Knowing what the customer wants

I know I said I wanted a Mustang, but I was really looking for the five-liter, turbocharged model...

You can't always get what you want...but the customer should!

Great software development delivers **what the customer wants**.

Orion’s Orbits is modernizing

- Orion’s Orbits provides quality space shuttle services to discerning clients, but their reservation system is a little behind the times, and they’re ready to take the leap into the 21st century. With the next solar eclipse just four weeks away, they’ve laid out some serious cash to make sure their big project is done right, and finished on time.

We need a web site showing our current deals, and we want our users to be able to book shuttles and special packages, as well as pay for their bookings online. We also want to offer a luxury service that includes travel to and from the spaceport and accommodation in a local hotel...

Here's one to get you started.

Title: Show current deals

Description: The web site will show current deals to Orion's Orbits users.

Title:

Description:

Title:

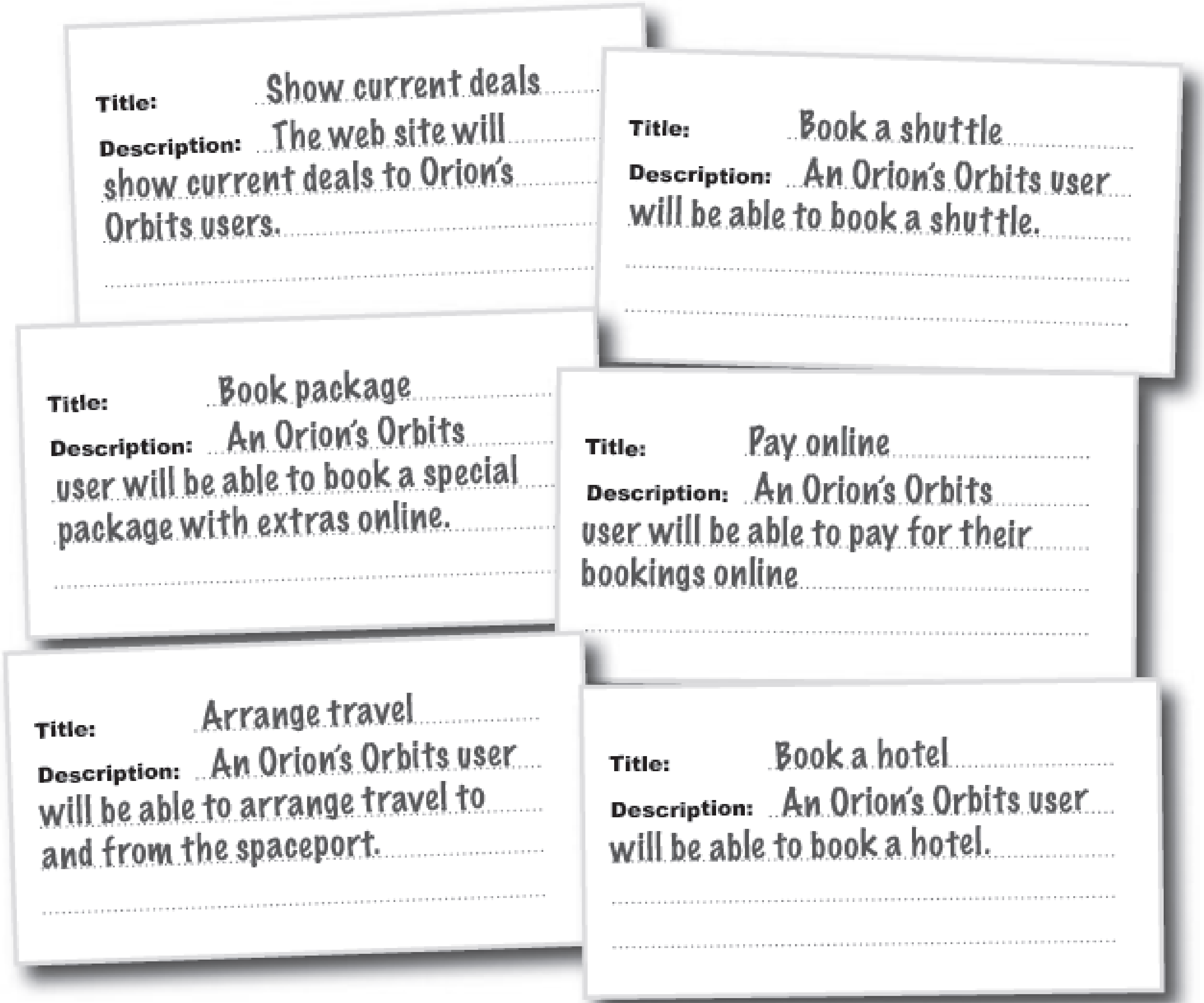
Description:

Title:

Description:

Remember, each requirement should be a single thing the system has to do.

If you've got index cards, they're perfect for writing requirements down.



Title: Show current deals

Description: The web site will show current deals to Orion's Orbits users.

Title: Book a shuttle

Description: An Orion's Orbits user will be able to book a shuttle.

Title: Book package

Description: An Orion's Orbits user will be able to book a special package with extras online.

Title: Pay online

Description: An Orion's Orbits user will be able to pay for their bookings online

Title: Arrange travel

Description: An Orion's Orbits user will be able to arrange travel to and from the spaceport.

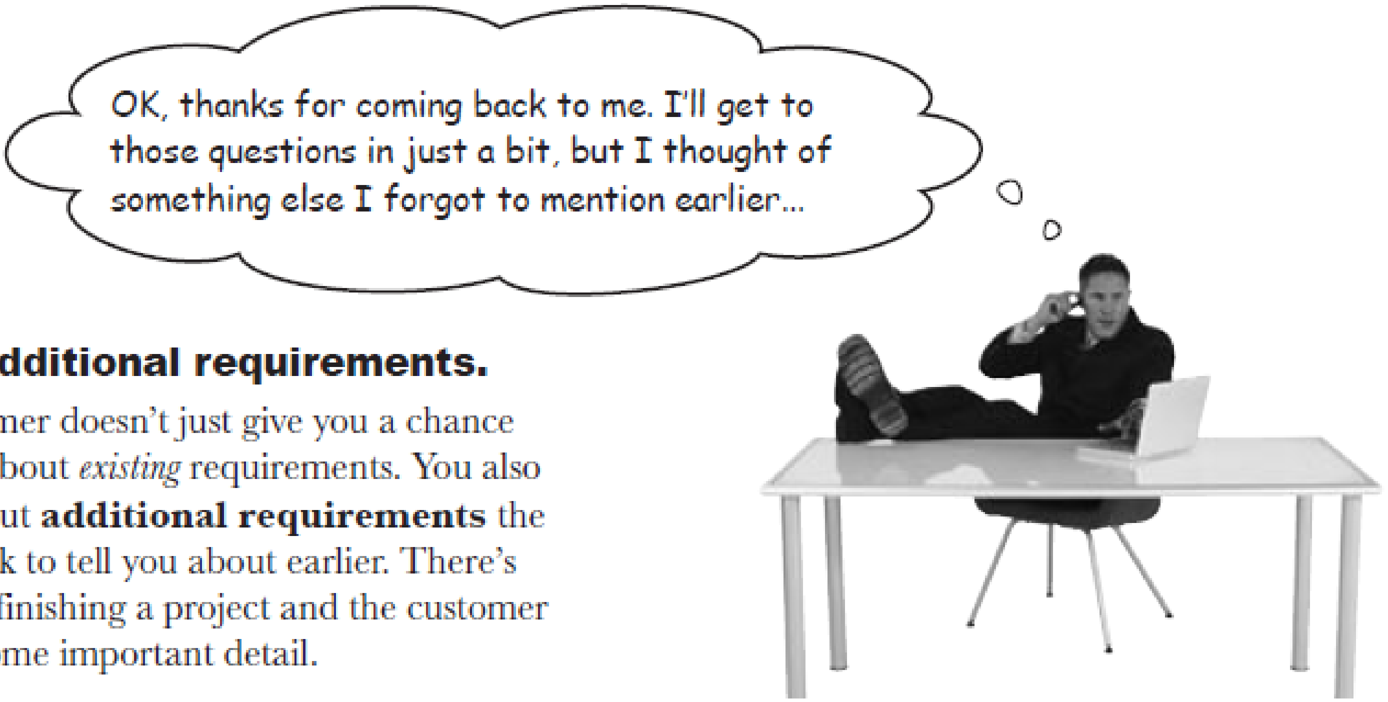
Title: Book a hotel

Description: An Orion's Orbits user will be able to book a hotel.

Each card captures one thing that the software will need to provide.

Talk to your customer to get MORE information

- How many different types of shuttles does the software have to support?
- Should the software print out receipts or monthly reports (and what should be on the reports)?
- Should the software allow reservations to be canceled or changed?
- Does the software have an administrator interface for adding new types of shuttles, and/or new packages and deals?
- Are there any other systems that your software is going to have to talk to,
- like credit card authorization systems or Air/Space Traffic Control?

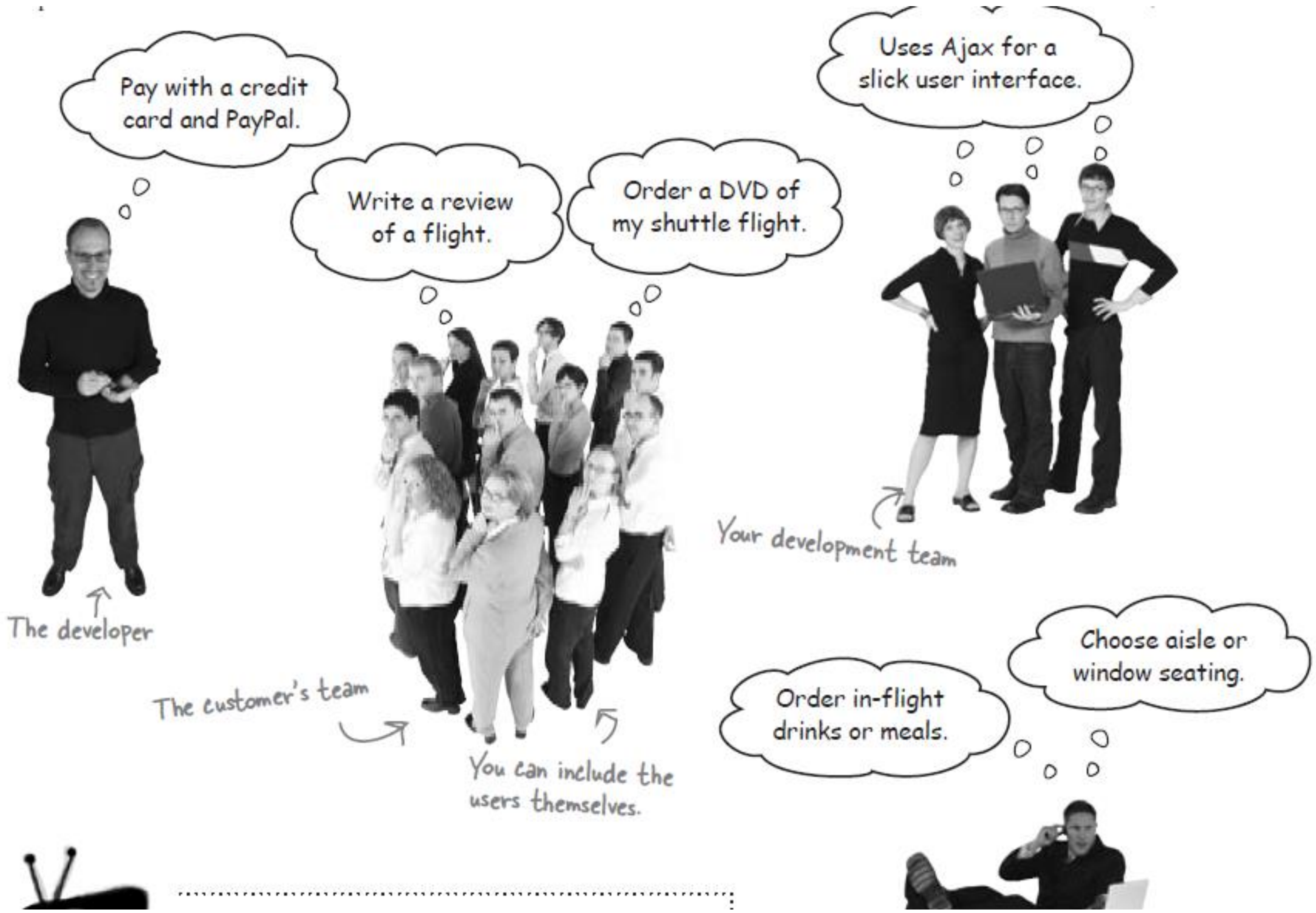

OK, thanks for coming back to me. I'll get to those questions in just a bit, but I thought of something else I forgot to mention earlier...

