



BITS Pilani presentation

BITS Pilani
Pilani Campus

K.Anantharaman
kanantharaman@wilp.bits-pilani.ac.in



SE ZG 544 , Agile Software Process

Module5 – Agile Requirements & Agile Estimation

Book References

1. The Agile Sketchpad, O'Reilly Media, Dawn Griffiths, David Griffiths
2. Writing User Stories By Ryan Harper, O'Reilly Media
3. The Agile Samurai by Jonathan Rasmusson Published by Pragmatic Bookshelf, 2010

Requirements Gathering in Waterfall Method



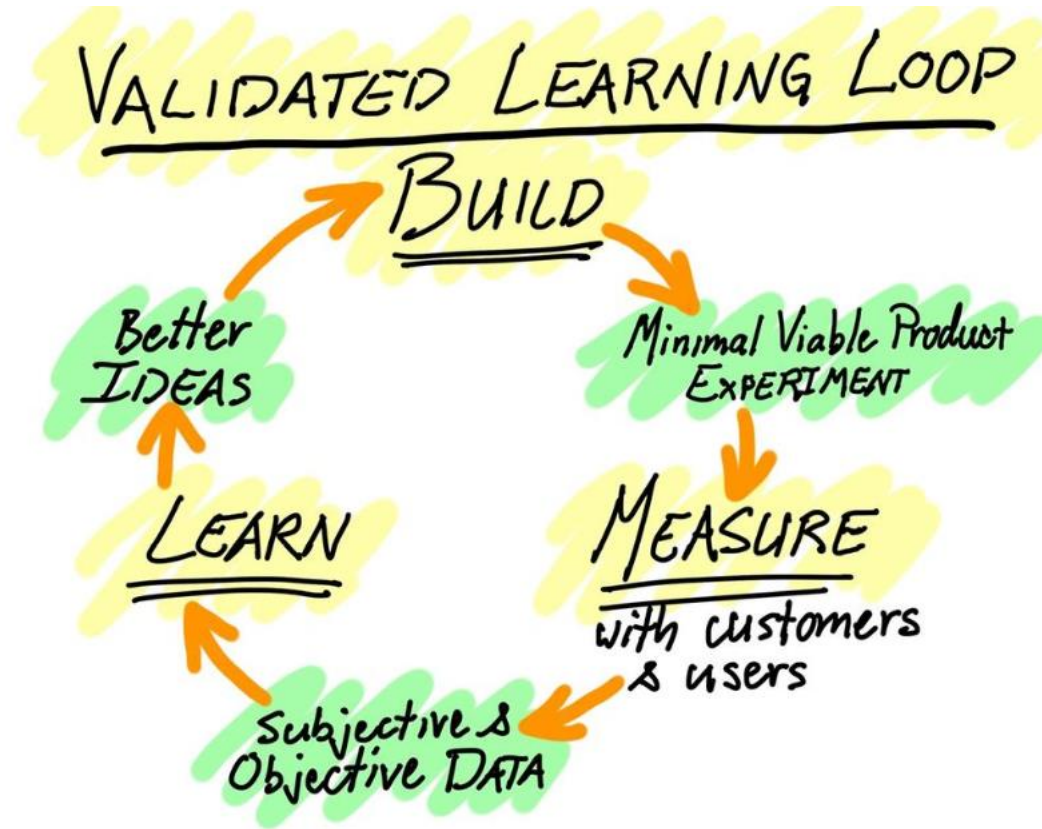
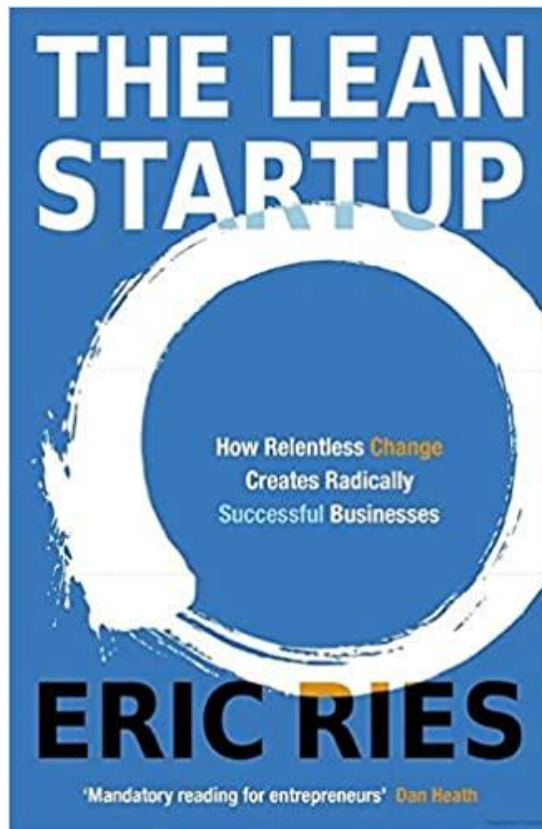
- In Waterfall model, We tend to describe upfront how the entire product/system will work and document them.
 - PRD or SRS or CRS
- The problem with gathering requirements as documentation isn't one of volume—it's **one of communication**. It's just really easy to misinterpret what someone wrote.
- Other Issues:
 - Lengthy Process (1 to 3 months), Sometime Project wont get started
 - Requirement change is hard, especially late in the cycle
 - Bad guesses and wrong assumptions and so on

Requirements Gathering in Agile



- In Agile Projects, **User Stories** are the main way to track the information or requirements, of the project.
- User Stories tell us about:
 - What customer the wants the team to do?
 - How valuable the work is?
 - How long it is likely to take completed?
- **User stories are fundamental unit** of product development in Agile environments
- User stories describe single feature **which enable rapid iteration.**

User stories at heart of the build, measure, learn cycle



Why Write User Stories?

Empathic design is a user-centered design approach

- User stories also enable empathic design and development as they are written from the perspective of the end user
- A well-written user story will communicate both how a feature will work and how it will benefit the end-user
- User stories ensure that teams are building features to meet a user goal or need instead of “building stuff to build stuff”

Product Management



- **We need to keep in mind the business, Technology, and User Interface in mind when we write user stories.**
- It is Ok, to help customer to write user stories

In a real situations, User stories are written also by:

- The Agile team
- Other Stake holders

What is the User Story and How it looks?



The User Story Template

As a <type of user>
I want <some goal>
so that <some reason>.



who is this story for
what they want to do
why they want to do it

The user role?

The value (the most difficult part to write in a user story)

An Example , showing the Front side of the card

User stories are written on Index cards or Sticky notes.

Why it is written on card? (Deliberately, not to provide more information, just a promise for a conversation with the customer and the team later.)



Another User Story format and Examples



Writing User Stories

The user story is written in the following format:

"As a [user], I want to do [x] so that I can accomplish [y]."

For example, "As a Gmail user, I want to be able to attach a photo to an email so that I can share it as part of my message."

Who?

What?

Why?

Note: If the user story involves a frontend (user-facing) design component, the design files should be included with the user story

User Story Examples

A user story for returning images in Google Image search.

“As a Google Images user, when I search for an image I want to see images that match my query so that I can find the image for which I’m looking.”

- Note that, user story does not focus on how the images will be returned or displayed, but rather on end user’s goal and needs.

Writing Acceptance Criteria for User Stories



- The second part of the user story , **the acceptance criteria**, explains how the feature will work.
- The acceptance criteria consists of series of **boolean statements (true or false)**, such as “ When [x] happens, [y] should happen

Acceptance Criteria/Conditions of Satisfaction



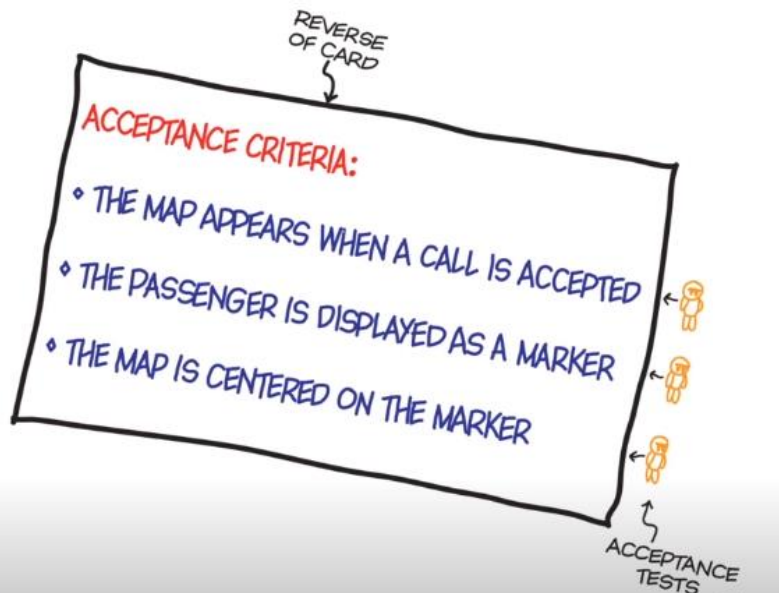
Creating clear acceptance criteria reduces ambiguity for the development team and allows a feature to be easily tested.

Acceptance criteria can also serve as a form of documentation once a feature has gone live, providing a written record of how a feature is intended to work.

Examples Acceptance Criteria



Back of the User Story card



For example for an Gmail user,

“ When the user saves an email that has not been sent, it should be stored in the user’s Drafts folder”

Outcome: Email stored in Drafts (Yes) or Not (No)

Acceptance Criteria for Google Image search



"As a Google Images user, when I search for an image I want to see images that match my query so that I can find the image for which I'm looking."

Some possible acceptance criteria:

"When the user inputs a query, such as 'cat', the image results should be returned in order of relevance."

"The image results should be returned in rows."

"When the user clicks/taps on an image, a detail view of that image should appear between that image's row and the row below."

