- Domain Experts

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.”
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

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
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?

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
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

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
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

“Who’s 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
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

- 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

- **Mary**
  - Frequent flier who never knows where she'll be

- **Jim**
  - Frequent flier who always to the same place

- **Laura**
  - Wants to schedule her family's annual vacation

- **Dominic**
  - Hotel chain Vice President; wants to monitor reservations

- **Howard**
  - Mary's assistant; books her reservations

- **Frequent Flier**
- **Repeat Traveler**
- **Infrequent Vacation Planner**

User Roles and Personas

User story / role matrix

<table>
<thead>
<tr>
<th>User Roles and Personas</th>
<th>Book Simple Trips</th>
<th>Save and reuse trips</th>
<th>Booking reports</th>
<th>Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent Flier</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Repeat Traveler</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheduler</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insider</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Vacation Planner</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
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

Document the user role

User Role: Infrequent Vacation Planner

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.
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).
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”

A sample persona

Jim lives in a 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

- 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


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

- “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

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

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

- **Hot Deal Details**
  - Location info
  - 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
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

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

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
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?

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
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:

<table>
<thead>
<tr>
<th>Task</th>
<th>Local Safety</th>
</tr>
</thead>
</table>

So this happens:

<table>
<thead>
<tr>
<th>Local Safety</th>
<th>Task</th>
</tr>
</thead>
</table>

My trip to the airport

- 1:05 Estimate
- 1:45 Buffer to 90%
- Total = 2:50 minutes
Trip to the airport with a project buffer

Why Plans Go Wrong

1 45
5 7 53

50% Estimate
Buffer
Time = 1:05

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
  - \(50\% \times 50\% \times 50\% \times 50\% \times 50\% = 3.125\%\)

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

All slides copyright 2001-3, Mountain Goat Software
Exercise

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

Estimating stories

- 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
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.'”

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
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)

“How long will this take?”

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

![Calendar Image](calendar.png)
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
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

- Gut feel
- Analogy
- Decomposition
- Wideband Delphi
Gut feel

- Good as a reasonableness check

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
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

Wideband Delphi—an example

<table>
<thead>
<tr>
<th>Estimator</th>
<th>Round 1</th>
<th>Round 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Jody</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Ann</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Sherri</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

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

Exercise

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

Different dimensions to prioritization

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

- 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

Customer cannot prioritize without knowing the cost of the stories

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

3—5 days

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
Sample buffer calculation

<table>
<thead>
<tr>
<th>Story</th>
<th>50%</th>
<th>90%</th>
<th>(90%—50%)²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story 1</td>
<td>2</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Story 2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Story 3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Story 4</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Story 5</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Story 6</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>17</td>
<td>28</td>
<td>27</td>
</tr>
</tbody>
</table>

\[ Schedule = 17 + \sqrt{27} = 17 + 5.2 = 22 \]

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

<table>
<thead>
<tr>
<th>Developer</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Thereafter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>.5</td>
<td>.6</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>Ann</td>
<td>.5</td>
<td>.5</td>
<td>.5</td>
<td>.5</td>
</tr>
<tr>
<td>Randy</td>
<td>.2</td>
<td>.3</td>
<td>.4</td>
<td>.4</td>
</tr>
<tr>
<td>Clark</td>
<td>.2</td>
<td>.3</td>
<td>.3</td>
<td>.4</td>
</tr>
<tr>
<td>Vlade</td>
<td>.5</td>
<td>.6</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>Chris</td>
<td>.8</td>
<td>.9</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Total</td>
<td>2.5</td>
<td>3.1</td>
<td>3.6</td>
<td>3.7</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Developer</th>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Thereafter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer 1</td>
<td>.5</td>
<td>.6</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>Programmer 2</td>
<td>.5</td>
<td>.6</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>Programmer 3</td>
<td>.3</td>
<td>.4</td>
<td>.5</td>
<td>.5</td>
</tr>
<tr>
<td>Programmer 4</td>
<td>.3</td>
<td>.4</td>
<td>.5</td>
<td>.5</td>
</tr>
<tr>
<td>Total</td>
<td>2.5</td>
<td>3.1</td>
<td>3.6</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Full example of planning a release