Try to gather additional requirements.

Talking to the customer doesn't just give you a chance to get more details about *existing* requirements. You also want to find out about **additional requirements** the customer didn't think to tell you about earlier. There's nothing worse than finishing a project and the customer saying they forgot some important detail.

Bluesky with your customer

- When you iterate with the customer on their requirements, THINK BIG.
- Brainstorm with other people; two heads are better than one, and ten heads are better than two
- Don’t rule out any ideas in the beginning—just capture *everything*


Pay with a credit card and PayPal.
Write a review of a flight.
Order a DVD of my shuttle flight.
Uses Ajax for a slick user interface.
The developer
The customer's team
You can include the users themselves.
Your development team
Choose aisle or window seating.
Order in-flight drinks or meals.

Take four of the ideas from the bluesky brainstorm and create a new card for each potential requirement. Also, see if you can come up with two additional requirements of your own.

Title: Pay with Visa/MC/PayPal

Description: Users will be able to pay for their bookings by credit card.

We can refer to each requirement easily by using its title.

Title: ....

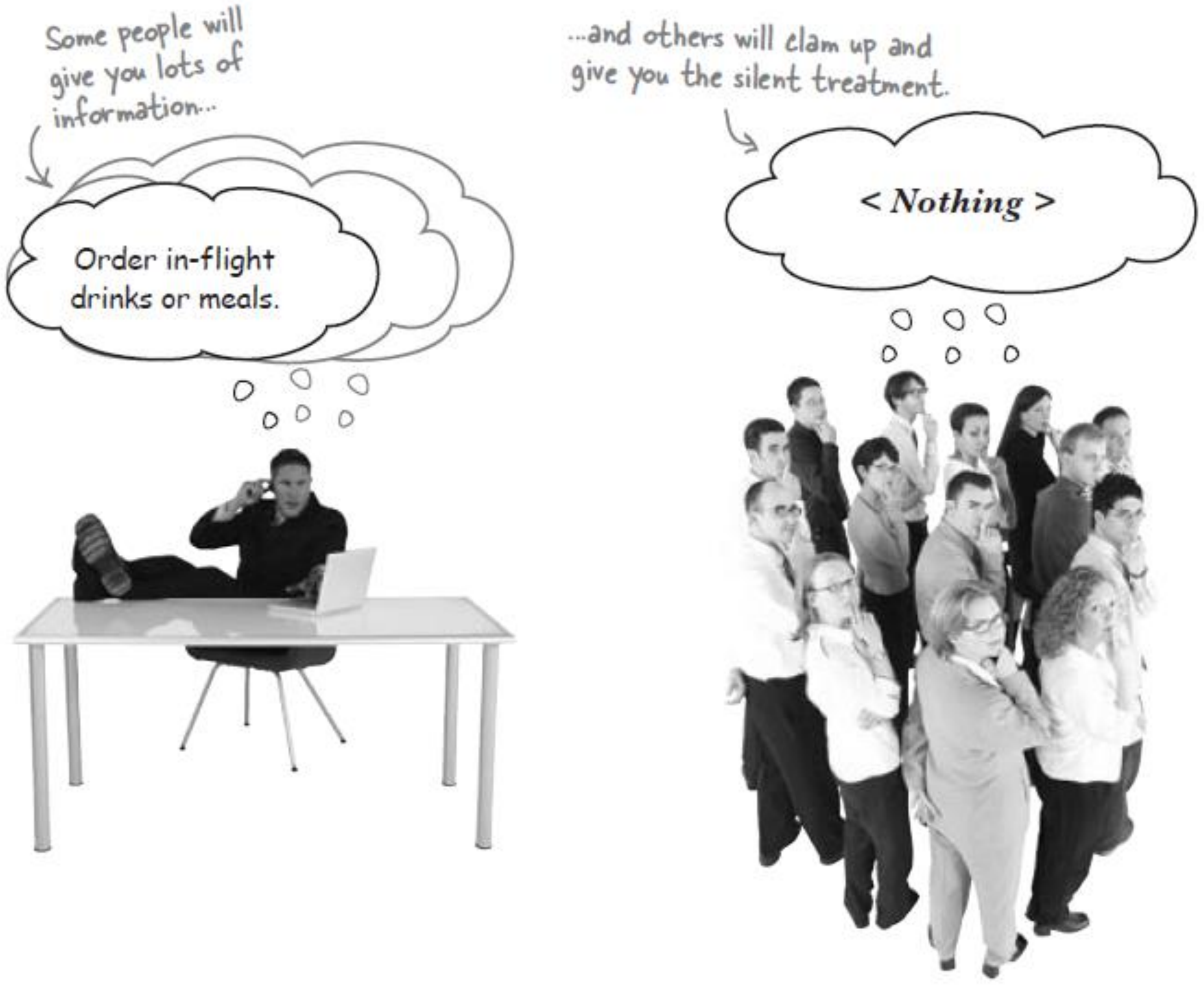
Description: ....

....

....

....

Sometimes your bluesky session looks like this...


Some people will give you lots of information...
Order in-flight drinks or meals.
...and others will clam up and give you the silent treatment.
< Nothing >

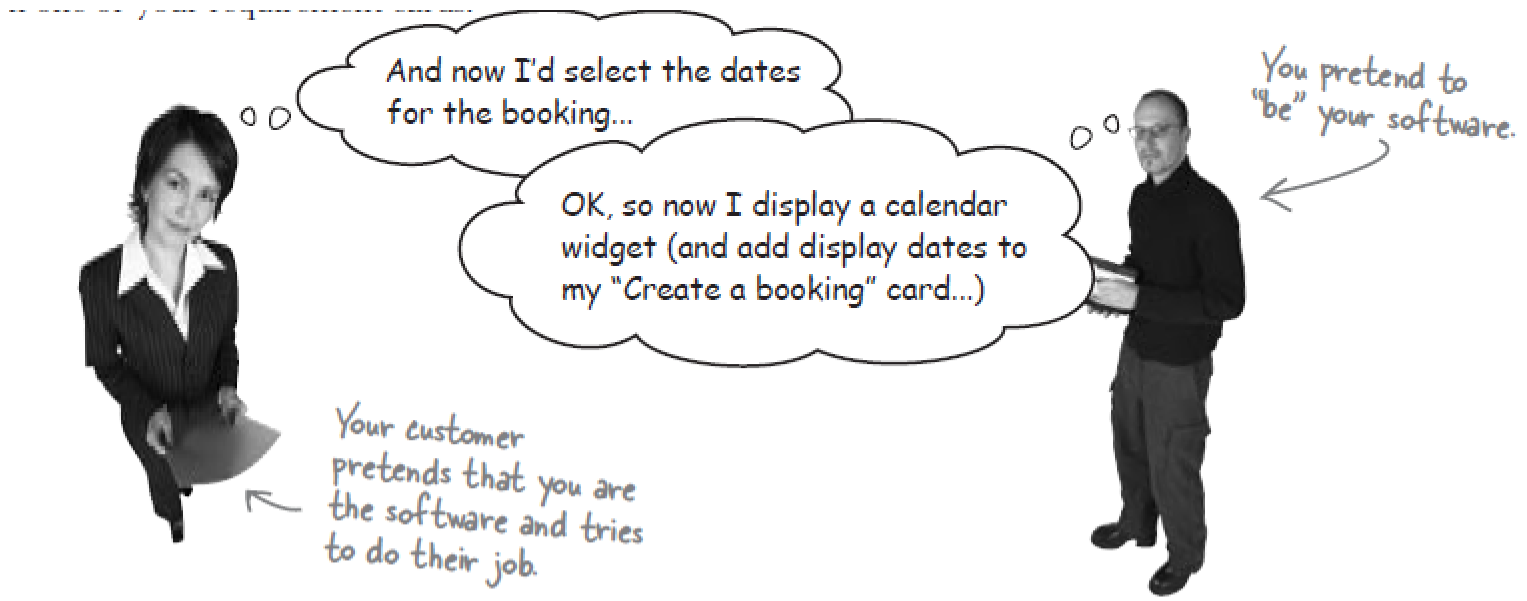
The zen of good requirements

The key to capturing good requirements is to get as many of the stakeholders involved as possible. If getting everyone in the same room is just not working, have people brainstorm individually and then come together and put all their ideas on the board and brainstorm a bit more. Go away and think about what happened and come back together for a second meeting.

There are LOTS of ways to gather good requirements. If one approach doesn't work, simply TRY ANOTHER.

