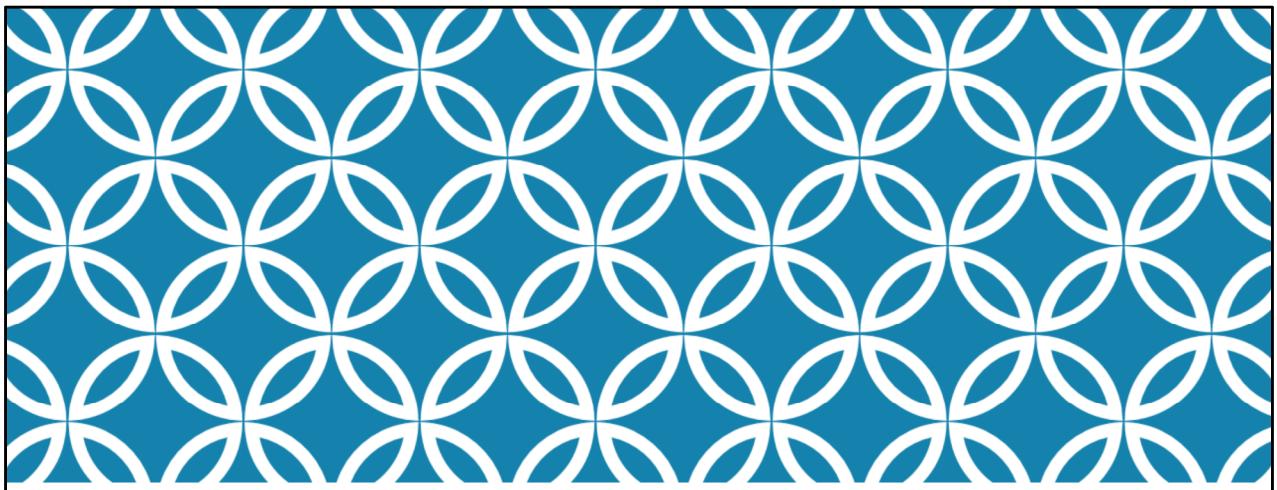


USER STORY WORKSHOP

WHAT BRINGS US HERE TODAY?

Who is our focus and why?

What are some techniques for structuring the work of the team?

How can the team define and refine their work?

DISCUSSION: TERMINOLOGY

Group Conversation

- How do you use these terms in your team?
- What are the similarities, differences, are they interchangeable?

