Building Better Products Using

User Story Mapping

Jeff Patton

jpatton@acm.org

www.agileproductdesign.com

Our goals and agenda today

Goal: Learn to use the user story backlog as a way to describe user's experience with your product

Part 1: Mapping user stories

- User story essentials
- Telling stories about the user experience
- Mapping user stories based on experience

Part 2: Planning valuable incremental releases

- Identifying product goals that delivery value
- Slicing the story map into valuable releases

Part 3: Iteratively constructing software (time permitting)

- *Identifying an iterative and incremental construction strategy*
- *Splitting and thinning stories for upcoming development iterations*

Starting with the **User Story**

What do you know about user stories?

What do you like about user stories?

What causes you trouble with user stories

User Stories are multi-purpose

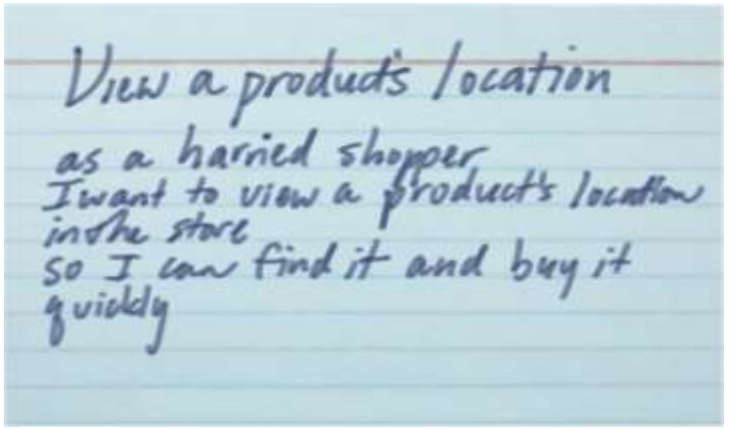
© NBC Studios

User Stories are multi-purpose

Stories are a:

- User's need
- Product description
- Planning item
- Token for a conversation
- Mechanism for deferring conversation

** Kent Beck coined the term user stories in Extreme Programming Explained 1st Edition, 1999*



Stories gain detail over time

Start with **a title**

Add a **concise description** often using this useful template:

As a [type of user]

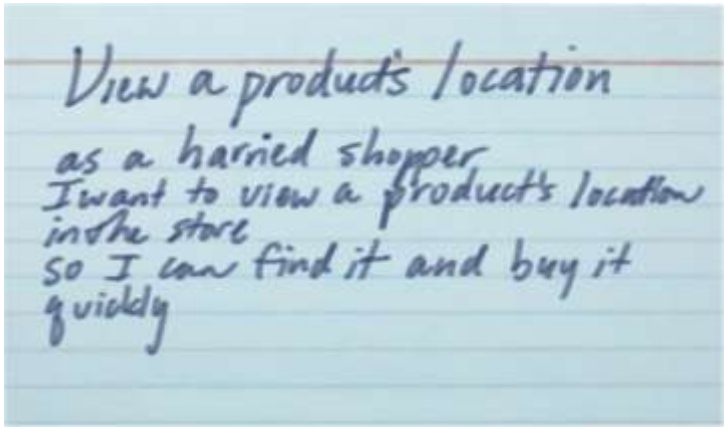
I want to [perform some task]

so that I can [reach some goal]

Remember — that's just a thinking template. No need to write all your stories this way.

Add other relevant **notes**, **specifications**, or **sketches**

Before building software write **acceptance criteria** *(how do we know when we're done?)*



Agile customers or product owner prioritize stories into a backlog

A collection of stories for a software product is referred to as the **product backlog**

The backlog is prioritized such that the most valuable items are highest

Let’s talk about the nature of multi-purpose things

(yes I’m going meta. I blame Brian.)

User Stories are **boundary objects**

"A boundary object is a concept in sociology to describe information used in different ways by different communities. They are plastic, interpreted differently across communities but with enough immutable content to maintain integrity" -- Wikipedia

"They are weakly structured in common use, and become strongly structured in individual-site use. They may be abstract or concrete. They have different meanings in different social worlds but their structure is common enough to more than one world to make them recognizable means of translation. The creation and management of boundary objects is key in developing and maintaining coherence across intersecting social worlds." -- Leigh & Griesemer

But size always matters…

How big is the story we want to talk about?

And, it's easy to get lost in the sheer number of them

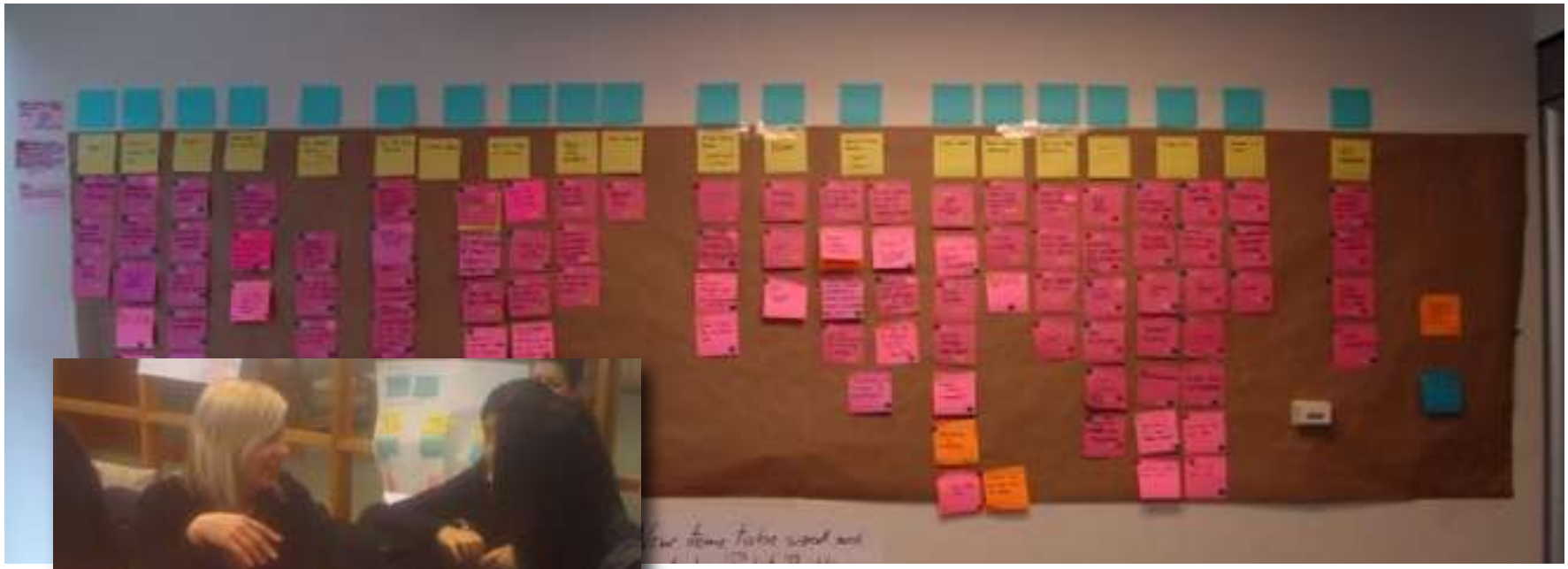
And, as we start moving forward, how do we stay on track?

User Story Mapping is an an approach to **Organizing** *and* **Prioritizing** user stories

Unlike typical user story backlogs, Story Maps:

- make **visible** the workflow or value chain
- show the **relationships** of larger stories to their child stories
- help confirm the **completeness** of your backlog
- provide a useful **context** for prioritization
- Plan releases in complete and **valuable slices** of functionality.

User Story Mapping is an an approach to **Organizing** *and* **Prioritizing** user stories



Story Maps support the primary intent of user stories, **rich discussion**

The foundational building block of a story map is the **user task**

First let's do a **warm-up exercise** to understand a few concepts

What were all the things you did to get ready to be here today?

- Starting from the moment you woke up until you arrived here
- Write one item per sticky note

First let's do a **warm-up exercise** to understand a few concepts

In a small group (3 to 5 people) merge these stickies into a single model

- **Arrange** them left to right in an order that makes sense to the group
- **Eliminate duplicates**
- **Cluster** items that seem similar and **create labels** for the clusters if items that seem to go together

What's common about the items each of you wrote down?

What was different?

How did the models created by different groups vary?

People achieve goals through interaction

problem or goal

How I'd like to feel, or what I'd like to achieve

goal evaluation

Is my goal met or problem resolved?

Take some action

action evaluation

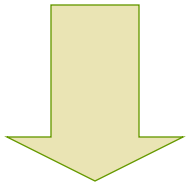
Did that action deliver the results I expected?

the world

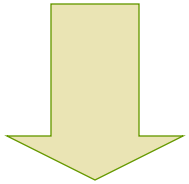
Information and tools

Think of three levels: goal, task, and tool

goal

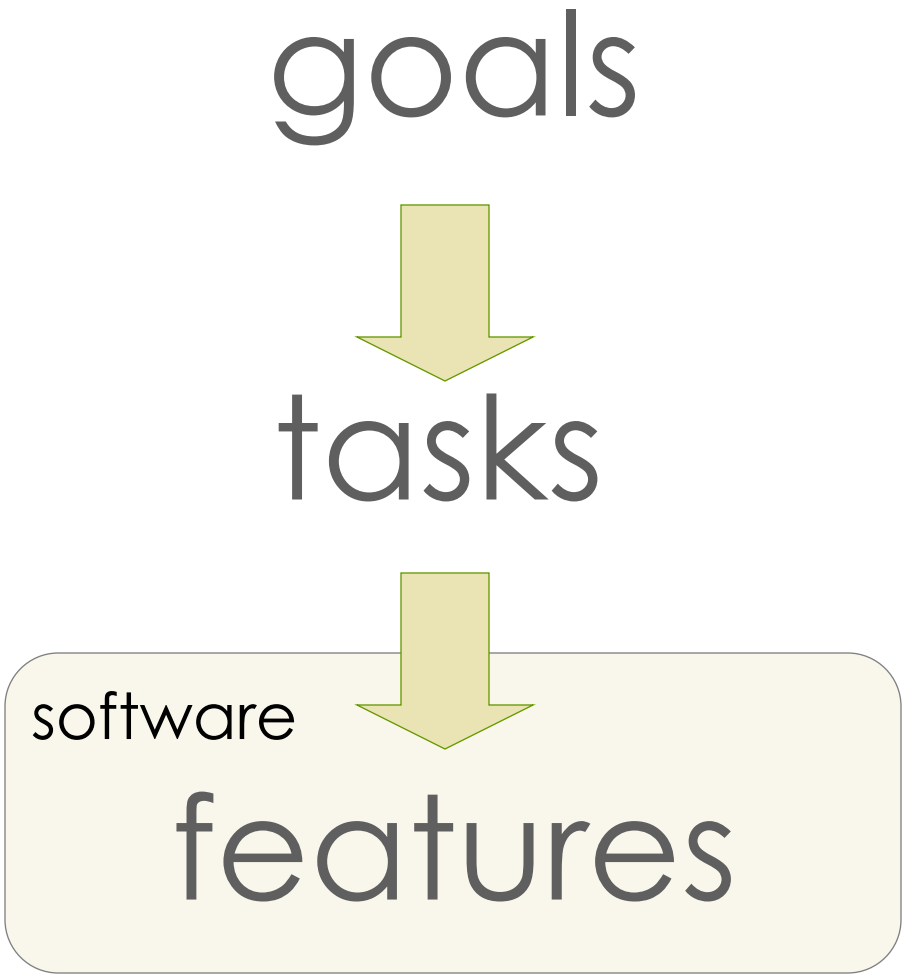


task



tool

Software contains features that support a variety of tasks and a variety of goals



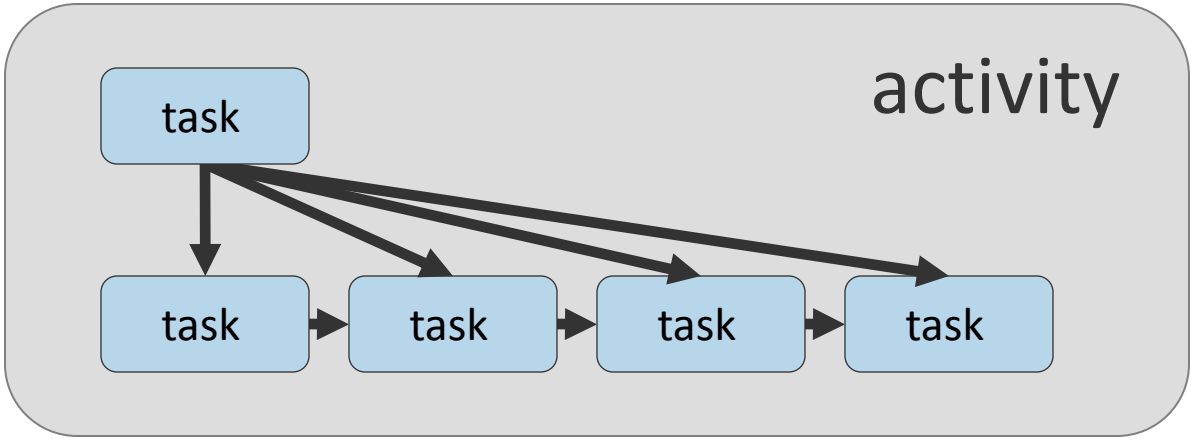
User tasks are decompose to smaller tasks and organize into activities

Tasks require i**ntentional action** on behalf of a tool's user

Tasks have an **objective** that can be completed

Tasks **decompose** into smaller tasks

Tasks often **cluster** together into activities of related tasks



User tasks are decompose to smaller tasks and organize into activities

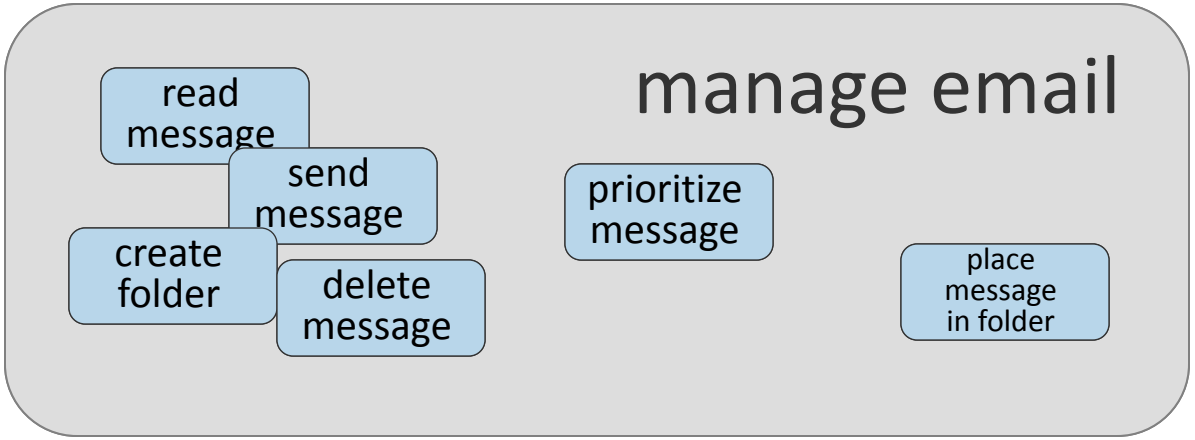
Tasks require **intentional action** on behalf of a tool's user

Tasks have an **objective** that can be completed

Tasks **decompose** into smaller tasks

Tasks often **cluster** together into activities of related tasks

"Read an email message" is a task, "Managing email" is an activity.



Be sensitive to your user task's "altitude"

Too abstract

Activity or "Kite level"

Longer term goals often with no precise ending. I'll perfo
several functional tasks in the context of an activity

Think about user experience at this level

Functional or "Sea level"

I'd reasonably expect to complete this in a single sitting

Sub-Functional or "Fish level"

Small tasks that by themselves don't mean much. I'll do several of these before I reach a functional level goal

Too detailed

* from Cockburn's Writing Effective Use Cases

User tasks make ideal user stories:

Title: Take a shower

As an **instructor**
I want to **take a shower**
So that **I don't offend my colleagues**

What are the goals & tasks?

Business goals or pain points?

Types of users using this system?

User's goals or pains?

How will users of the system reach their goals?

Start with an understanding or user experience

Write user scenarios to think through user experience

Steven

Credit Card Marketing Field Manager

Steven is a field manager working at the local shopping center. He's in the middle of a long workday supervising 13 reps who are busy talking to people trying to convince them to apply for a credit card.

Field Manager enters daily performance reports

The shift has just ended and his reps are coming up with their totals. They have printed sheets with totals written on them. Steve quickly looks them over and signs them off. His assistant manager brings him other sheets with totals he's signed off.

In between visits by reps, Steve opens his Field Manager Workbench on his laptop. After logging in he sees today's date and the planned number of applications his reps should be gathering – 180 for today.