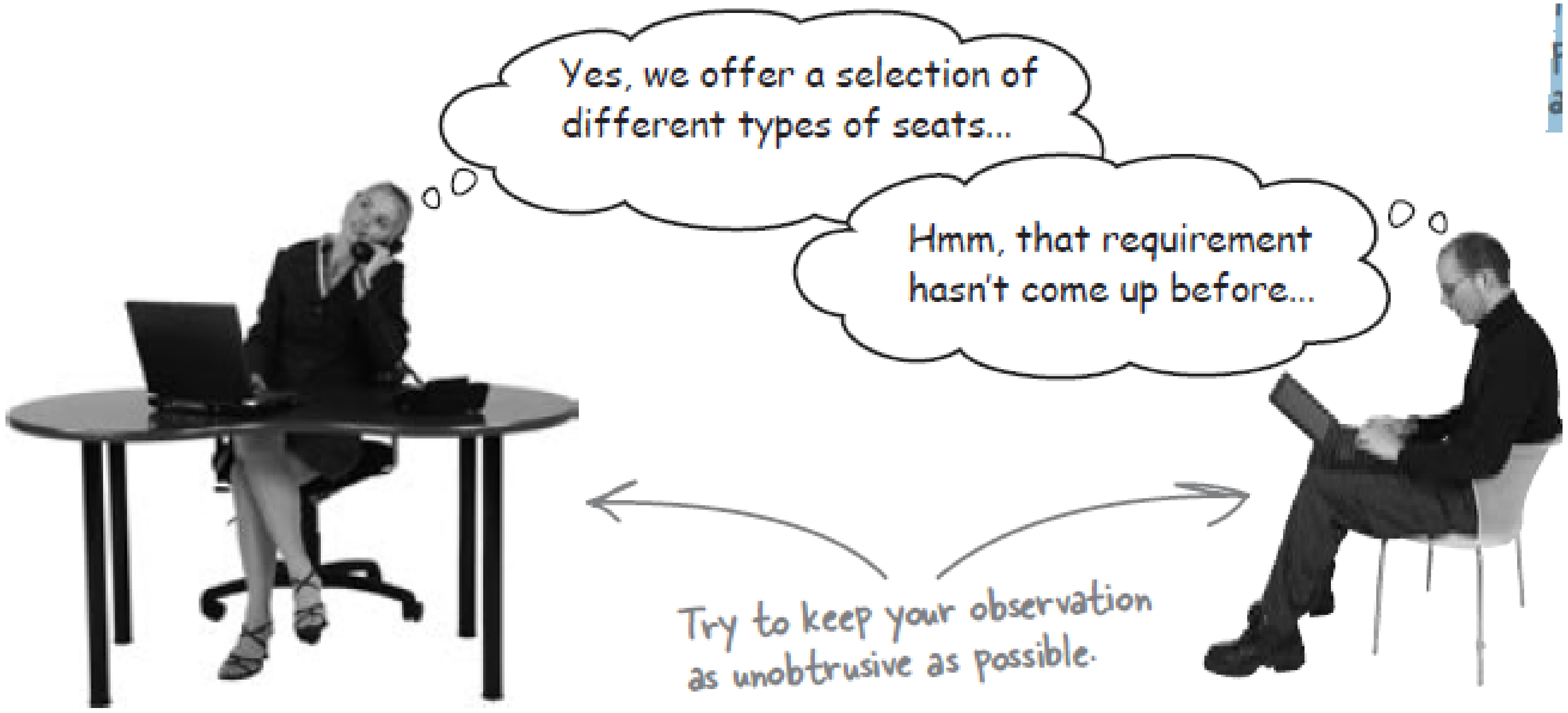
Find out what people REALLY do

- Role playing


And now I'd select the dates for the booking...
OK, so now I display a calendar widget (and add display dates to my "Create a booking" card...)
You pretend to "be" your software.
Your customer pretends that you are the software and tries to do their job.

Find out what people REALLY do

- Observation


Yes, we offer a selection of different types of seats...
Hmm, that requirement hasn't come up before...
Try to keep your observation as unobtrusive as possible.

Your requirements must be CUSTOMER-oriented

- A great requirement is actually written **from your customer’s perspective**
- Describing what the software is going to do **for the customer**
- A requirement should be written in the customer’s language and read like a **user story**: a story about how their users interact with the software you’re building

User stories SHOULD...

- ☐ ... describe **one thing** that the software needs to do for the customer.
- ☐ ... be written using language that **the customer understands**.
- ☐ ... be **written by the customer**.
- ☐ ... be **short**. Aim for no more than three sentences.

Think "by the customer, for the customer"

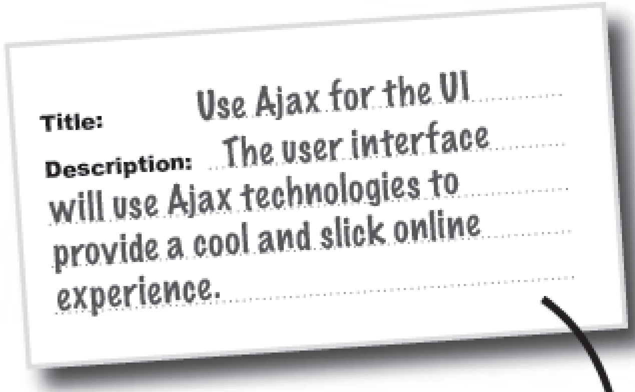
This means the customer drives each one, no matter who scribbles on a notecard.

You should be able to check each box for each of your user stories.

User stories SHOULD NOT...

- ☐ ... be a long essay.
- ☐ ... use technical terms that are unfamiliar to the customer.
- ☐ ... mention specific technologies.

If a user story is long, you should try and break it up into multiple smaller user stories (see page 54 for tips).

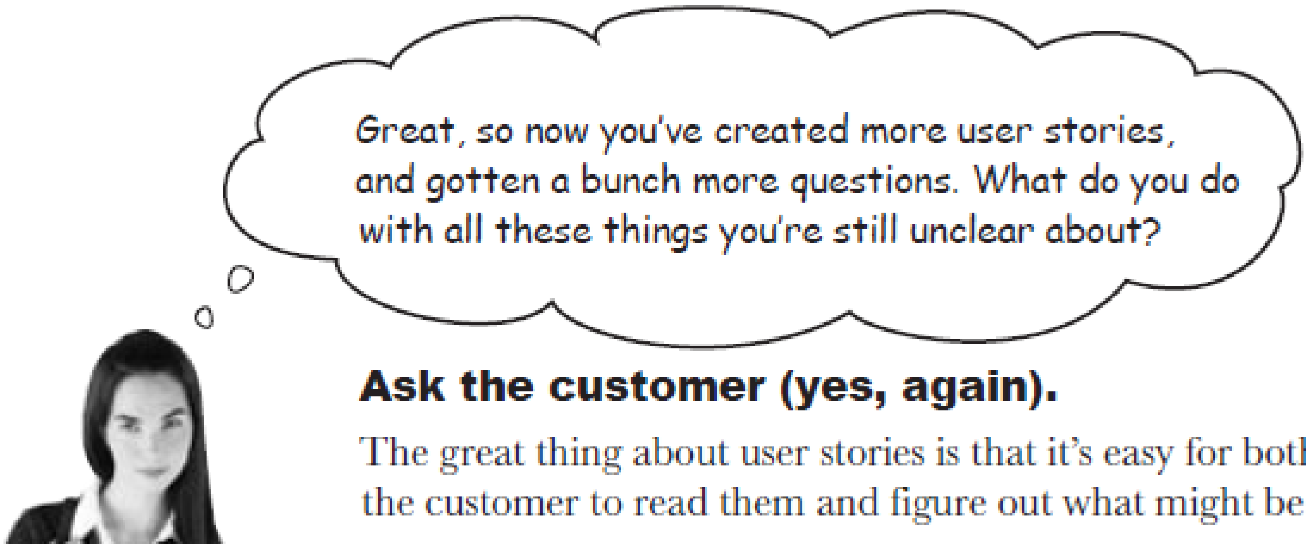


Title: Use Ajax for the UI

Description: The user interface will use Ajax technologies to provide a cool and slick online experience.

This card is not a user story at all; it's really a design decision. Save it for later, when you start implementing the software.

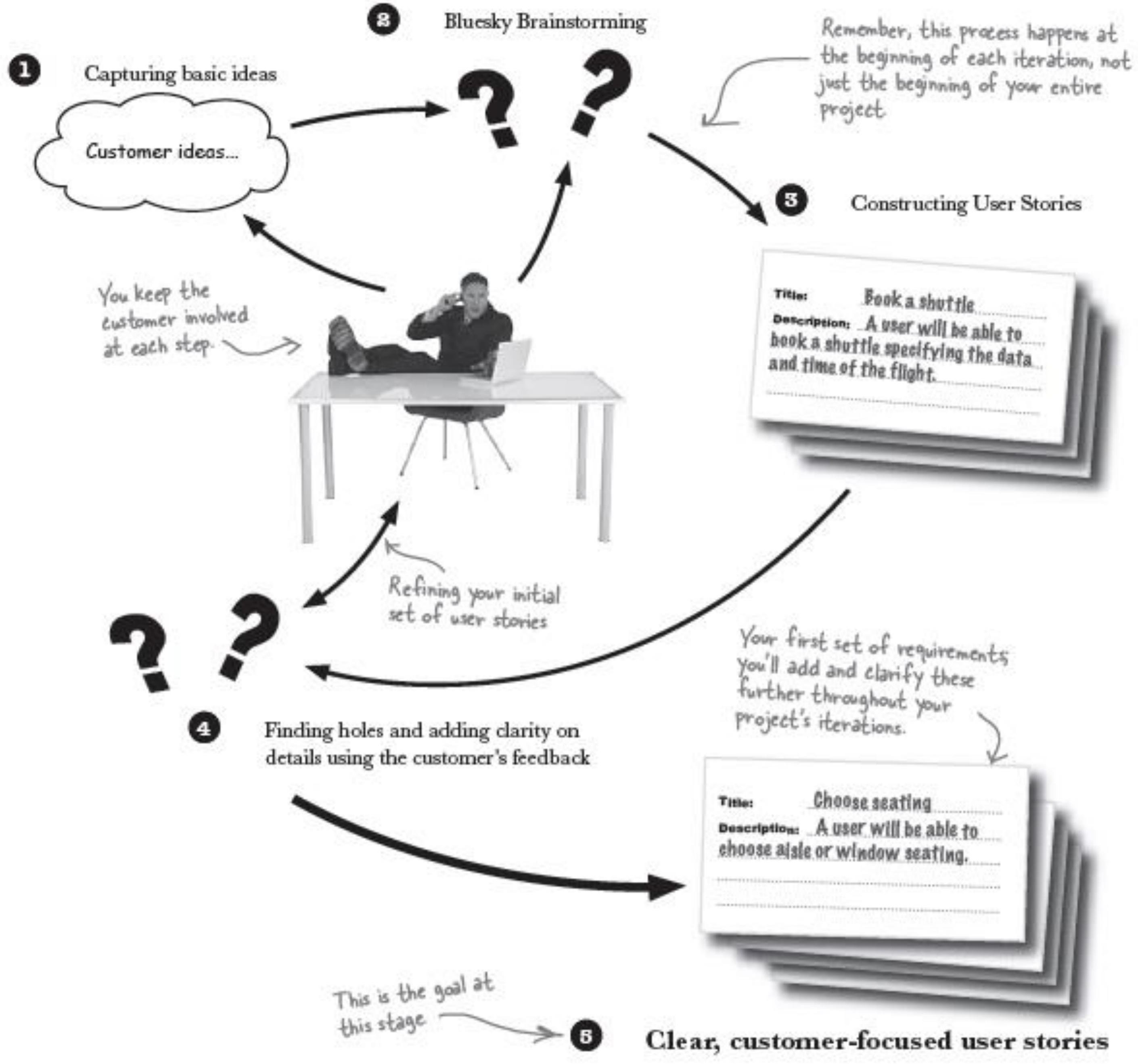
A user story is written from the CUSTOMER'S PERSPECTIVE. Both you AND your customer should understand what a user story means.


Great, so now you've created more user stories, and gotten a bunch more questions. What do you do with all these things you're still unclear about?