User Story

Feature

Product Backlog Item

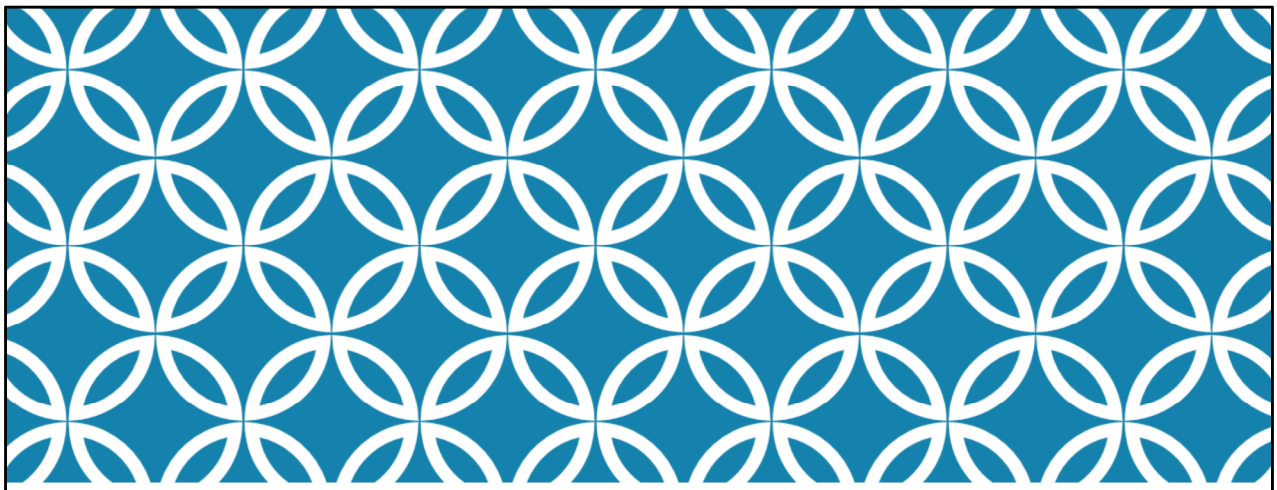
Work Item

Ticket

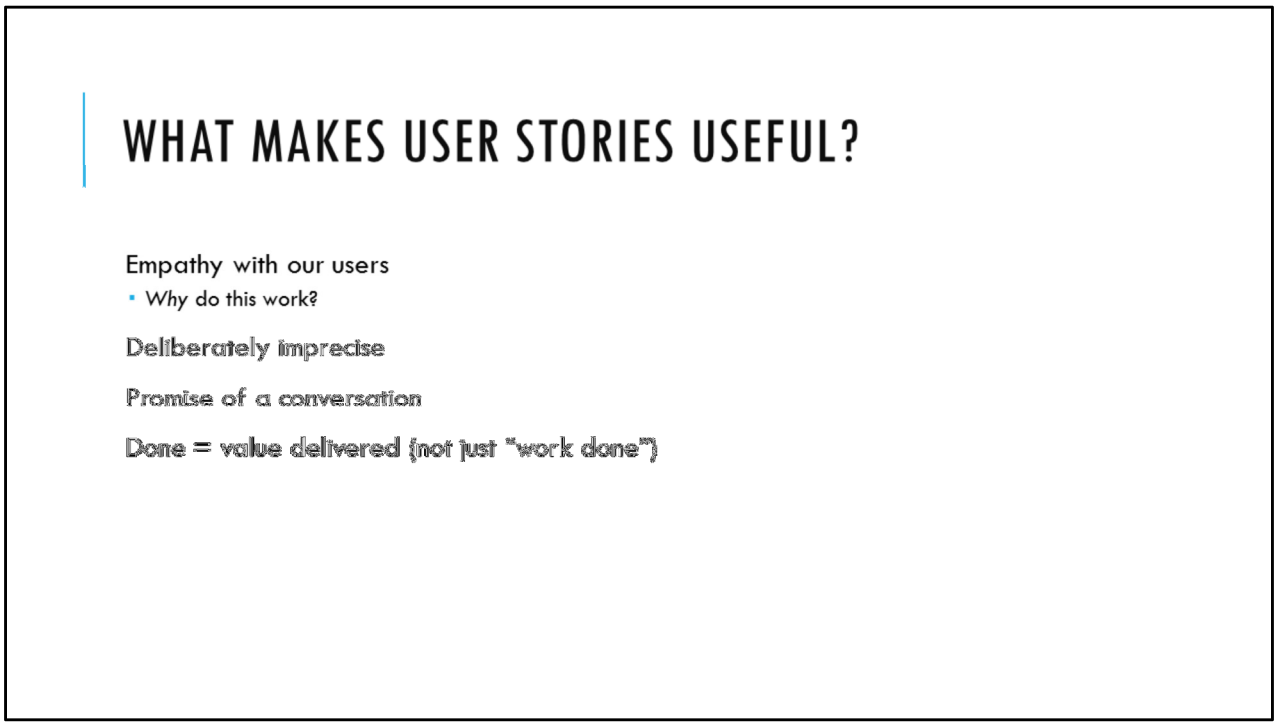
Requirement

Use Case

This helps set the stage for what everyone means when they use certain terms in other discussions throughout the course (including the instructors!)



WHO IS OUR FOCUS AND WHY?



Empathy with our users

This is the *user's story*. They've communicated the 'ending' (outcome), and we need to fill in the action to make it happen!

This is our opportunity to understand the value the user sees in the 'ending' (outcome). Refining the user's desires into a user story builds that understanding.

Deliberately imprecise – don't add detail until it is needed (one to two sprints ahead of time)

Why? Because if you add detail too early, you will probably have to rework it later (wastes your time and energy)

Why? Because if you add detail too early, you will have to revisit it later anyway to make sure you understand it again before estimating/bringing it into the team's workflow (redundant work)

Placeholder (just enough detail to get you off on the right foot) until the work is prioritized to be taken in in the next iteration or two

Promise of a conversation

A face-to-face conversation is a low-cost, high-bandwidth way to communicate.

Done = value delivered to the user, not just 'checked off' requirements

Because it is a *user* story, the conversation is more about the value brought to the user, and therefore not just a series of tasks but really a series of steps to deliver

value at the end of the story.

Let the participants "interpret" what is happening here…and provide them alternate interpretations if they aren't playing along. ☺

British Left Waffles on Falkland Islands
- Did the British leave waffles on the Falkland Islands? Or did the British 'political left' waffle on the issue of the Falkland Islands?

Shot Off Woman's Leg Helps Nicklaus to 66
- Did someone detach a woman's leg with a gunshot to help golfer Jack Nicklaus's score? Or did a golf ball bouncing off of a woman's leg help him out?

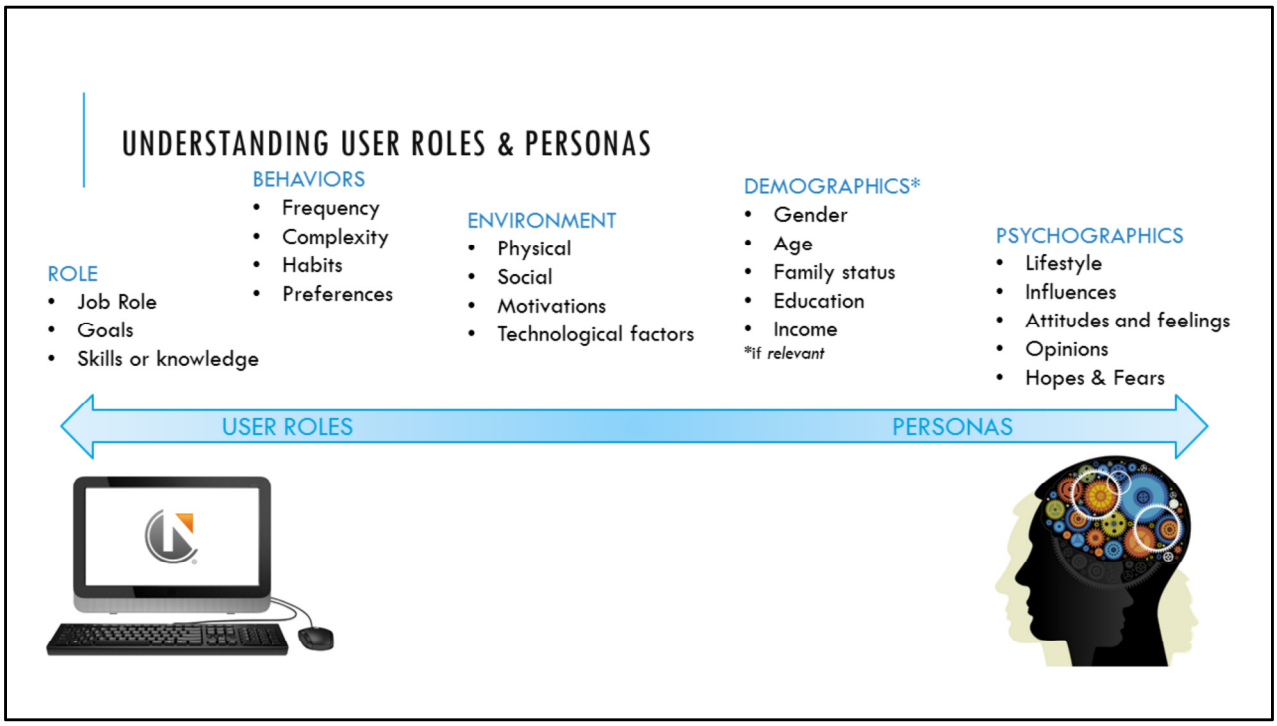
Stolen Painting Found by Tree
- Did a sentient tree help out with a criminal matter? Or was the stolen painting discovered in close proximity to a (normal) tree?

Enraged Cow Injures Farmer with Axe
- Did a cow with a temper take an axe to a farmer? Or did a farmer with an axe have a rough encounter with an angry heifer?

Why are we talking about cows and legs? The written word leaves a lot to interpretation
- it lacks the context of spoken conversation, including nonverbal signals (gestures, tone, facial expression, body language)
- No quick answers to clarifying questions or questioning looks from the recipient of the

message

- as the person delivering the message, you get more of a chance to correct if you see your audience getting confused or off track when you're *verbally* communicating



User Roles = Understand the Job

Personas = Understand the Psychology (This isn't *always* needed – be practical about what will influence how you build the software)

Don't just SAY "As a sales rep…" – KNOW what a sales rep does and who she is.

A note on "technical stories", or "developers" or "system as the user

Is the work important if there isn't a non-developer user? How do you know? How does it rank against work with a non-developer user? What factors come into play?
Who benefits from the change you're proposing? Are *they* your user?
How do you "sell" the value of work that doesn't appear to benefit a user (at first glance)?
How often does your team have "technical" stories?