He also sees yesterday's numbers, and last week's numbers, and the last 30 days in graph that shows applications relative to approval rate. Last week's numbers were bad, and it's the last week of the month, so Steve knows he's got to do well today.

Steve clicks "enter rep performance data." He shuffles his reps performance sheets and grabs the first one.

5. The date is defaulted to today, and the shift is defaulted to 'morning' since he hasn't yet entered info for today. Steve begins to enter the reps name, but after a few characters the system auto-completes his name.
6. The rep's ID is already filled in, along with the code for the credit card promotion they're working on today.
7. Steve fills in the shift information for his rep. He then enters the total number of applications taken.
8. It looks like from the notes on this sheet that this rep left sick two hours early. Steve adds a note about this in the system.
9. Time passes as more reps bring in their sheets and Steve completes entering them in between conversations.
10. After all the sheets are done, Steve looks at a summary screen for the day. It looks like he's close to his goal. If the next shift continues at this rate he'll beat the plan by 5% or so. That's good.
11. Steve validates that the base pay is correct at $5 per app, and that he's set an individual bonus giving reps $50 each if they reach 20 apps. Next to each rep he sees the calculated pay, base, bonus, and total pay for the day.
12. The annual sale at Macy's has brought a lot of people in today. Steve chooses a "sale increases mall foot traffic" code to add to his shift data sheet. Frank, his boss, has pestered him to make sure he includes this type of information in his daily summaries.

A user scenario is a simple way to think through user experience concretely

A user scenario tells a story about a main character with a problem or goal

- Describes how that character reaches their goal
- contains important facts
- describes external context
- describes goals and concerns of our character

include interesting plot points that help us envision important aspects of the system

A scenario can gloss over uninteresting details

Imagine the user experience as a scenario

Separate your team of four into two pairs

One person imagine out loud the experience of someone using the kiosk. Think about:

- Typical use, including typical problems
- Interesting plot points
- Goals and pains of your user

The other ask questions to better understand

After 5 minutes of discussion write out the scenario

Pair **one** think about **Carol**:

casual browser

"I want to find something good from a band I heard on the radio."

Goal: : have fun finding something interesting

Pair **two** think about **Isaac**:

Impatient buyer

"I wan to find a specific hard to find foreign film."

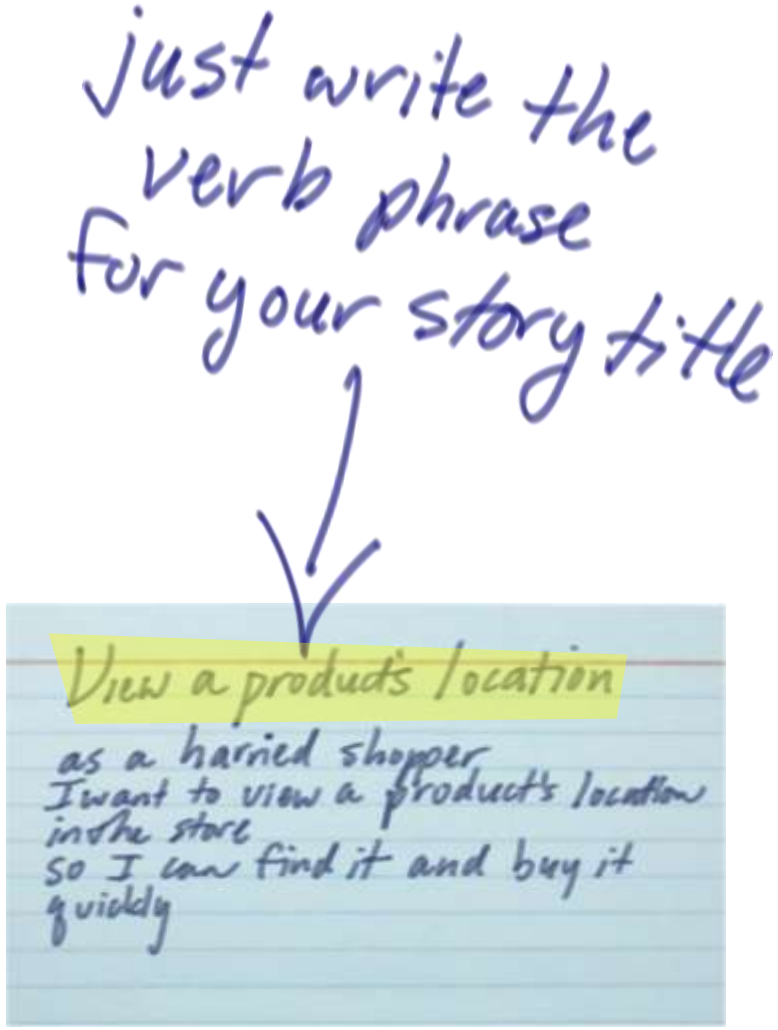
Goal: : Find it and get out quickly without wasting my time

Extract task-centric user stories from your user scenarios

Come back together as a team

Reviewing each others scenarios, extract task-centric stories using yellow stickies

Write only the tasks **verb-phrase** for your title



Organize user stories into a map that communicates experience

By arranging activity and task-centric story cards spatially, we can tell bigger stories

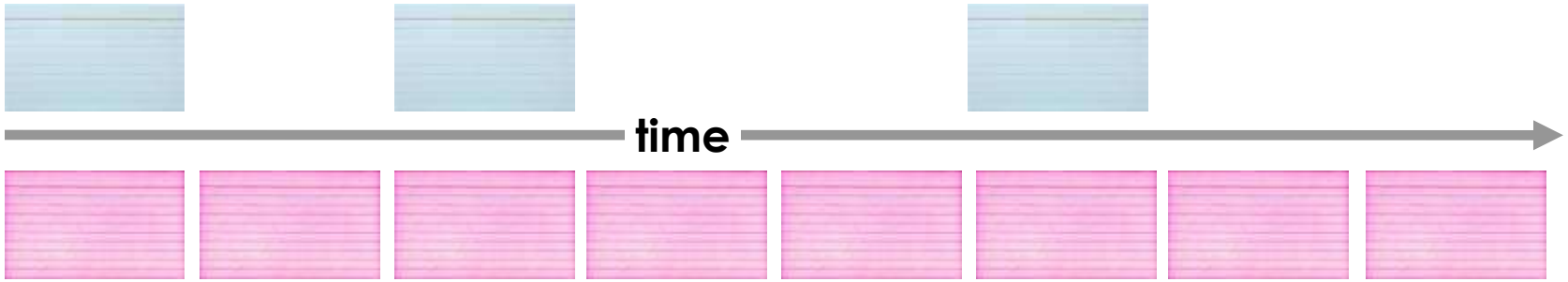
Tell a big story of the product by starting with the major user activities the kiosk will be used for

- Arrange activities left to right in the order you'd explain them to someone when asked the question: "What do people do with this system?"

By arranging task-centric story cards spatially, we can tell bigger stories

Add task-centric stories in under each activity in workflow order left to right.

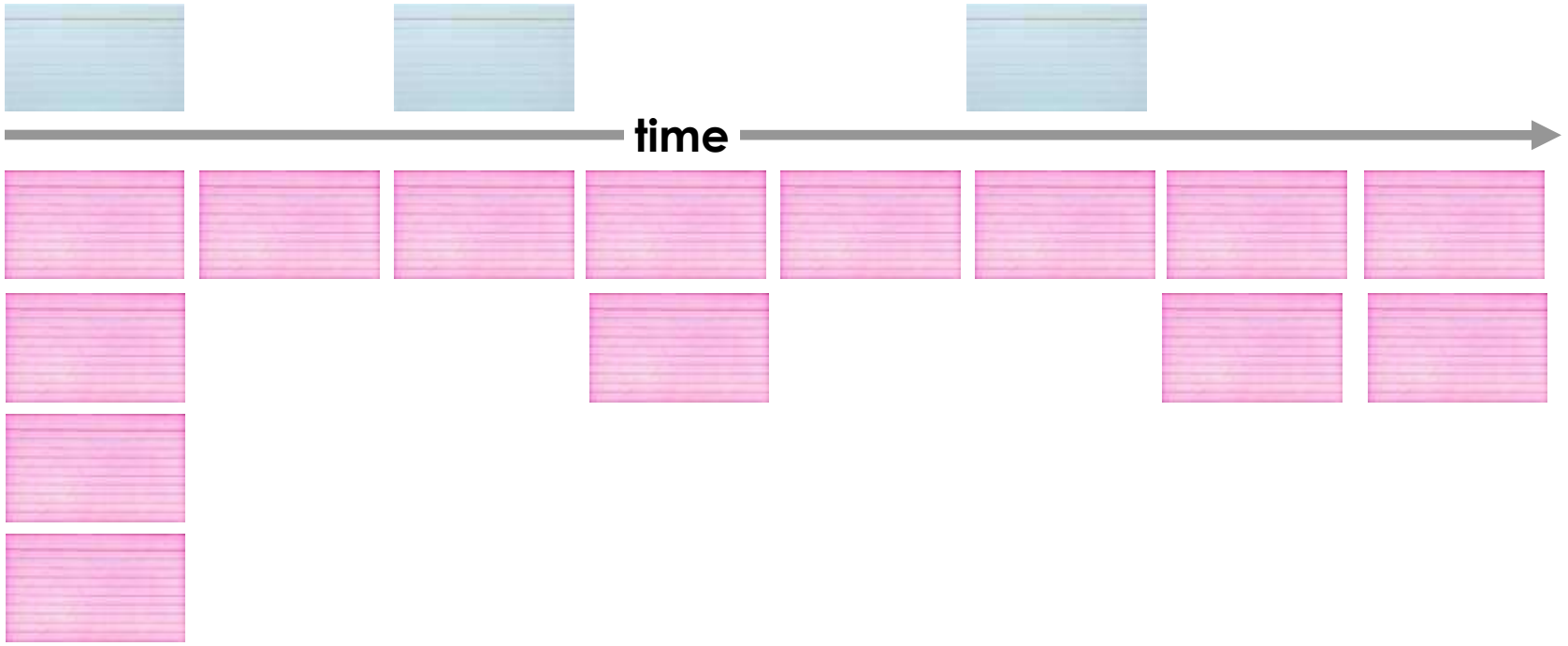
- If you were to explain to someone what a person typically does in this activity, arrange tasks in the order you'd tell the story. Don't get too uptight about the order.



By arranging task-centric story cards spatially, we can tell bigger stories

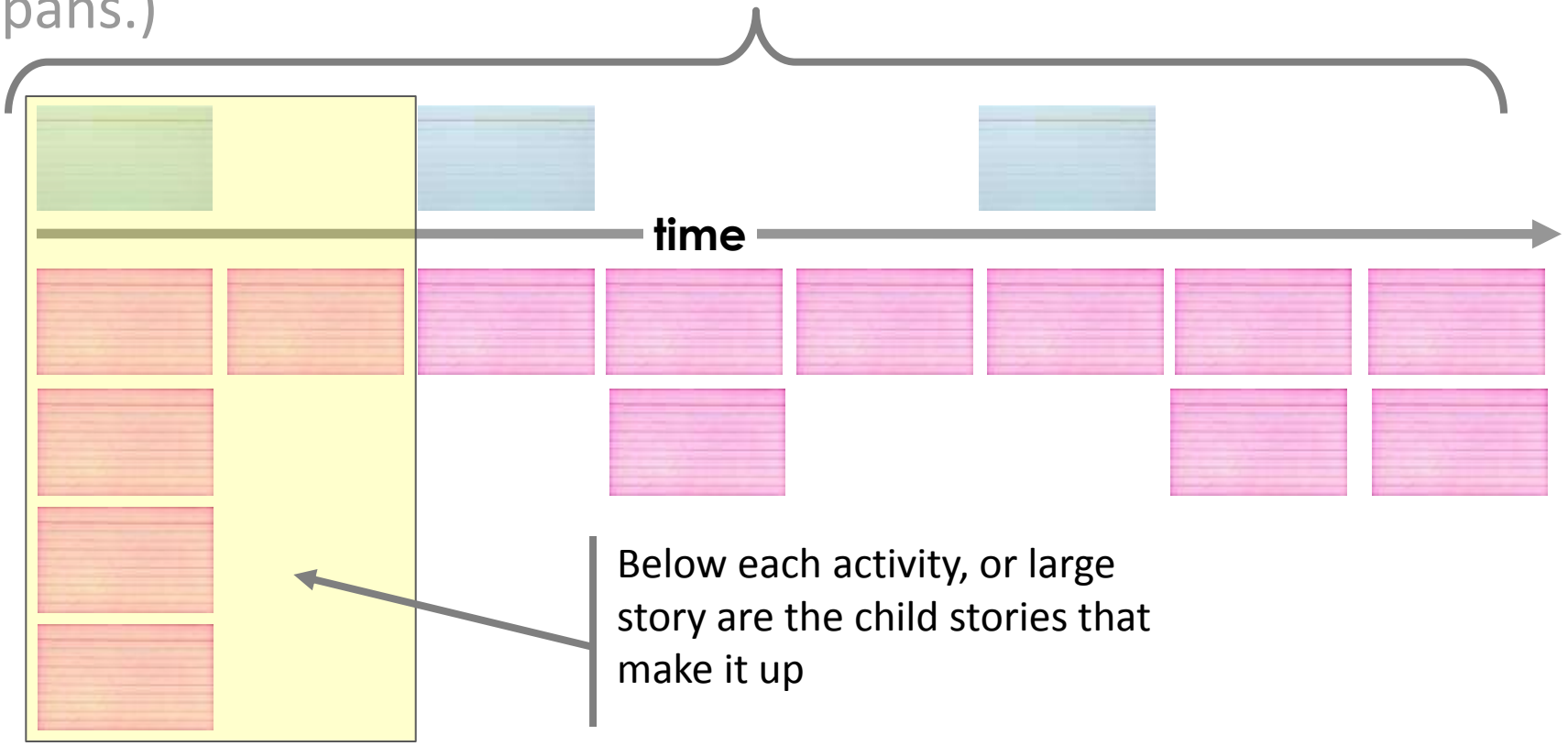
Overlap user tasks vertically if a user may do one of several tasks at approximately the same time

- If in telling the story I say the systems' user typically "does this or this or this, and then does that," "or's" signal a stacking vertically, "and then's" signal stepping horizontally.

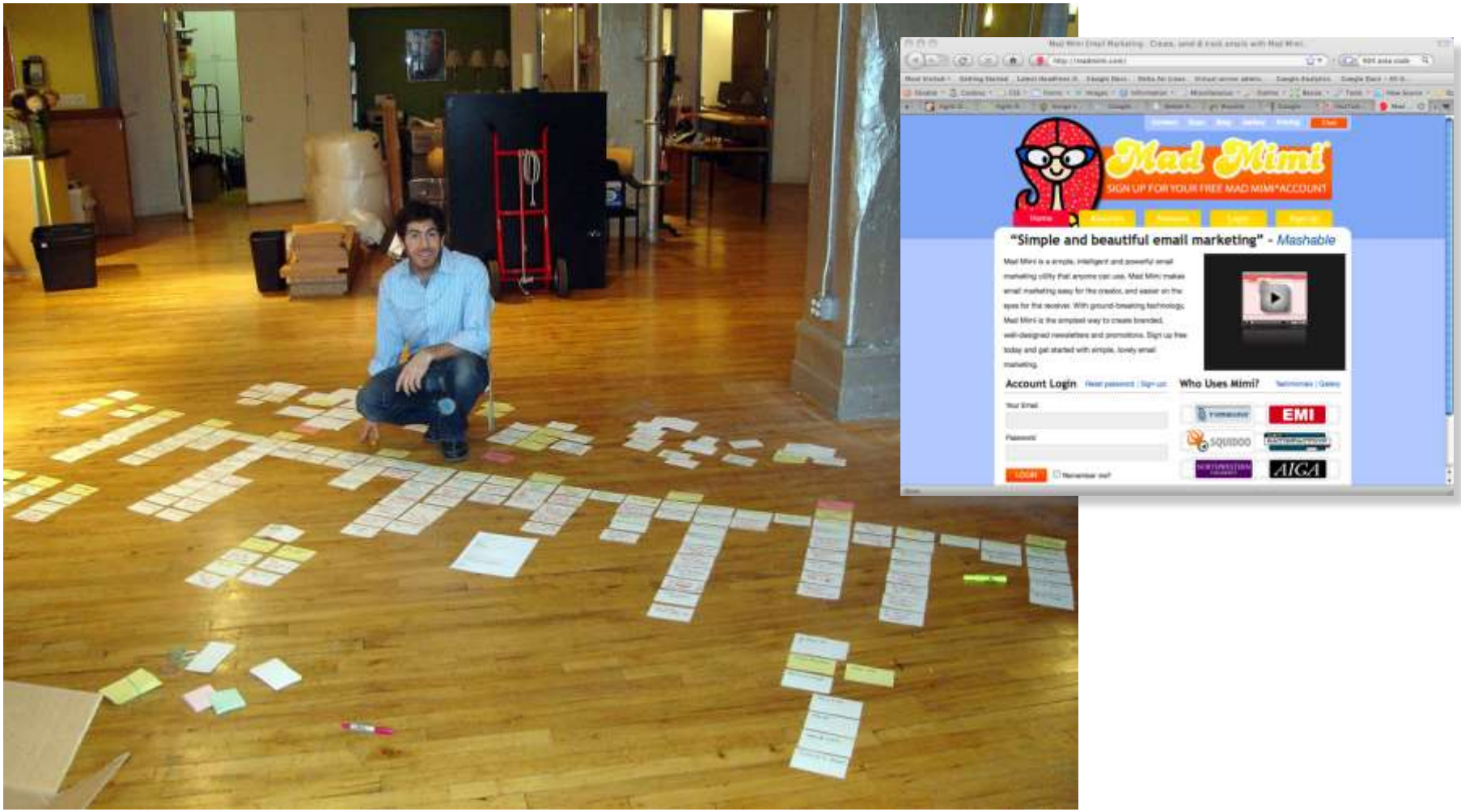


The map shows decomposition and typical flow across the entire system

Reading the activities across the top of the system helps us understand end-to-end use of the system. (Talk through just these when talking with people with short attention spans.)

