

COPENHAGEN BUSINESS ACADEMY



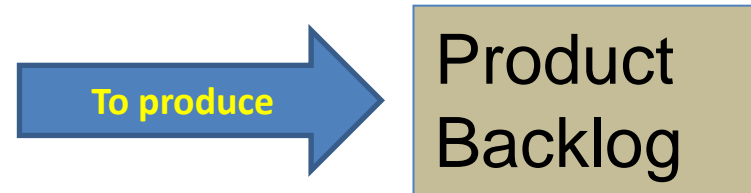
Systems Development – Day 2

Agenda for Scrum Day 2

- Poker game
- User stories
- Estimation
- Done criteria
- Elephant Carpaccio exercise

Stories – how to write GOOD stories!

- How to identify stories (PO)
- How to write stories (PO)
- How to estimate stories (Team)



OBS! We must remember INVEST for each story

- I – Independent
- N – Negotiable
- V – Valuable
- E – Estimable
- S – Small
- T – Testable

Independent Stories

- Stories are easiest to work with if they are independent.
- We'd like them to not overlap in concept
- We'd like to be able to schedule and implement them in any order.

Negotiable... .. and Negotiated Stories

- A good story is negotiable.
- It is not an explicit contract for features; Rather, details will be co-created by the Product Owner and Team.
- A good story captures the essence, not the details.

Valuable Stories 1

- A story needs to be valuable to the customer
- What about Tech Stories? (H. Kniberg p. 39)
 - Examples:
 - Install server
 - Write a system design overview
 - Refactor the data layer
- Transform into normal story
- Be a task in another story
- Define and keep in separate list
 - Let Product Owner see, but not edit
 - Negotiate with Product Owner

Valuable Stories 2

- Valuable – to who?
 - Customer (purchaser & user)
 - Secondarily developer

Examples:

Valued by purchaser, but maybe not individual users:

"All configuration information is read from a central location"

"The development team will produce the software in accordance with CMM Level 3"

Valued by both customer and developer... if changed from

"All error handling and logging is done through a set of common classes" into this text:

"All errors are presented to the user and logged in a consistent manner"

Estimable Stories 1

- A good story can be estimated.
- We don't need an exact estimate, but just enough to help the Product Owner rank and schedule the story's implementation.
- Being estimable is partly a function of being negotiated, as it's hard to estimate a story we don't understand.
- It is also a function of size: bigger stories are harder to estimate.
- And of the team: what's easy to estimate will vary depending on the team's experience

Estimable Stories 2

- Why difficult to estimate stories?
 1. Developers lack domain knowledge.
 2. Developers lack technical knowledge.
 3. The story is too big.

Solutions

1. Discuss with customer
2. Turn into two stories:
 - a quick spike to gather information
 - a story to do the real work.
3. Disaggregate it into smaller, constituent stories

Small Stories

- Good stories tend to be small.
- Stories typically represent at most a few person-weeks worth of work.
 - Some teams restrict them to a few person-days of work.
- Above this size, it seems to be too hard to know what's in the story's scope.

Testable Stories

- A good story is testable.
- Writing a story card carries an implicit promise: "I understand what I want well enough that I could write a test for it."
- Several teams have reported that by requiring customer tests before implementing a story, the team is more productive

User Story

- ... is short, simple description of a feature told from the perspective of the person who desires the new capability (typically user or customer)

- User stories can be written informally:

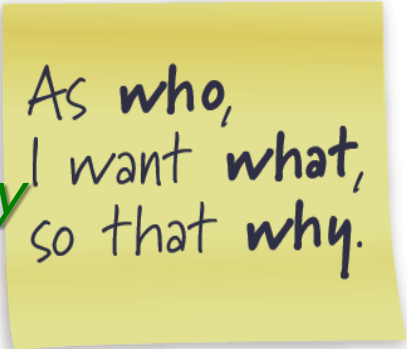
Registered users can reset their password

- Or use a bit more formal template

As a registered user,

I want to reset my password,

so that I can get back into the site if I forget my password



As who,
I want what,
so that why.