Mention addressing TD – again no direct tie to user features now, but makes future user features easier to add

Role:

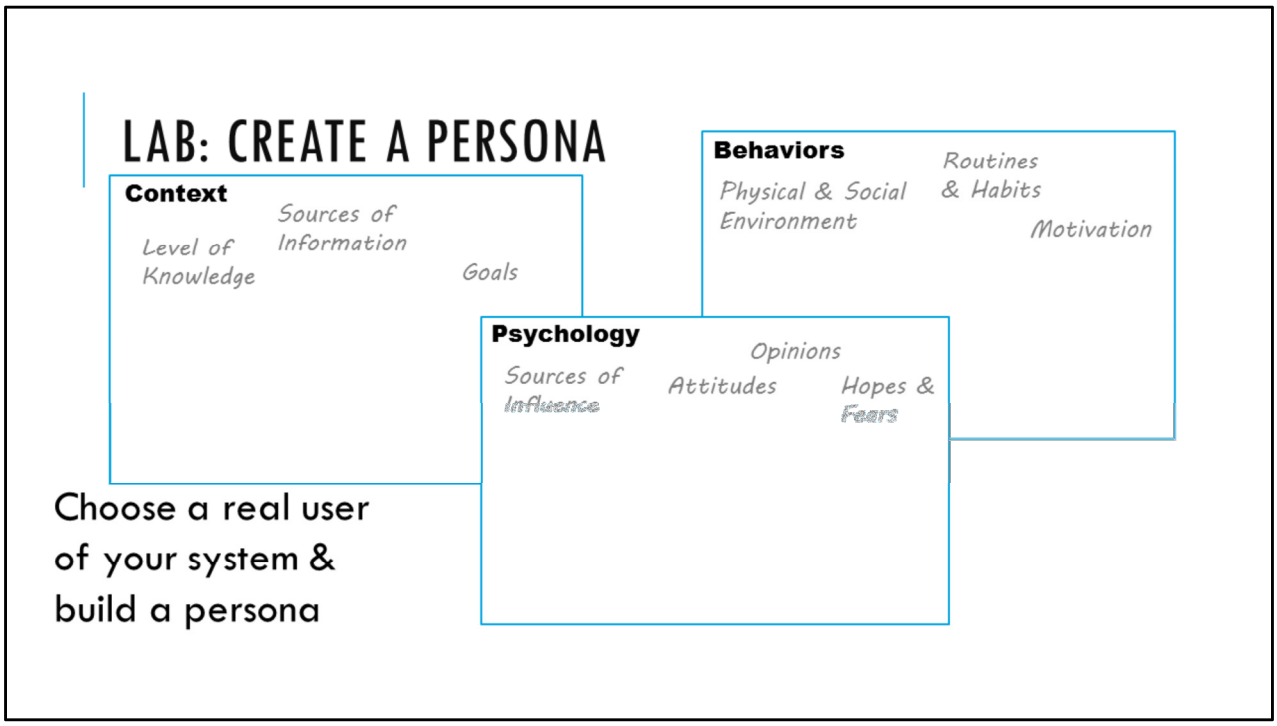
- How do they interact with the system?
- How does the system play a role in their natural/habitual workflow? How often do they interact with the system?
- What information is useful to them at the various stages of their workflow?
- How proficient and knowledgeable are they (in their jobs and with the system)?

Behaviors:
- What is their volume of work? Does this make them use the system differently?
- What ingrained habits exist, and are we trying to change them?
- What's the preferred way to do work and is that contextual?

Environment:
- What is their physical/social environment?
- Does it contribute to their ability (or desire) to use the system?
- What are the motivating factors within the environment (incentives, competition)?
- What gets this person excited / what drags them down?

Demographics:
- Keep in mind that including demographic factors can also lead to unhelpful assumptions or conversations…
- Most of these demographics don't apply to our users, but are there any that do?
- Is education relevant?
- Is there a certain kind of person that is routinely hired into this role? (graduating from XX role, or fresh hires from college)

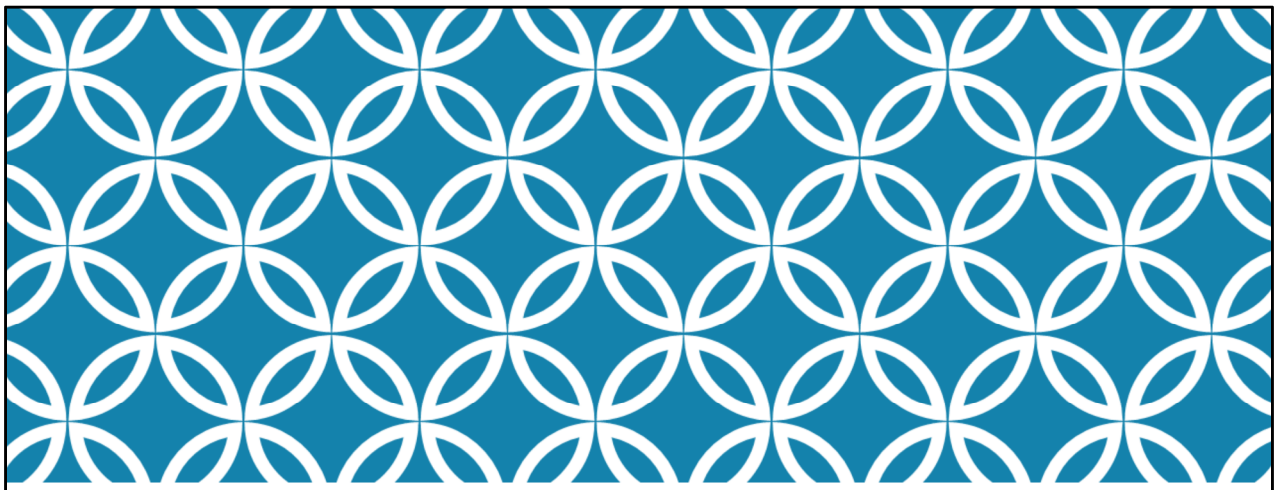
Psychographics
- Who are the influencers in their work and home life?
- What motivates them as a person?
- What are their life and career goals?
- How are they inspired to excel?