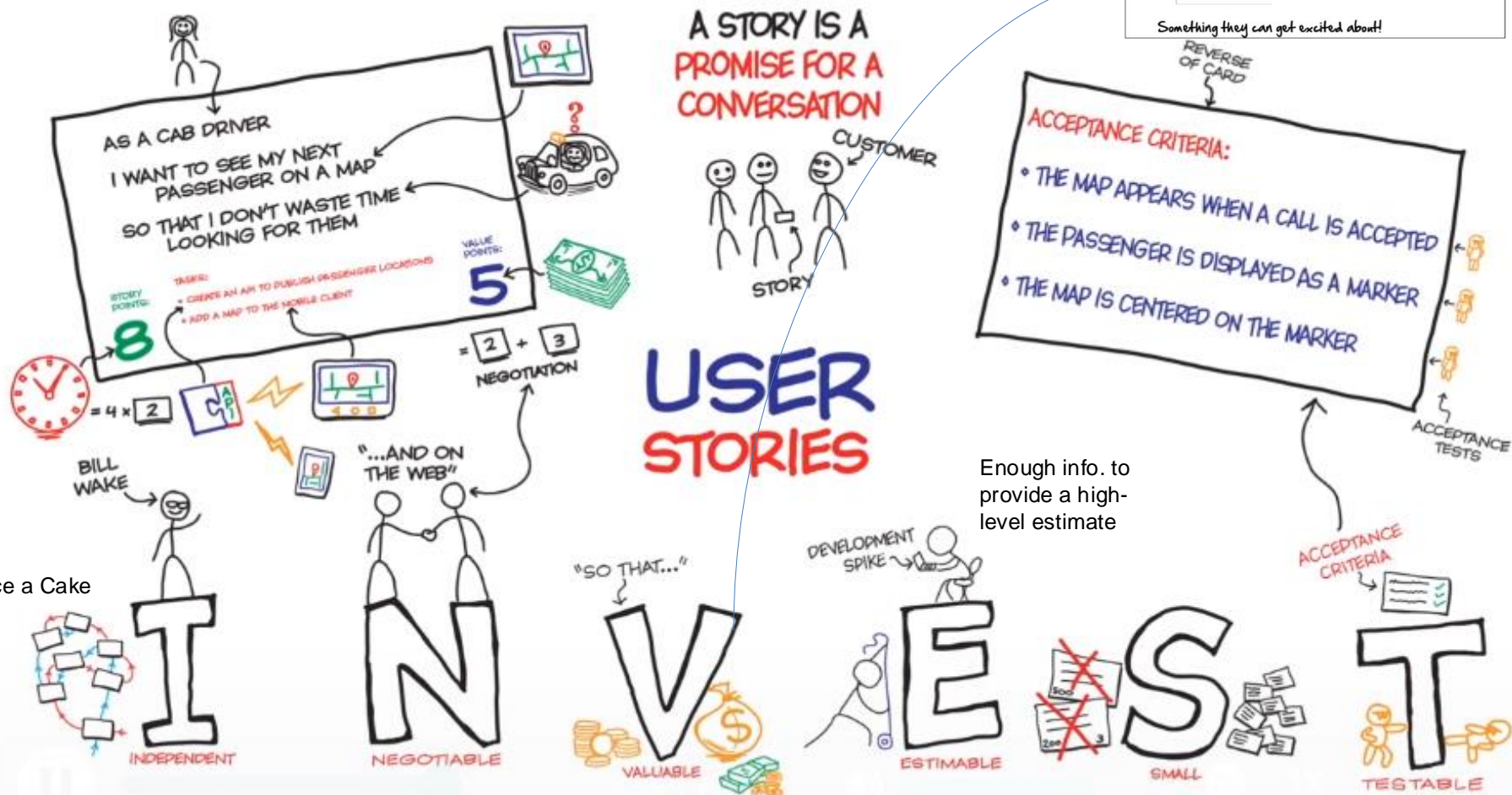
1. Relevance: The image most related to user query appears first, the images least related to user query appears last
2. Rows: Layout function
3. Tap on Image: how user will interact with the image

Elements of Good User Stories



- Bill Wake came up with the **INVEST** acronym for good user story.
- Good user stories also have the following characteristics:
- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimatable
- **S**mall
- **T**estable

Characterterics of a Good User Story (INVEST)



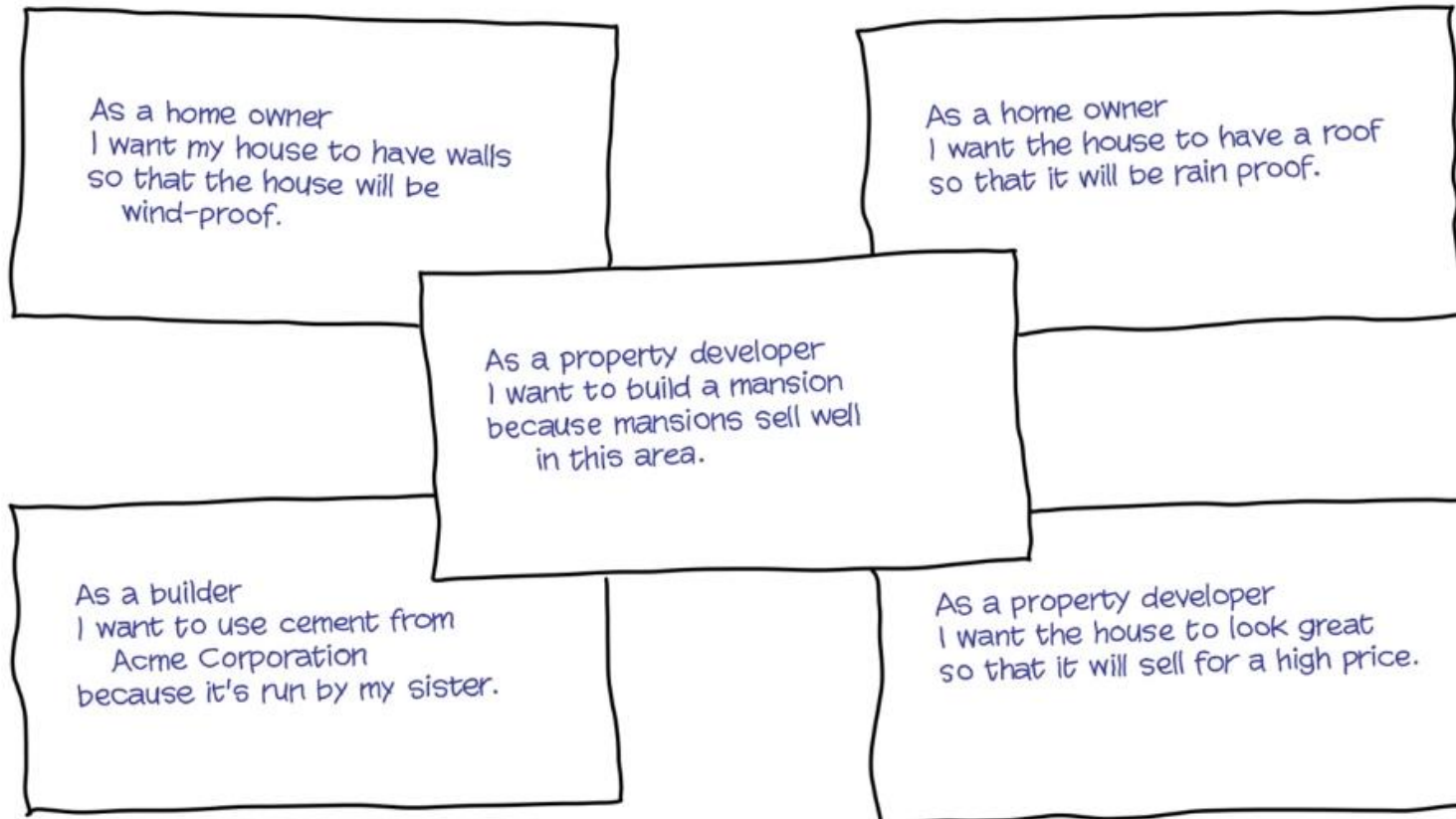
User Stories – Exercise (Customer is a Property Developer)