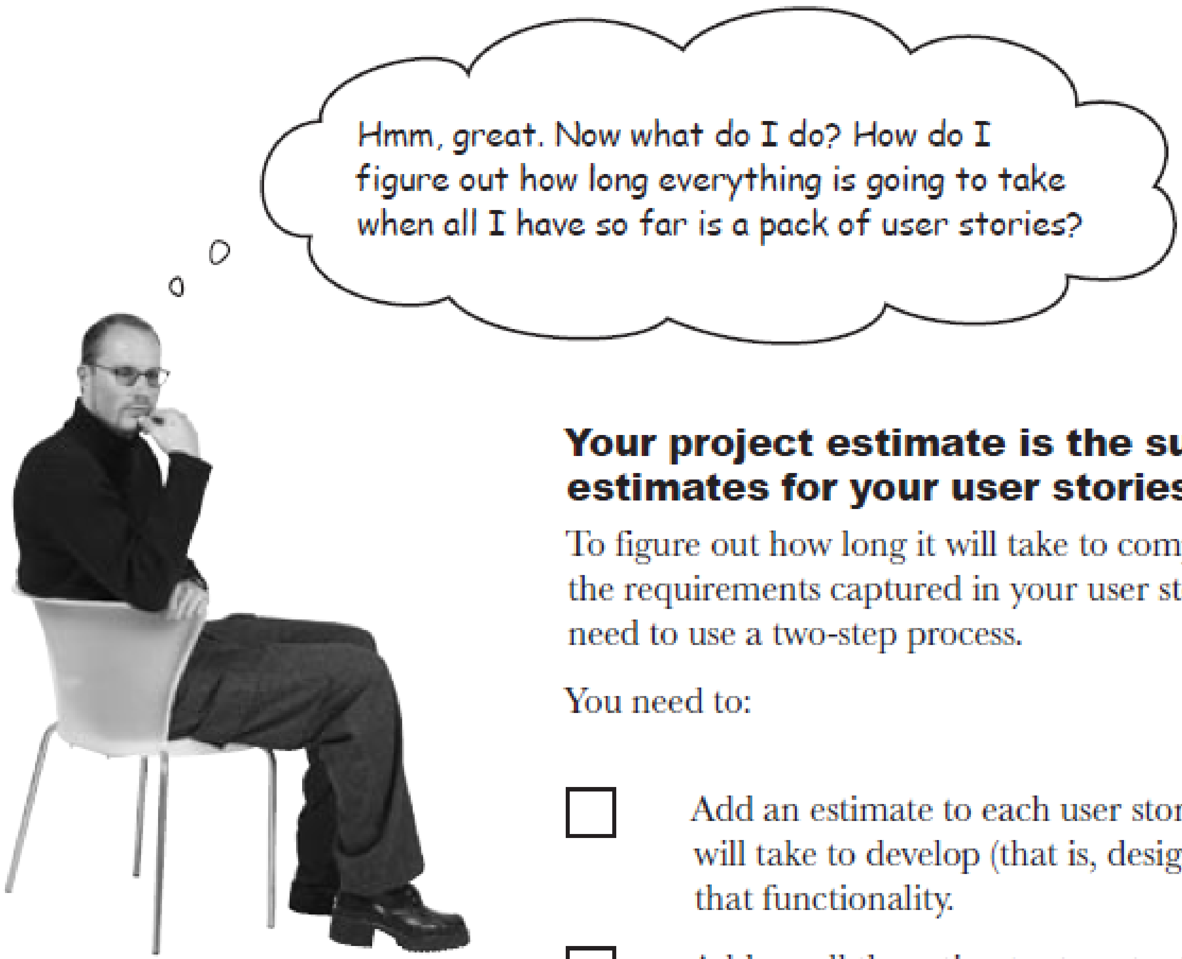
Ask the customer (yes, again).

The great thing about user stories is that it's easy for both you and the customer to read them and figure out what might be missing.

Develop your requirements with customer feedback


1 Capturing basic ideas
Customer ideas...
2 Bluesky Brainstorming
Remember, this process happens at the beginning of each iteration, not just the beginning of your entire project
3 Constructing User Stories
Title: Book a shuttle
Description: A user will be able to book a shuttle specifying the data and time of the flight.
You keep the customer involved at each step.
Refining your initial set of user stories
4 Finding holes and adding clarity on details using the customer's feedback
Your first set of requirements; you'll add and clarify these further throughout your project's iterations.
Title: Choose seating
Description: A user will be able to choose aisle or window seating.
This is the goal at this stage
5 Clear, customer-focused user stories

User stories define the WHAT of your project...
estimates define the WHEN


Hmm, great. Now what do I do? How do I figure out how long everything is going to take when all I have so far is a pack of user stories?

Your project estimate is the sum of the estimates for your user stories

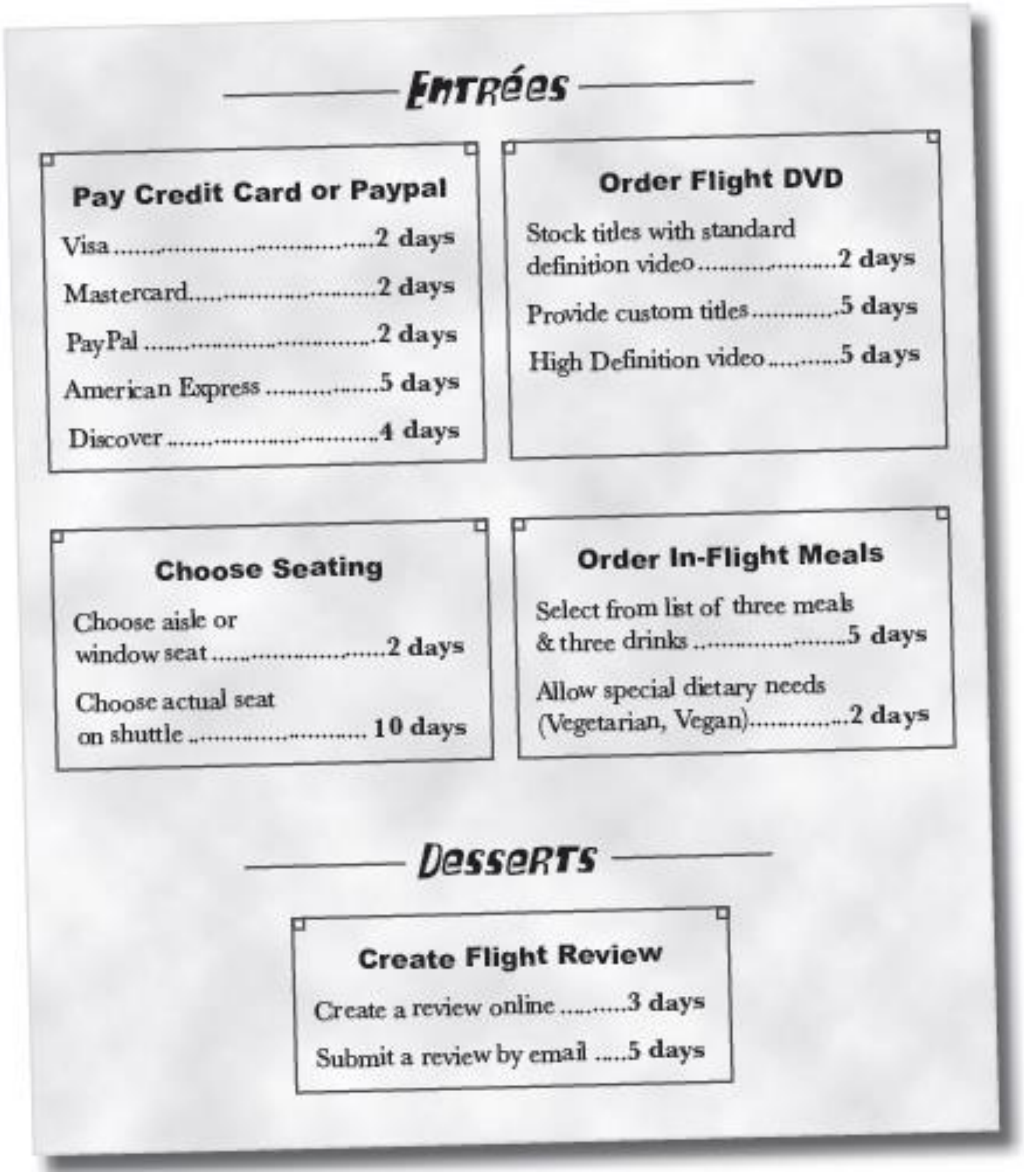
To figure out how long it will take to complete all of the requirements captured in your user stories, you need to use a two-step process.

You need to:

If you can get this figured out...

- [ ] Add an estimate to each user story for how long you think it will take to develop (that is, design, code, test, and deliver) that functionality.
- [ ] Add up all the estimates to get a total estimate for how long your project will take to deliver the required software.

...then this will be a piece of cake.



Entrées

Pay Credit Card or Paypal

Visa	2 days
Mastercard	2 days
PayPal	2 days
American Express	5 days
Discover	4 days

Order Flight DVD

Stock titles with standard definition video	2 days
Provide custom titles	5 days
High Definition video	5 days

Choose Seating

Choose aisle or window seat	2 days
Choose actual seat on shuttle	10 days

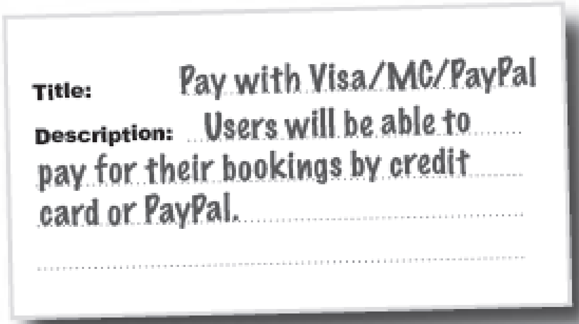
Order In-Flight Meals

Select from list of three meals & three drinks	5 days
Allow special dietary needs (Vegetarian, Vegan)	2 days

Desserts

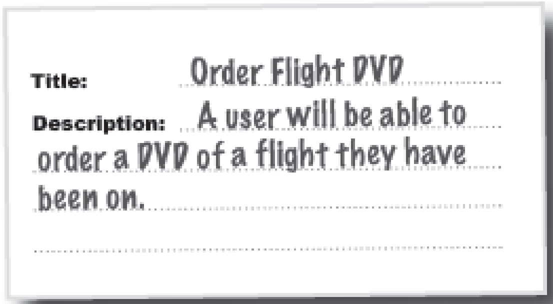
Create Flight Review

Create a review online	3 days
Submit a review by email	5 days



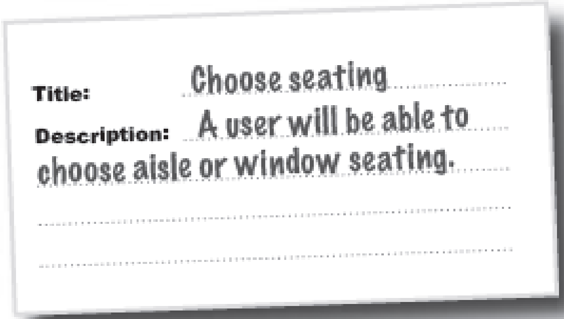
Title: Pay with Visa/MC/PayPal

Description: Users will be able to pay for their bookings by credit card or PayPal.



Title: Order Flight DVD

Description: A user will be able to order a DVD of a flight they have been on.



Title: Choose seating

Description: A user will be able to choose aisle or window seating.

Estimate for each user story in days

Write your estimate for the user story here.