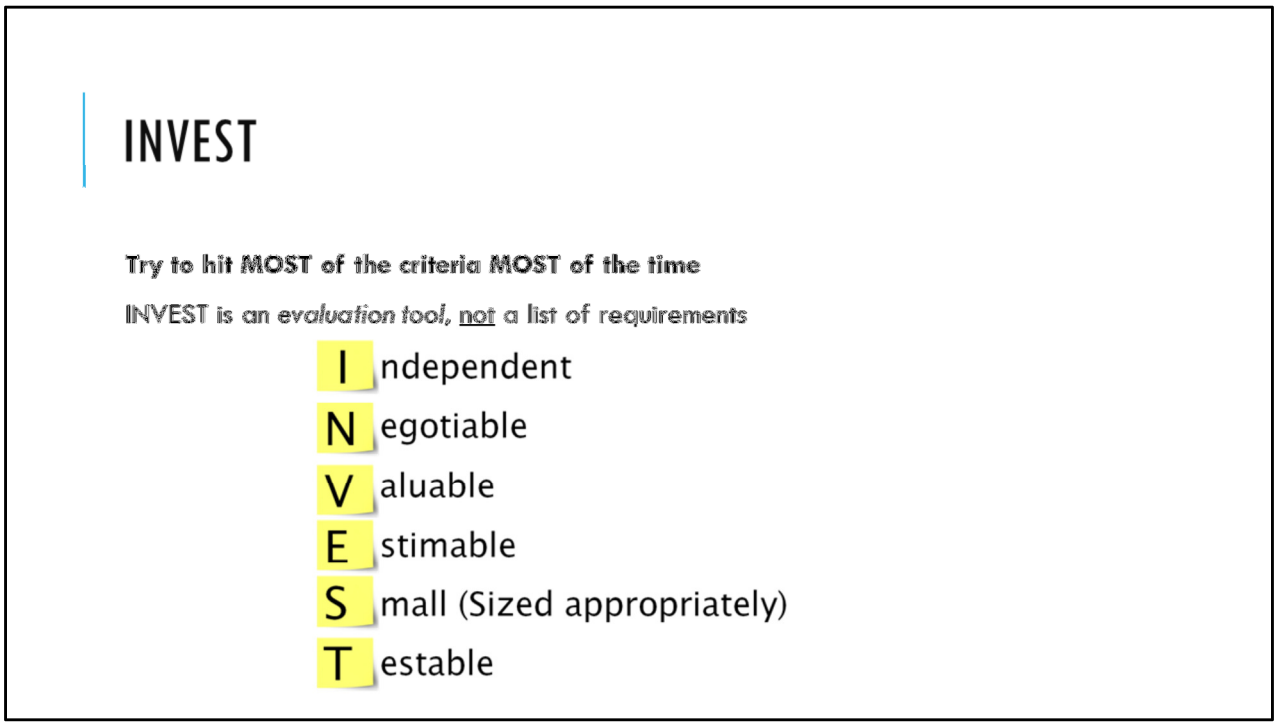
Context:
What do we know about their job?

Behaviors:
What about their physical or social environment encourages (or discourages) the desired behaviors?
What habits or routines are ingrained? Does this impact how we would build for them?
What motivates this person – what intrinsic motivations do they have? What extrinsic or structural motivations are in place in the work environment?

Psychology:
What drives this person – beyond their work life. Who and what influence the course of their life and career?
How does that bleed in to their work, and what adaptations can we make to create the best experiences for this person?
What are their attitudes about work? About life?
What opinions do they hold about their peers, leaders, work environment?
What hopes do they have for themselves – inside and outside of work?
What do they fear or become anxious about?



WHAT ARE SOME TECHNIQUES FOR STRUCTURING THE WORK OF THE TEAM?



http://www.scrumhint.com/invest-in-good-stories-and-smart-tasks/

INDEPENDENT
If a story is not independent, is there another way to split it?

Having a dependency because of the natural order of the way user's work is done isn't considered a "dependency" in this case

NEGOTIABLE
Having leeway to chose how something is done leaves room for different options in the face of problems, and more creativity in solving problems and designing the solution.

Be clear about what's absolutely necessary vs nice to have - It's great to know what can be sacrificed in a pinch.

VALUABLE
Why do work that doesn't bring the most value right now?

ESTIMABLE
Can you tell how big this piece of work is? Are there too many open questions, or is the scope too large?

SIZED APPROPRIATELY

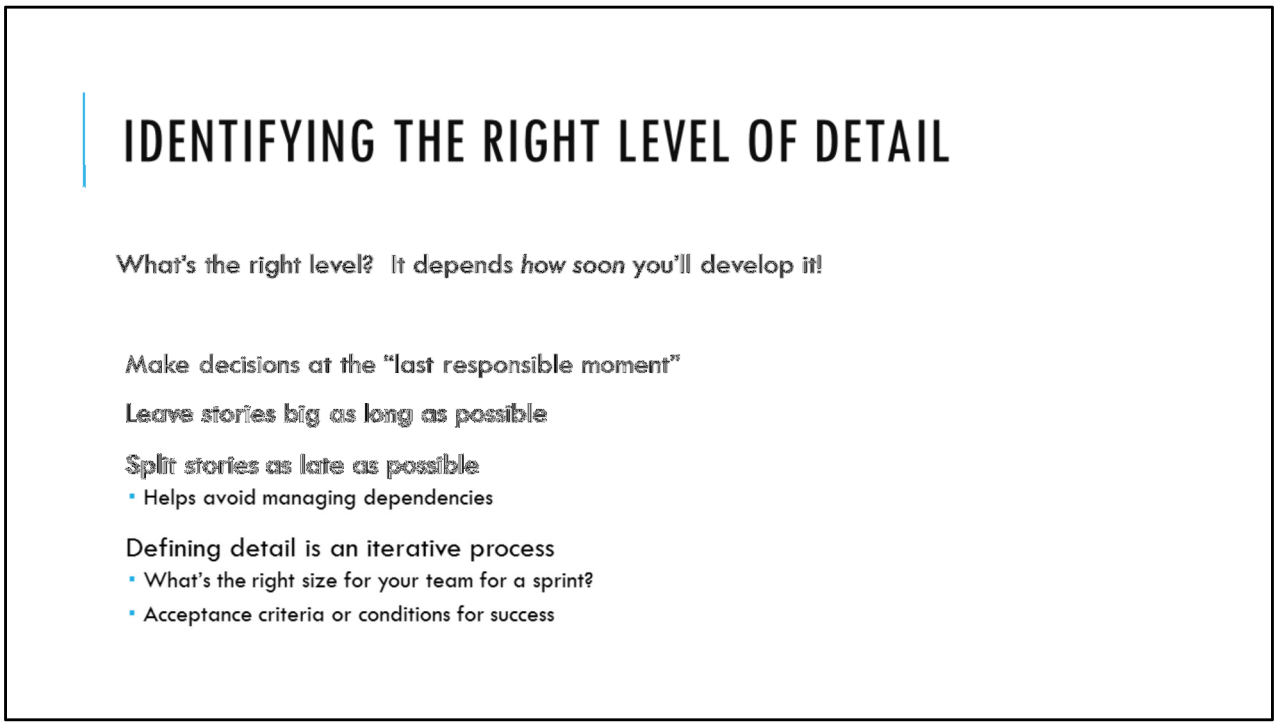
Why estimate? (Helps us plan based on past metrics of stories completed, know when a work item might be too large)