innovate

achieve

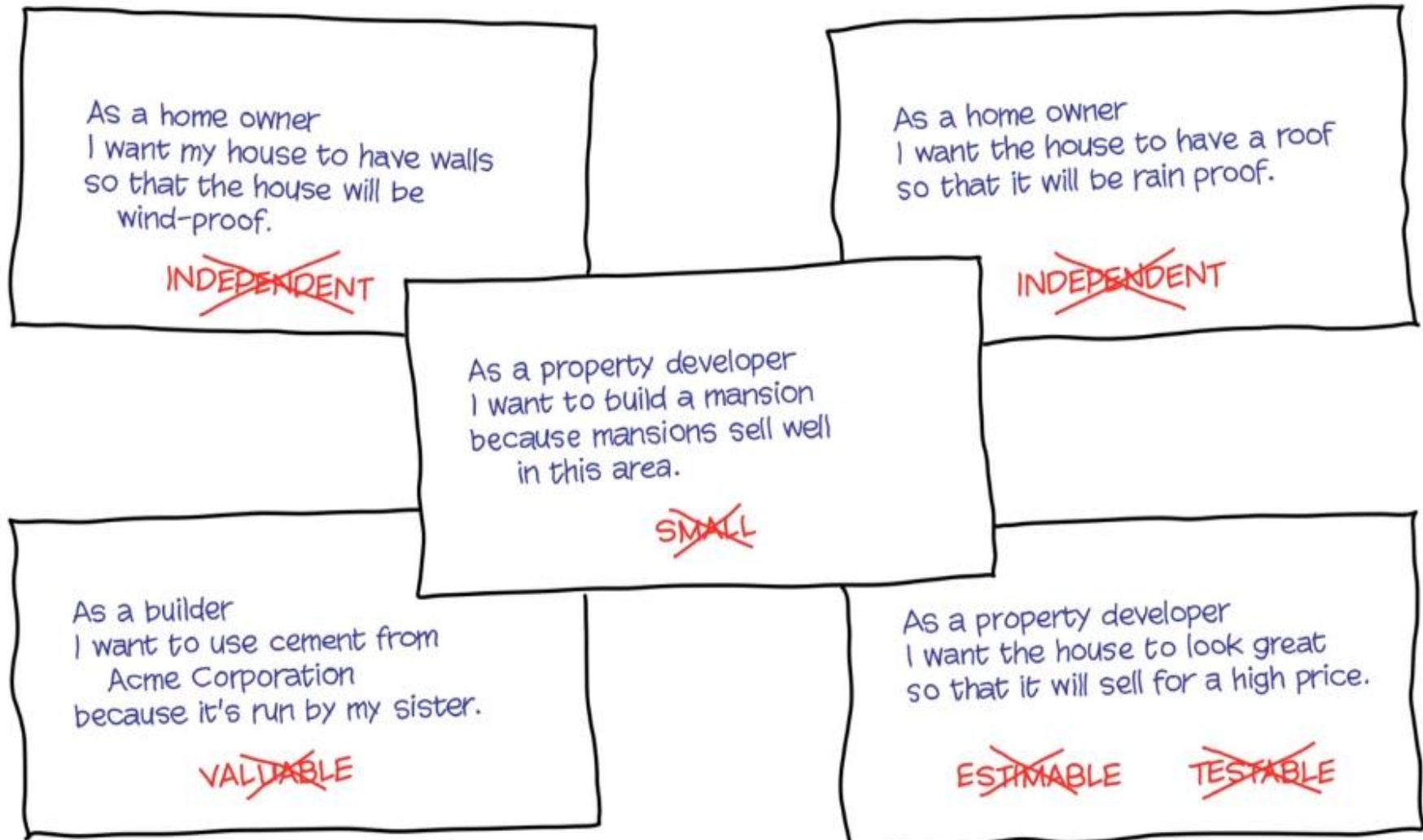
lead

Project is to build a House



Apply INVEST test and check if these stories are good or not?

User Stories – Exercise (INVEST Test)



The 3 Cs- Story Process



- In the book *Extreme Programming Installed*, Ron Jeffries et al. (Addison-Wesley Longman Publishing) **describe the story process** best:

- Card:**

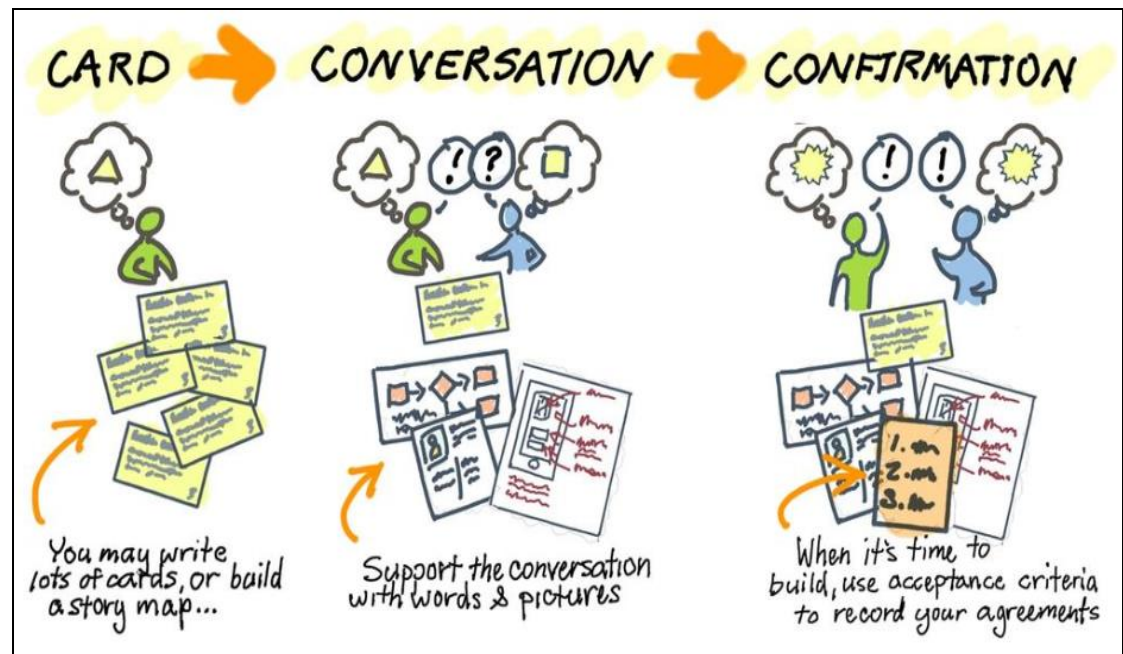
- Write what you'd like to see in the software on a bunch of index cards.

- Conversation**

- Get together and have a rich conversation about what software to build.

- Confirmation**

- Together agree on how you'll confirm that the software is done.



Difference Between User Story, Bugs, Constraints



Features vs. Bugs

Unlike features, bugs (problems with how a live feature is working) are not written using the user story format.

Instead, bugs are documented with a descriptive title and clear steps for how to reproduce the issue.

For example, if tapping a pause button on a video player in Mobile Safari wasn't working, the bug could be written as follows:

Title: Tapping Pause on Video Player Doesn't Work

1. In Mobile Safari on iOS 10.3 / iPhone 6S, go to myvideoplayer.com/videoexample.
2. The video will play automatically on mute.
3. Tap the pause button. The pause button does not become a play button, and the video is not paused.

Constraints

Story: Website must be superfast

Story: Design should look really good
– Constraint Card

- **Stories like these, we call constraints.**
- But they are important because they describe characteristics our customers would like to see in their software.
- For example, The Website must be superfast can be written like this.

All web pages must
load in less than 2 sec



A constraint card

How to take care of Frontend Design?



- If a user story includes a new design, the design files illustrating that design (including any interactions) are included with user story.
 - Design tools: Adobe Photoshop, Sketch and Invision
 - Design files: Wireframes, Mockup, Prototypes

<https://justcoded.com/blog/wireframe-mockup-and-prototype-whats-the-difference/>

How User Stories Promote Empathy



- One benefit of the user story format is that it requires the product manager to think about the feature from the end user's perspective.
- How, when, why, and where the user will interact with the feature will affect how the feature is surfaced.
- What elements are included in the feature, and how the success of the feature will be determined

Example from WebMD apps



Case Study: WebMD



Case Study: WebMD



This design is not optimal for user, based on the feedback from app store because no priority order, though the conditions are prioritized

After adding relevance indicator, user able to decide whether to seek medical condition or not

Story Grooming Meeting



- Story grooming meetings give the engineering team a chance to review user stories **before they are scheduled for a development sprint.**
- Story grooming meetings are **critical for securing the development team's buy-in.**
- The team is given a chance to ask the questions that would normally arise during sprint planning:
 - What should we do if the user enters invalid data here?
 - Are all users allowed to access this part of the system?
 - What happens if...?
- The team provides feedback on the feasibility, viability, and size of each feature and may provide alternate solutions/ or identify previously unforeseen prerequisites or roadblocks for the user needs identified in the user stories.



Agile Estimation

Absolute Estimation



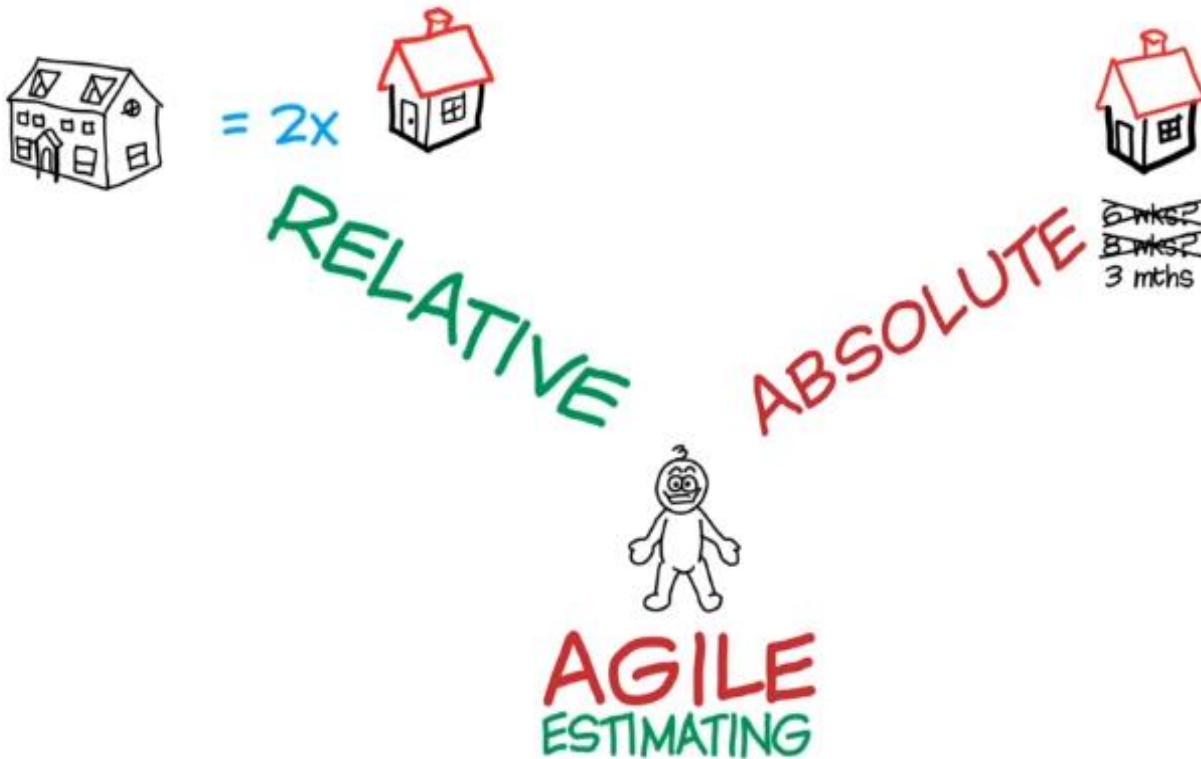
- We estimate our work in hours, days, and weeks.
- We use all the knowledge and experience at hand to make a guess about the amount of time it is going to take.
- Estimation is **approximate and not accurate**
- Absolute estimation:
 - Estimating in absolute values (Examples, days, weeks, months or KMs, Miles)
 - **Absolute values are not easier to judge**
 - **People are not good at absolute estimation**

Relative Estimation



- Agile team use relative estimation.
- Relative estimating compares what you don't know against what you do know.
 - For example, You might not be able to guess how much a truck weighs, but if you saw the truck, you can probably guess how many cars equal a truck.
 - A “customer search” story, as it probably involves double the effort to implement than a simple “Login user” story.
- Relative estimation is easier to judge than absolute values.
 - This means judging how big or complex tasks are with respect to other tasks
- This estimation is not designed to be precise.
 - But that doesn't mean it's useless. Instead it gives you a starting point, a way to start the discussion on what it takes to deliver your stories.