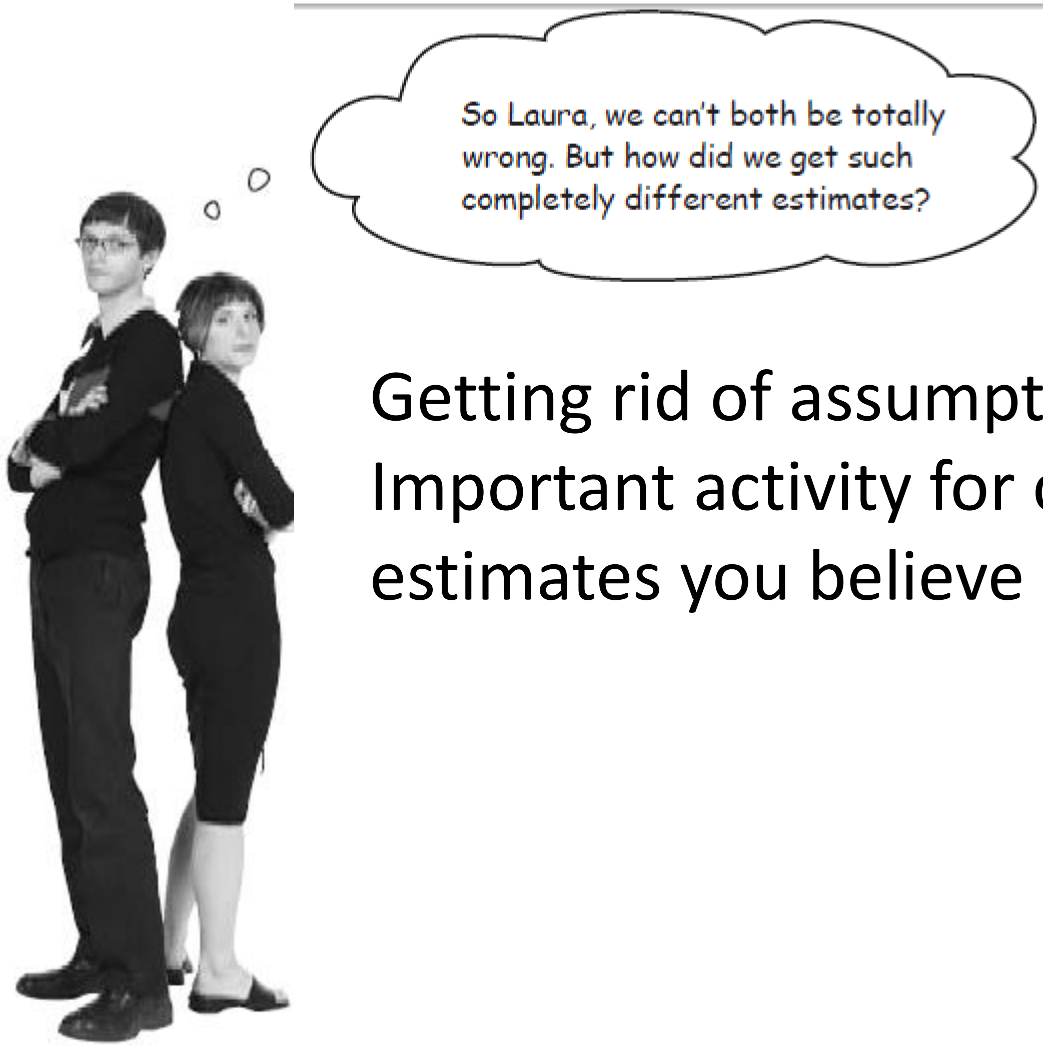
Assumptions?

Jot down any assumptions you think you're making in your estimate.

Solution

Put your estimates here.

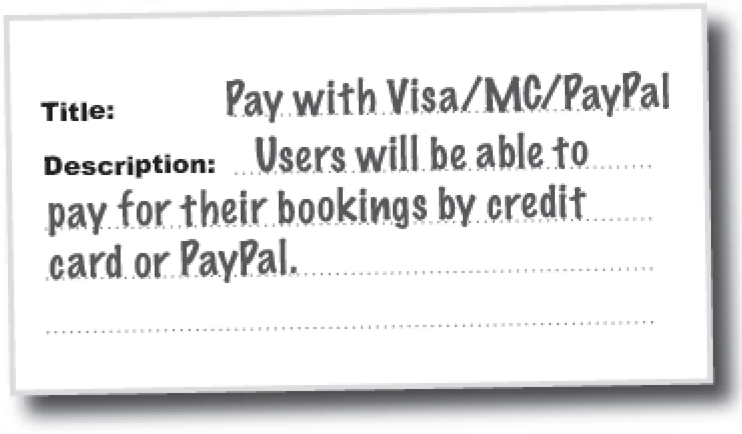
Title	Your estimates	Bob's estimates	Laura's estimates
Pay with Visa/MC/PayPal		10	15
Order Flight DVD		2	20
Choose seating		12	2
Order in-flight meals		2	7


So Laura, we can't both be totally wrong. But how did we get such completely different estimates?

Getting rid of assumptions is the most Important activity for coming up with estimates you believe in.

Playing planning poker

1 **Place a user story in the middle of the table**
This focuses everyone on a specific user story so they can get their heads around what their estimates and assumptions might be.

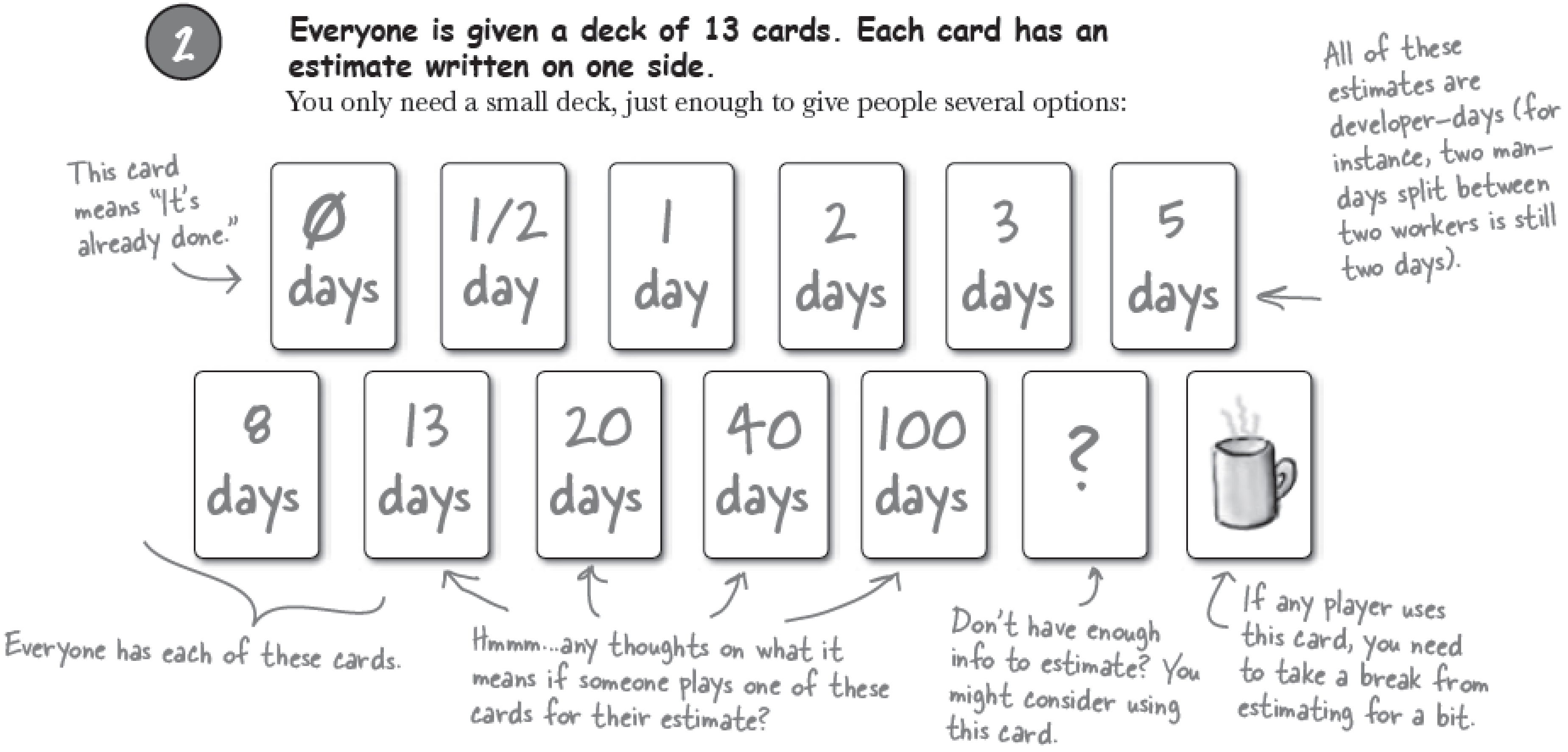


Title: Pay with Visa/MC/PayPal

Description: Users will be able to pay for their bookings by credit card or PayPal.

We want a solid estimate for how long it will take to develop this story. Don't forget that development should include designing, coding, testing, and delivering the user story.

Playing planning poker



2 **Everyone is given a deck of 13 cards. Each card has an estimate written on one side.**

You only need a small deck, just enough to give people several options:

This card means "It's already done."
Ø days
1/2 day
1 day
2 days
3 days
5 days
All of these estimates are developer-days (for instance, two man-days split between two workers is still two days).
8 days
13 days
20 days
40 days
100 days
?
Everyone has each of these cards.
Hmmm...any thoughts on what it means if someone plays one of these cards for their estimate?
Don't have enough info to estimate? You might consider using this card.
If any player uses this card, you need to take a break from estimating for a bit.

Playing planning poker

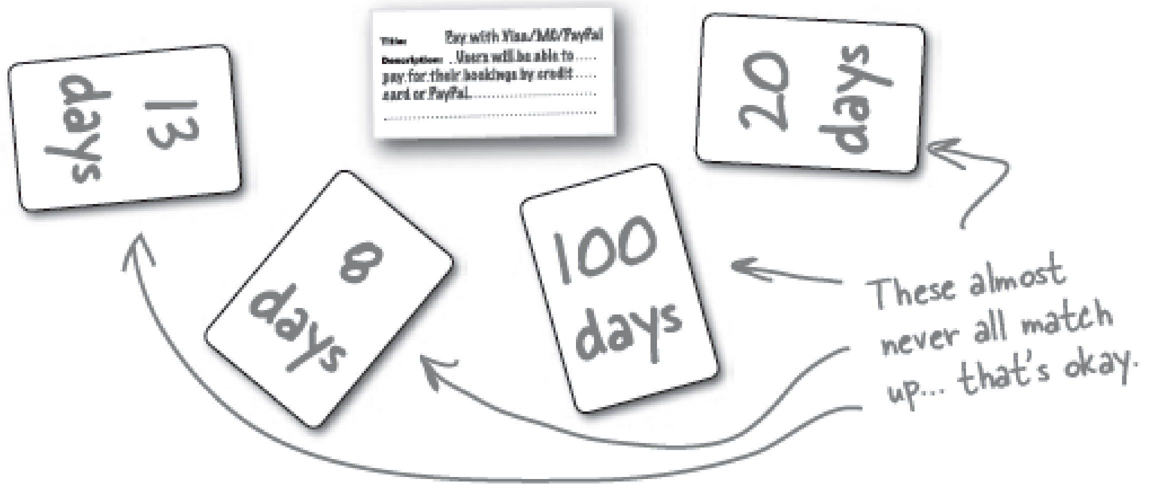
3 **Everyone picks an estimate for the user story and places the corresponding card face down on the table.**
You pick the card that you think is a reasonable estimate for the user story. Don't discuss that estimate with anyone else, though.