Sized appropriately also means trying as hard as you can to break down your pieces of work so that they can be completed in one iteration or less – if you carry work from iteration to iteration, it is hard for your VO/PO to reprioritize

BUT – the reason we say "sized appropriately" instead of just "small" is because sometimes small is *not* better – sometimes larger pieces of work are more efficient, easier to understand, make for fewer dependencies, etc

TESTABLE

Can it be tested? If not, how will we know it's "good to go"? Is there a way to make it testable (shrink it, make it bigger, combine it with something else, use a special testing tool)?



The "Last Responsible Moment" is that point in time where delaying a decision any further will result in too much risk for the reward of additional information.
But to make the decision sooner means a lost opportunity to have more (time to gather) knowledge about the decision.

What's the right size and detail for YOUR sprint?
- Half your velocity? A quarter?
- What size of stories seem to flow smoothest?
- General advice on size/detail:
  - Is no more than ½ the team's velocity
  - Small enough that each person can do at least 1-2 stories per sprint
  - Identifies the crucial "Acceptance Criteria" or "Conditions for Success"
  - Is something your user can actually use

There are differing opinions on how small stories should be (in ideal scenario) – some say 1/6 to 1/10 of velocity. Do some math based on number of team members – 4 people x 2 stories = 8 per sprint (good workflow), so each story should ideally be smaller than 1/8 of velocity

What should be in Acceptance Criteria:

- If “these” things were missing, a Value Owner would reject the work

What **shouldn’t** be in Acceptance Criteria:

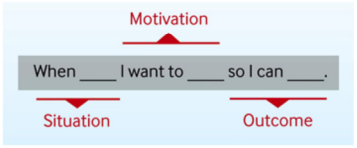
- Things that are apparent by a screen shot
- Things that are negligible to the functioning of a story and that people can make good judgements on if they know the context of the work.
- Things that are obvious – but it bears discussing with the team what is “obvious” to different people!

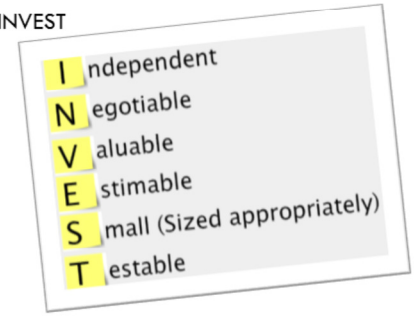
LAB: TRY INVEST & USER STORY FORM

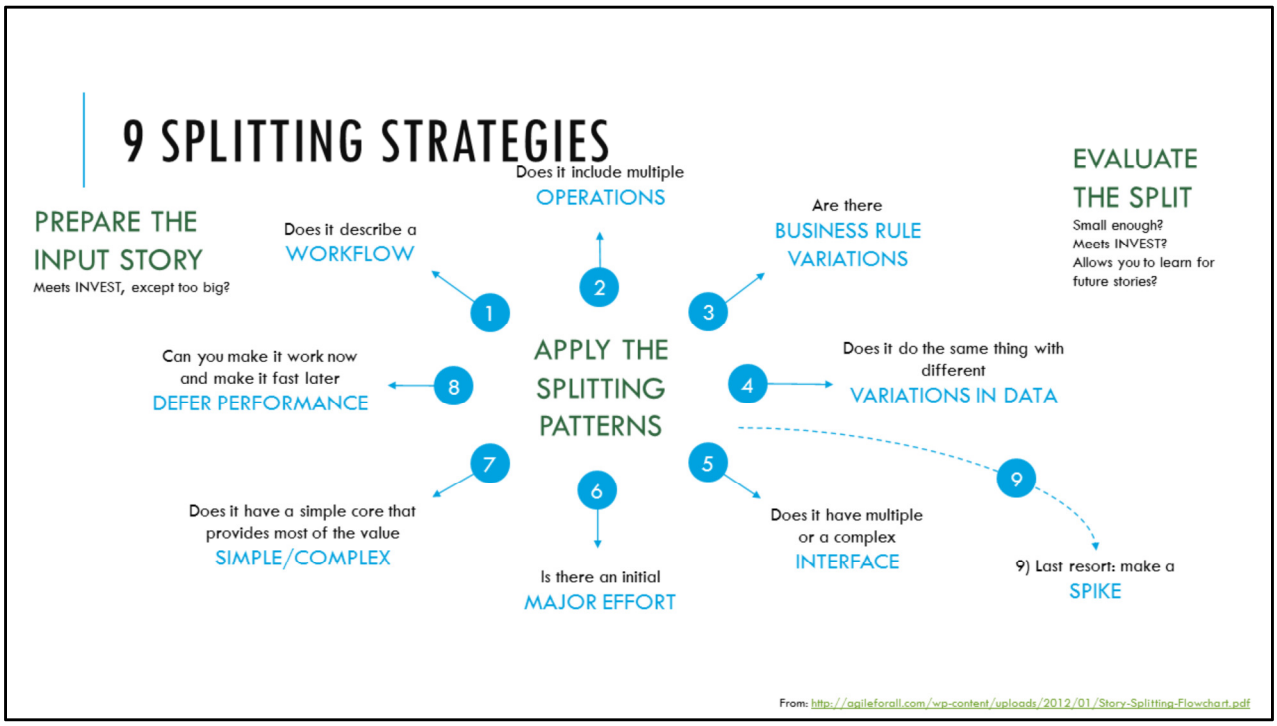
As a customer, I want to be able to run your product on all versions of Windows from Windows XP on.