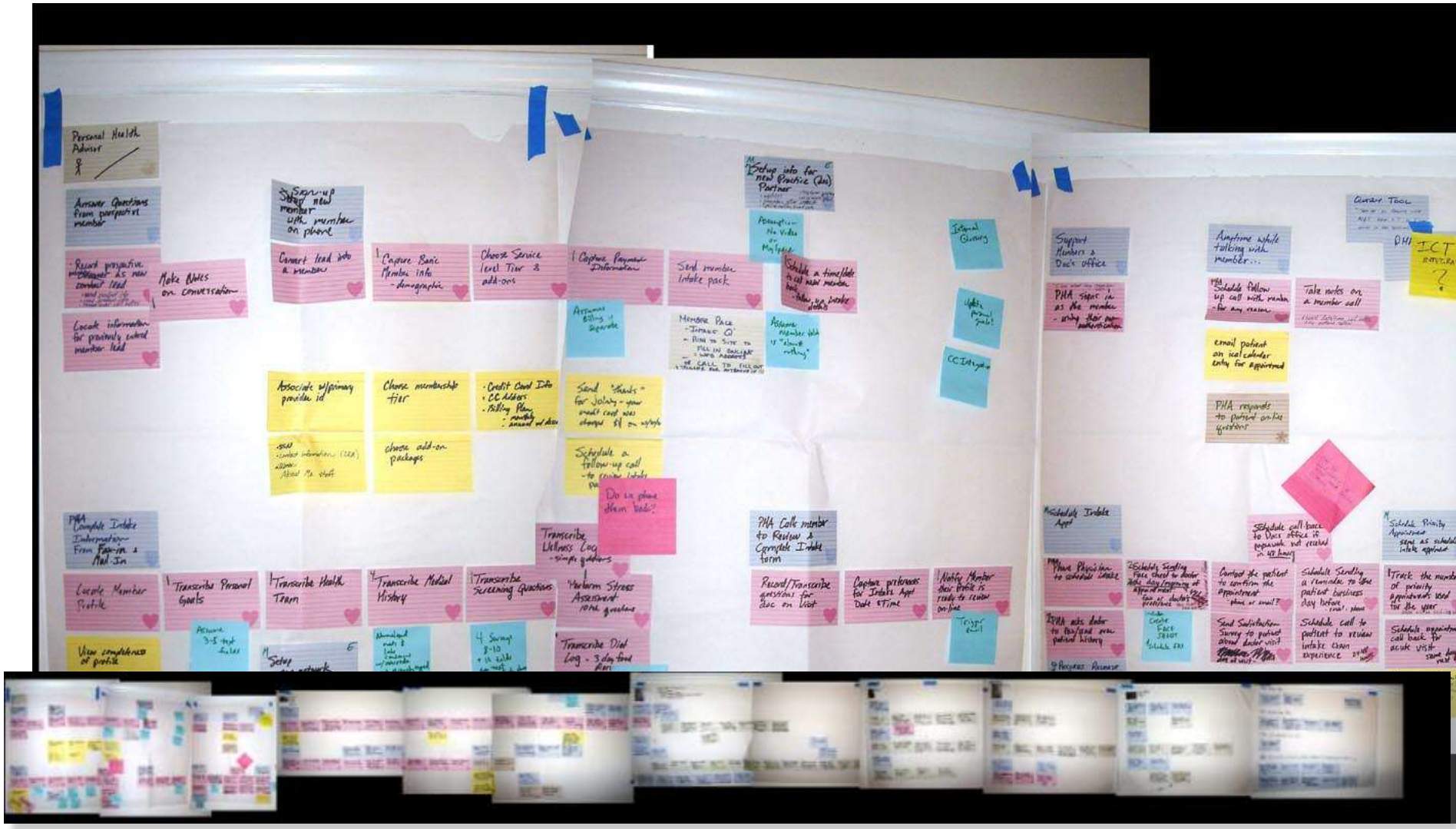


Building a story map helps facilitate discussion – but requires a bit of space



Gary Levitt, owner & designer of Mad Mimi

A story map for a reasonable sized system can fill a room

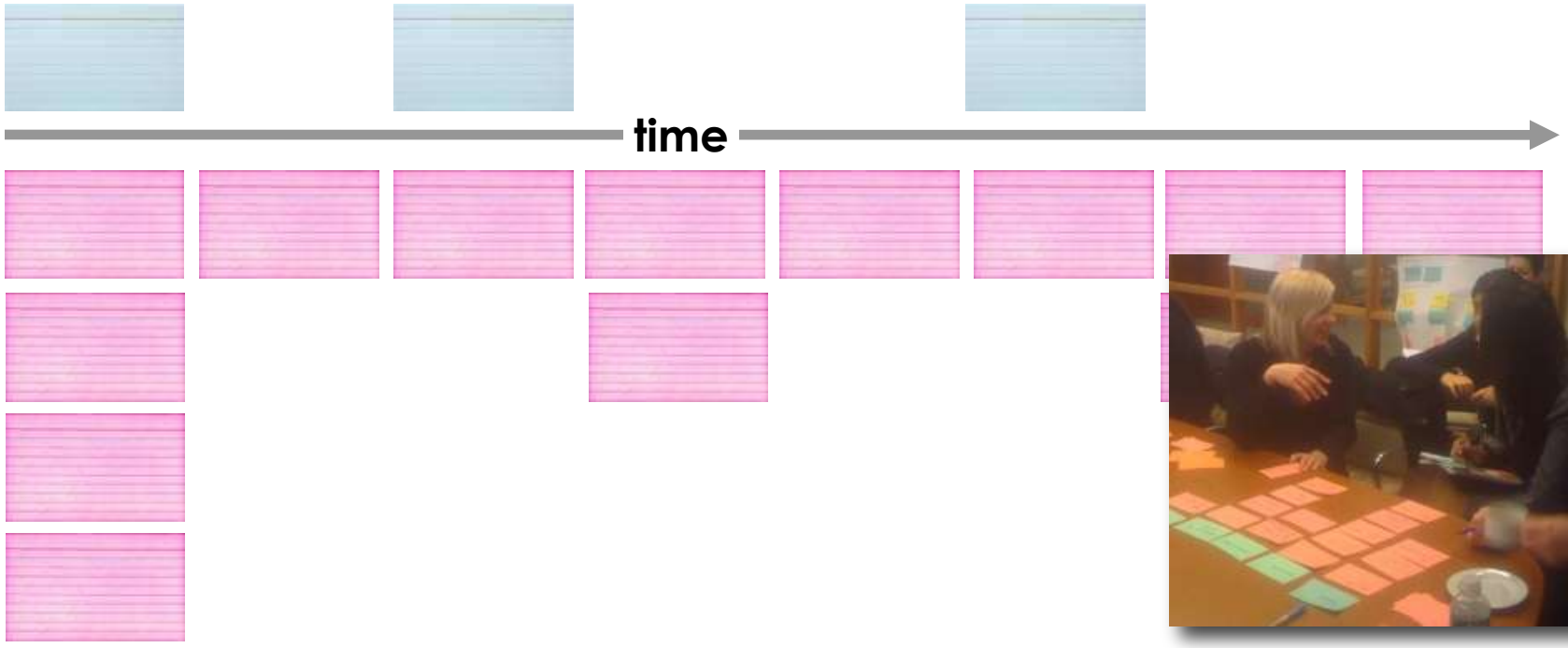


Assemble your harvested task-centric stories into a simple story map

Work as a team to organize your stories

Look for activities: tasks that seems similar, are done by similar people in pursuit of similar goals

Use a different color sticky to label activities

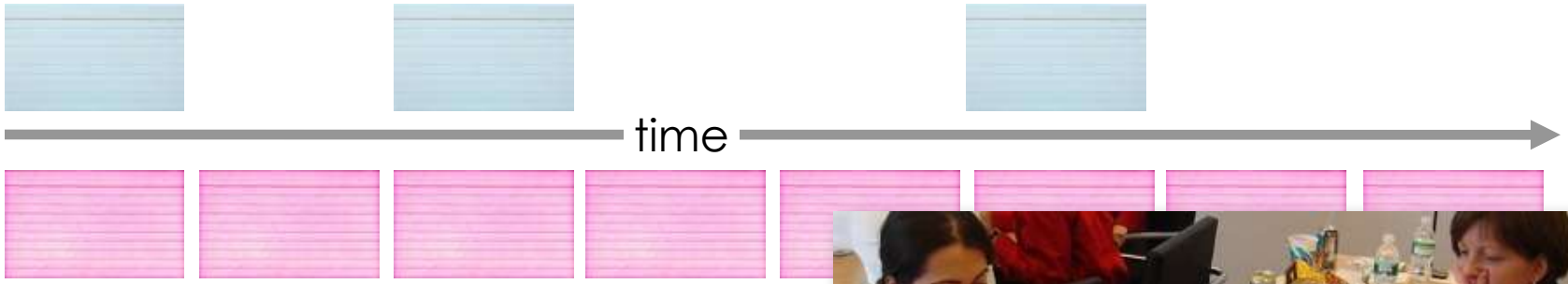


Discuss, fill in, refine the map, and test for completeness

Once you've got the idea down, it's quick to record stories as you discuss the experience with users

Discuss the steps of the process with candidate users

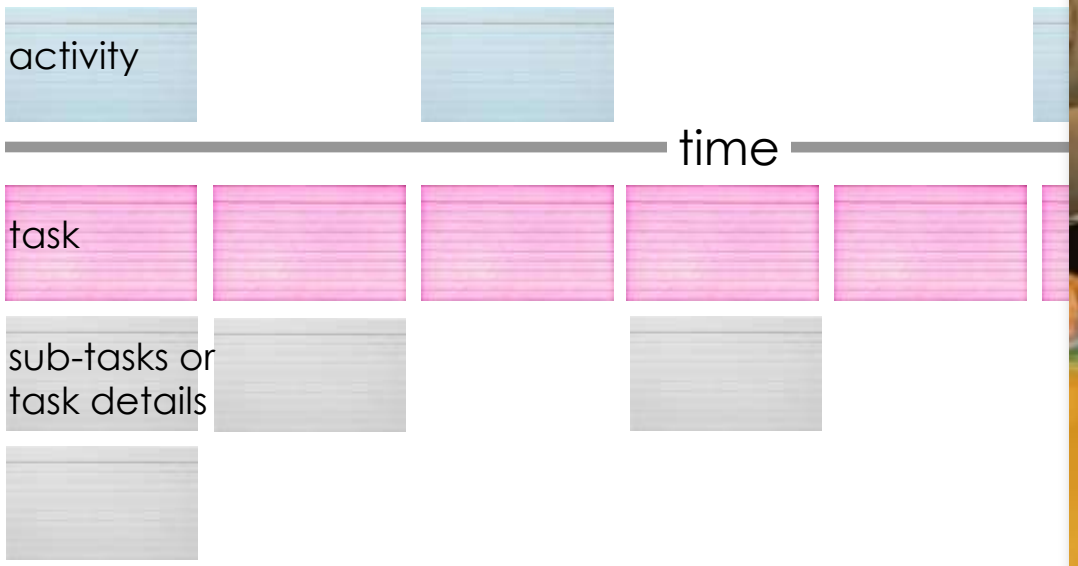
- Record tasks as they say them
- Rearrange tasks and insert tasks as you clarify the big story
- Add activities as you identify them from discussion



As details emerge in conversation, trap them under their associated task cards

Record details so they're not lost, and so those who you're working with know that you're listening

- Consider tucking them under tasks cards to "hide them" from the discussion

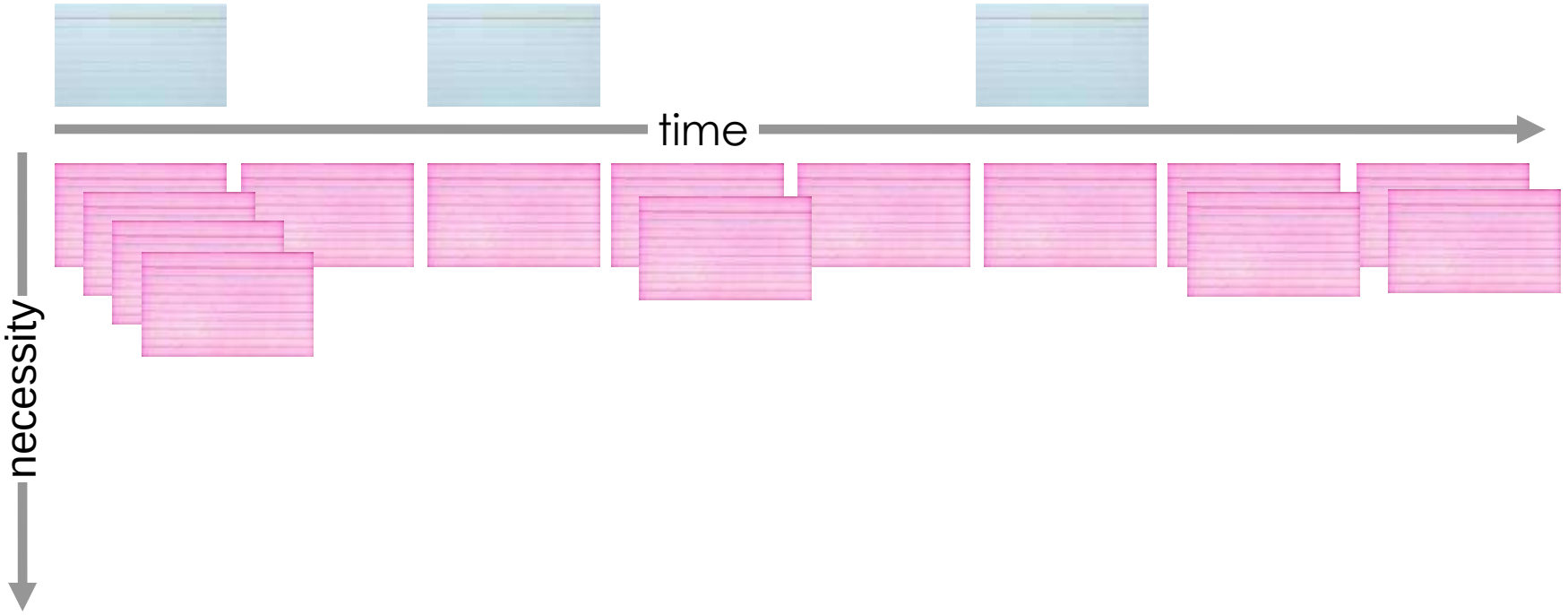


By arranging activity and task-centric story cards spatially, we can tell bigger stories

Add a vertical axis to indicate necessity

Move tasks up and down this axis to indicate how necessary they are to the activity.