Make sure your estimate is for the <u>whole</u> user story, not just a part of it.

Title: Pay with Visa/MC/PayPal
Description: Users will be able to pay for their bookings by credit card or PayPal
Place your choice face-down so you keep your estimate from everyone else
The user story is still in the middle... still the focus.

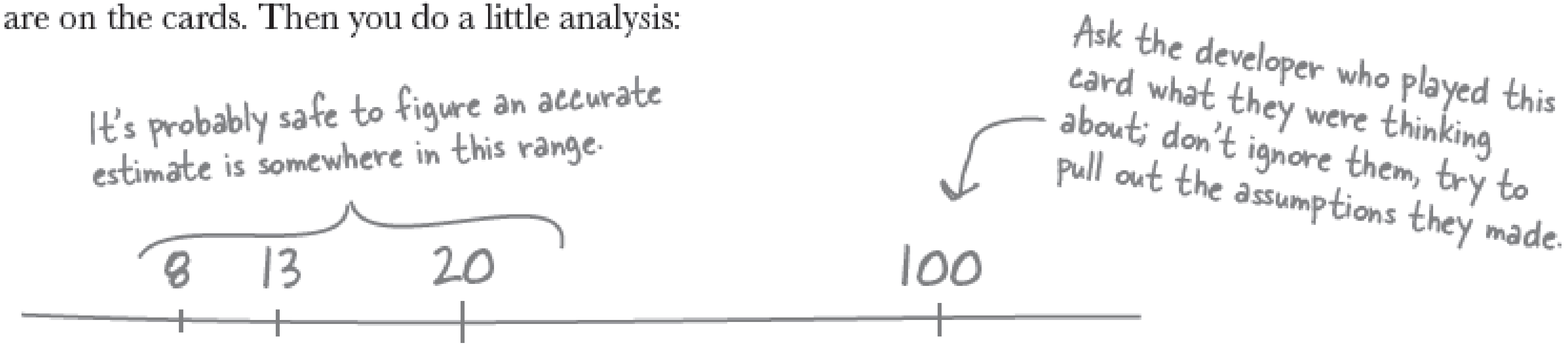
Playing planning poker

4 **Everyone then turns over their cards at exactly the same time.**
Each player at the table shows their hand, which gives their honest estimate for the user story.


Title: Pay with Visa/MC/PayPal
Description: Users will be able to pay for their bookings by credit card or PayPal.
13 days
8 days
100 days
20 days
These almost never all match up... that's okay.

Playing planning poker

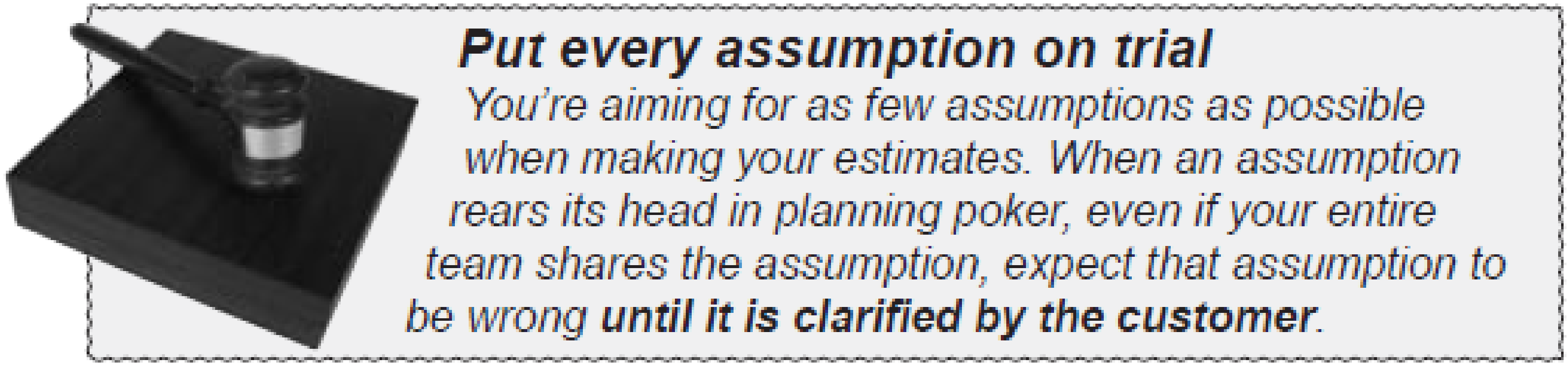
5 **The dealer marks down the spread across each of the estimates.**
Whoever is running the game notes the spread across each of the estimates that are on the cards. Then you do a little analysis:


It's probably safe to figure an accurate estimate is somewhere in this range.
Ask the developer who played this card what they were thinking about; don't ignore them, try to pull out the assumptions they made.
8
13
20
100

The larger the difference between the estimates, the less confident you are in the estimate, and the more assumptions you need to root out.

Put assumptions on trial for their lives

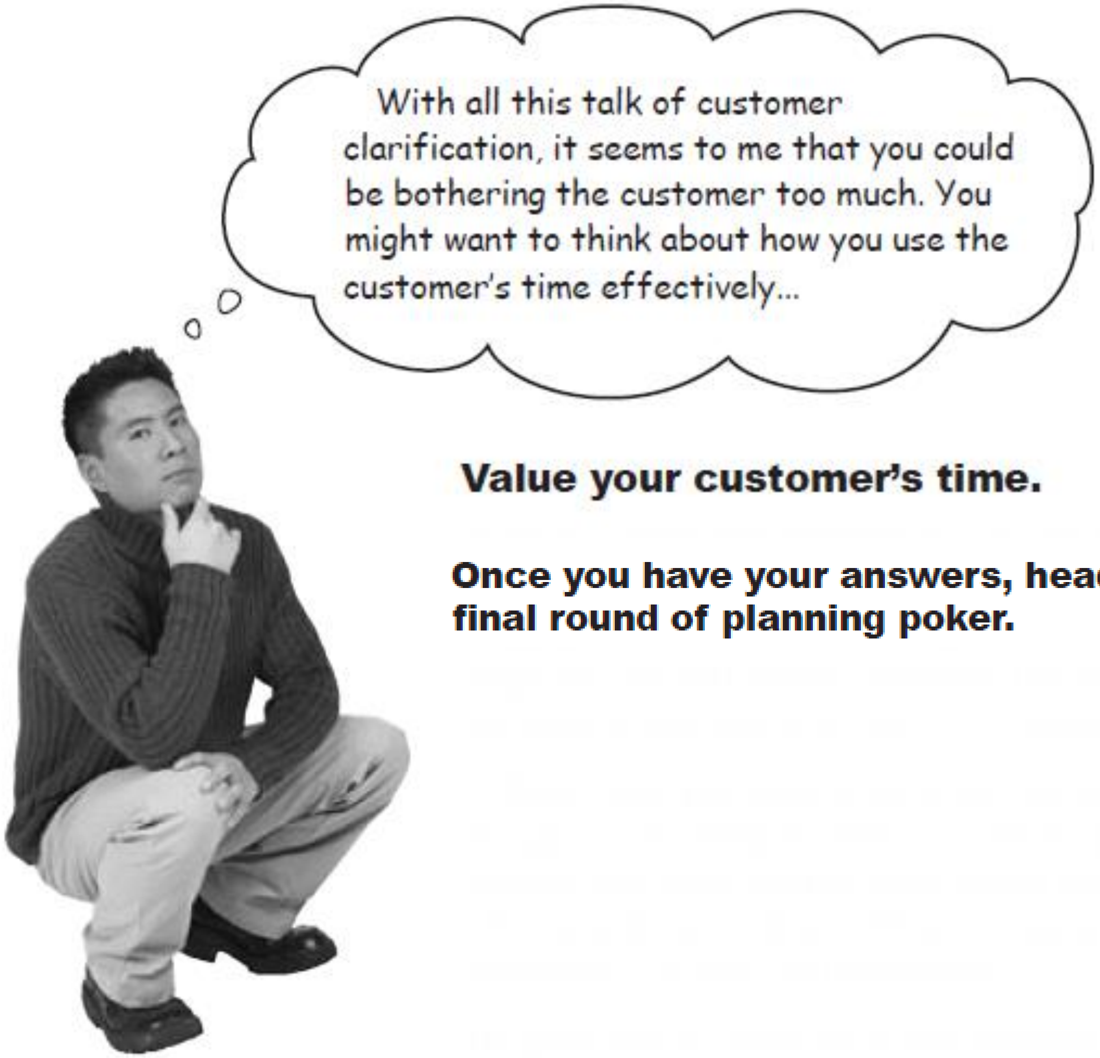
- When it comes to requirements***, no assumption is a good assumption***.



Put every assumption on trial

You're aiming for as few assumptions as possible when making your estimates. When an assumption rears its head in planning poker, even if your entire team shares the assumption, expect that assumption to be wrong ***until it is clarified by the customer****.*

While you can't always get rid of all assumptions, the goal during estimation is to eliminate as many assumptions as possible by clarifying those assumptions with the customer. Any surviving assumptions then become risks.

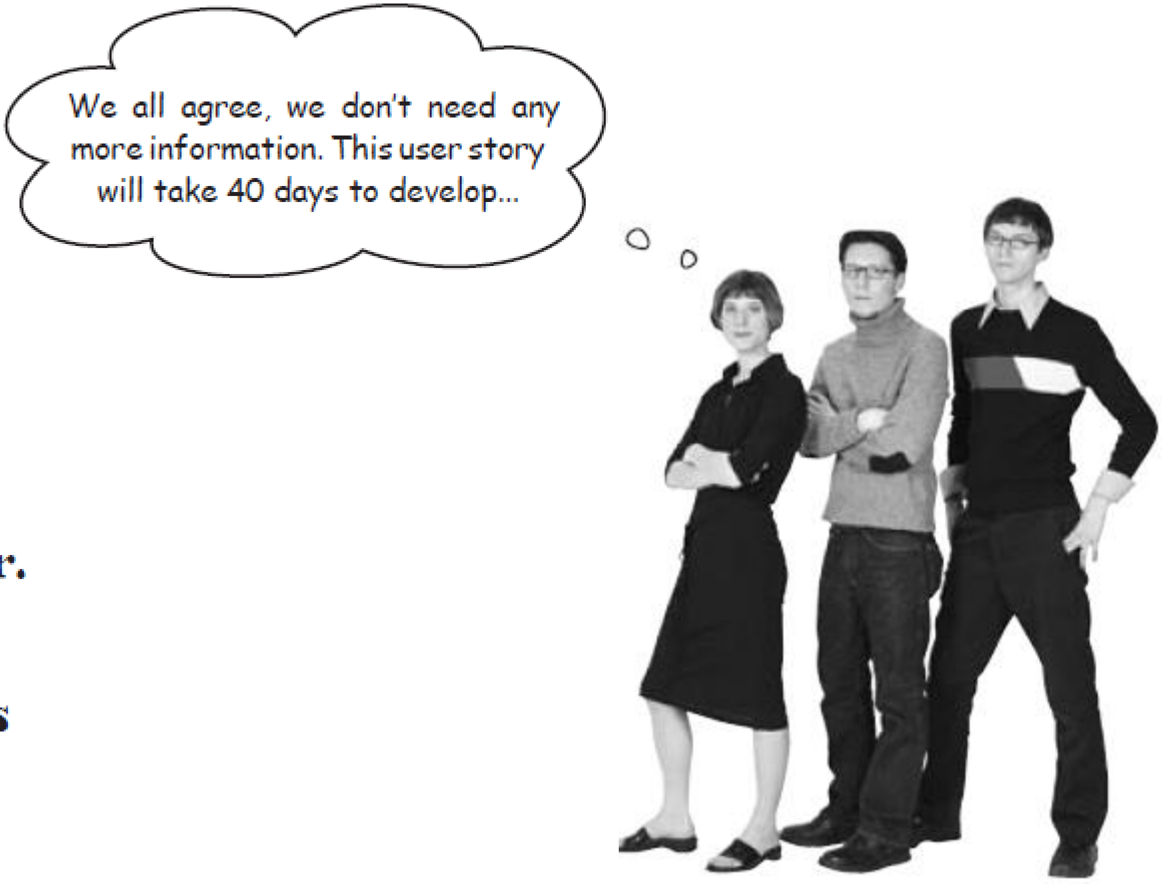

With all this talk of customer clarification, it seems to me that you could be bothering the customer too much. You might want to think about how you use the customer's time effectively...