Using an item from your team's backlog…

1. Write it in User Story format
2. Add some acceptance criteria
3. Evaluate it against INVEST







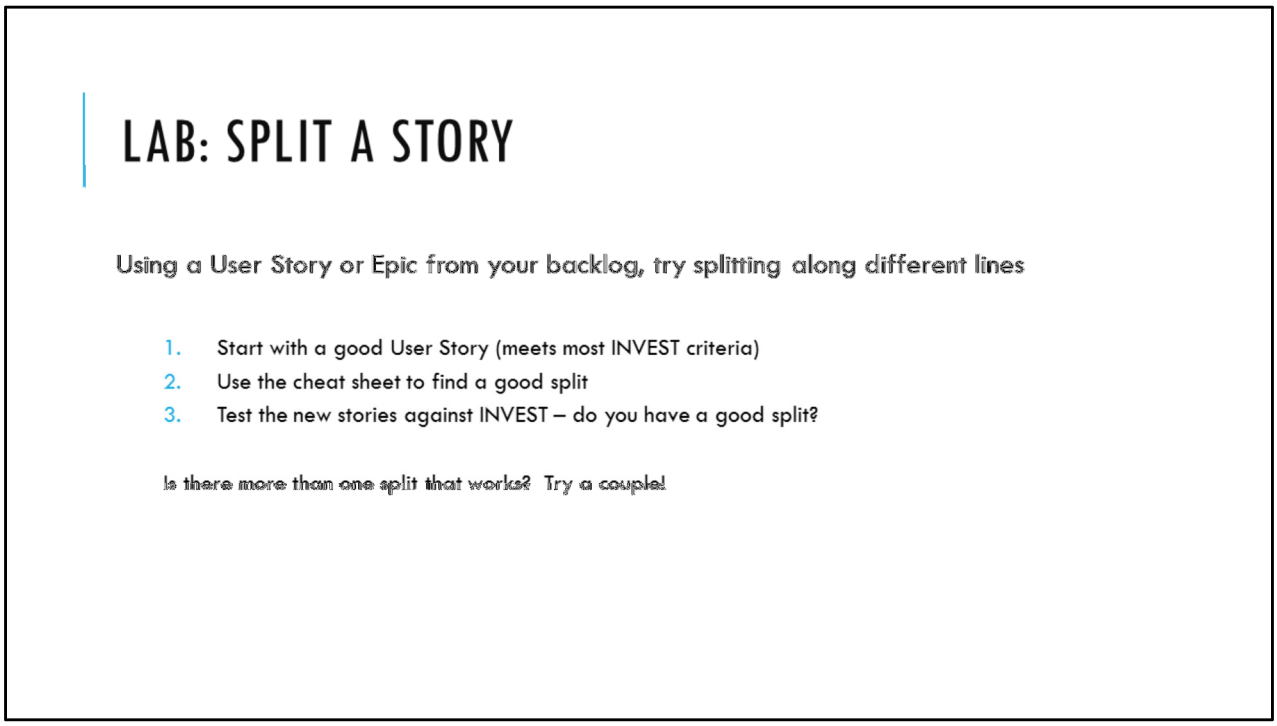
See handout for more details. http://agileforall.com/wp-content/uploads/2012/01/Story-Splitting-Flowchart.pdf

About Major Effort: Mike Cohn would disagree with the Major Effort split, but it does seem logical if that's the way you have to do it.

- Idea is you would do the first thing that builds the infrastructure, and then the other story can have "do all the others" as a single, somewhat equally sized story.
    - "I can pay with 1 credit card type (choose which as late as possible)"
    - "I can pay with all credit card types"

About Spikes: Spike doesn't build functionality, it builds *knowledge*

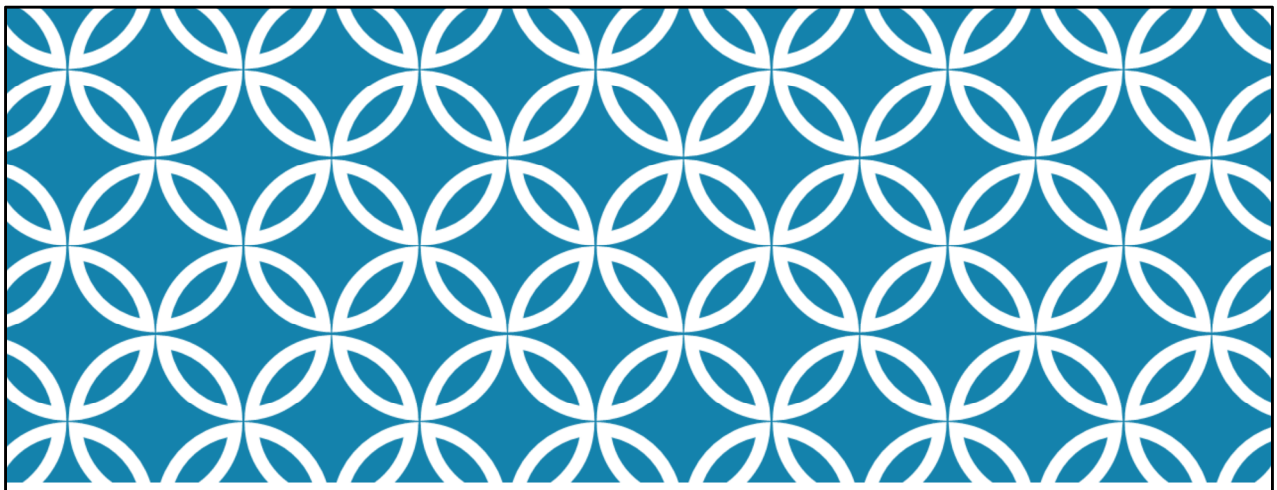
- Go into the spike with 1-3 questions that are most holding you back – do the minimum to answer those
- Time box spikes - avoid analysis paralysis.
- Spikes are not meant to remove ALL uncertainty. Use them sparingly.



