Agile Estimating and Planning

Mike Cohn

10 November 2009

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
Scrum

Sprint goal
Return

Cancel
Coupons
Gift wrap
Product backlog

What’s a good plan?

• A good plan is one that supports reliable decision-making
• Will go from
  • We’ll be done in the third quarter
  • We’ll be done in August
  • We’ll be done August 18th

“It’s better to be roughly right than precisely wrong.”
~John Maynard Keynes
What makes planning agile?

- Is more focused on planning than the plan
- Encourages change
- Results in plans that are easily changed
- Is spread throughout the project

Product, release, iteration planning

Release 1 | Release 2 | Release 3

Release Plan

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iterations 4–7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We’ll focus here today

<table>
<thead>
<tr>
<th>Task</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task A</td>
<td>8 hours</td>
</tr>
<tr>
<td>Task B</td>
<td>16 hours</td>
</tr>
<tr>
<td>Task C</td>
<td>5 hours</td>
</tr>
<tr>
<td>Task D</td>
<td>8 hours</td>
</tr>
</tbody>
</table>
Agenda

- Product backlog estimation units
  - Story points
  - Ideal time
- Techniques for estimating
- Iteration planning
- Release planning

Estimating in Story Points
How long will it take...

• ...to read the latest Harry Potter book?
• ...to drive to Minneapolis?
Estimate size; derive duration

- Traditional and agile measure size differently

Traditional measures of size
- Lines of Code
- Function Points

Agile measures of size
- Story points
- Ideal days
Story points

- The “bigness” of a task
- Influenced by
  - How hard it is
  - How much of it there is
- Relative values are what is important:
  - A login screen is a 2.
  - A search feature is an 8.
- Points are unit-less

As a user, I want to be able to have some but not all items in my cart gift wrapped.

5

Dog points

Assign “dog points” to the following breeds:

- Labrador retriever
- Dachshund
- Great Dane
- Poodle
- German Shepherd
- Terrier
- St. Bernard
- Bulldog
One order of magnitude

- We’re good over one order of magnitude
- So think about where to place it on your product backlog

[Diagram showing a scale from 1 to 10 with a box labeled "A typo" on the left and "The largest new feature" on the right]

Estimating in Ideal Time
Ideal time

- How long something would take if
  - it’s all you worked on
  - you had no interruptions
  - and everything you need is available
- The ideal time of a football game is 60 minutes
  - Four 15-minute quarters
- The elapsed time is much longer (3+ hours)

Ideal time vs. elapsed time

- It’s easier to estimate in ideal time
- It’s too hard to estimate directly in elapsed time
  - Need to consider all the factors that affect elapsed time at the same time you’re estimating
Comparing the approaches

- Story points help drive cross-functional behavior
- Story point estimates do not decay
- Story points are a pure measure of size
- Estimating in story points is typically faster
- My ideal days cannot be added to your ideal days
- Ideal days are easier to explain outside the team
- Ideal days are easier to estimate at first

Three levels of planning...

Release plan

Iteration plan
- Daily plan
- Daily plan
- Daily plan

Iteration plan
- Daily plan
- Daily plan
- Daily plan
...three levels of precision

<table>
<thead>
<tr>
<th>Product Backlog</th>
<th>Iteration Backlog</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>Code the UI</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>Write test fixture</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>Code middle tier</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>Write tests</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td>As a frequent flyer, I want to...</td>
<td>Automate tests</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

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

What I usually do

- I prefer story points
- ...but they make some teams uncomfortable, so I’ll
  - Start with ideal time
    - Gives the team a nice foundation for the initial stories
    - Helps team get started
  - Define “1 story point = 1 ideal day”
  - Then
    - Gradually convert team to thinking in unit-less story points
    - “This story is like that story.”
    - Stop talking about how long it will take
Estimate by analogy

- Comparing a user story to others
  - “This story is like that story, so its estimate is what that story’s estimate was.”
- Don’t use a single gold standard
- Triangulate instead
  - Compare the story being estimated to multiple other stories
Triangulation

- Confirm estimates by comparing the story to multiple other stories.
- Group like-sized stories on table or whiteboard

![](image)

Disaggregation

- Breaking a big story into smaller stories or tasks
  - You know how long the smaller tasks take
  - So, disaggregating to something you know lets you estimate something bigger you don’t know
- Sometimes very useful
- But disaggregating too far causes problems
  - Forgotten tasks
How much effort?

- A little effort helps a lot
- A lot of effort only helps a little more

Use the right units

- Can you distinguish a 1-point story from a 2?
- How about a 17 from an 18?
- Use a set of numbers that make sense; I like:
  - 1, 2, 3, 5, 8, 13, 20, 40, 100
- Stay mostly in a 1-10 range
- Nature agrees:
  - Musical tones and volume are distinguishable on a logarithmic scale
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

<table>
<thead>
<tr>
<th>Estimator</th>
<th>Round 1</th>
<th>Round 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Susan</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Vadim</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Ann</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Chris</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>
Estimate these