User stories

- Eliciting requirements from customers/potential user
- Requirements specification
- Expresses one specific need that the user has
- Example
 - As a customer, I want to register into the system.
 - As a customer, I want to login and access my account.
 - As a customer, I want to browse the available bikes.
 - As an administrator, I want to....
 - As a manager, I wan to...

Acceptance Criteria ...

- Bring the project from *"It Works as Coded"* to *"It Works as Intended"*
- Are conditions that a story must satisfy to be accepted by a user, customer or other stakeholder (PO in Scrum)
- Are a set of statements, each with a clear pass/fail result, that specify both functional and non-functional requirements
 - Functional example: *When a user clicks on the 'Reports' dropdown, a list of available reports will be displayed.*
 - Non-functional example: *Form edit buttons will be blue, and Form workflow buttons will be green.*

Source: <http://www.seguetech.com/blog/2013/03/25/characteristics-good-agile-acceptance-criteria>

Story Estimation Technique

- **S, M, L and XXXXL**
- Each estimator have four cards S, M, L and XXXXL (epic)
- Each estimator privately selects one card to represent their estimate for a story. All cards are revealed at the same time
- If consensus, that will be the estimate
- If not, discussion will lead to re-estimation until consensus

Story Estimation Technique

- **Planning Poker** is a consensus-seeking approach
- Each estimator have a deck of Planning Poker cards with values like 1, 2, 5, 8, 13, 20, 40, 100 and ? (I don't know), coffee cup (I want a break)
 - The values represent number of story points, ideal days, hours, or other unit in which the team estimates.
- Each estimator privately selects one card to represent their estimate for a story. All cards are revealed at the same time
- If consensus, that will be the estimate
- If not, discussion will lead to re-estimation until consensus

Product Backlog Example

- Story example where feature description is specified in “How to demo” field, i.e. test steps description (acceptance criteria) (Kniberg p. 10)

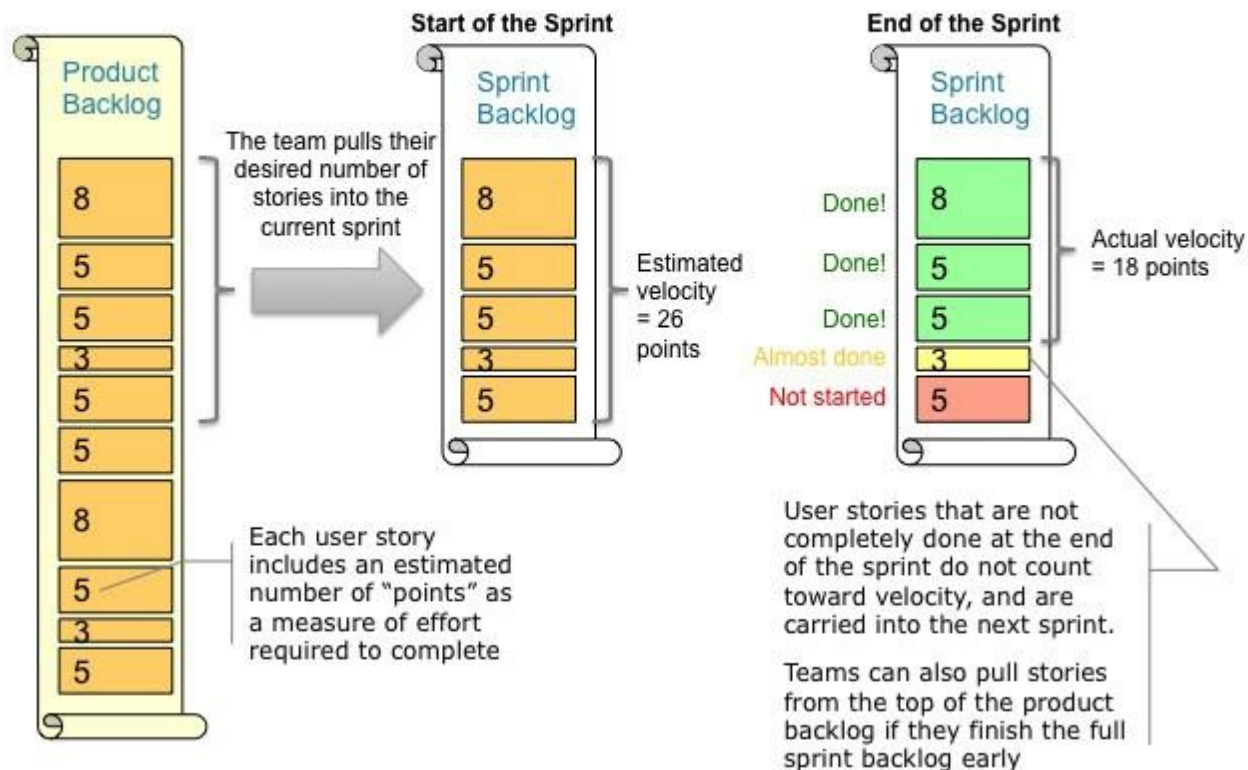
PRODUCT BACKLOG (example)					
ID	Name	Imp	Est	How to demo	Notes
1	Deposit	30	5	Log in, open deposit page, deposit €10, go to my balance page and check that it has increased by €10.	Need a UML sequence diagram. No need to worry about encryption for now.
2	See your own transaction history	10	8	Log in, click on “transactions”. Do a deposit. Go back to transactions, check that the new deposit shows up.	Use paging to avoid large DB queries. Design similar to view users page.