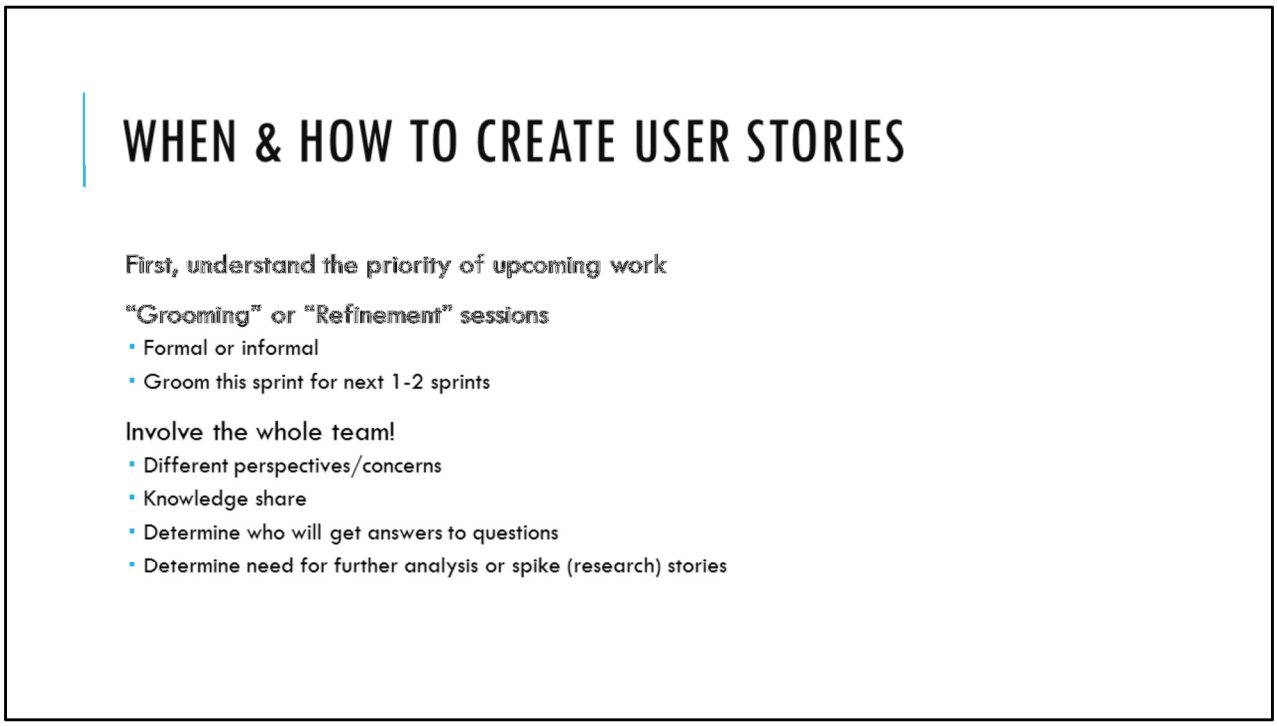
When you think about the absolute simplest way to implement something, postpone the judgement about value. Just identify the simplest way.
https://gojko.net/2012/01/23/splitting-user-stories-the-hamburger-method/

When you can't Split, Mike Cohn says:

1. Try harder (do it again if you've already tried) - Get help
2. Take multiple iterations to mull it over and see stories flow through your team - evaluate now that you know more if there really was a way to split it.
3. Feel Guilty ☺



HOW CAN THE TEAM DEFINE AND REFINE THEIR WORK?



If you don't know the priority, it will be harder to refine the right stories in the right timeframe.
To improve your understanding of priority, work with value owner/product owner/delivery lead/users

You don't want to add detail too far ahead – remember, we want to reduce re-learning and rework

- You learn a lot in 1-3 sprints, some of which affects future stories and how they are split or documented

Additionally, if you refine too late, you lose opportunities to get good answers, vet the work with the team, and understand dependencies/technical approaches

Grooming/Refinement sessions

- When looking to add detail, generally be only 1-2 sprints ahead of when you think you'll take the work in

Getting the whole team involved is ideal

- creative questions from different perspectives (roles)
- senior people can pass knowledge to less senior
- SMEs can share knowledge with non-SMEs

- determine what questions need to be answered for the team to feel comfortable estimating and taking the work on
- Determine if spikes are needed for technical analysis

STORY MAPPING

Organize a larger effort with a Story Map

Who? VO/PO/DL, Team

When? Whenever it makes sense.

- Common at the beginning of projects
- Also useful to organize smaller, complex parts of a project.

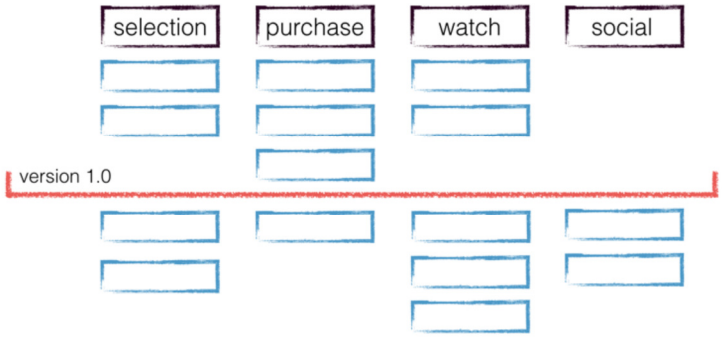
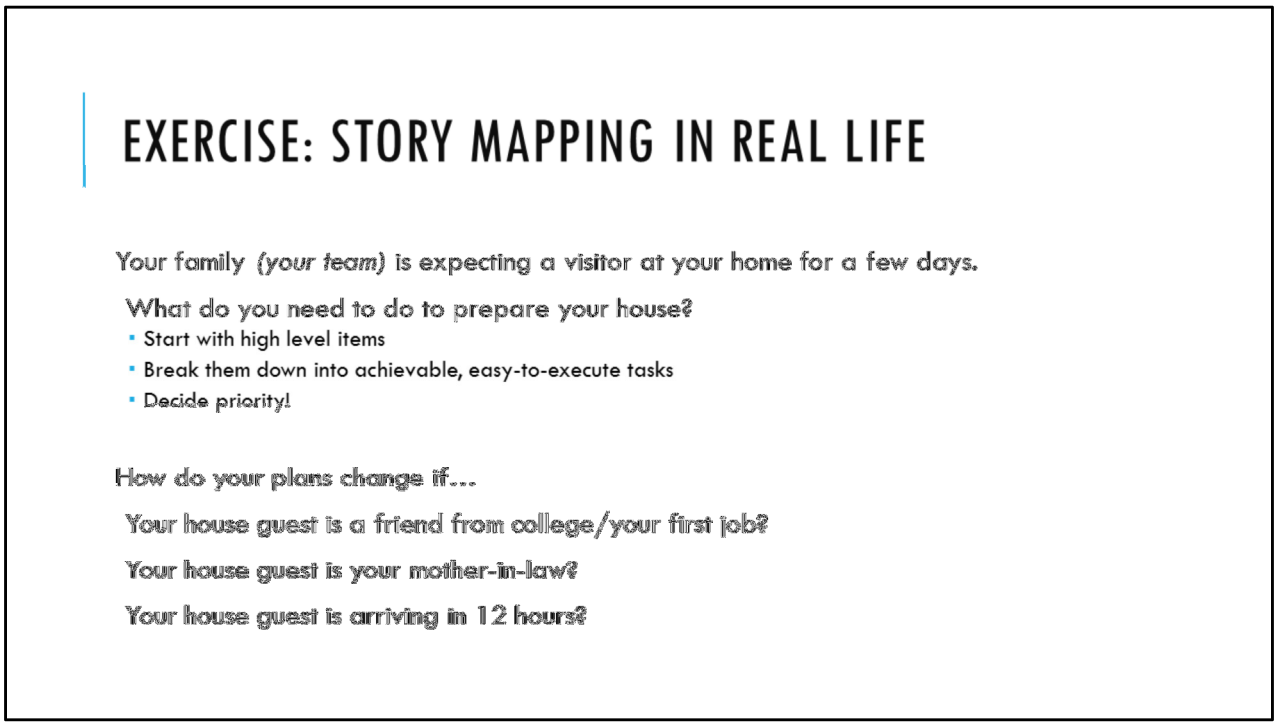