Value your customer's time.

Once you have your answers, head back for a final round of planning poker.

Don't make
assumptions about
your assumptions...
talk about
EVERYTHING.

A **BIG** user story estimate is a **BAD** user story estimate


We all agree, we don't need any more information. This user story will take 40 days to develop...

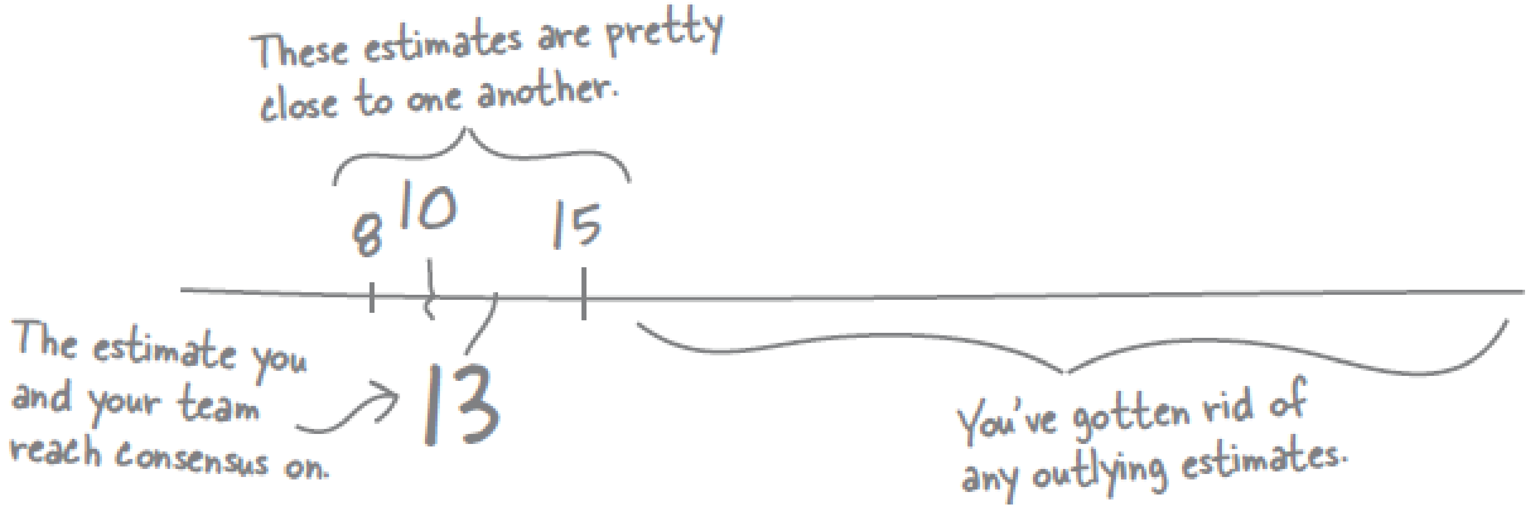
Estimates greater than 15 days per user story allow too much room for error.

When an estimate is too long, apply the AND rule to break the user story into smaller pieces.

When a user story’s estimate breaks the 15-day rule you can either:

- **Break your stories into smaller, more easily estimated stories**
- **Talk to your customer...again.**

The goal is convergence


These estimates are pretty close to one another.
8
10
15
The estimate you and your team reach consensus on.
13
You've gotten rid of any outlying estimates.

Run through this cycle of steps till you reach a consensus:

1. **Talk to the customer**
   First and foremost, get as much information and remove as many assumptions and misunderstandings as possible by talking to your customer.

2. **Play planning poker**
   Play planning poker with each of your user stories to uproot any hidden assumptions. You'll quickly learn how confident you are that you can estimate the work that needs to be done.

3. **Clarify your assumptions**
   Using the results of planning poker, you'll be able to see where your team may have misunderstood the user stories, and where additional clarification is needed.

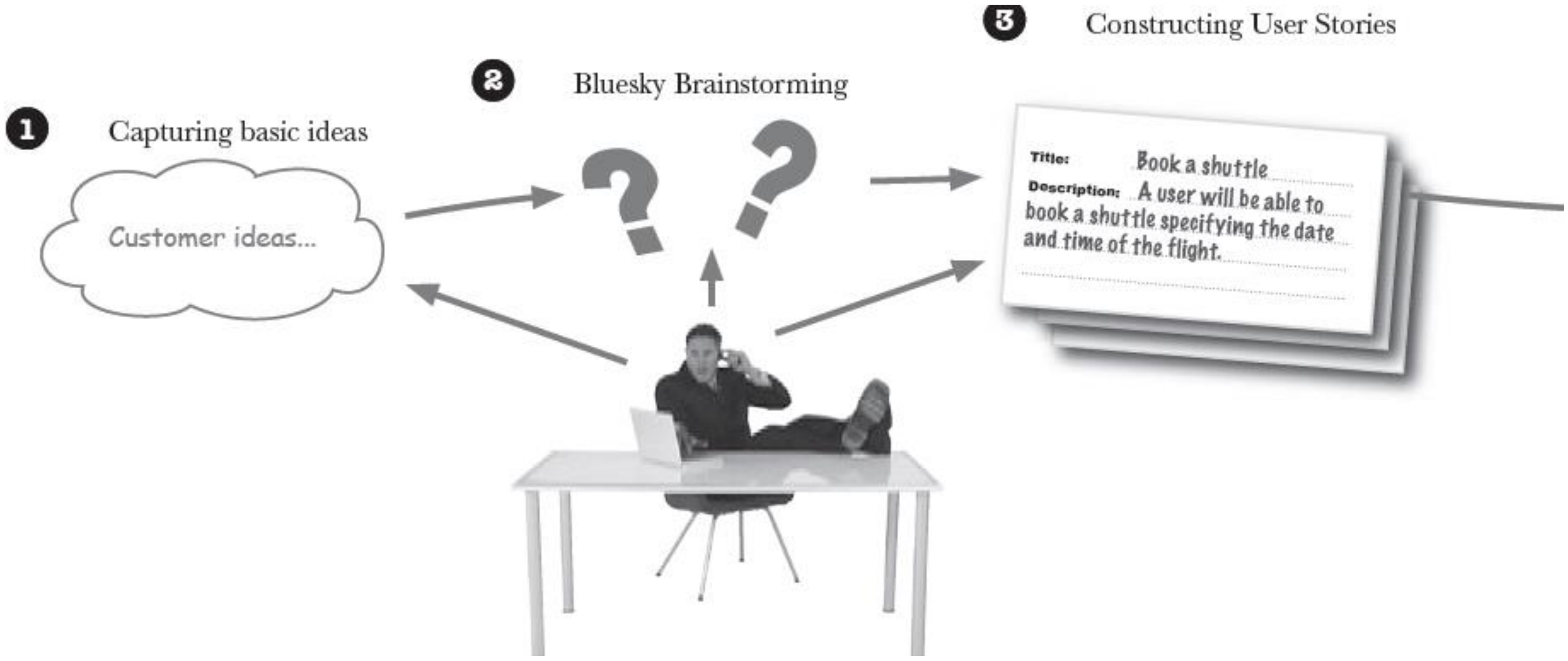
4. **Come to a consensus**
   Once everyone's estimates are close, agree on a figure for the user story's estimate.

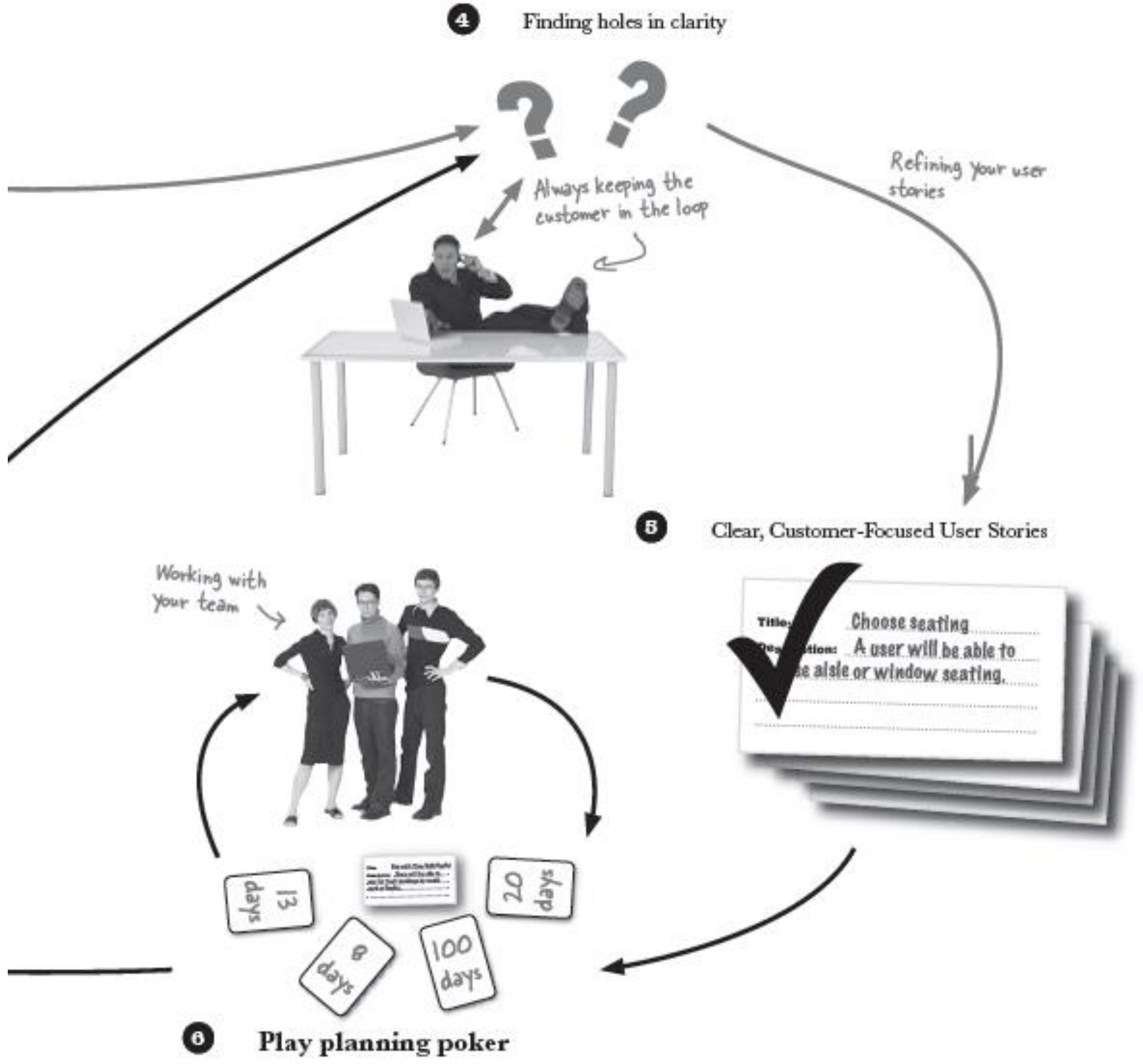
Head back to Step 1 if you find assumptions that only the customer can answer.