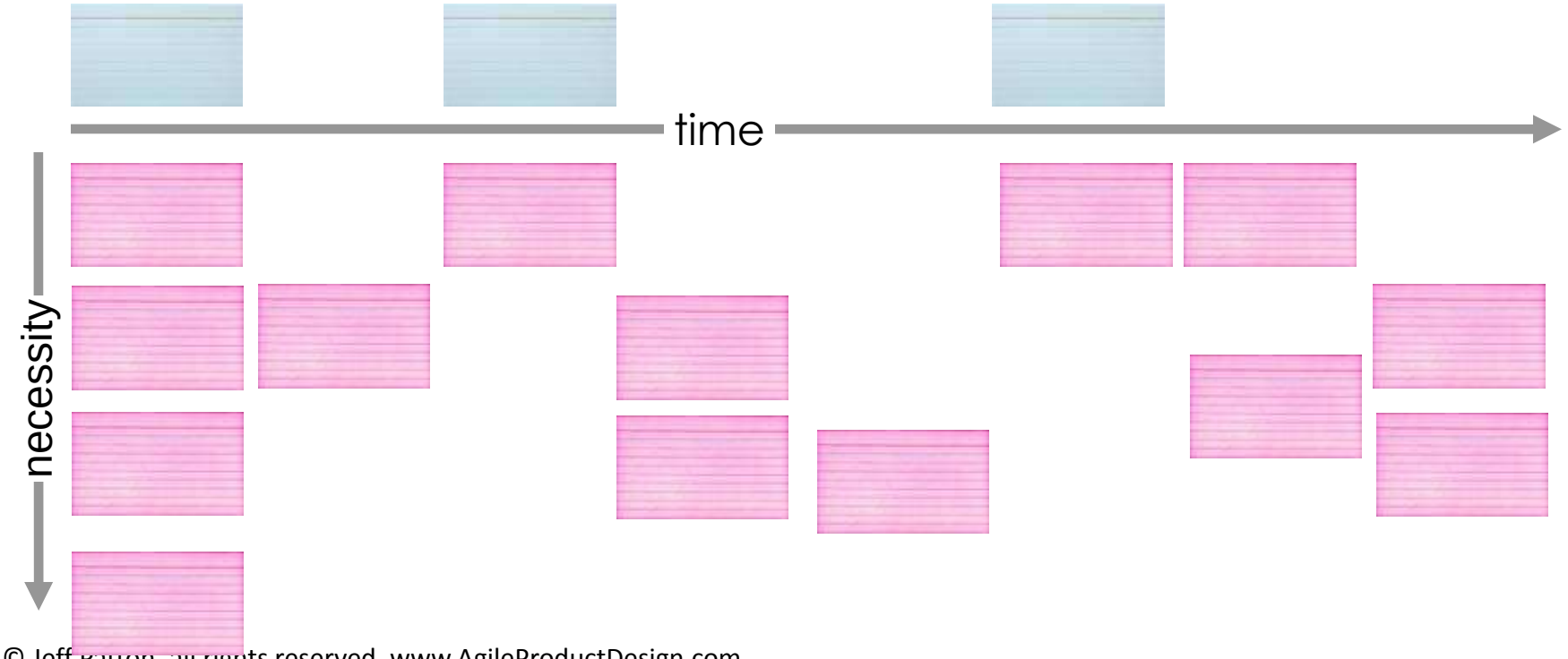
- For a user to successfully engage in this activity, is it necessary they perform this task? If it's not absolutely necessary, how critical is it?



By arranging activity and task-centric story cards spatially, we can tell bigger stories

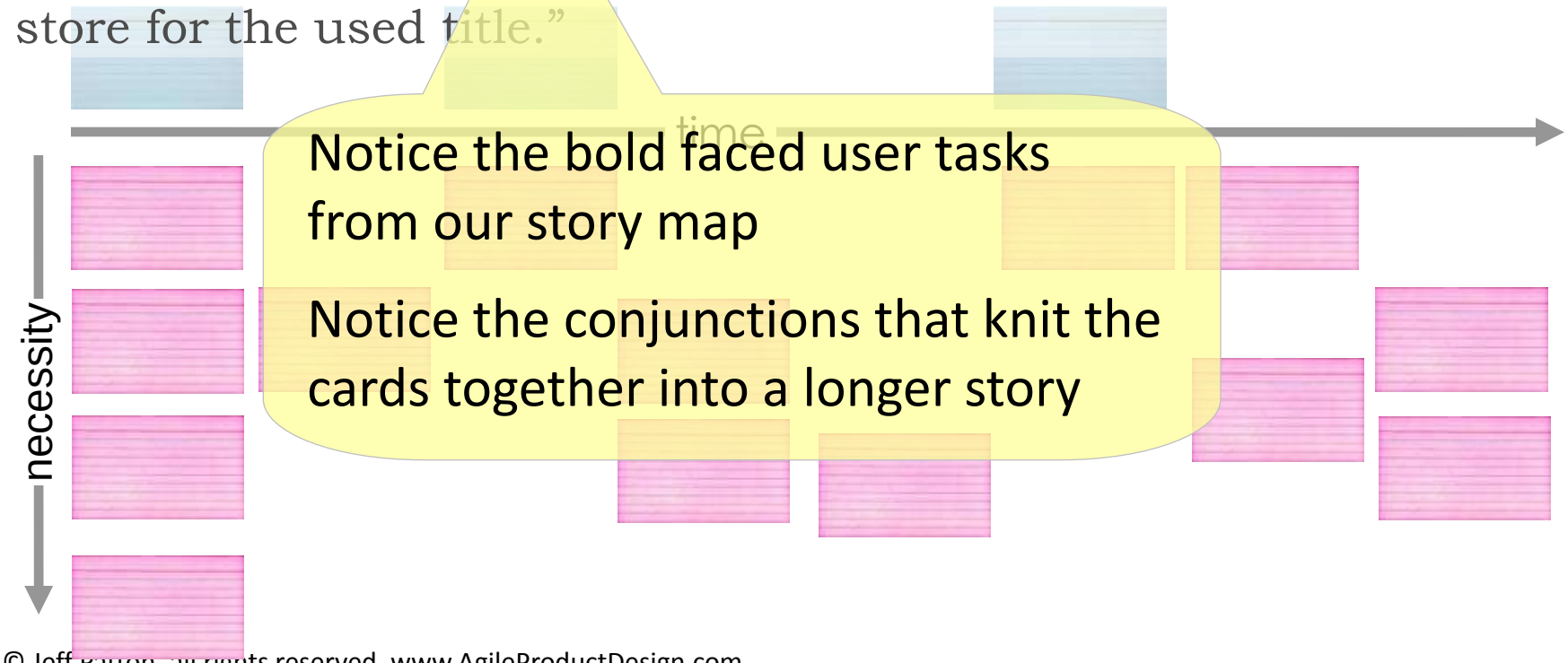
Test the Story Map by telling bigger stories with it

- Choose an activity to start with
- When reading left to right use the conjunction “and then” to connect cards in the story
- With cards in the same row use “or” to connect cards in the story
- For cards below the top, “absolutely necessary” axis, use the phrase “might optionally” to communicate optionality
- Chose a concrete user name to help tell the story



By arranging activity and task-centric story cards spatially, we can tell bigger stories

"Steve knows the title of what he's looking for. He steps up to the kiosk and **searches by title**. Optionally he might have **searched by artist**. After seeing titles that match what he typed in, Steve **views the price new and used**, and then **views the status** – whether it's in stock or not. He notices it's in stock as both new and used, so then Steve **views the location** in the store for the used title."



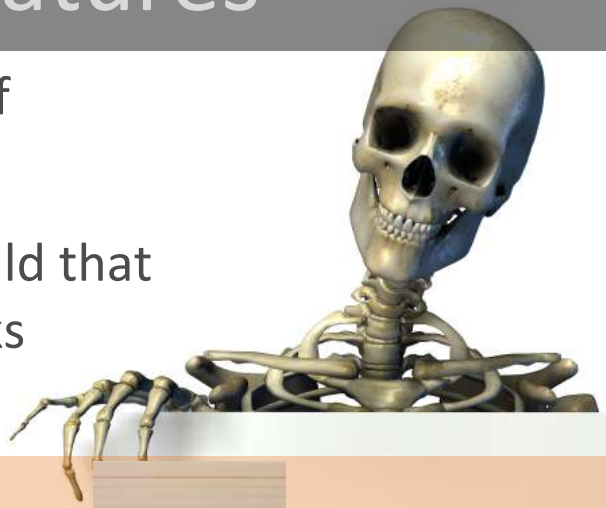
Discussions over story maps help drive out more details

Repeated review of the story map with multiple users and subject matter experts will help test the model for completeness

The user story map contains two important anatomical features

The **backbone** of the application is the list of essential activities the application supports

The **walking skeleton** is the software we build that supports the least number of necessary tasks across the full span of user experience



Using discussion, fill in your story map

Work together as a team

Look for **alternative tasks**

- What else might users of the system have done that didn't come up in your scenarios?

Look for **exceptions**

- What could go wrong, and what would the user have to do to recover?

Consider **other users**

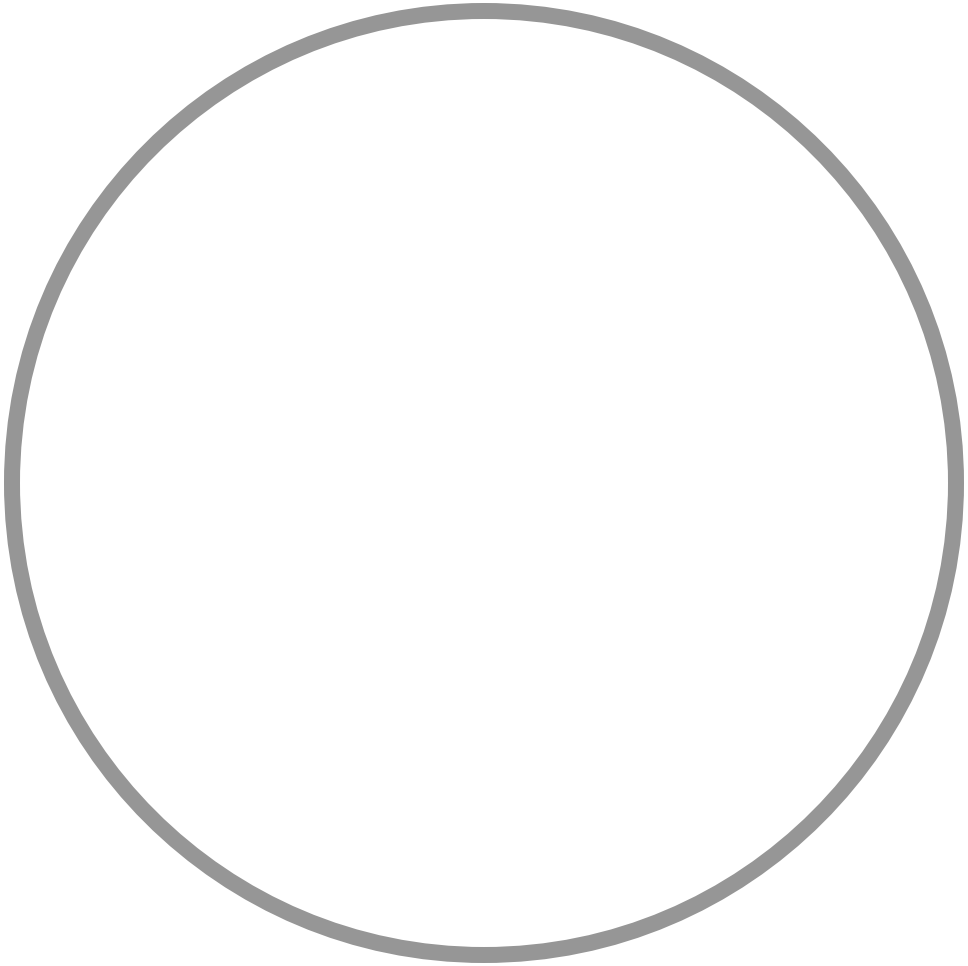
- What might other types of users do to reach their goals?

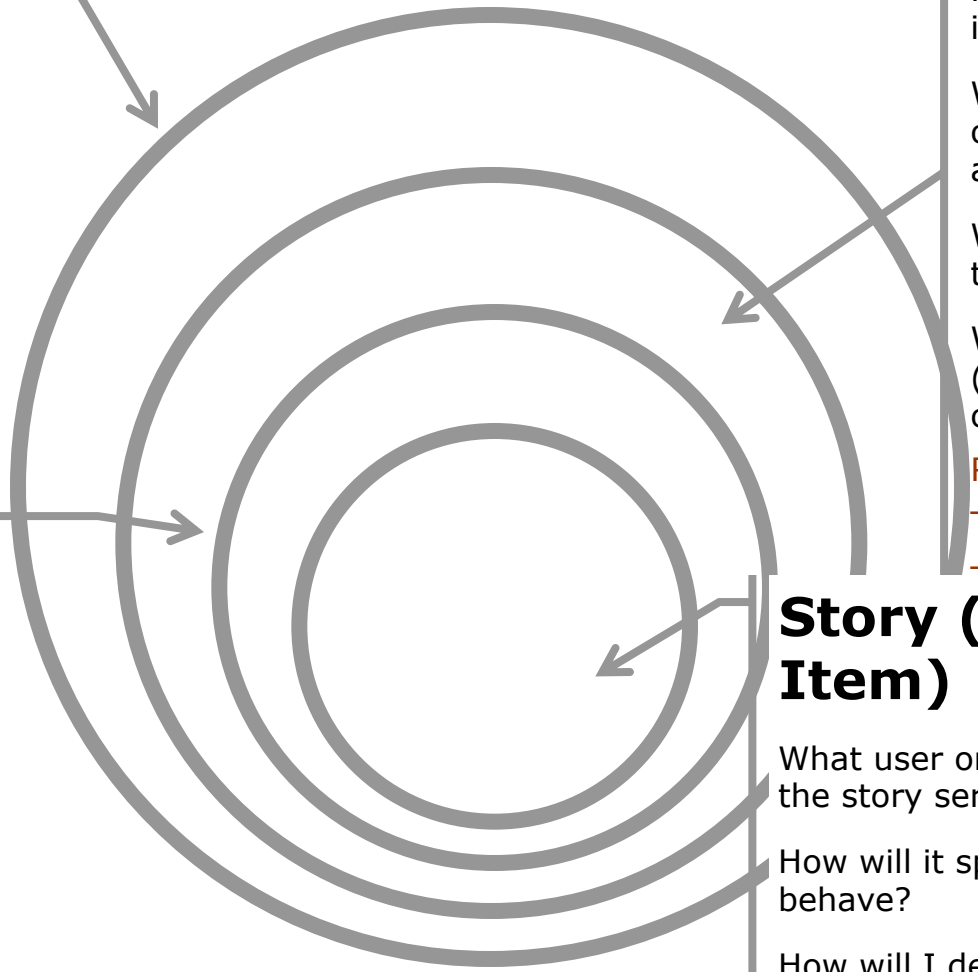
Knit all these additional stories into your map

Slice the map to find ideal incremental releases

Agile teams plan product construction in layers

Agile teams plan product construction in layers





Release

How can we release value incrementally?

What subset of business objectives will each release achieve?

What user constituencies will the release serve?

What general capabilities (big stories) will the release offer?

Release Roadmap

Target Customers

Target Personas

Product or Project

What business objectives will the product fulfill?

Product Goals

Product Charter

Customers

User Personas

Iteration or Sprint

What specifically will we build? (user stories)

How will this iteration move us toward release objectives?

Iteration Goal

Development or Construction Tasks

Story (Backlog Item)

What user or stakeholder need will the story serve?

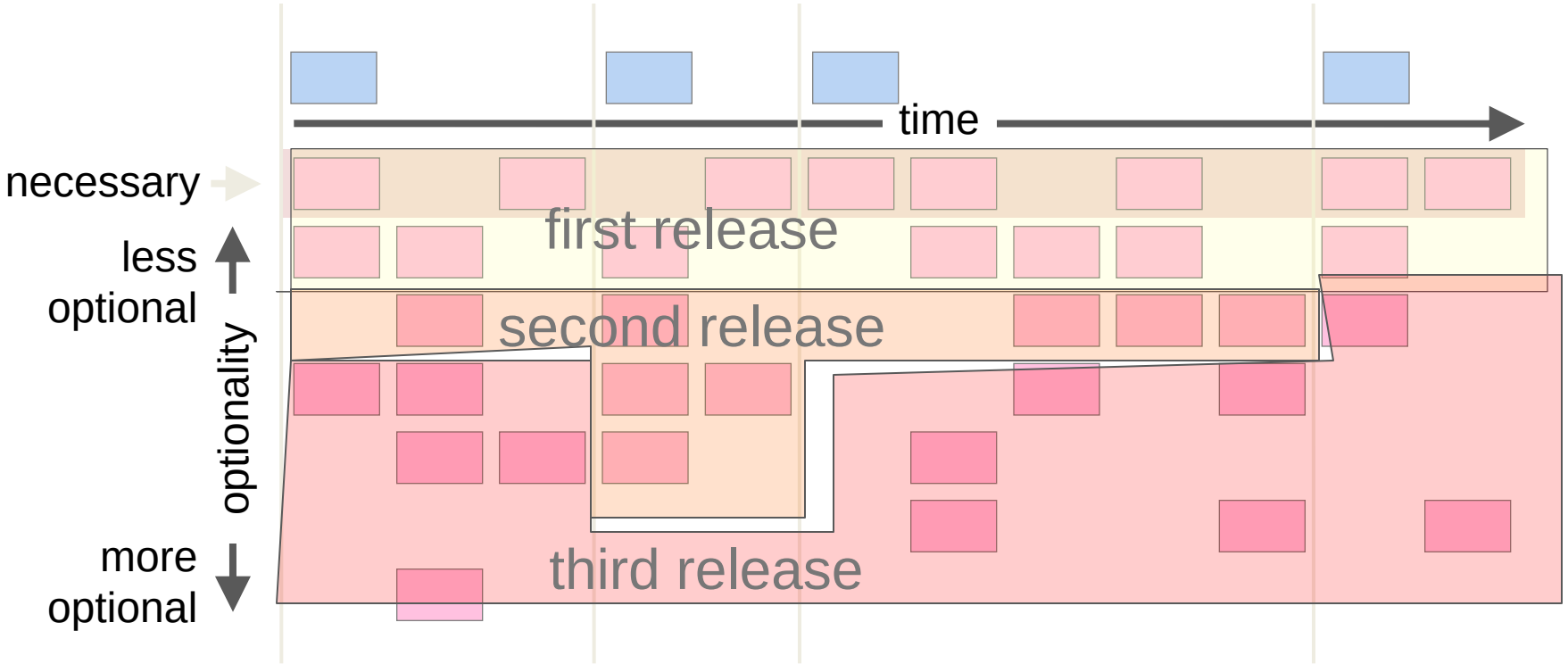
How will it specifically look and behave?