<table>
<thead>
<tr>
<th>Story</th>
<th>50%</th>
<th>90%</th>
<th>(90%—50%)^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story 1</td>
<td>2</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>Story 2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>117</td>
<td>200</td>
<td>1089</td>
</tr>
</tbody>
</table>

$$117 + \sqrt{1089} = 117 + 33 = 150$$
Example, continued

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Duration (Days)</th>
<th>Daily Velocity</th>
<th>Story Points in Iteration</th>
<th>Cumulative Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>10</td>
<td>2.5</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>10</td>
<td>3.1</td>
<td>31</td>
<td>56</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>9</td>
<td>3.6</td>
<td>32</td>
<td>88</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>10</td>
<td>3.7</td>
<td>37</td>
<td>125</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>10</td>
<td>3.7</td>
<td>37</td>
<td>162</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Duration (Days)</th>
<th>Daily Velocity</th>
<th>Story Points in Iteration</th>
<th>Cumulative Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>10</td>
<td>2.5</td>
<td>25</td>
<td>22</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>10</td>
<td>3</td>
<td>23</td>
<td>45</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>9</td>
<td>3.1</td>
<td>22</td>
<td>66</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>10</td>
<td>3</td>
<td>25</td>
<td>91</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>10</td>
<td>3</td>
<td>25</td>
<td>116</td>
</tr>
<tr>
<td>Iteration 6</td>
<td>10</td>
<td>25</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>Iteration 7</td>
<td>10</td>
<td>25</td>
<td>166</td>
<td></td>
</tr>
</tbody>
</table>

What if only 22 story points got done?

- 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
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—Cosmodemonic Biotech

Staff Size

Cash

$132M


306 44

Revenue

Sales

Interest

$2M

$7M

$2M

$7M
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

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?
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

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the formula editor</td>
<td></td>
</tr>
<tr>
<td>Type floor(2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Formula is OK” is displayed</td>
</tr>
<tr>
<td>Type floor(2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Invalid formula” is displayed</td>
</tr>
<tr>
<td>Type floor (2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Invalid formula” is displayed</td>
</tr>
<tr>
<td>Type floor ((2 * age))</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Formula is OK” is displayed</td>
</tr>
</tbody>
</table>
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.

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
An example

<table>
<thead>
<tr>
<th>Step</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the formula editor</td>
<td></td>
</tr>
<tr>
<td>Type floor(2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Formula is OK” is displayed</td>
</tr>
<tr>
<td>Type floor(2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Invalid formula” is displayed</td>
</tr>
<tr>
<td>Type floor 2 * age</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Invalid formula” is displayed</td>
</tr>
<tr>
<td>Type floor (2 * age)</td>
<td></td>
</tr>
<tr>
<td>Press “check formula”</td>
<td>Message “Formula is OK” is displayed</td>
</tr>
</tbody>
</table>

User can add parentheses to a formula.

A huge amount of detail was left out

Do left/right parens need to match?

User can add parentheses to a formula.

Can the user add redundant parentheses?

Can parentheses be nested?
Our only user story documentation

Case Study

**Admin**
- Admin user can create new user accounts. A new account requires a first name, last name, username, password.
- Admin user can modify existing user accounts.
- Admin user can deactivate existing user accounts.
- Admin user can delete data within the authenticated user account.
- Admin can transfer data within the authenticated user account to an existing active user account.
- Admin user can manage connections on all user accounts.

**General User Functions**
- User can log in with own password.
- User can navigate to password reset page.
- Permissions to access Data Collection/Security Settings.
- Permissions to view, Read, Write, Read Only, and Edit Access.
- Permissions to perform login and logout.
- Data Collection/Security Settings as otherwise account.
- Permissions to View, Read, Write, and Edit Access.

**Effort comparisons**

<table>
<thead>
<tr>
<th>Calendar Months</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person Months</td>
<td>540</td>
</tr>
</tbody>
</table>

**Waterfall**

**Agile (Scrum)**
Size and productivity comparisons

<table>
<thead>
<tr>
<th></th>
<th>Waterfall</th>
<th>Agile (Scrum)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-Comment Source Statements (NCSS)</strong></td>
<td>58,000</td>
<td>51,000</td>
</tr>
<tr>
<td><strong>NCSS / Developer-Month</strong></td>
<td>120</td>
<td>840</td>
</tr>
</tbody>
</table>

Where to go next?

- **User Stories**
  - [www.userstories.com](http://www.userstories.com)
  - [groups.yahoo.com/group/userstories](http://groups.yahoo.com/group/userstories)

- **Agile in General**
  - [www.agilealliance.com](http://www.agilealliance.com)

- **Usage-Centered Design**
  - [www.foruse.com](http://www.foruse.com)
My contact information

- **Email**
  - mike@mountaingoatsoftware.com
  - mike@userstories.com

- **Websites**
  - www.mountaingoatsoftware.com
  - www.userstories.com