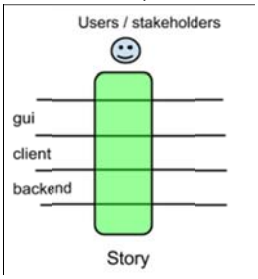
SCRUM EXERCISE – ELEPHANT CARPACCIO

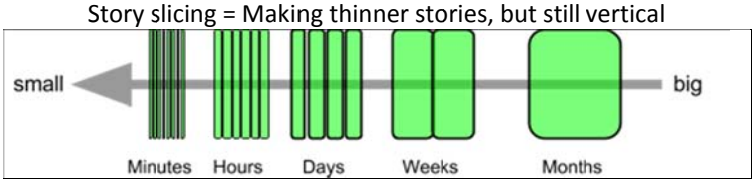
Learn how to do small work slices and continuously make progress
Purpose is to learn to split stories really small

WHY SPLIT STORIES?

Story = Vertical, testable, user-valuable, across multiple architectural layers.



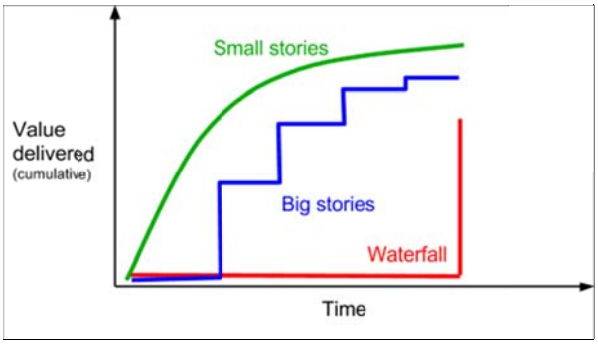
Story slicing = Making thinner stories, but still vertical



Story = a few days
Task = a few hours
Commit = several times per hour

Why split stories?
The value / benefits from dividing user stories into very thin vertical slices

- Learn faster
- Deliver more often
- Happier stakeholders
- Better prioritizations



Small stories curve end up with higher cumulative value

EXERCISE

Laptop / Handout

2 hours

- 15-20m Discussion about user stories
- 20-30m Breaking down backlog
- 40m Coding
- 15-20m Debrief

Goal

- Build a simple application in 40 minutes, divided into 5 iterations x 8 minutes.
- Most people would build this app in 2-3 slices, we will do it in 15-20.
- Elephant carpaccio = Very thin elephant-shaped slices, that together form the whole elephant.

Details

- Use any programming language.
- Interface could be console, command line, web, gui, whatever.
- Build a retail calculator

A runnable application with user interface, three inputs and one output.

3 inputs:

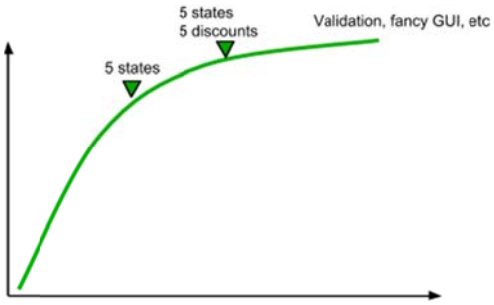
How many items
Price per item
2-letter state code

1 output:

Total price of the order

Give a discount based on the total price
Add state tax based on the state code and discounted price

Priorities



Target is 5 discounts, 5 states
Break into 10-20 slices
A slice is only valid if it has a UI, input & output, and is visibly different from last slice.

5 states before doing anything with discounts.

Why?

So it can be deployed! State tax is a legal requirement, discount is not

Validation & fancy GUI comes after 5 states & 5 discounts

Don't spend any time on that before!

Build the highest-value thing first, all the time

For example, before you have 5 states, don't waste a single keystroke on discount-related code!

It is not important how far you get on this curve / The important thing is to practice microslicing.
If you get all the way to the end with only 3 slices, you've wasted your time and missed the point of this exercise.

CREATE THE BACKLOG

Split into groups of 2
10 minutes: Write your backlog on paper
Write 10-20 demo-able user stories ("slices") that will take you from nothing to 5 states & 5 discounts.
Valid slice guideline:

- Implementable (including user interface) in 2-6 minutes.
- Noticeable different from last slice
- More valuable to customer than last slice
- No slice is just a mockup or UI, a data structure, or test case.

What is your first slice?

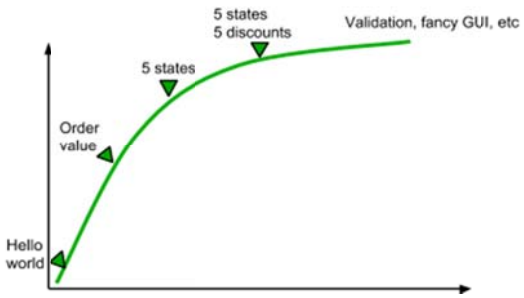
- Anything bigger than hello-world or echo-input-to-output is too big

The "order value" milestone

- 2 inputs ("# of items" & "price per item") and multiply them.
- No discounts/states.
- Order value. 2 inputs, 1 output. No state tax, no discounts.

Next slice, after order value?

- Hard-coded state tax.
  - Still 2 inputs, 1 output.
  - Utah state tax added automatically.
    - (Now we can go live in Utah!).
- 3 inputs (price per item, # of items, state tax %).
  - Output is order value with state tax added.
  - User inputs actual tax rate rather than state code.
  - This gives simpler code (no internal data structure to map state code to tax rate), so we get faster delivery.
- 3 inputs (price per item, # of item, state tax), but only two states are supported. Any other gives error message.
- Add the other 3 states.
  - There is limited value in adding 1 state at a time at this point
  - There is no risk reduction value, and it takes literally just a few seconds more to all 3 states at once



BUILDING

40 minutes
5 iterations
8 minutes per iteration.

At the end of each iteration, you will be alerted for "demo time!".
That means stop coding, and demonstrate your app to another team.
Don't spend too much time on demo.
Shout "slice" whenever you finish a slice.

REVIEW

How far did you get?
Mark each team's approximate position on the value curve

Typical result: some teams get past 5 states & 5 discounts.
Most teams at least get to hard-coded state tax.
How many slices did you have?

Acceptance test:
Start your application and enter these values: I am in Utah, I'm buying 978 items, each item costs $270.99.
No fiddling around, just run the app and enter the values and see what comes out.
Each team provides output!
(265028,22 * 0,85) = 225273,987 * 1,0685 = 240705,2551095
Compare the results.

DEBRIEF

How is your code quality, how proud are you of your code?
Discuss importance of sustainable pace & quality, and how developers are responsible for this
How much work to pull in
Any other questions or reflections?

Round-robin:

- What did you learn?
- What will you do?
- Name one take-away insight from today, and one thing you will do differently in the future.