How will I determine if it's completed?

Story Details

Acceptance Tests

Given story map organized vertically by necessity, we need only slice to plan

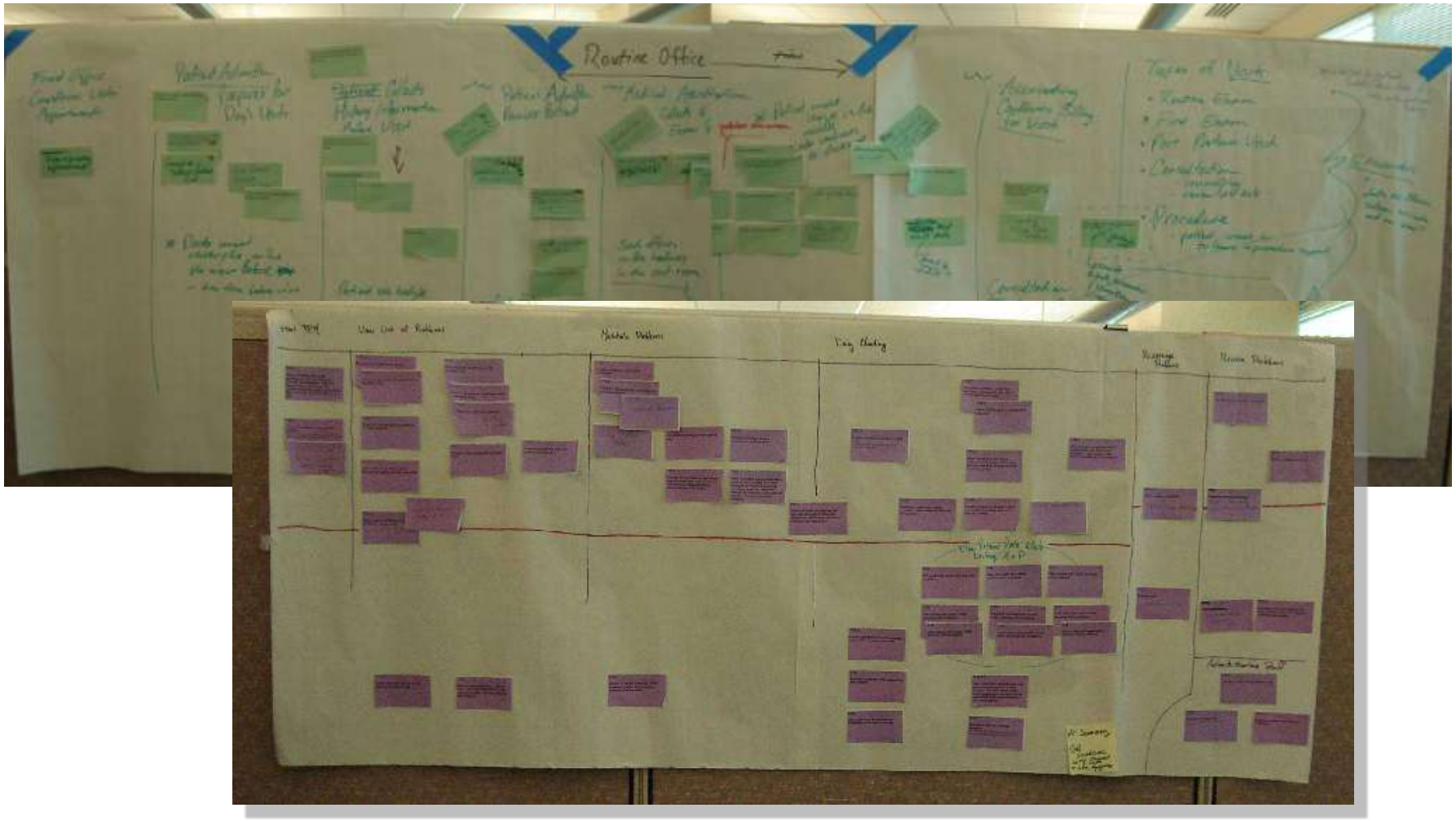


Choose coherent groups of features that consider the span of business functionality and user activities

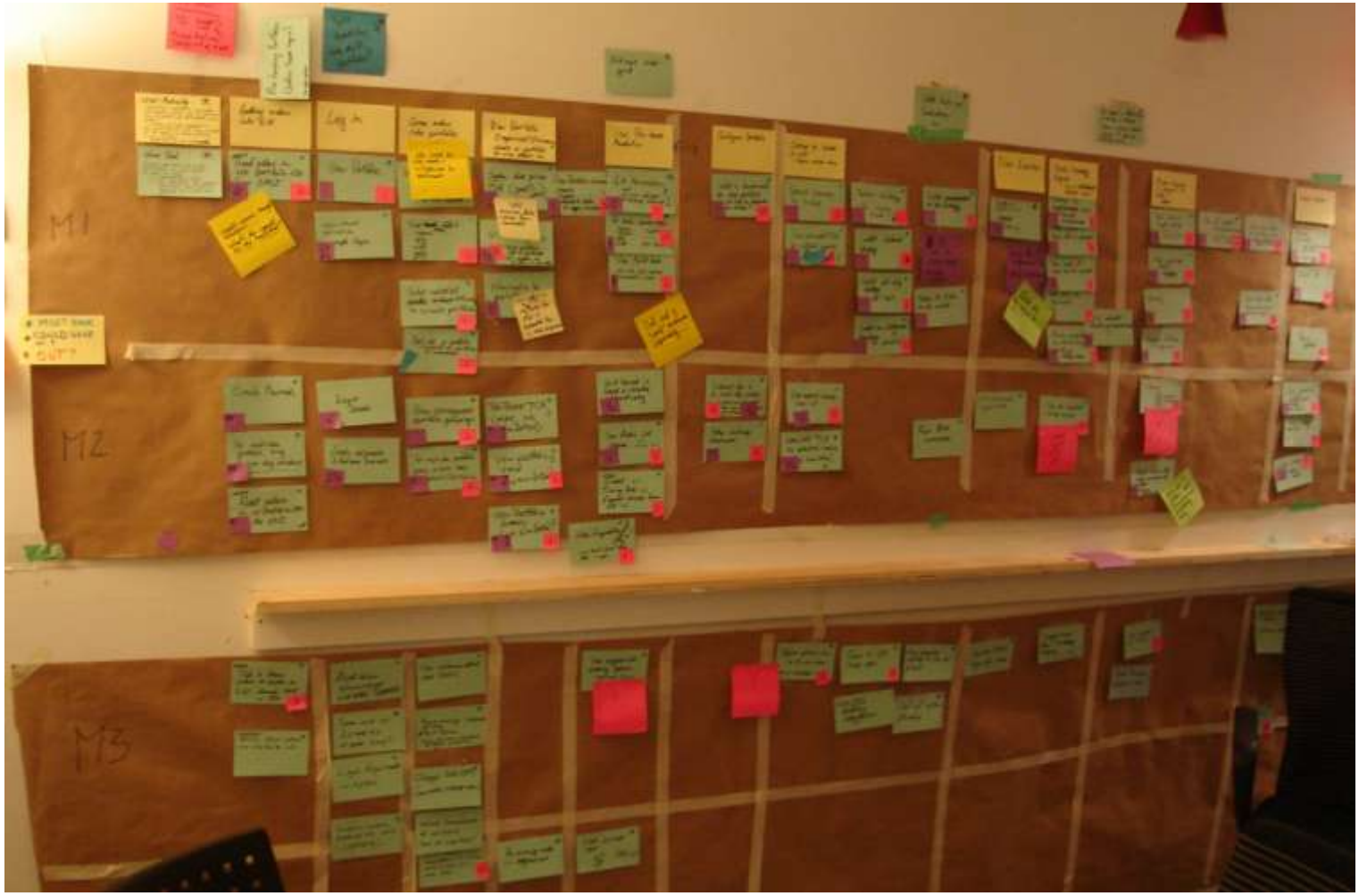
Support all necessary activities with the first release

Improve activity support and add additional activities with subsequent releases

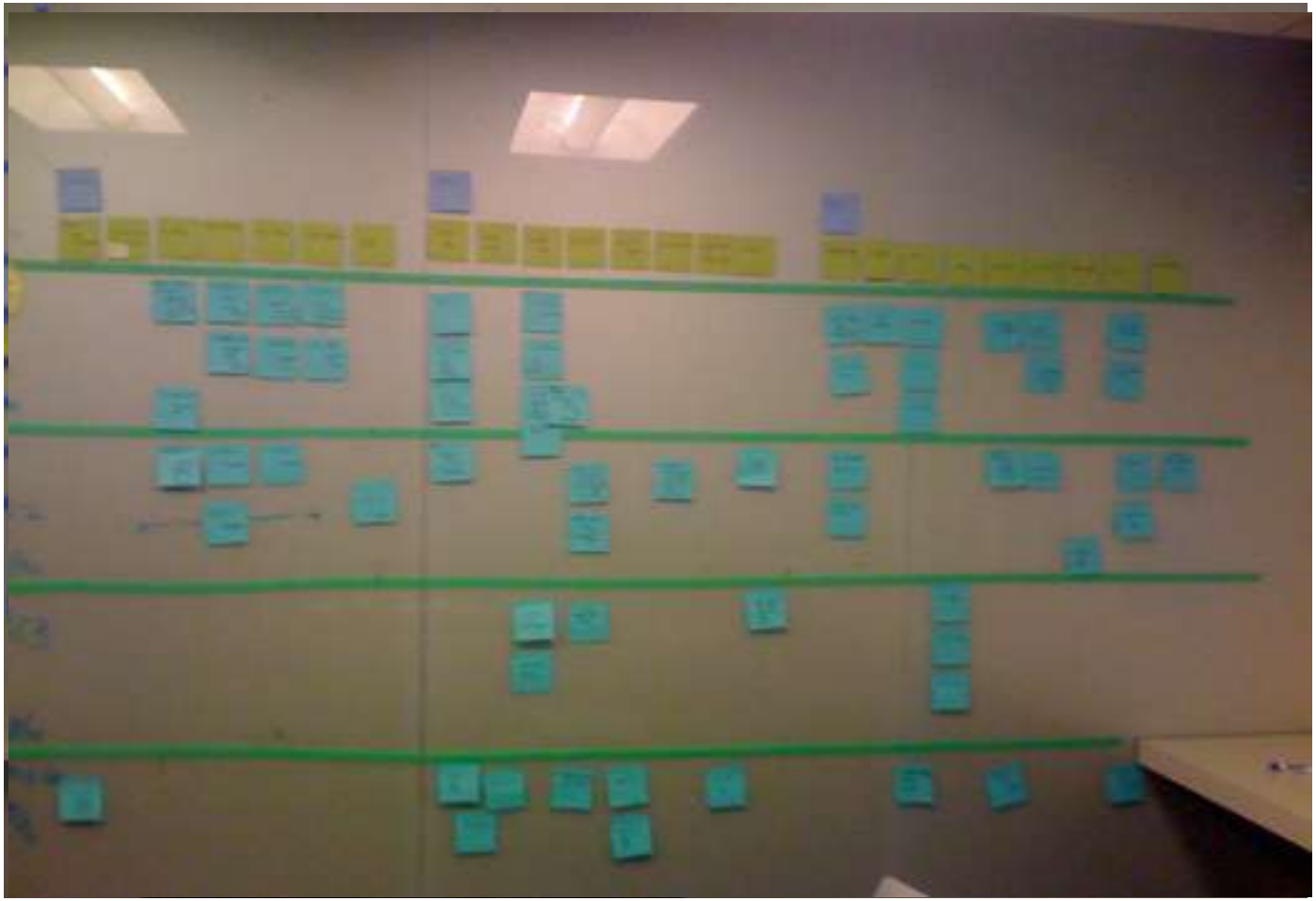
Given story map organized vertically by necessity, we need only slice to plan



Adding tape lines to the wall lets participants organize stories into layers



Adding tape lines to the wall lets participants organize stories into layers



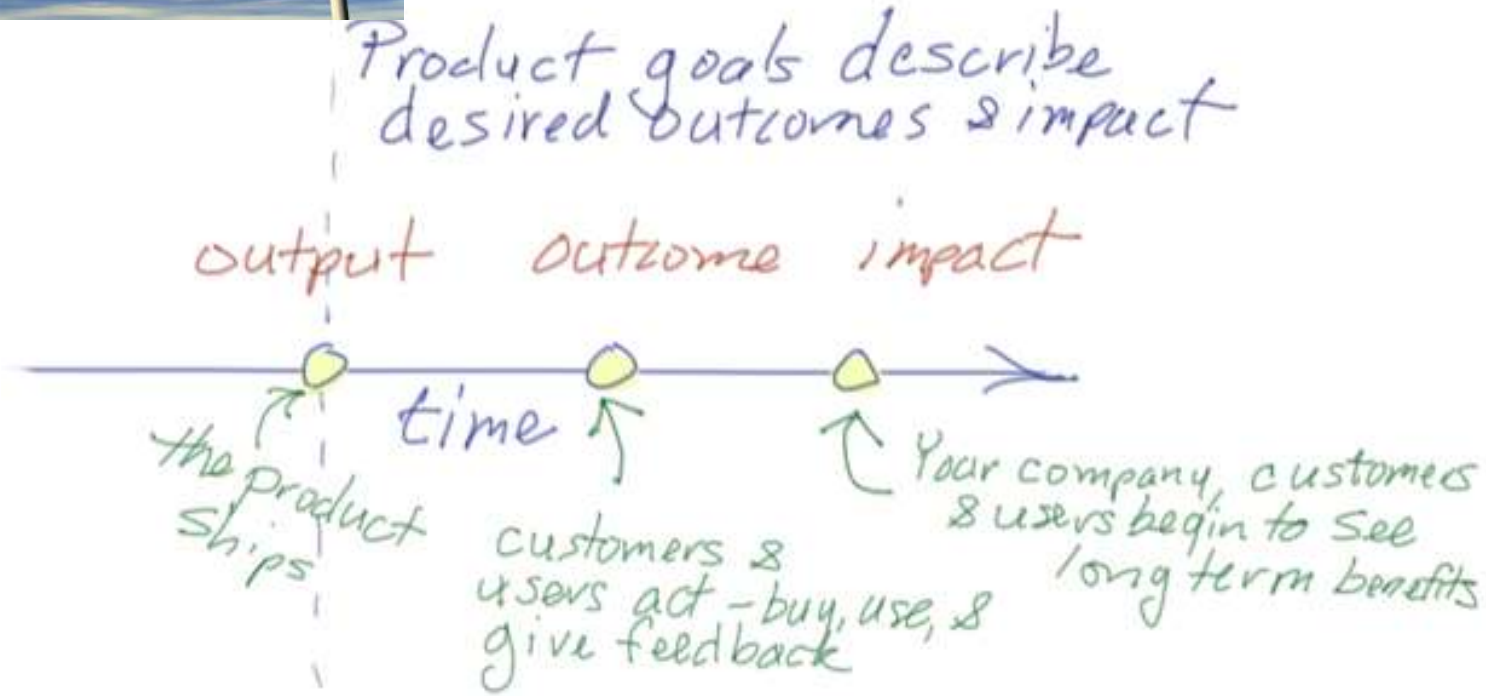
Planning incremental releases can be facilitated as a collaborative event

There's a secret to effective prioritization

(Before I give my ideas about what it is, what's worked for you?)

