Mike Cohn - background

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

© Mountain Goat Software, LLC
Imagine...

- That you’re fed up with software development as a career
- And you decide to go into the landscaping business
- Your first job is moving this pile of rock from the front of my house to the back

How might you estimate this?

- One way:
  - Look at the pile of rock and estimate how many wheelbarrow loads it represents
  - After an hour, see how many wheelbarrow loads you’ve moved then extrapolate the total duration

- I think that’s 80 wheelbarrow loads
- After an hour I’ve moved 20 loads
- So, I’ll be done in a total of 4 hours
My landscaping

- An iteration is a short, constrained period of time
- Typically 1-4 weeks

Velocity is the amount of work planned or completed in an iteration.

A release typically comprises more than one iteration.
The planning onion

- Agile teams plan at the innermost three levels.
- Others (on the team in the company) plan at the outer levels.

Relating the different planning levels

### Product Backlog

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>3</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>5</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>5</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>2</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>2</td>
</tr>
</tbody>
</table>

### Iteration Backlog

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the UI</td>
<td>8</td>
</tr>
<tr>
<td>Write test fixture</td>
<td>6</td>
</tr>
<tr>
<td>Code middle tier</td>
<td>12</td>
</tr>
<tr>
<td>Write tests</td>
<td>5</td>
</tr>
<tr>
<td>Automate tests</td>
<td>4</td>
</tr>
</tbody>
</table>

“Yesterday I started on the UI; I should finish before the end of today.”
Product, release, sprint planning

We’ll focus here today

Release Plan

<table>
<thead>
<tr>
<th>Sprint</th>
<th>Task A</th>
<th>8 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Task B</td>
<td>16 hours</td>
</tr>
<tr>
<td>2</td>
<td>Task C</td>
<td>5 hours</td>
</tr>
<tr>
<td>3</td>
<td>Task D</td>
<td>8 hours</td>
</tr>
<tr>
<td>4-7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Agenda

☐ Estimating

☐ Release planning
**Story points**

- Probably the most commonly used estimating unit among agile teams today
  - Name is derived from agile teams commonly expressing requirements as “user stories”
- Based on a combination of the size and complexity of the work
- Unitless but numerically relevant estimates
  - A 10-point user story is expected to take twice as long as a 5-point user story

---

**Dog points**

Assign “dog points” to the following dogs

Labrador retriever
Dachshund
Great Dane
Terrier
German Shepherd
Poodle
St. Bernard
Bulldog
Consider these two piles of work

What story point values might we put on these?

Three key advantages

- Estimating in story points:
  1. Forces the use of relative estimating
     - Studies have shown we’re better at this†
  2. Focuses us on estimating the size, not the duration
     - We derive duration empirically by seeing how much we complete per iteration
  3. Puts estimates in units that we can add together
     - Time based estimates are not additive

Comparing apples to apples

Product Backlog

<table>
<thead>
<tr>
<th>Story</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>30</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>50</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>50</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>20</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>20</td>
</tr>
</tbody>
</table>

Sprint Backlog

<table>
<thead>
<tr>
<th>Task</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the UI</td>
<td>8</td>
</tr>
<tr>
<td>Write test fixture</td>
<td>6</td>
</tr>
<tr>
<td>Code middle tier</td>
<td>12</td>
</tr>
<tr>
<td>Write tests</td>
<td>5</td>
</tr>
<tr>
<td>Automate tests</td>
<td>4</td>
</tr>
</tbody>
</table>

“Yesterday I started on the UI; I should finish before the end of today.”

Planning poker

- An iterative approach to estimating

Steps

- Each estimator is given a deck of cards, each card has a valid estimate written on it
- Customer/Product owner reads a story and it’s discussed briefly
- Each estimator selects a card that’s his or her estimate
- Cards are turned over so all can see them
- Discuss differences (especially outliers)
- Re-estimate until estimates converge
Planning poker - an example

Why planning poker works

- Those who will do the work, estimate the work
- Estimators are required to justify estimates
- Focuses most estimates within an approximate one order of magnitude

---

Why planning poker works

- Combining of individual estimates\(^6\) through group discussion\(^7\) leads to better estimates
- Emphasizes relative rather than absolute estimating
- Estimates are constrained to a set of values so we don’t waste time in meaningless arguments
- Everyone’s opinion is heard
- It’s quick and fun


Reduces likelihood of anchoring

**Control group**
- Given a product spec
  - 456 hours

**High anchor group**
- Given the same product spec
- Told the customer thinks 500 hours is a reasonable estimate but that
  - The customer knows very little about the implications of his spec on the estimate
  - You shouldn’t let his number influence you
  - 555 hours

**Low anchor group**
- Same as high but customer thinks 50 hours
  - 99 hours

Source: How to avoid impact from irrelevant and misleading information on your cost estimates, Magne Jørgensen and Stein Grimstad, Simula Research Laboratory, Simula Research Labs Estimation Seminar, Oslo, Norway 2006.
Before Planning Poker

Overcommit by 186 hours per iteration

Graphs show hours of task estimate error

A 43% reduction in task estimate error per iteration

After Planning Poker

Overcommit by 106 hours per iteration

Agenda

- Estimating
- Release planning
Release planning

Purpose
To answer questions such as:
• How much will be done by 30 June?
• When can we ship with this set of features?
• How many people or teams should be on this project?

Inputs
• Velocity
• The length of the project
• Prioritized product backlog

An example with velocity=14
Updating the release plan

• Use multiple views of observed velocity

Extrapolate from velocity

Assume:
There are five sprints left.

At our slowest velocity we’ll finish here (5×28)
At our long-term average we’ll finish here (5×33)
At current velocity we’ll finish here (5×36)
Fixed-date planning

How much can I get by <date>?

1. Determine how many sprints you have
2. Estimate velocity as a range
3. Multiply low velocity \times \text{number of sprints}
   - Count off that many points
   - These are “Will Have” items
4. Multiply high velocity \times \text{number of sprints}
   - Count off that many more points
   - These are “Might Have” items

Fixed-date planning: an example

<table>
<thead>
<tr>
<th>Desired release date</th>
<th>30 June</th>
</tr>
</thead>
<tbody>
<tr>
<td>Today’s Date</td>
<td>1 January</td>
</tr>
<tr>
<td>Number of sprints</td>
<td>6 (monthly)</td>
</tr>
<tr>
<td>Low velocity</td>
<td>15</td>
</tr>
<tr>
<td>High velocity</td>
<td>20</td>
</tr>
</tbody>
</table>

Will have
6 \times 15

Might have
6 \times 20

Won’t have
Fixed-date contracting

Will have

6x15

Might have

6x20

Won’t have

If you write a contract for just the will haves:
- You won’t likely win the contract
- But you’ll probably make money if you do

If you write a contract that includes the might haves:
- You will likely win the contract
- But probably not make money on it

It’s a risk issue
Where do you want to be?

Mike Cohn contact info

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