<table>
<thead>
<tr>
<th>Product backlog item</th>
<th>Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read a high-level, 10-page overview of agile software development in <em>People</em> magazine.</td>
<td></td>
</tr>
<tr>
<td>Read a densely written 5-page research paper about agile software development in an academic journal.</td>
<td></td>
</tr>
<tr>
<td>Write the product backlog for a simple eCommerce site that sells only clocks.</td>
<td></td>
</tr>
<tr>
<td>Recruit, interview, and hire a new member for your team.</td>
<td></td>
</tr>
<tr>
<td>Create a 60-minute presentation about agile software development for your coworkers.</td>
<td></td>
</tr>
<tr>
<td>Wash and wax your boss’ Porsche.</td>
<td></td>
</tr>
<tr>
<td>Read a 150-page book on agile software development.</td>
<td></td>
</tr>
<tr>
<td>Write an 8-page summary of this conference for your boss.</td>
<td></td>
</tr>
</tbody>
</table>

Why planning poker works

- Those who will do the work, estimate the work\(^1\)
- Estimators are required to justify estimates\(^2, 3\)
- Focuses most estimates within an approximate one order of magnitude\(^4, 5\)


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


Iteration Planning

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></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>

Creating this is iteration planning
Two approaches

- Velocity-driven iteration planning
  - "We finished 15 story points last time, let’s plan on 15 story points this time.”
  - Very unreliable in what will be accomplished during an iteration
    - Velocity is mostly useful over the long term
- Commitment-driven iteration planning

Commitment-driven iteration planning

- Discuss the highest priority item on the product backlog
- Decompose it into tasks
- Estimate each task
  - Whole team estimates each task
- Ask ourselves, “Can we commit to this?”
  - If yes, see if we can add another backlog item
  - If not, remove this item but see if we can add another smaller one
Estimate availability

<table>
<thead>
<tr>
<th>Person</th>
<th>Hours per Day</th>
<th>Hours per Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sergey</td>
<td>4-6</td>
<td>40-60</td>
</tr>
<tr>
<td>Yuri</td>
<td>5-7</td>
<td>50-70</td>
</tr>
<tr>
<td>Carina</td>
<td>2-3</td>
<td>20-30</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>110-160</strong></td>
</tr>
</tbody>
</table>

It looks something like this

As a user, I want ...

2

• Code the abc class (8 hours)
• Code the user interface (4)
• Write test fixtures (4)
• Code the xyz class (6)
• Update performance tests (4)

Team can commit, so they continue...

As a user, I want ...

3

• Prototype the UI (8 hours)
• Demo UI to 3 outside users (3)
• Code new UI (12)
• Update documentation (3)
A caution

- The purpose of the iteration planning meeting is to arrive at a commitment to an iteration goal or set of product backlog items.
- The purpose of the meeting is not to come up with a list of tasks and hours.
- The tasks and estimates are a tool for determining what we can commit to.
Release planning

Release planning meeting

Release plan

- Iteration 1
- Iteration 2
- Iteration 3
- Iterations 4–7
Velocity

- To do a release plan, you need to know or have an estimate of velocity
- Three ways to get velocity:
  1. Use historical averages
  2. Run 1-2 iterations and see what you get
  3. Forecast it
- Should be expressed as a range
  - Size of range depends on familiarity of team, domain, and technologies

Forecasting velocity

- Just like commitment-driven iteration planning
  - Estimate available hours for the iteration
  - Repeat until full:
    - Pick a story, break into tasks, estimate each task
An example

- Estimating available hours

<table>
<thead>
<tr>
<th>Person</th>
<th>Hours per Day</th>
<th>Hours per Iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sergey</td>
<td>4-6</td>
<td>40-60</td>
</tr>
<tr>
<td>Yuri</td>
<td>5-7</td>
<td>50-70</td>
</tr>
<tr>
<td>Carina</td>
<td>2-3</td>
<td>20-30</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>110-160</td>
</tr>
</tbody>
</table>

An example

At 110-160 available hours per iteration, what is the team’s velocity?

<table>
<thead>
<tr>
<th>Code the UI</th>
<th>Code the UI</th>
<th>Code the UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Write tests</th>
<th>Write tests</th>
<th>Write tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

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

© Mountain Goat Software, LLC
Predicting release contents

- Determine your median velocity
- Put a 90% confidence interval around it
- Predicts “best case” and “worst case”

An example

<table>
<thead>
<tr>
<th># of Historical Sprints</th>
<th>n^{th} Highest &amp; Lowest Sprint to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>26</td>
<td>9</td>
</tr>
</tbody>
</table>

Median

90% confidence interval
Extrapolate from velocity

Assume:
There are five iterations left.

- We’ll almost certainly get here \((5 \times 34)\)
- At our median velocity we’ll get here \((5 \times 39)\)
- The most we could realistically expect \((5 \times 41)\)

Upcoming public classes

<table>
<thead>
<tr>
<th>Date</th>
<th>What</th>
<th>Where</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 1–2</td>
<td>Certified ScrumMaster Certified Scrum Product Owner</td>
<td>Dallas</td>
</tr>
<tr>
<td>Feb 3–4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>March 1</td>
<td>User Stories for Agile Requirements Certified ScrumMaster Agile Estimating &amp; Planning</td>
<td>Boulder</td>
</tr>
<tr>
<td>March 2–3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>March 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>April 12</td>
<td>User Stories for Agile Requirements Certified ScrumMaster Succeeding with Scrum (new!)</td>
<td>San Diego</td>
</tr>
<tr>
<td>April 13–14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>April 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See mountaingoatsoftware.com for details.