Identify and prioritize desired outcomes before prioritizing the backlog

Product goals describe what outcome or benefit received by the organization **after the product is in use**



Product goals suggest how the business will earn value from the product, and how we can tell we're getting it

Software built for internal use usually **saves money** or helps improve service to customers indirectly earning money

Software built for use by customers **earns money** through direct sales, improved customer retention, or improved customer loyalty

Product goals are **specific** to that product - not generic to any product. A goal to earn more money isn't useful

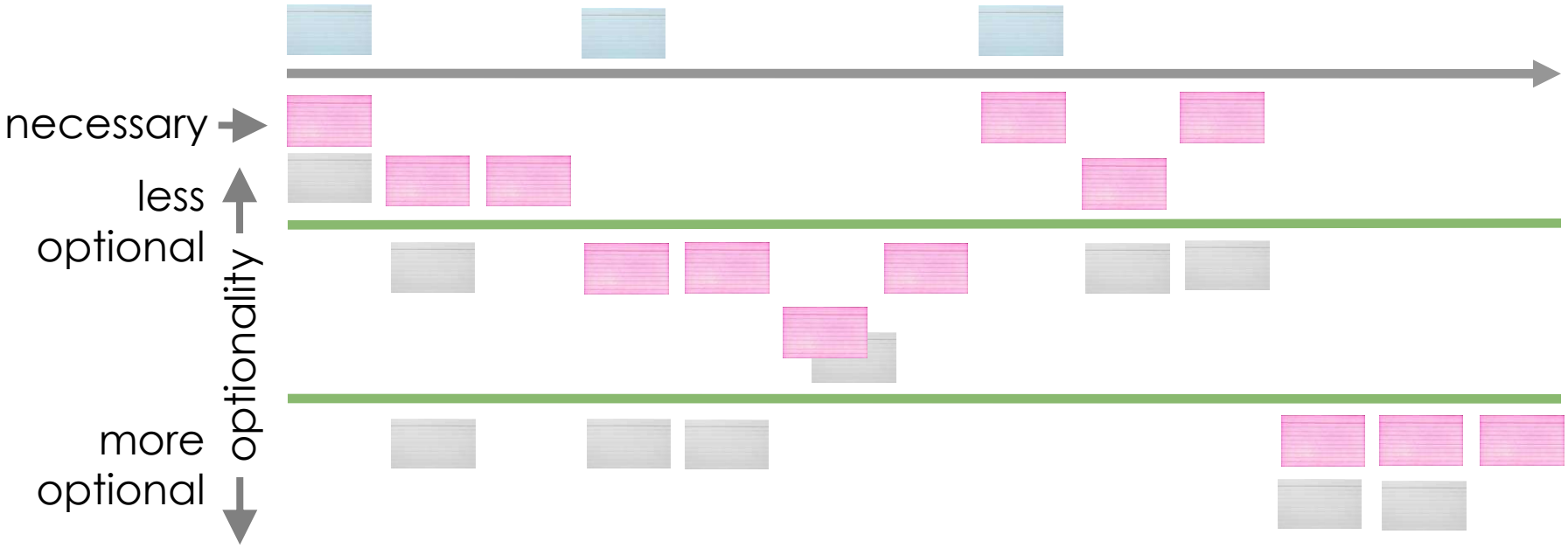
Given a product goal, ask:

"if we're making progress towards this goal, how would we know it? What would we observe in our organization that indicates success?"

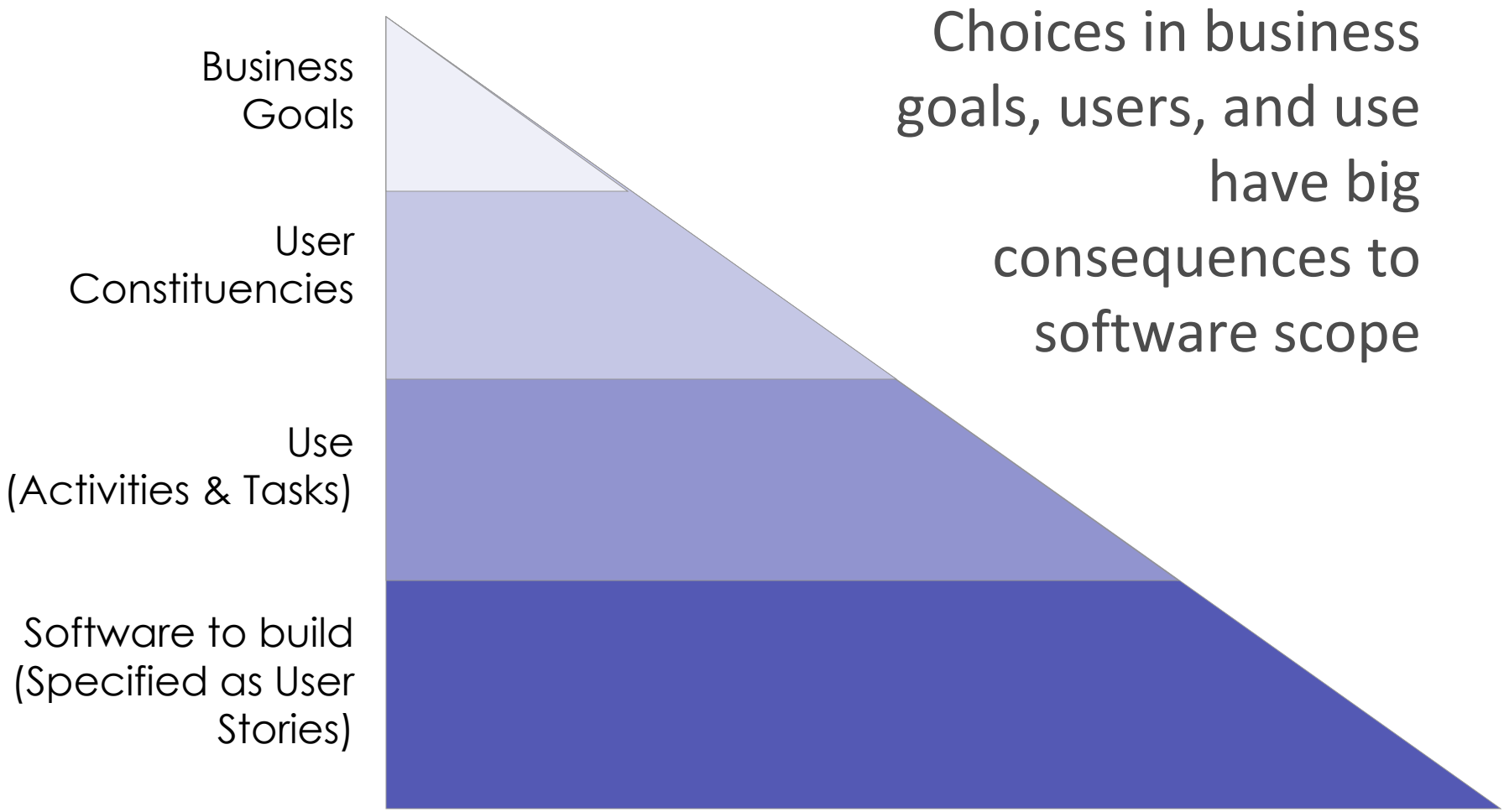
The answer to these questions are useful **metrics**

Use product goals to identify candidate incremental releases, where each release delivers benefit

- Create horizontal swim-lanes to group features into releases
- Arrange features vertically by necessity from the user's perspective
- Split tasks into parts that can be deferred till later releases
- Use the product goals from your handouts to identify slices that incrementally realize product goals

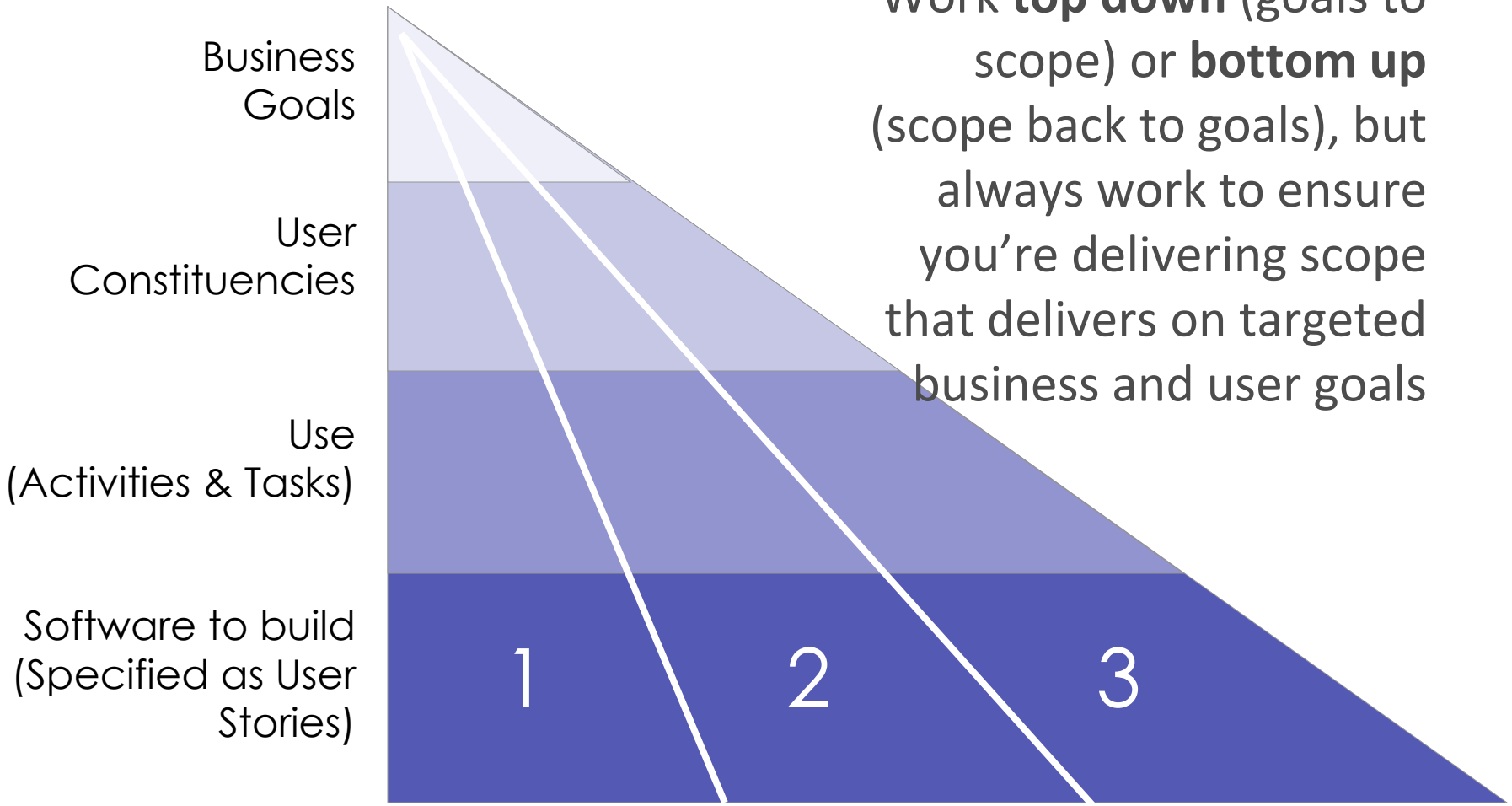


The software we choose to build is a consequence of smaller upstream choices



Choices in business goals, users, and use have big consequences to software scope

Segmenting scope into incremental releases also segments use, user goals, and business goals



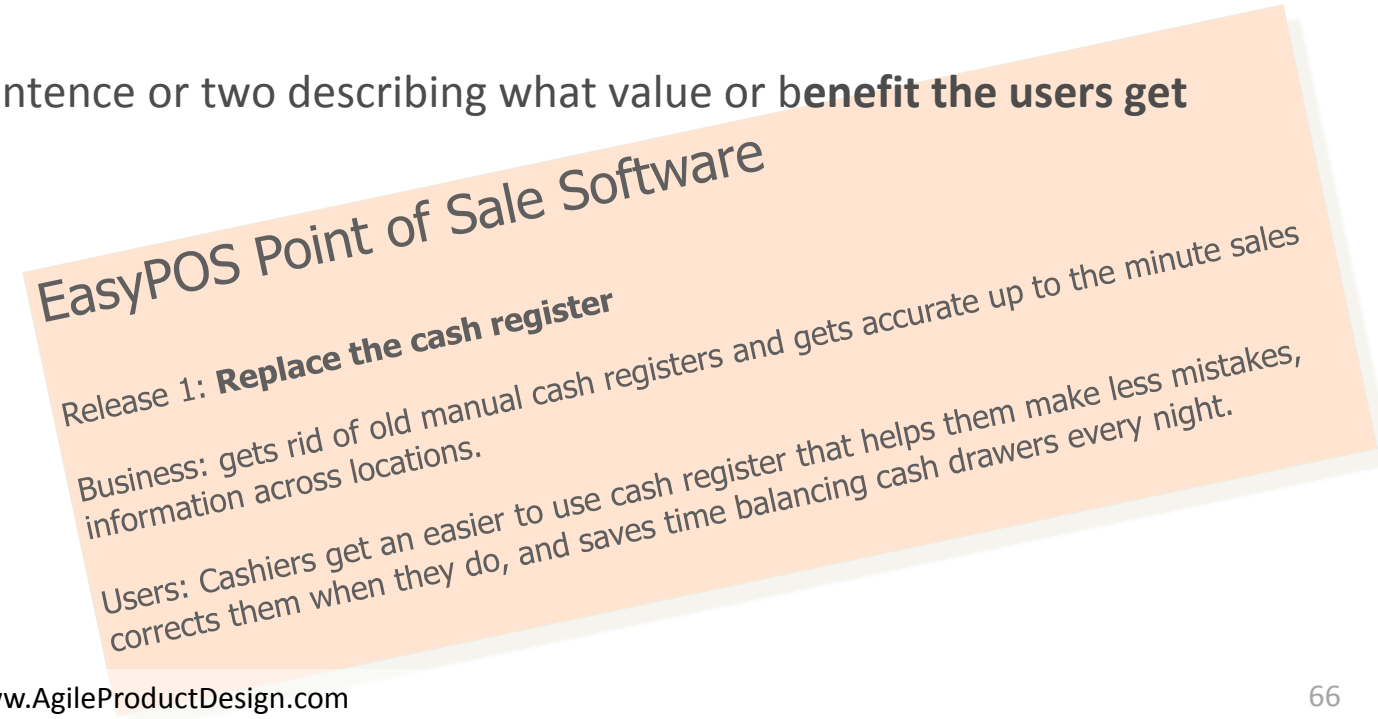
Work **top down** (goals to scope) or **bottom up** (scope back to goals), but always work to ensure you're delivering scope that delivers on targeted business and user goals

A product release roadmap targets benefit delivered over time

A roadmap serves to clearly communicate release level product goals and benefits to stakeholders

For each incremental release:

- Give the release a **name** or simple statement describing its purpose – think mantra
- Write a short sentence or two describing what value or **benefit the business gets**
- Write a short sentence or two describing what value or b**enefit the users get**



Turn your sliced story map into a roadmap

For each slice in your release:

- Give the release a **name** or simple statement describing its purpose – think mantra
- Write a short sentence or two describing what value or **benefit the business gets**
- Write a short sentence or two describing what value or b**enefit the users get**

EasyPOS Point of Sale Software

Release 1: **Replace the cash register**

Business: gets rid of old manual cash registers and gets accurate up to the minute sales information across locations.

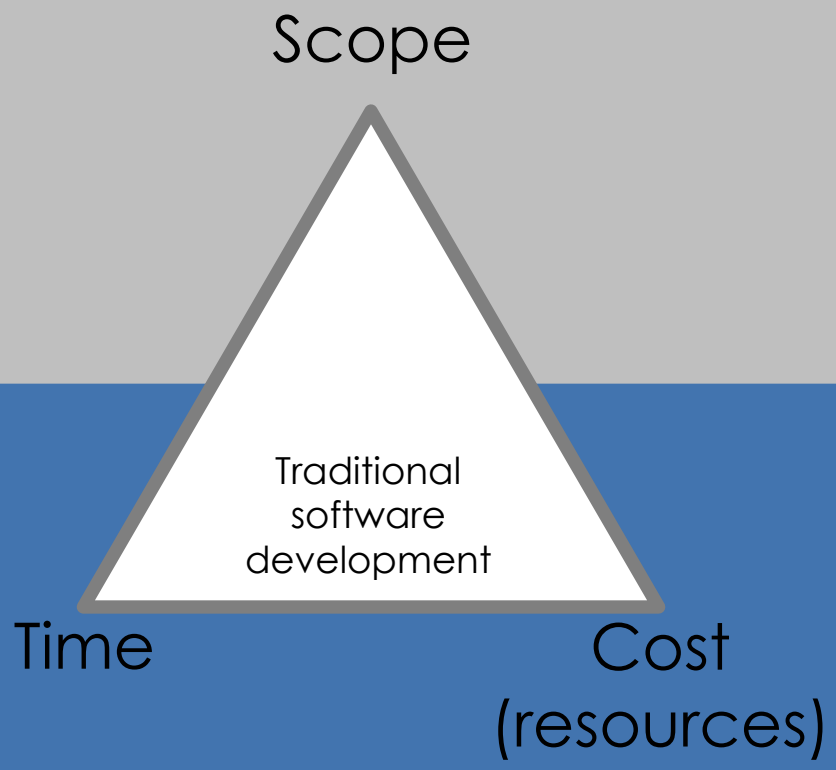
Users: Cashiers get an easier to use cash register that helps them make less mistakes, corrects them when they do, and saves time balancing cash drawers every night.