Absolute vs Relative Estimation



As an analogy, it is much easier to say that Delhi to Bangalore is twice the distance of Mumbai to Bangalore than saying that the distance from Delhi to Bangalore is 2061 kms.

Why Agile team use Relative Estimation?

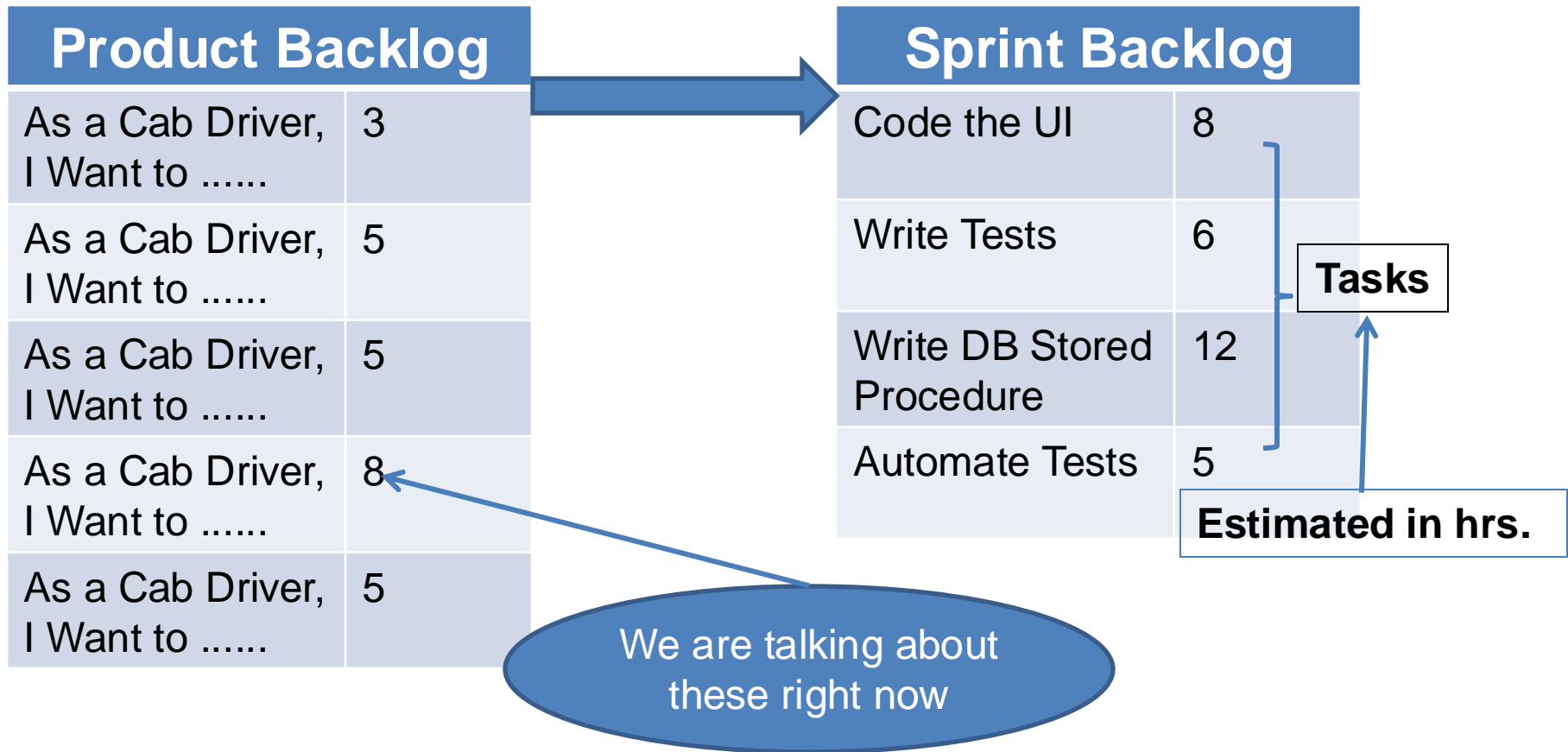


1. Relative estimation takes away from the false comfort of precision.
 - The team is accepting the fact that the estimates will be imprecise.
 - That way we can start talking about what it takes to deliver this story instead spending too much time on estimates.
2. Agile uses relative estimating is that it keeps the team from **confusing estimates from commitments**.
 - An estimate is the useful information you might give a co-worker. A commitment is something that you usually give to a supervisor. An estimate is a best guess. A commitment is often a worst case scenario. That's why for Agile planning, you want estimates and not commitments.
3. Relative sizing across stories tends to be much more accurate over a larger sample, than trying to estimate each individual story for the effort (in hours) involved.



Story Point Estimation

What are we Estimating?



Story Point for Estimation

- In Agile, we use relative estimation
- We do this by comparing the time to take one story vs time to take another story without using absolute estimates
- We do this by using Story points.
- We will have an exponential number sequence.
 - Something like 1,2,3,5,8, 13 These are the points for each of the stories.
- When we estimate with story points, we assign a point value to each item.
 - **The raw values we assign are unimportant.** What matters are the **relative values**. A story that is assigned a 2 should be twice as much as a story that is assigned a 1. A 2 point story is 2/3 of 3 point story.
 - Instead of assigning 1, 2 and 3, that team could instead have assigned 100, 200 and 300. Or 1 million, 2 million and 3 million. It is the **ratios that matter**, not the actual numbers.

What does a Story Point represent ?



- **Represents the amount of effort or fixed period of time** required to implement a user story. **(Size)**
- Story Point is not an estimate of the amount of time it takes to implement a Story.
- Some argue that it is **a measure of complexity**, but that is only true if the complexity or risk involved in implementing a user story translates into the effort involved in implementing it.

Fibonacci series as Story points



- The most common way is to estimate a user story is to use the **Fibonacci series** (1, 2, 3, 5, 8, 13, 21, 34, 55..... with each number the sum of the preceding numbers.
- Why Fibonacci?
 - It's because numbers that are too close to one another are impossible to distinguish as estimates.
- In Fibonacci series, after the 2 (which is 100% bigger than one), each number is about **60% larger** than the preceding value.

Predictability of User Stories Estimation



- Small stories tend to result in a more accurate and reliable estimates.
- Small stories reduces variability and improves predictability.
- So, a 13 or 20-point story is likely much less predictable than several 2, 3, or 5-point stories.
- Relative story point estimates using the Fibonacci sequence are, by design, increasingly **less accurate for larger estimates** – like the “cone of uncertainty”

How User Stories are estimated by the team?

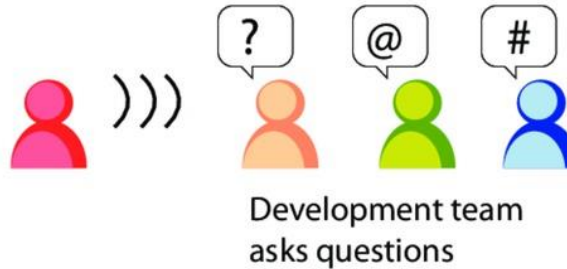


- One method is to play **Planning poker** game.
 - Planning poker helps give everyone a voice.
 - Combining of individual estimates through group discussion leads to better estimates
 - Combat Groupthink-meaning, the way that people tend to agree with the most popular idea.
- Planning poker is a game where the development team estimates stories individually first (using a deck of cards with numbers like **1, 2, 3, 5, 13** ...on them) and then compares the results collectively together after.
- If everyone's estimate is more or less the same, the estimate is kept. If there are differences, however, the team discusses them and estimates again until consensus is reached.