It can also be useful to note the low, converged, and high estimates to give you an idea of the best and worst case

Your estimates are your **PROMISE** to your customer about how long it will take you and your team to **DELIVER**.

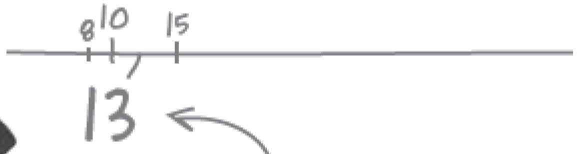
The requirement to estimate iteration cycle


1 Capturing basic ideas
Customer ideas...
2 Bluesky Brainstorming
3 Constructing User Stories
Title: Book a shuttle
Description: A user will be able to book a shuttle specifying the date and time of the flight.

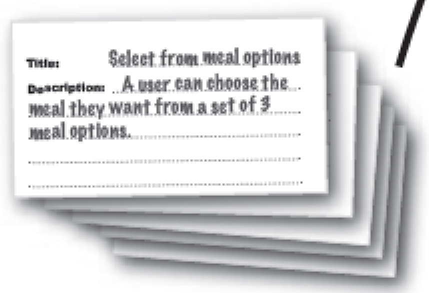

4 Finding holes in clarity
Always keeping the customer in the loop
Refining your user stories
5 Clear, Customer-Focused User Stories
Title: Choose seating
Description: A user will be able to choose aisle or window seating.
Working with your team
13 days
20 days
8 days
100 days
6 Play planning poker

We're now ready to estimate how long the project as a whole is going to take.

Estimate!



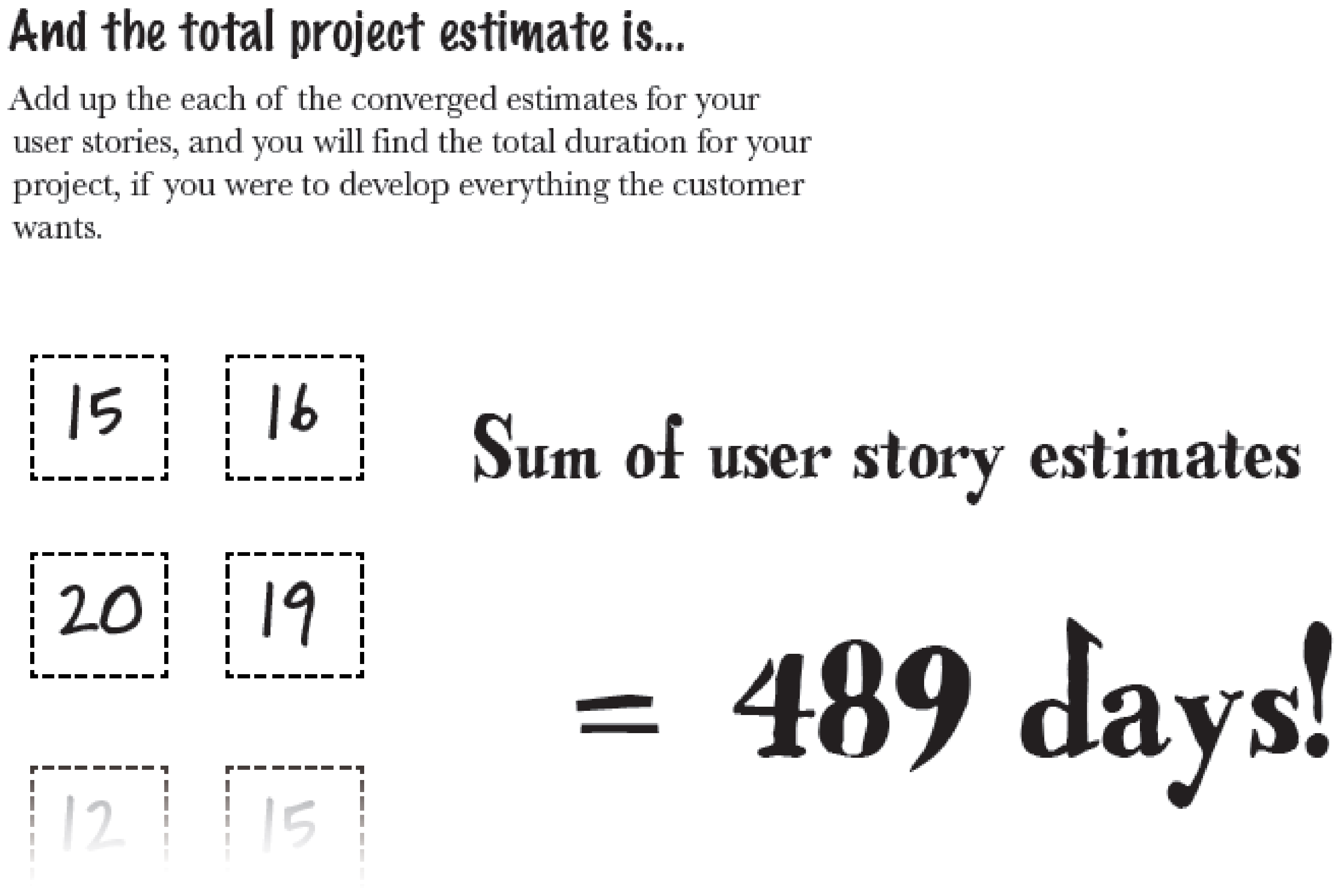
Your spreads are now converged, and you have an estimate for each user story.


Title: Select from meal options
Description: A user can choose the meal they want from a set of 3 meal options.

❽ **Estimate how long all of the customer's requirements will take**

❼ **Get any missing information from the customer, and break up large user stories**

Finally, you're ready to estimate the whole project...



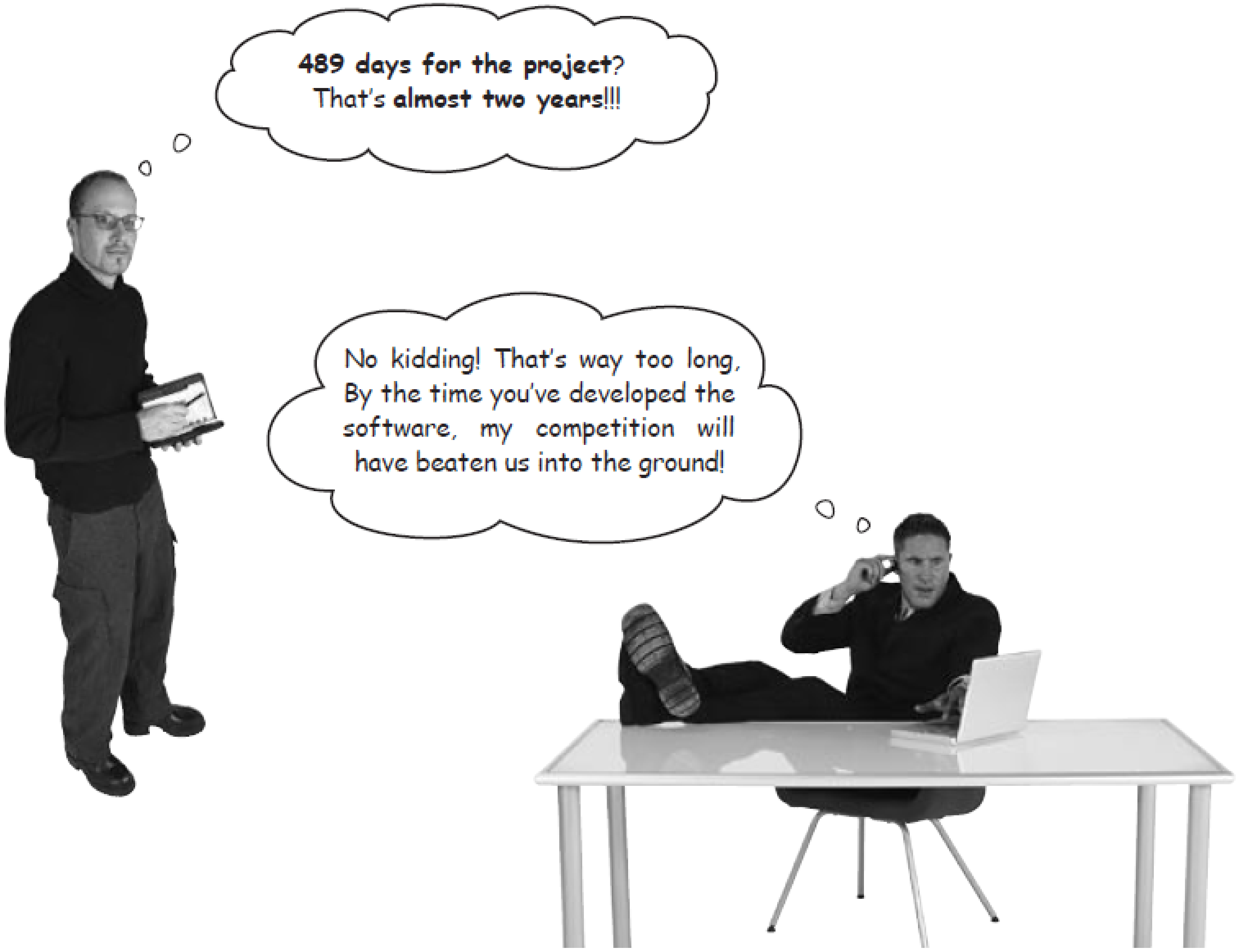
And the total project estimate is...

Add up the each of the converged estimates for your user stories, and you will find the total duration for your project, if you were to develop everything the customer wants.

15	16
20	19
12	15

Sum of user story estimates

= 489 days!


489 days for the project? That's **almost two years**!!!
No kidding! That's way too long, By the time you've developed the software, my competition will have beaten us into the ground!

What do you do when your estimates are WAY too long?

Development Techniques

Bluesky, Observation and Roleplay

User Stories

Planning poker for estimation

Development Principles

The customer knows what they want, but sometimes you need to help them nail it down

Keep requirements customer-oriented

Develop and refine your requirements iteratively with the customer