How to estimate?

- Committing stories into a sprint
- Gut feeling
- Velocity calculation

Velocity Calculations

"Velocity" is the Key Metric in Scrum



scruminc.

Adapted from materials by Henrik Kniberg

© 2012 Scrum Inc.

Velocity Calculation

The best way to determine a reasonable focus factor is to look at the last sprint (or even better, average the last few sprints).

LAST SPRINT'S FOCUS FACTOR:

$$(\text{FOCUS FACTOR}) = \frac{(\text{ACTUAL VELOCITY})}{(\text{AVAILABLE MAN-DAYS})}$$

Actual velocity is the sum of the initial estimates of all stories that were completed last sprint.

Elephant Carpaccio



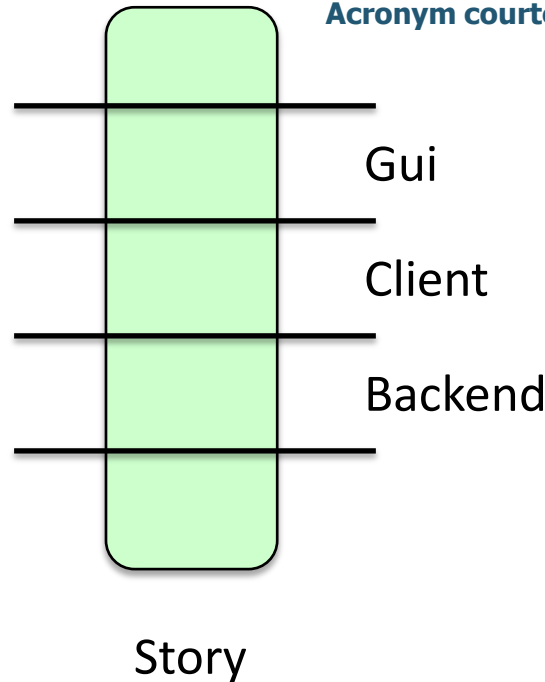
A (User) Story

Vertical
Testable
User-valuable

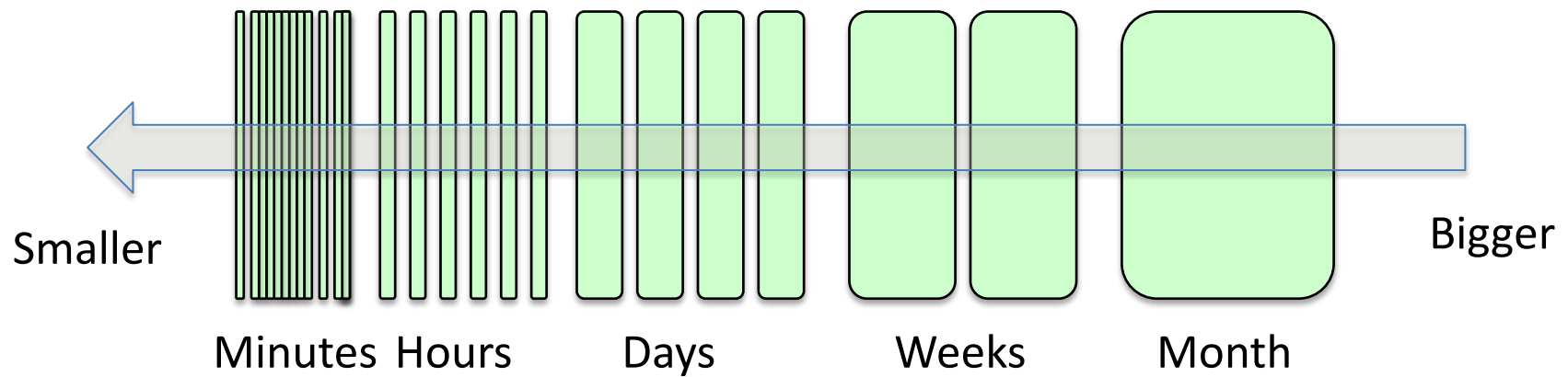
Kniberg

Independent
Negotiable
Valuable
Estimable
Small
Testable

Acronym courtesy of Bill Wake – www.xp123.com

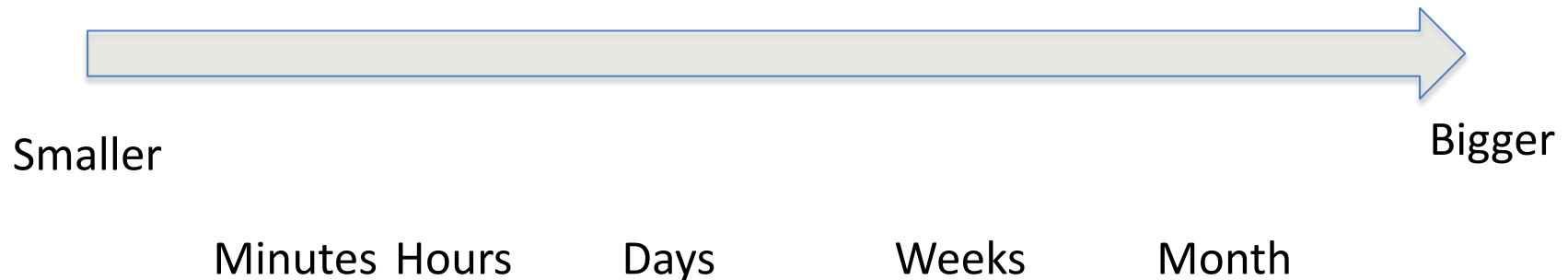


Story slicing – making thinner but vertical



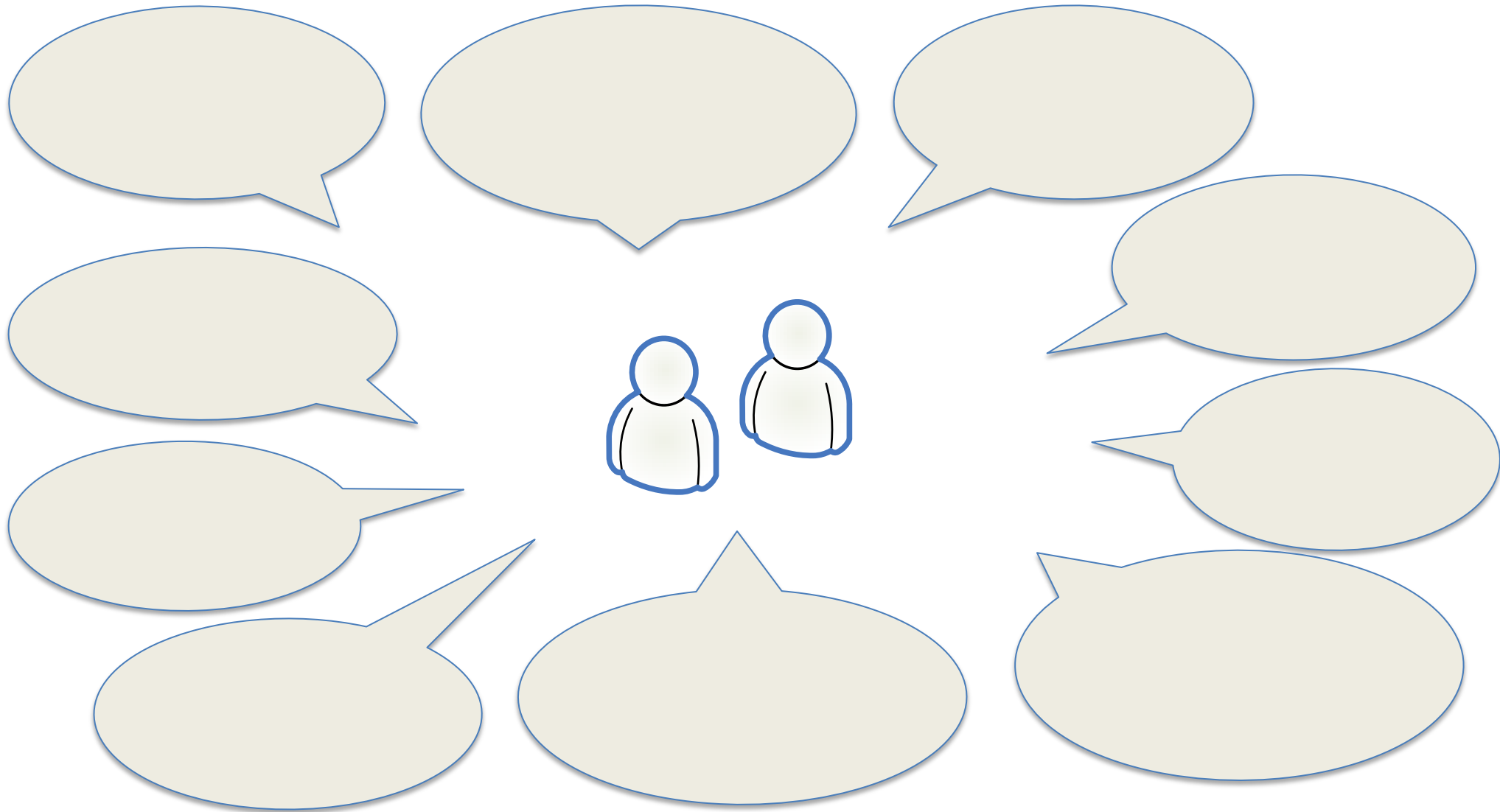
How big are your user stories?