Diagram from: https://blog.seibert-media.com/2017/01/30/agile-user-story-maps-understanding-clients-needs/

Black items are high level goals, in natural order for completion (if that's possible) ("super-epics" or "epics")
Blue items are separate work items that would need to happen to make (the beginnings of "user stories", or perhaps still a little more on the "epics" side)

- Blue items should be listed in order of importance or value

Draw a line (red here) to indicate MVP (minimum viable product) or MMF (minimum marketable feature) or Phase 1 or Version 1 or whatever your team likes to say



It'll take too much prep to do an actual story map in class (need users defined, a well define project, a stack of user stories) – but we could follow up with them after they've had a chance to build that stuff out.

Simulation ideas:
Building a house (a bit overdone in our analogies, but works)
Planning an event
Washing your car
Preparing for a house guest – this could be fun as far as "what's minimally viable?" and who is the user/guest (MIL, friend, sibling, family with kids)?

For some ideas of MVP (or not) these are actually pretty funny:

From the 50's: https://nonfictioness.com/2016/01/14/how-to-be-a-good-hostess-1950s-manual/

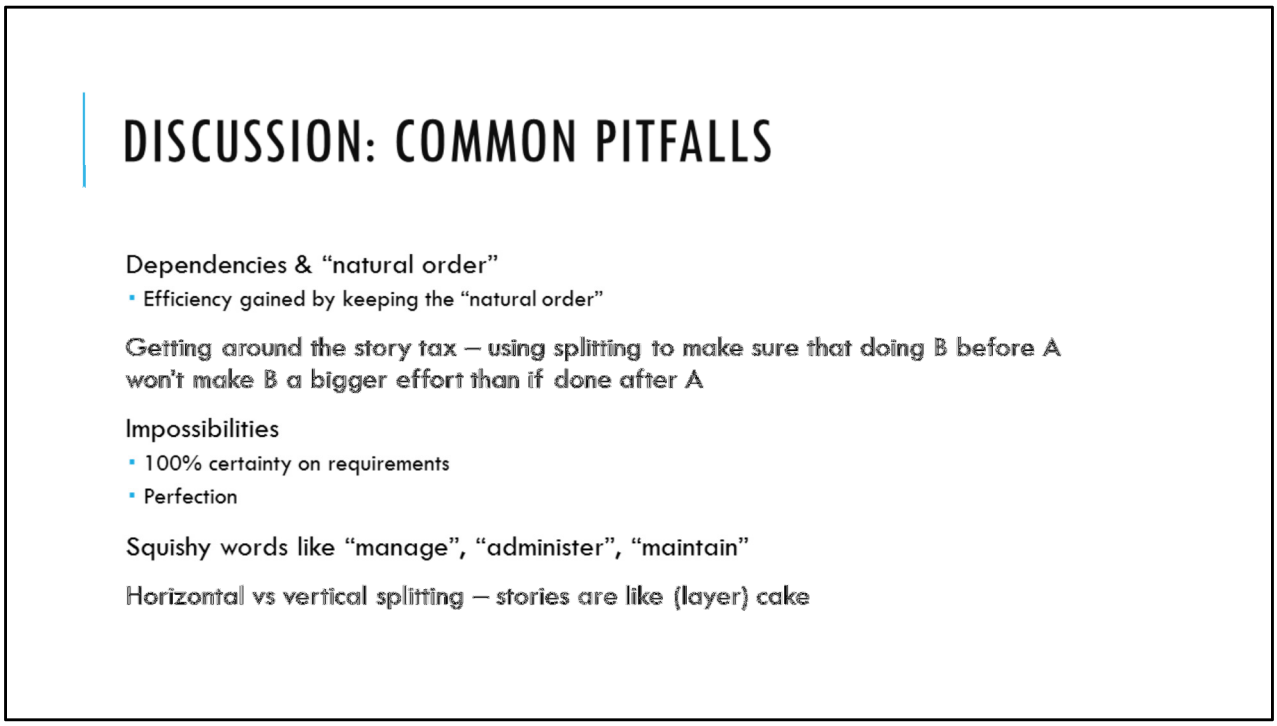
From 2010: http://lifehacker.com/5606282/how-to-be-the-perfect-host-in-the-21st-century

MVP vs MTF vs MMF – just a different way to frame it

Audience – 1 house guest or 10 house guests makes a big difference

Risk – test earlier on (in production, with users) if the risk is higher
- Jurgen Appelo: I'm terribly embarrassed to release this (an early version of an agile coaching app, Mind Settlers), but I need feedback! I know there are shortfalls, but I need to learn something from you, so I'm putting it out there for you to interact with it and review it.
- Mother in law is a more high pressure audience than a friend

Market window – 12 hours is a lot shorter than a week – have to be really smart about what you choose to do in that time (and where, and how)



Respect the natural order – better to have an item catalog before a shopping cart – hard to add an item to a cart when there are no items. ☺ Or better to have ability to select a carrier before you have the ability to book the carrier, because otherwise what exactly are you booking? Natural order is not the same as dependency.

Story tax
- In building Feature X, have to lay out Foundation B. So if we want to accept credit cards, we have to lay out the functionality to accept any credit card FIRST, which means that the first credit card (say, Visa) we wire up necessitates more effort than the other types of credit cards (say Mastercard and Discover), or we might group the later credit card work together because otherwise they're uselessly separated and don't deliver much value on their own. Have to be clear with PO that choosing a smaller future "non-taxed" item first shifts the tax to the one done first, no matter which one that is.
- Think DevOps, too – the first time you put something through the pipeline, there's a lot of setup that you don't have to do for future enhancements/features for that same service. So the first effort is, by nature, larger.

NEXT STEPS

Try incorporating refinement sessions into your team workflow!

- Cadence and format vary widely team to team

When refining and estimating, reference INVEST and story splitting worksheet

- Don't forget you can adapt these techniques for projects and epics too!

Try story mapping to break a project or epic into helpful pieces, and determine how to get the most value to the user soonest