Planning Poker



1. Customer reads story.



2. Team estimates.
This includes testing.



3. Team discusses.



4. Team estimates again.
Repeat until consensus reached.

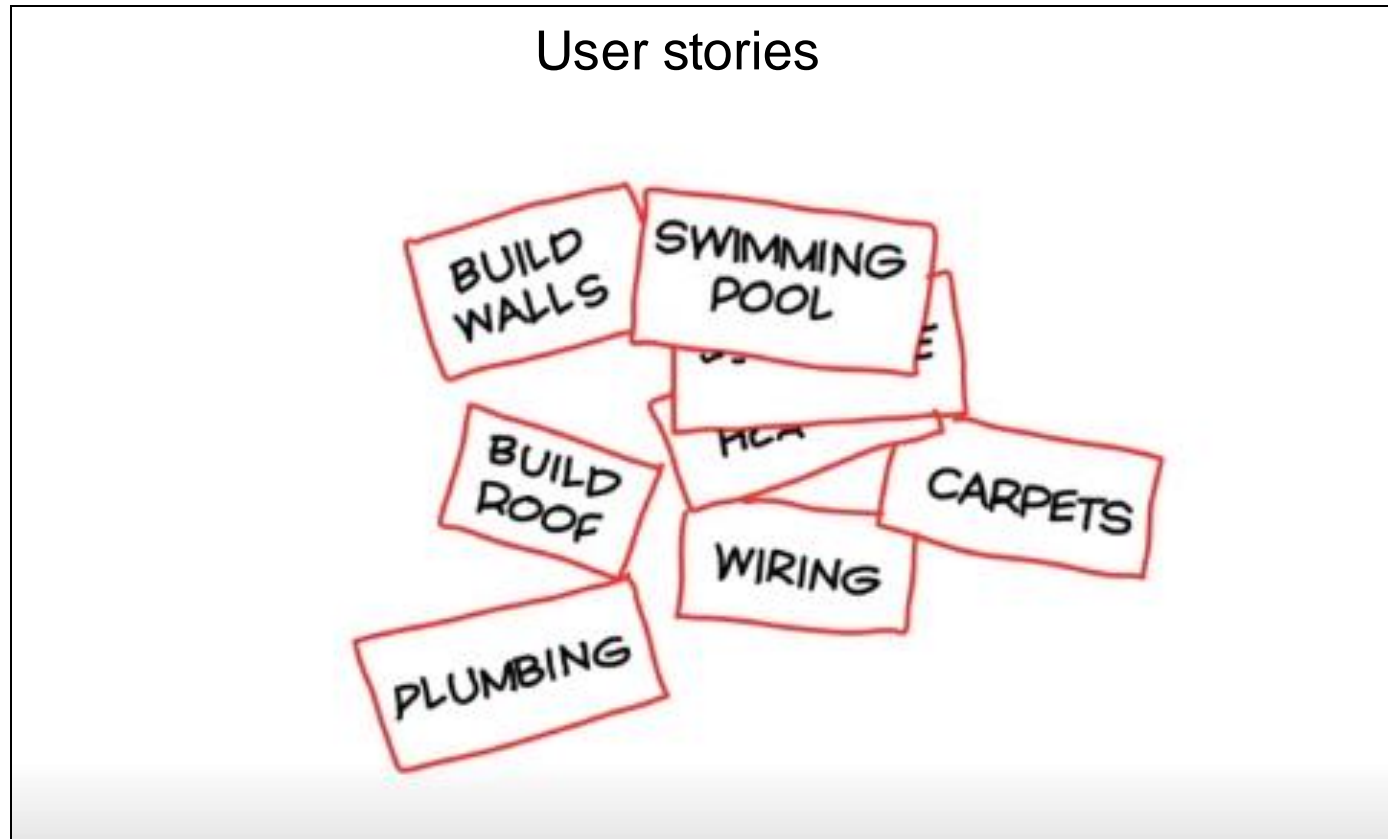


Estimator	Round1	Round 2
Team member-1	3	5
Team member-2	8	5
Team member-3	2	5
Team member-4	5	8

Story Point Estimation – An Example



Project : Build a House, Suppose we have the following bunch of user stories to be estimated. How do we start?

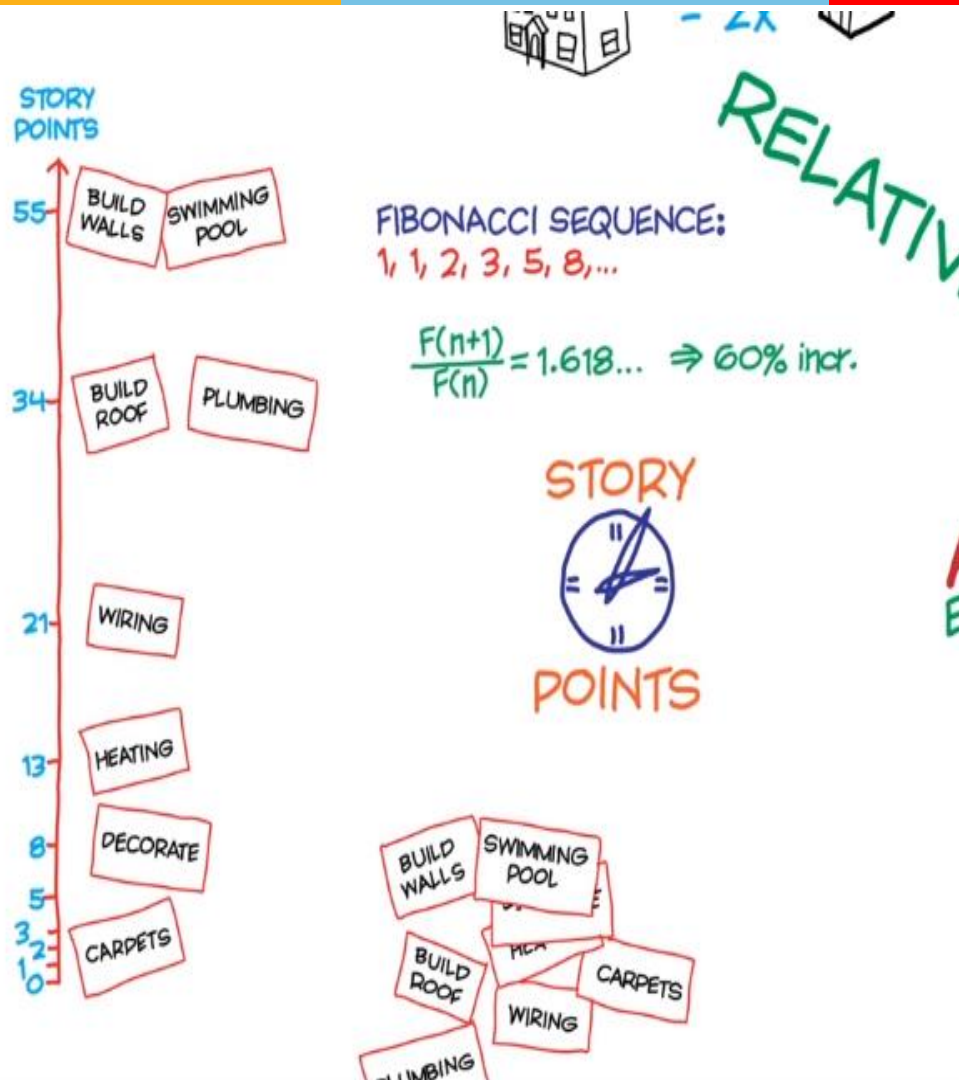


Story Point Estimation – Example

innovate

achieve

lead



Steps:

1. The team may start with the smallest - Carpets Story , assign 2 points.
2. Next, for example, we may discuss the Build wall story. We agree on, it is 30 times larger than Carpet story. Hence, we assign 55 points in Fibonacci scale.
3. Then relatively, assign story points to other stories, Decorate, Heating etc....

At this point, We do not know the effort of each story

Value Point



- A user Story has two estimates.
 - Story point – estimate of time
 - Value Point – estimate of value
- Developers , the people who do the work, estimate User Stories in Story points,
- Customer / Product owner estimates User Stories in Value Point, in the same way.