Target
Story = A few days
Task = a few hours
Commit = several a day

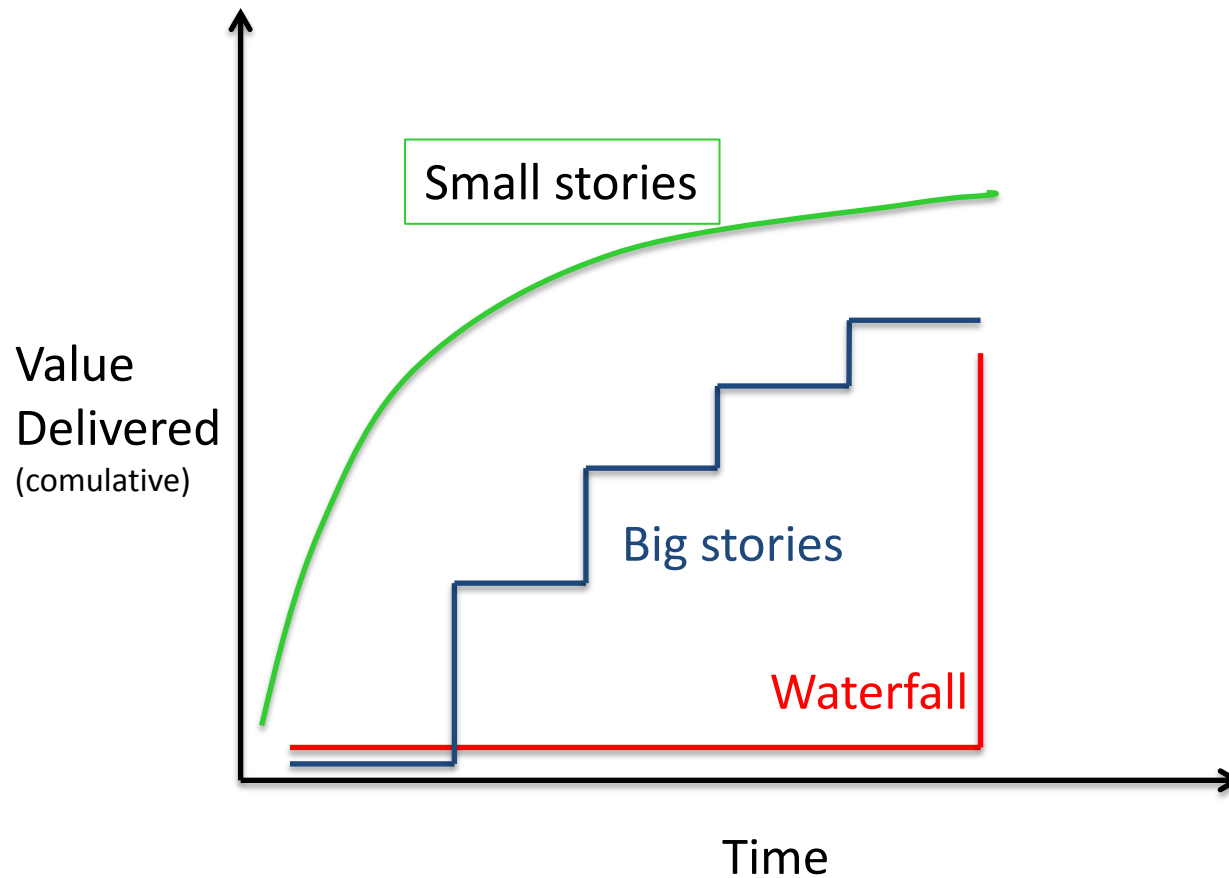


Why split user stories?

Discuss in pairs

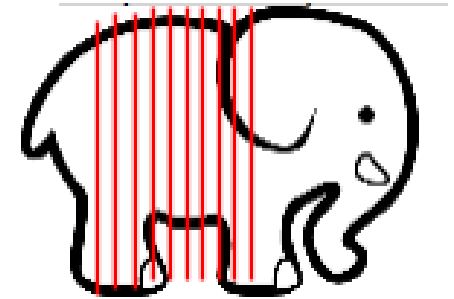


Value curve



We will build

- A a simple retail calculator
- Using any programming language
- It takes input, and omit result.
 - Any interface is OK (command, web, gui)
- 40 minutes in 5 iterations x 8 minutes
- We will do it by slicing the story in 15 – 20 pieces
- Elephant carpaccio = very thin slices, each one still elephant-shaped. Together they form the whole elephant.



Elephant Carpaccio

Background

Your sales department has asked you for a quoting application. They wrote this user story

“As a sales person, I want to quickly get the final price of an order”

Order value	Discount rate		State	Tax rate
\$ 1,000	3 %		UT	6.85 %
\$ 5,000	5 %		NV	8.00 %
\$ 7,000	7 %		TX	6.25 %
\$10,000	10 %		AL	4.00 %
\$50,000	15 %		CA	8.25 %