Iteratively and incrementally construct software

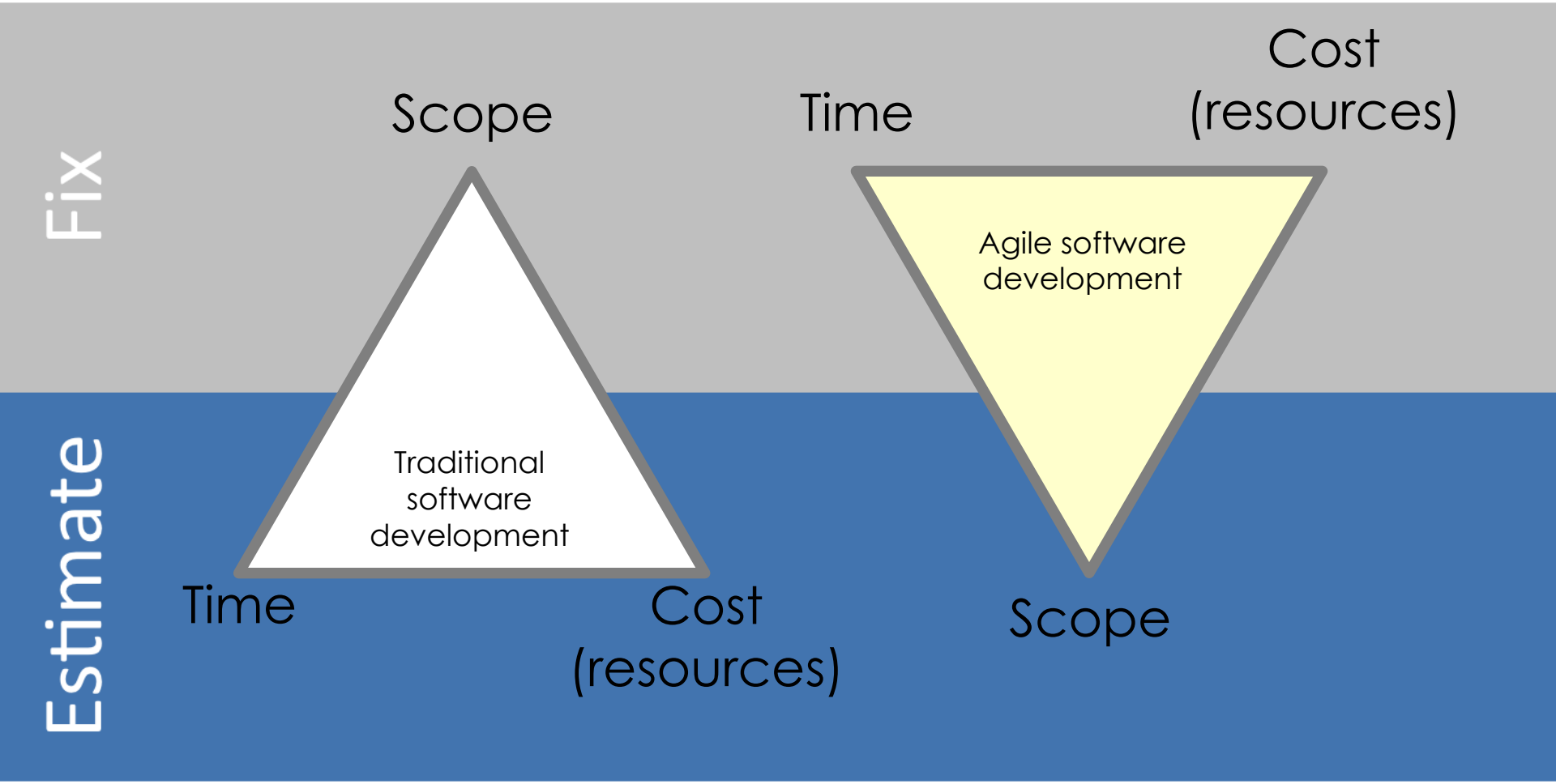
Traditional software development fixes scope then estimates, and attempts to fix time and cost

Fix

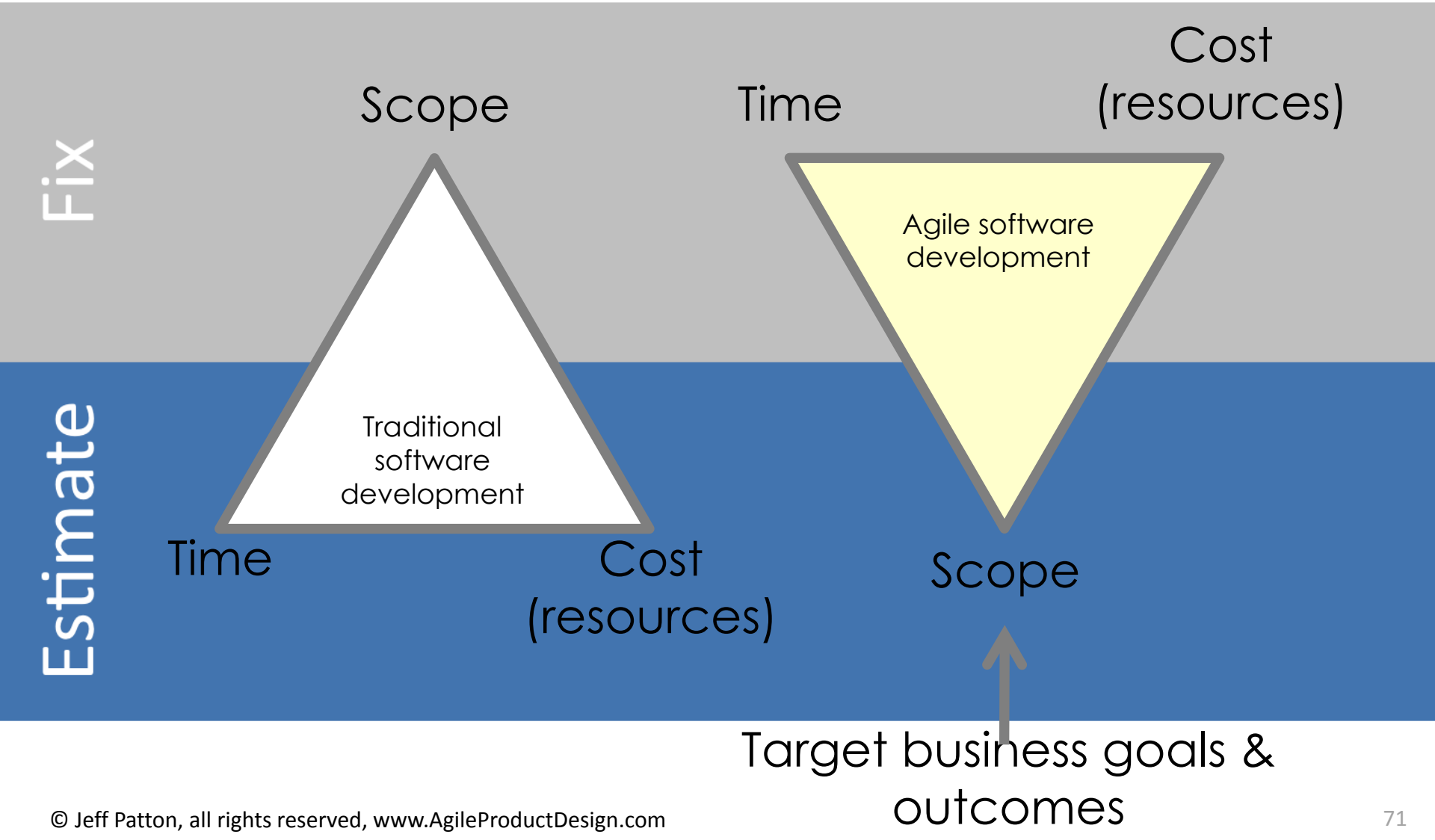
Estimate



Agile development fixes time and cost, then leverages iteration and incrementing to maximize scope



Leverage a shared understanding of desired product goals to minimize scope while maximizing value



To release benefit on a schedule we'll need to leverage incremental and iterative thinking

(What's the difference?)

“incrementing” builds a bit at a time

Incrementing calls for a fully formed idea.

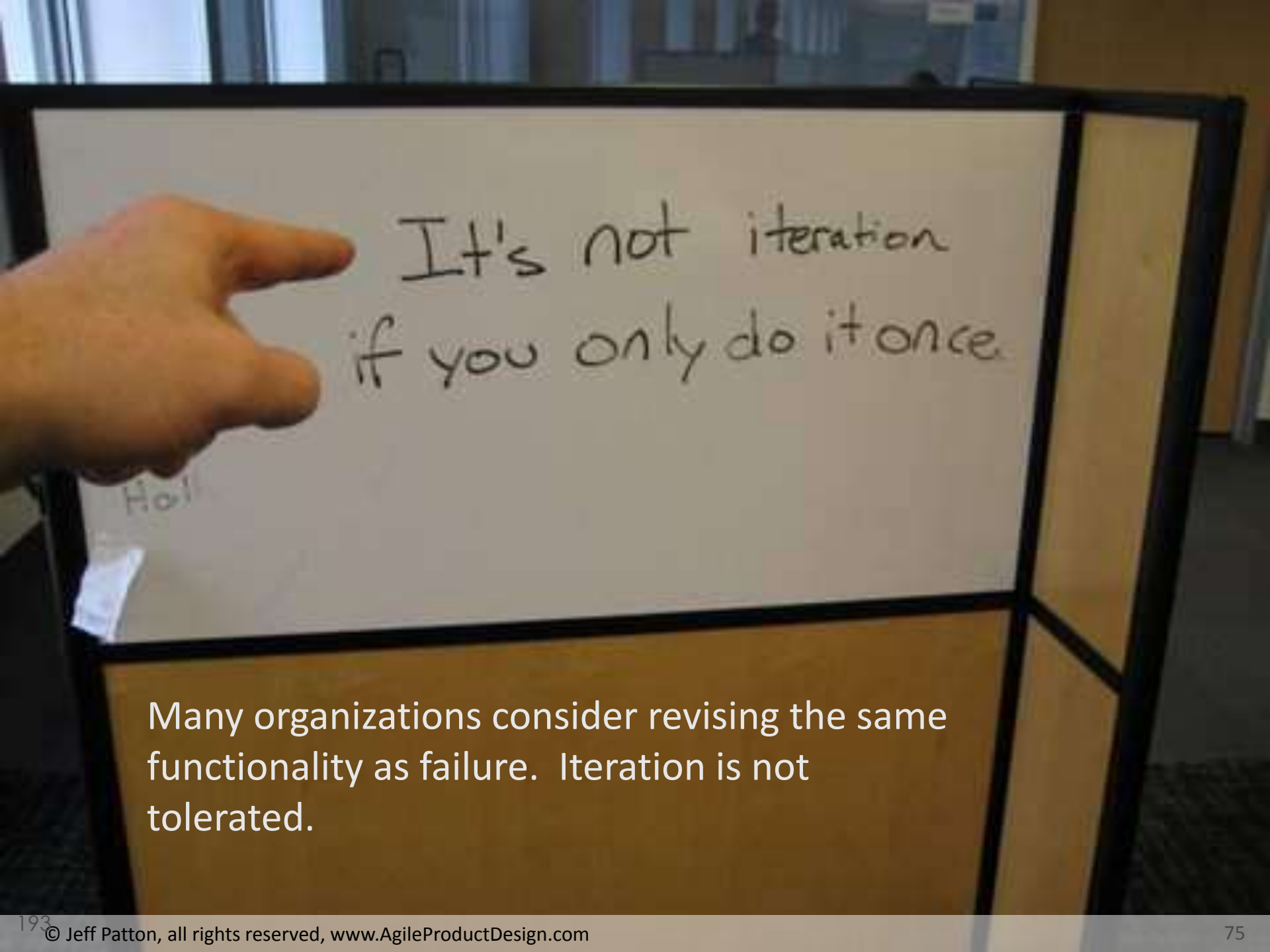
And, doing it on time requires dead accurate estimation.

1 2 3 4 5

“iterating” builds a rough version, validates it, then slowly builds up quality

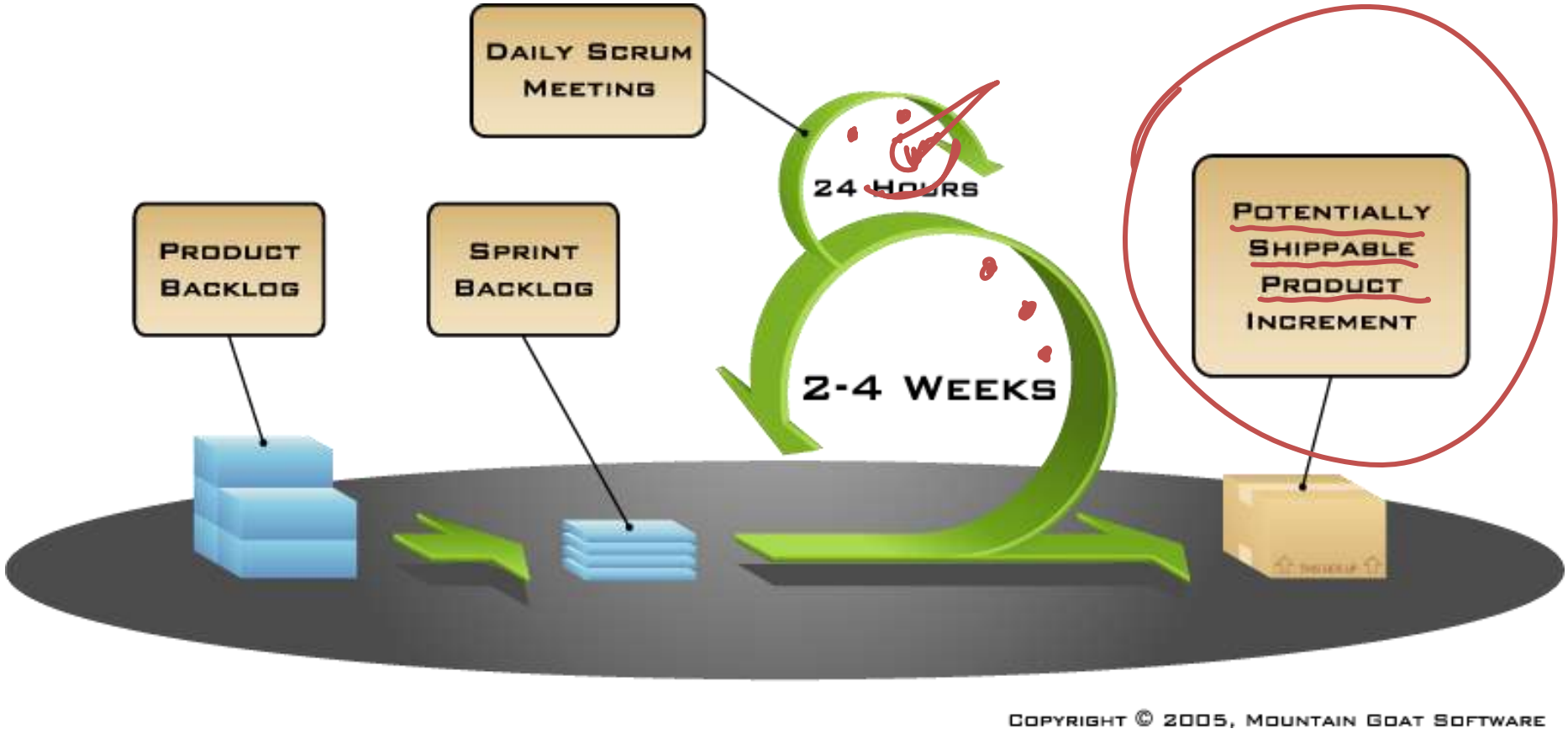
A more iterative allows you to move from vague idea to realization making course corrections as you go.

1 2 3 4 5



Many organizations consider revising the same functionality as failure. Iteration is not tolerated.