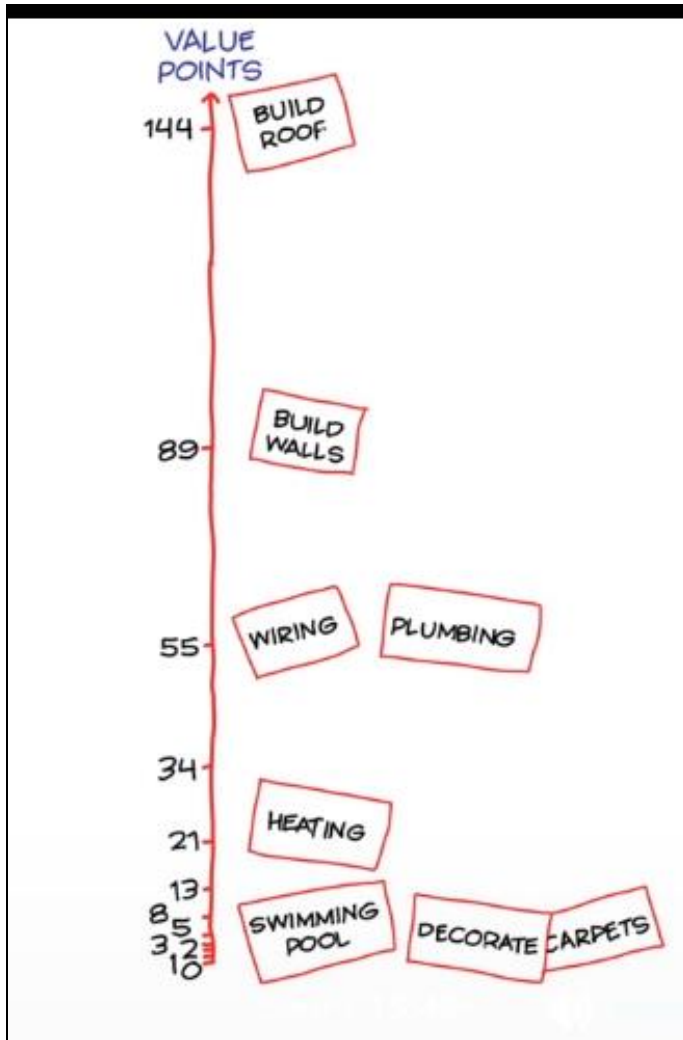
Story Card

Story Point : Amount of time/effort
Value Point : Worth to Customer

**Agile is about delivering
value early.**



Value Point Estimation – use the previous example

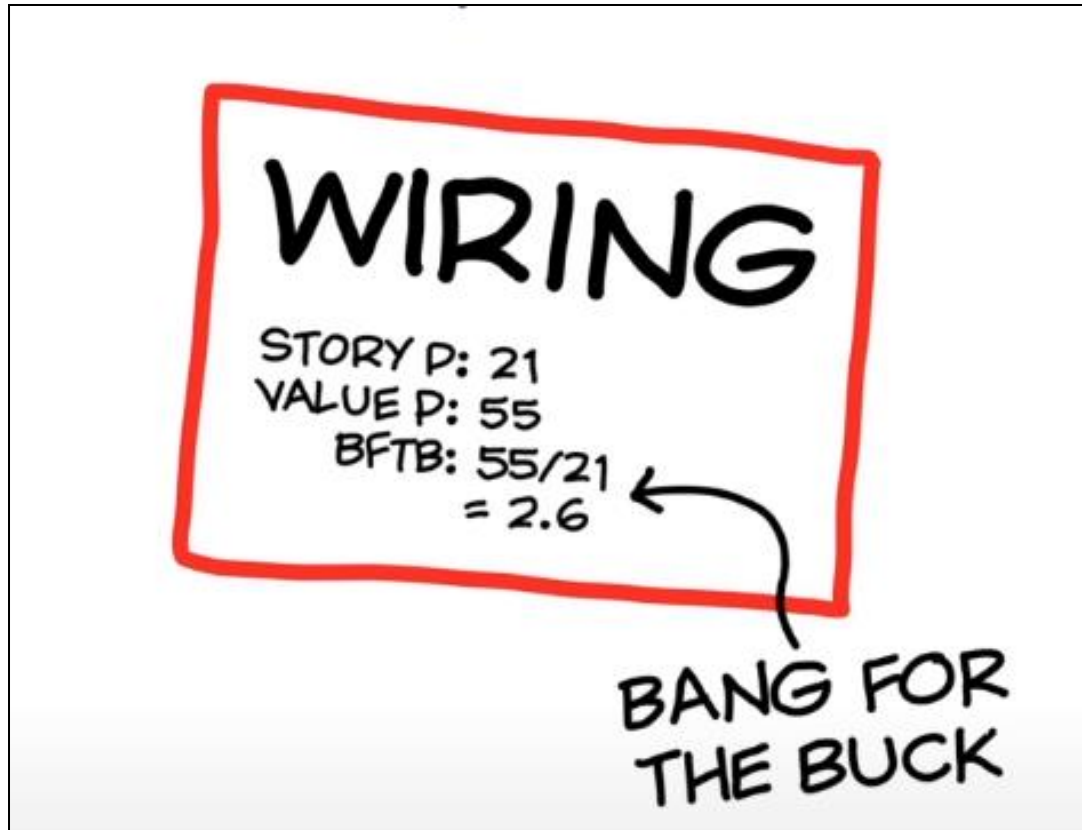


Highest Valued Stories at the top of Product backlog

Relatively Valued Stories

Lowest Valued Stories at the bottom

Bang For the Buck (BFTB) (OR) Priority



BFTB = Value Point divided by Story Point for each story

How stories are prioritized for each Iteration – by BFTB



Product Backlog

	STORY	VALUE	BFTB
BUILD ROOF	34	144	4.2
WIRING	21	55	2.6
BUILD WALLS	55	89	1.6
HEATING	13	21	
PLUMBING	34	55	
CARPETS	2	2	1
DECORATE	8	2	0.25
SWIMMING POOL	55	1	0.01

This story needs to go first (with an arrow pointing to BUILD ROOF)

Velocity



- **Velocity** = Number of story points the team can deliver in an iteration/Sprint (OR)
- The rate at which the team can complete the work within a time frame.
- **Calculating Velocity:**
- For example,

Team Velocity = Average values of the three sprints = $(16+15+17) / 3$
= 16 Story points per Sprint

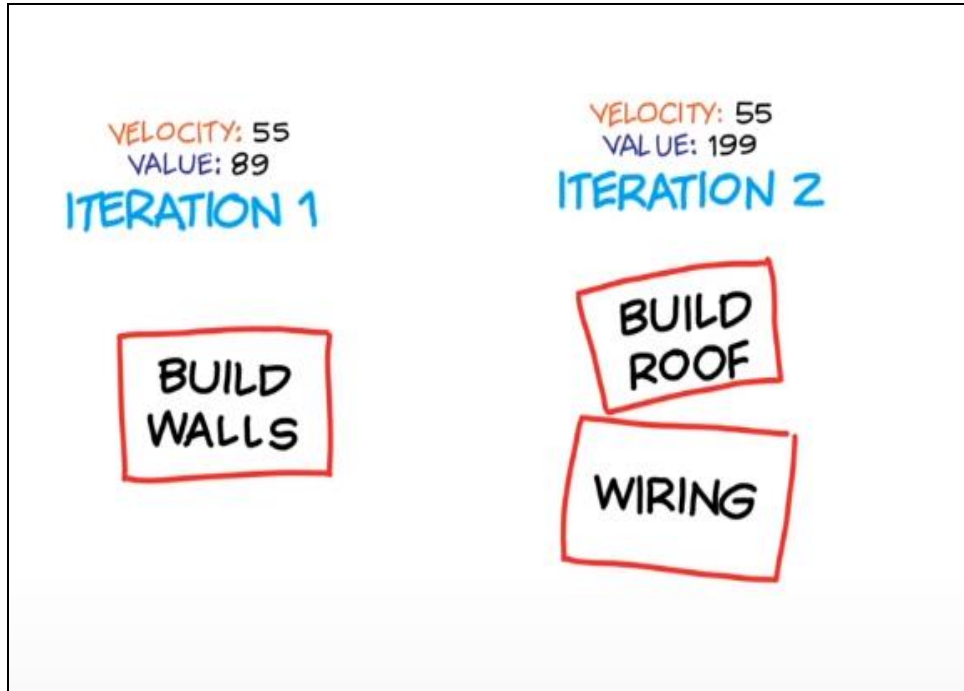
Sprints	Number of Story points Delivered
Sprint-1	16
Sprint-2	15
Sprint-3	17

- Velocity is a rolling average. That means that the velocity may increase or decrease depending on what happens with the team.
- After some iterations the velocity will become stable.

Highest value delivered in early Iterations



Product Backlog

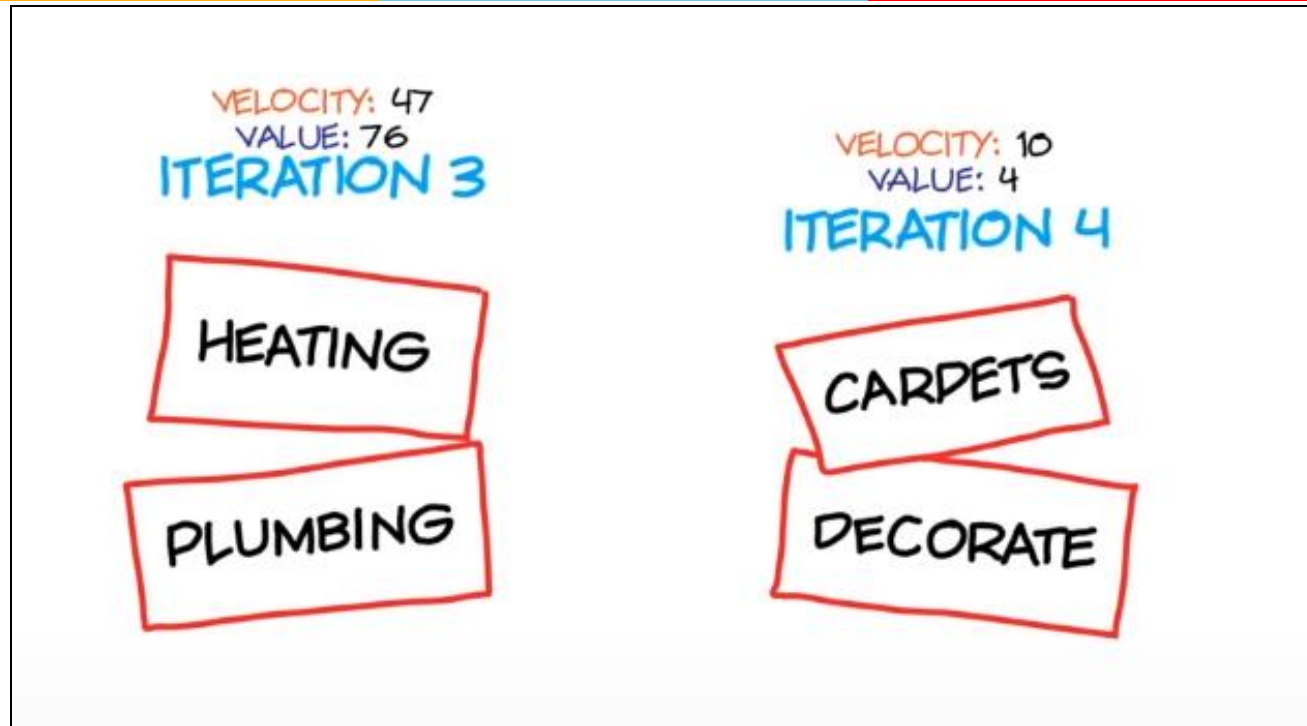


STORY	VALUE	BFTB
BUILD ROOF	34	144 4.2
WIRING	21	55 2.6
BUILD WALLS	55	89
HEATING	13	21 1.6
PLUMBING	34	55
CARPETS	2	2 1
DECORATE	8	2 0.25
SWIMMING POOL	55	1 0.01

This story needs to go first

Suppose, Cost of this iteration-1 is 20000\$;
= $20000/89 \sim 225\$$ Per Value Point.
For Iteration-2 = $225 \times 199 \sim \$44,000$ (Highest value delivered)

Value delivered decreases as iteration progress



Iteration-3 value = $76 * \$225 = \$17,100$; Iteration-4 = $4 * 225 = 900$

- **This is an example**, but In reality, the value will decrease after many iterations, then customer can take a call to continue the project or not.
- Over the period of time, after some iterations the velocity will become stable and value delivered will decrease.