Mattis Skarin

Usage

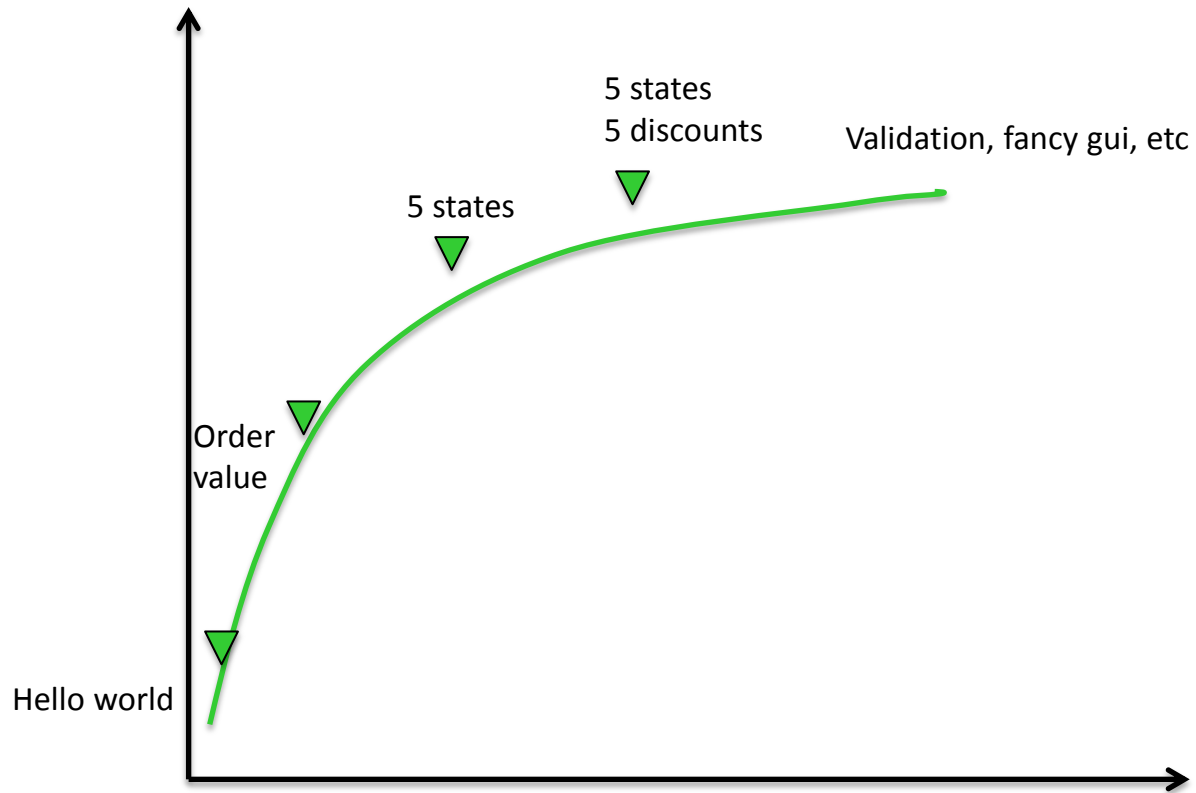
Our sales support system should accept three inputs

- ❑ number of items
- ❑ a price per item
- ❑ a 2-letter state code

Compute the price of the order, given the discount and state tax.



Priorities



Create backlog

- In pairs
- 10 + 5 minutes
- Write 10 – 20 demoable user stories
- Each should be implementable in 2 – 6 minutes
- Real input and output and different from last

What is your first slice?

When have you an order value?

Whats next slice after order value?

Minimum key strokes per slice

One way of slicing it

1. Echo input

Verify we can read input

2. Walking skeleton

3. Order value

2 inputs, 1 output. No state tax, no discounts.

Order total

4. Hard-coded state tax

Still 2 inputs, 1 output. Utah state tax added automatically.
(Now we can go live in Utah!).

Enable tax in one state

5. Manual tax (3 input: price per item, # of items, state tax %).

Output is order value with state tax added.

User inputs actual tax rate rather than state code.

Enable manual tax

This gives simpler code (no internal data structure to map state code to tax rate), so we get faster delivery.

6. State ID (3 input: price per item, # of item, state tax)

but only two states are supported. Any other gives error message.

Enable tax. Ship all states

7. Add the other 3 states

(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)

Discount for core customers

8. Once discount

9. 2 discounts

Discounts for all levels

10. All discounts

Decimal numbers?

Build it

- 5 iterations, 8 minutes per iteration
- I will shout “demo time” after each iteration
 - Do a fast demo to another team
 - Next iteration starts immediately
 - Shout slice for each slice you have done

Review

- How far did you get on the value curve?
- How many slices do you have?
- Acceptance test
 - Start the application
 - I am in Utha
 - Buying 978 items
 - Each cost \$270.99
- Compare result

Order Value: \$ 265 028.22
Tax: \$ 283 182.65
Tax & Rebate \$240 705.26

Debrief

- What was it like?
- Round robin
 - What did you learn?
 - What will you do?