As a product owner, you need a more refined understanding of “shippable”



Keeping our user stories task-centric allowed us to defer solution decisions

Make feature decisions in the context of time and budget limitations

hole

(to put the flower in)

dig hole

hold my options open

Commit to satisfying user needs, not to specific features

hole

(to put the flower in)

dig hole

Products with similar features often vary substantially in the price we pay

Think about the high-level features in a car - well a bus in our example

At a high level, all features are necessary

But we know that all buses don't have the same price

Each essential feature varies in subjective quality affecting the final price

engine
transmission
brakes
suspension
seats
steering wheel
…


low cost


moderate cost


high cost

Kano cautions us to consider quality as being composed of **objective** and **subjective** elements

There's more to me than that silly survey technique!

"Discussions of quality have revolved around the two aspects of subjectivity and objectivity since the time of Aristotle.

Embedded in this objective-subjective split is the idea that **objective quality pertains to the 'conformance to requirements'** while **subjective quality pertains to the 'satisfaction of users.'"**

--Noriaki Kano

Kano explains three general classifications for product features: **must-haves**, **one-dimensionals**, and **delighters**

"This car has many flaws. Buy it anyway. It's so much fun to drive"

-- from a NY Times review of the Mini Cooper

Must-haves

The products must have this features for me to be consider the product acceptable

One-dimensionals

The more of this I get, the better

Delighters

I love this element of the product!

Separate objective quality from subjective quality

Objective quality refers to the visible measurable, and easily validated characteristics of the product usually in relation to the products' specifications.

- Does the product perform bug free as specified?
- Expect objective quality to be high.

In Scrum, this is non-negotiable- objective quality is shipable every sprint.

Subjective quality refers to the specification or product design choices you make as a product owner. These choices affect the product users' perception of quality

- Is the product simple to use?
- Is the product efficient to use?
- Do I like using the product?

Use the Kano classifications to both prioritize and split

Brakes
(must have)

Basic brakes
(must have)

Stopping distance
(one dimensional)

Anti-locking
(delighter)

Cool dashboard light when slipping
(delighter)

Keep in mind: you must know your customers and users to determine subjective value.

One person's delighter may leave others apathetic.

Another's must have is useless to other customers

Let's look at what happens if we take a naive incremental aproach to construction

Let's start with the basic features of our bus.

Product goal: (**in 4 sprints**) be driving the coolest bus in town

We can leverage iteration to build up quality

Iterating affords building up quality over time

1

2

3

Consider these four story splitting heuristics that build up quality

Bare Necessity

For the feature to be minimally demonstrable – but not releasable, what is the minimal functionality

Example: A form with only necessary fields and no validation

Capability & Flexibility

What would add the ability to perform the user task in different ways? Adding in sub tasks that are optionally performed?

Example: a form with optional fields, date lookup tools, input translation on dates

Safety

What would make this feature safer for me to use? For both the user, and for the business paying for the software?

Example: input validation, enforcement of business rules such as credit card validation

Usability, Performance, Sex Appeal

What would make this feature easier to use? More desirable to use? Faster to use?

Example: auto-completion, sexy visual design, speed keys

* Adapted from Gerard Meszaros' "Storyotypes"

Building up quality iteratively and incrementally ships the best product possible

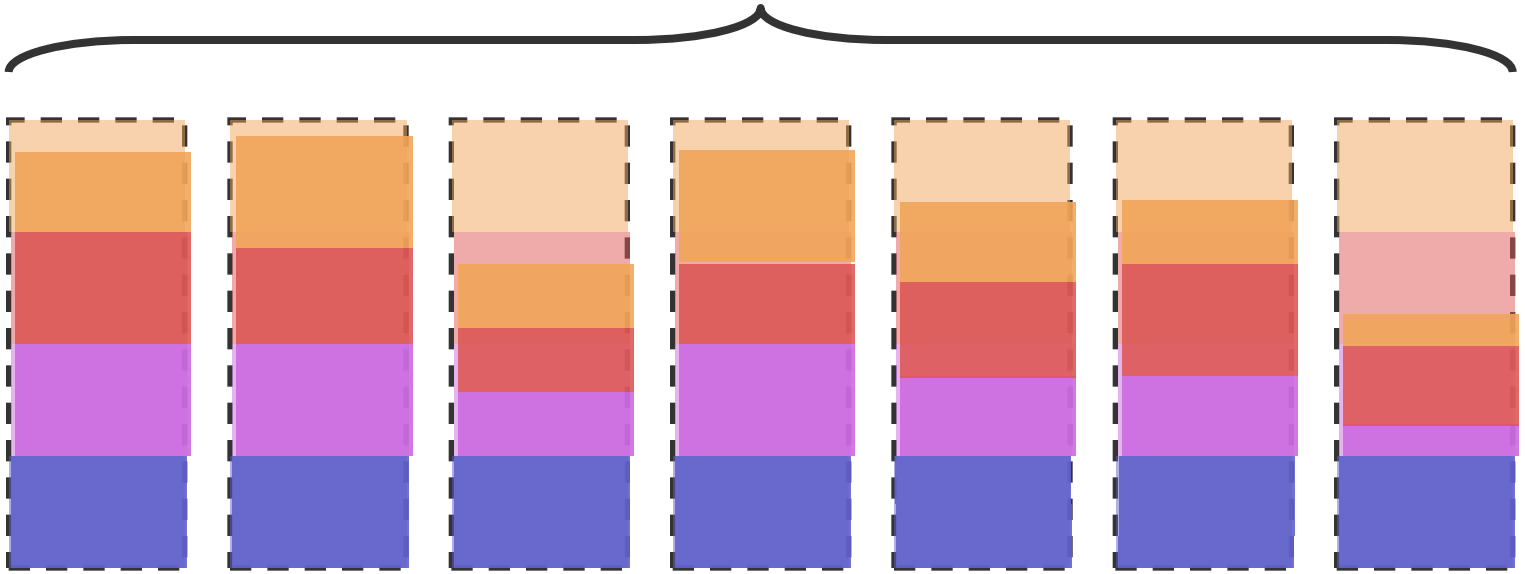
We know each story can be split into at least four parts

Early iterations strive to build bare necessities, later iterations build up quality

Evaluating readiness based on subjective quality to understand doneness

sprint 4

A- A B A A- A- B-



Product goal: (**in 4 sprints**) be driving the highest quality bus possible

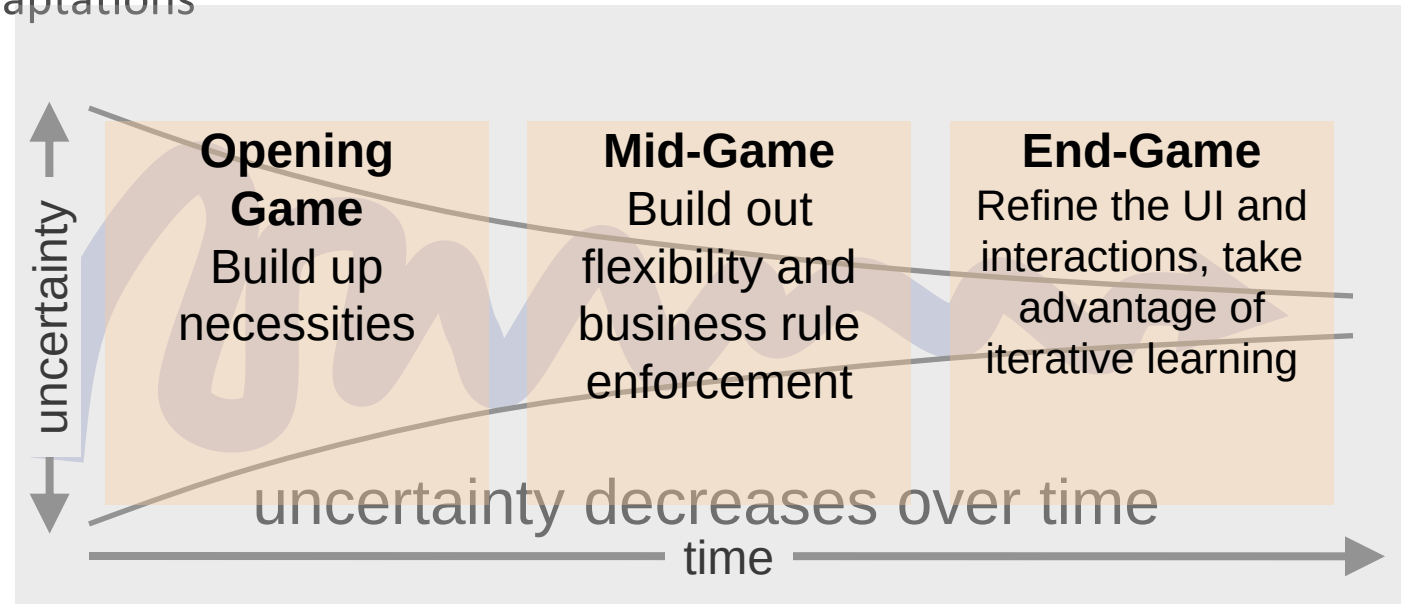
Divide release design & development into three phases

Opening Game: Build a simple system span of necessary features first – the walking skeleton

Mid-Game: Add flexibility and safety next

End Game: Finish with comfort, performance, and luxury

Reserve time in the remaining third for unforeseen additions and adaptations

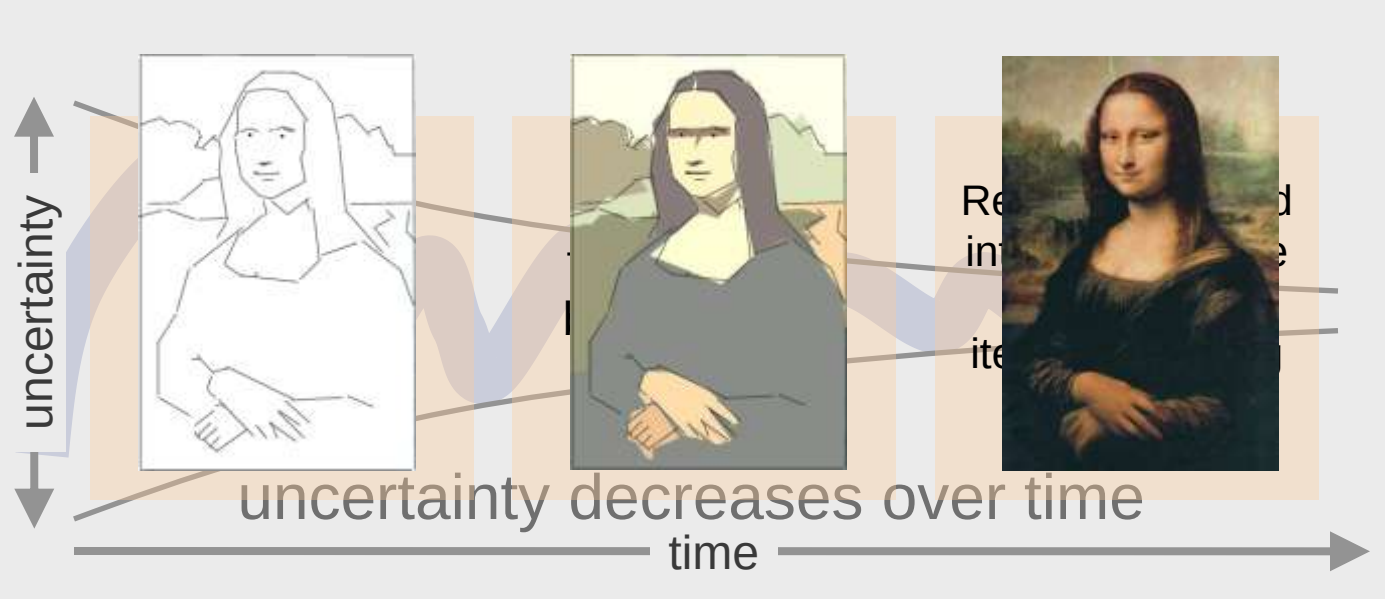


Construx on the Cone of Uncertainty: http://www.construx.com/Page.aspx?hid=1648
Visdos on the cone: http://www.implementingscrum.com/2008/02/19/vegas-hangover-enlightenment/

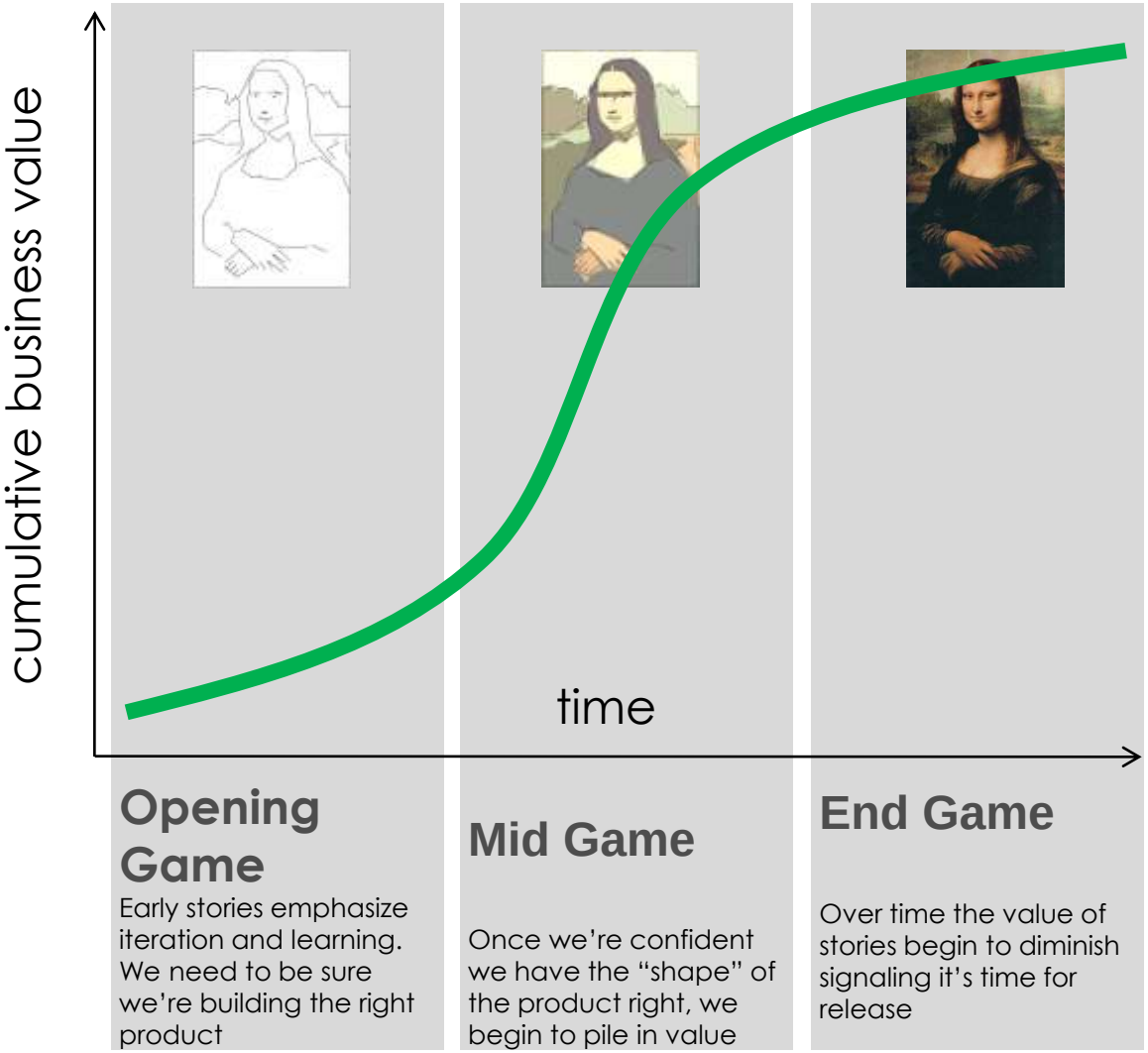
This product growing strategy slowly brings the product into focus

An artist envisions an entire painting by starting with a sketch or an under-painting and slowly building up detail

Apply the same strategy to learn about the product domain as quickly as possible – to chase out uncertainty before too heavily investing



Looking at the release of business value over time lets us see what's going on here



To finish on time we may "trim the tail" by deferring stories of modest value

Split a task-centric backlog item into smaller iteration stories using the 4 quality heuristics

Work in small groups of 2-3 people.

Review the Story Map – the organized backlog – for the Barney's problem

Choose a story you'd like to work with, one where your group can imagine a prospective user interface.

Brainstorm the smaller stories that could build up the larger story to its full quality.

Arrange your brainstormed stories into 3 pile: opening, mid, and end game.

Guidelines for releasing on time

Thin stories aggressively during early sprints to build all essential functionality early.

Build up functionality only after all necessities are in place.

Protect time in the final sprints for product refinement.

Assess release readiness at the end of each sprint as part of product review.

Parting thoughts

Questions?

Building Better Products Using

User Story Mapping

Jeff Patton

jpatton@acm.org

www.agileproductdesign.com