Estimation Exercise (Assume, 2 week Iteration)

innovate

achieve

lead

Iteration 7 (complete)

As a technical specialist
I want to adjust the
turboencapsulator
So that signals will be in
phase

Story points: 5 Value points: 13

As a customer
I want my hydrocoptic
vanes serviced
So that they will last
longer

Story points: 8 Value points: 8

1. Iteration 7 velocity?

$8 + 5 = 13$ story points

2. How long is a story
point?

$80 \text{ hrs} = 13 \text{ story pts}$

$1 \text{ story pt} = (80/13) \text{ hrs}$
 $= 6.15 \text{ hrs}$

3. Which stories in the
next iteration?

4. How long is the rest
of the backlog?

Total sp = $5 + 3 + 5 + 8 + 8 + 21$
 $= 50$

Time = $50 \times 6.15 \text{ hrs}$
 $= 307.5 \text{ hrs}$

Iteration 8 (new)

As a pilfrometer engineer
I want to order grommets
online
So I can stay mobile

Story points: 5 Value points: 5
BFTB: $5/5 = 1$

BFTB: $2/3 = 0.67$
Story points: 3 Value points: 2

BFTB: $2/5 = 0.4$

Story points: 5 Value points: 2

BFTB: $3/8 = 0.38$

Story points: 8 Value points: 3

BFTB: $3/8 = 0.38$

Story points: 8 Value points: 3

BFTB: $3/21 = 0.14$

Story points: 21 Value points: 3

Story Points – Real Examples



Pointing User Stories

Pointing Rubric at iHeartMedia
(two week development sprints)

- 1: Text Change
- 2: Text Change + Small Functionality Change
- 3: One Day of Work for One Developer
- 5: One Week of Work for One Developer
- 8: Two Weeks of Work for One Developer
- 13: Two Weeks of Work for Two Developers

Pointing Rubric at Condé Nast Entertainment
(one week development sprints)

- 1: Text Change
- 2: Text Change + Small Functionality Change
- 3: One Day of Work for One Developer
- 5: One Week of Work for One Developer
- 8: One Week of Work for Two Developers
- 13: Must Be Broken Down Into Smaller Stories



Other Estimation Techniques

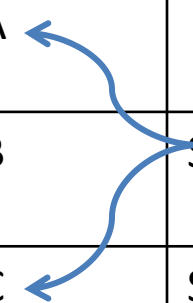
Estimate by Analogy

- Comparing a user story to others
 - “This story is like that story, so its estimate is what that story’s estimate was.”
- Don’t use a single gold standard
 - Triangulate instead
 - Compare the story being estimated to multiple other stories

Triangulation

- Confirm estimates by comparing the story to multiple other stories.
- Group like-sized stories on table or whiteboard

3 points	Story A		
2 points	Story B	Story E	Story F
1point	Story C	Story D	



Ideal Time



- How long something would take:
 - If it's all one person worked on
 - Had no interruptions
 - And everything you need is available.
- The ideal time of a football game is 90 minutes
 - Four 15-minute quarters
 - The elapsed time is much longer (3+ hours)
- It's easier to estimate in ideal time.
- It's too hard to estimate directly in elapsed time.
 - Need to consider all the factors that affect elapsed time at the same time you're estimating

Story Points Vs Ideal Time



- Story points help drive cross-functional behavior
 - Story point estimates do not decay
 - Story points are a pure measure of size
 - Estimating in story points is typically faster
-
- My ideal days cannot be added to your ideal days
 - Ideal days are easier to explain outside the team
 - Ideal days are easier to estimate at first

T-Shirt Sizing, Disaggregation



- Level of Effort (LOE) or T-Shirt Sizing
 - T-shirt size,” “level of effort” (LOE), or “small, medium, large.” (Easy, but lack precision, inability to add up several stories into a meaningful measure.)

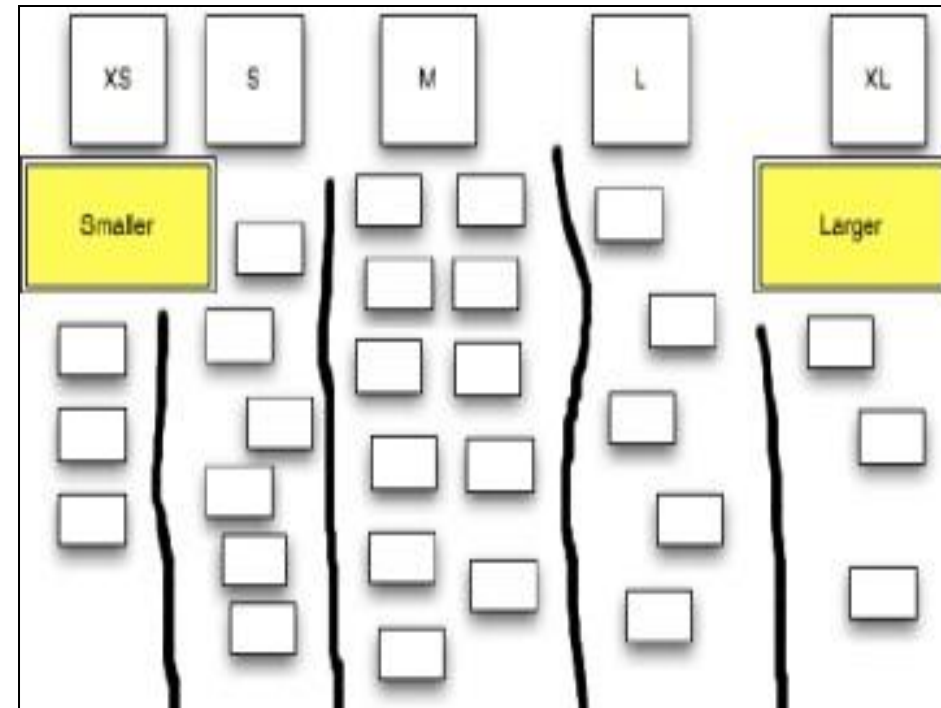
Extra small	Small	Medium	Large	Extra Large	Extra Extra Large
1 point	2 points	3 points	5 points	8 points	13 points

- Disaggregation
 - Breaking a big story into smaller stories ,we know how long the smaller stories take, So, disaggregating to something we know lets us estimate something bigger we don't know

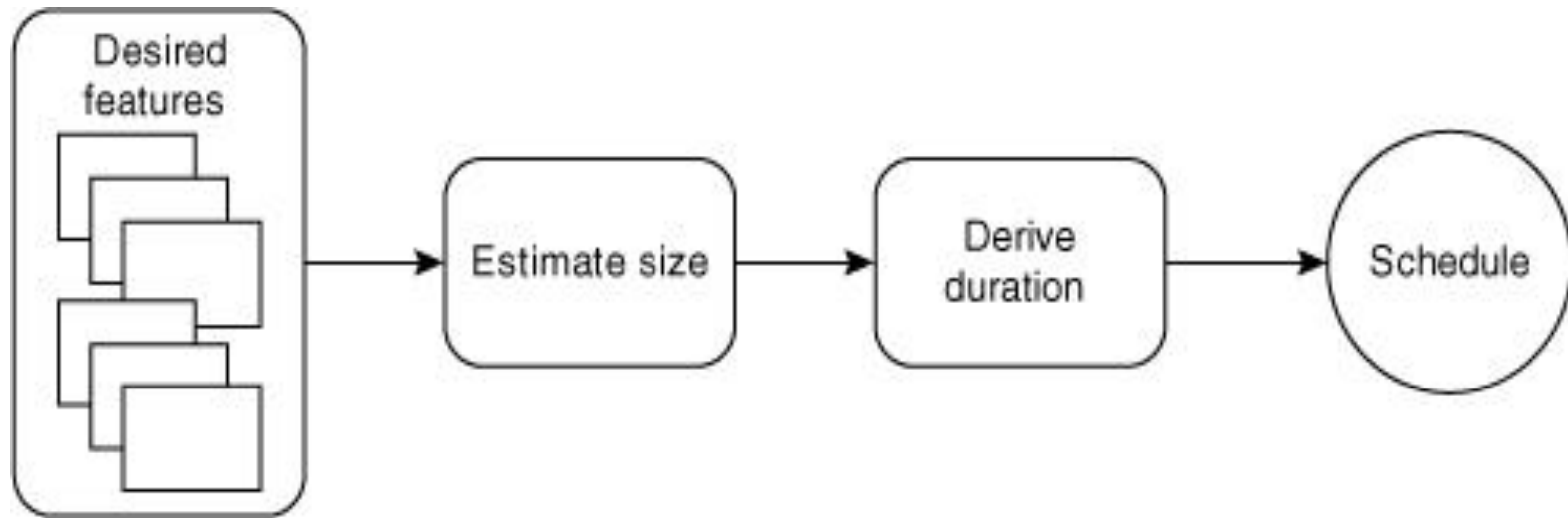
Affinity Grouping



- Team members simply group items together that are like-sized, resulting in configuration similar to the one in figure.



Estimating the duration of a project begins with estimating its size.



- Sum the story-point estimates for all desired features we come up with a total size estimate for the project.
- If we know the team's velocity we can divide size by velocity to arrive at an estimated number of iterations.
- We can turn this duration into a schedule by mapping it onto a calendar.

Source: Agile Estimating and Planning by Mike Cohn
Published by Addison-Wesley Professional, 2005

Re-estimating



- Remembering that story points and ideal days are estimates of the size of a feature helps you know when to re-estimate.
- You should re-estimate only when your opinion of the relative size of one or more stories has changed.
- Do not re-estimate solely because progress is not coming as rapidly as you'd expected.
- Let velocity, the great equalizer, take care of most estimation inaccuracies.

Source: Agile Estimating and Planning by Mike Cohn
Published by Addison-Wesley Professional, 2005



Supplementary Notes on User Stories

Let's start by looking at how we used to gather requirements and some of the challenges that come with trying to write everything down.

The problem with upfront requirement documentation



- **Heavy documentation** as a means of **capturing requirements** has **never really worked** for software projects.
- Here are some other problems teams run into when they rely too heavily on documentation for their software requirements:

THEY CAN'T HANDLE CHANGE

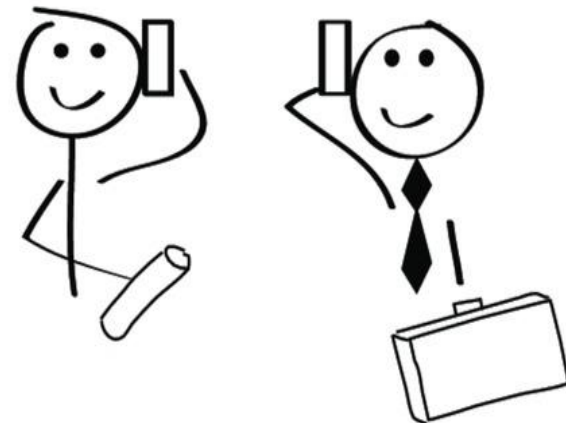
*I KNOW WHAT I SAID,
BUT THAT WAS SIX MONTHS
AGO !!!*



*THEY BUILD ACCORDING TO SPEC ...
INSTEAD OF WHAT THE CUSTOMER WANTS*

*GOOD NEWS !!!
I JUST FIGURED OUT
WHAT OUR CUSTOMERS WANT.*

*SUPER. CAN YOU PARK IT ?
WE JUST FROZE THE SPEC.*

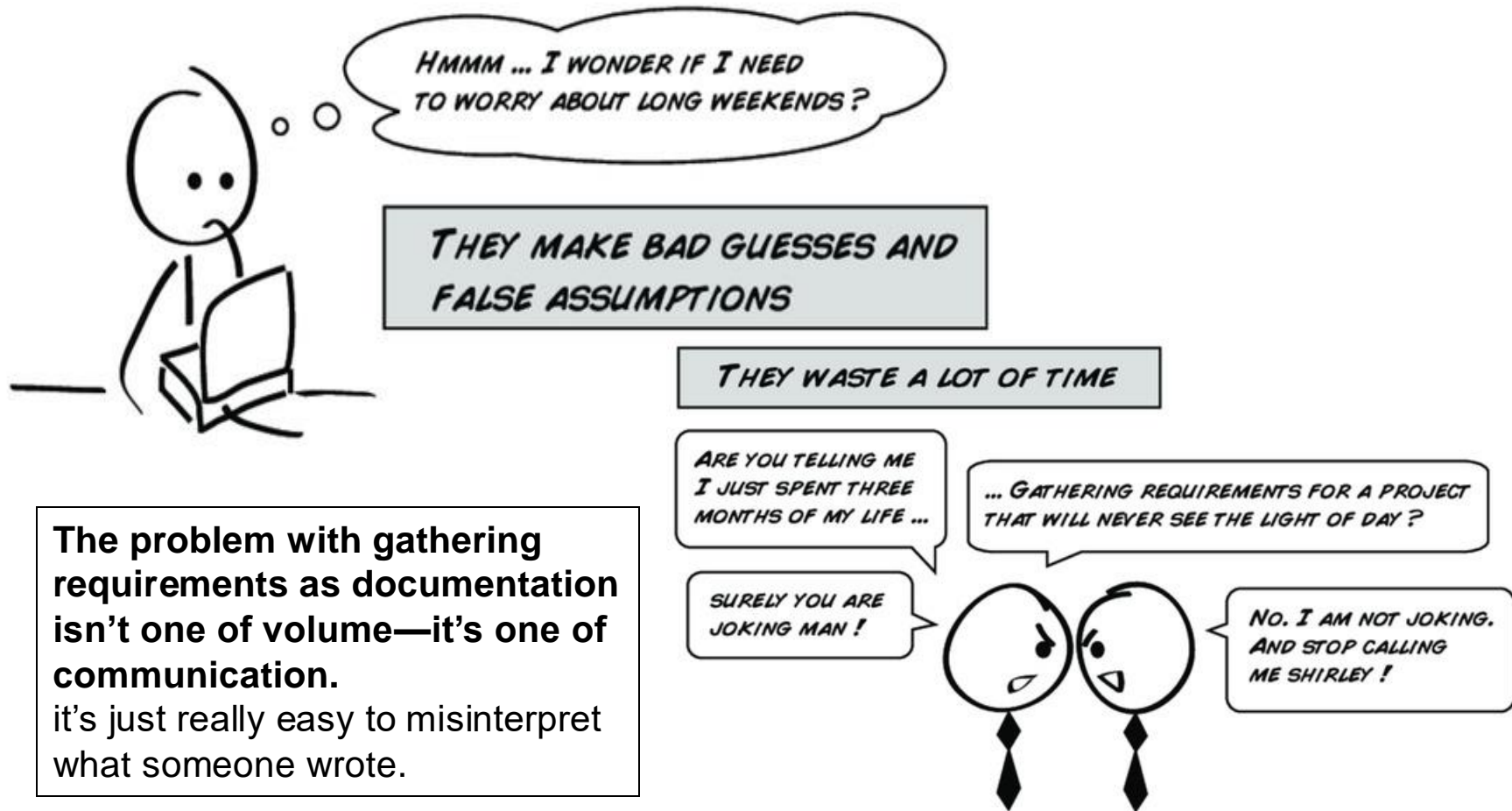


The Agile Samurai by Jonathan Rasmusson Published by Pragmatic Bookshelf, 2010

More Documentation Issues

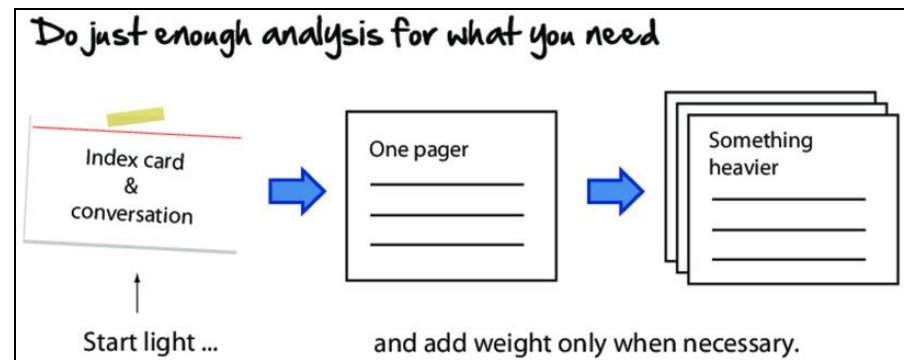
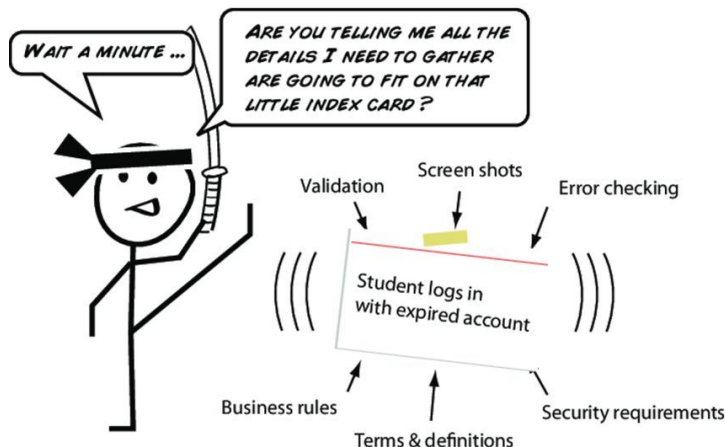


■ ■ ■



What is a User Story?

- Agile **user stories** are **short descriptions of features** our customer would like to one day see in their software
- Why capture just a few key words and not go to town on the full requirement?
 - Think of a user story **as a promise of a conversation**. At some point, we'll do the deep dive and get in there. But we're not going to do it until we are sure we're going to need it.
 - We **defer diving into the low-level details until later**

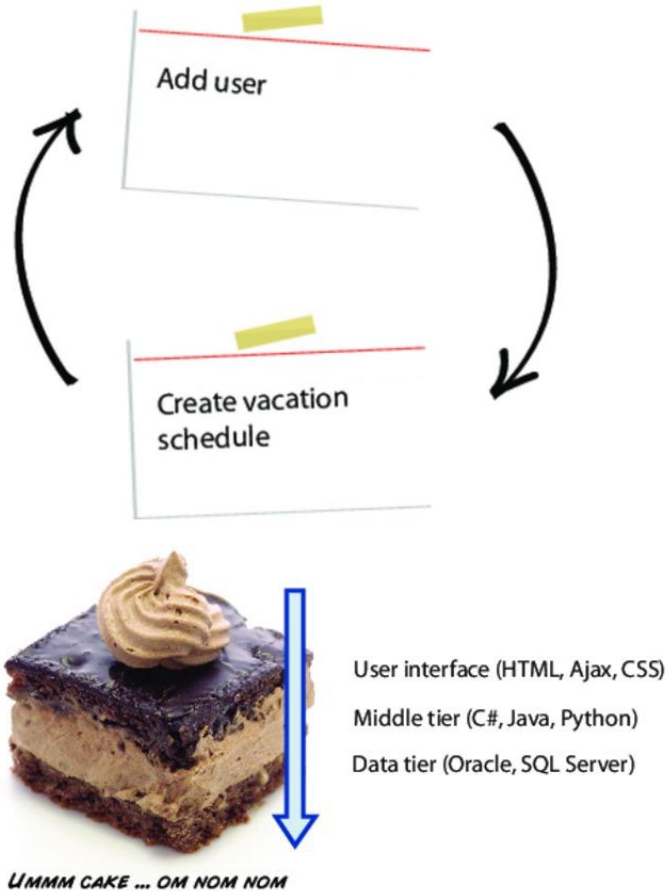


Elements of Good User Stories



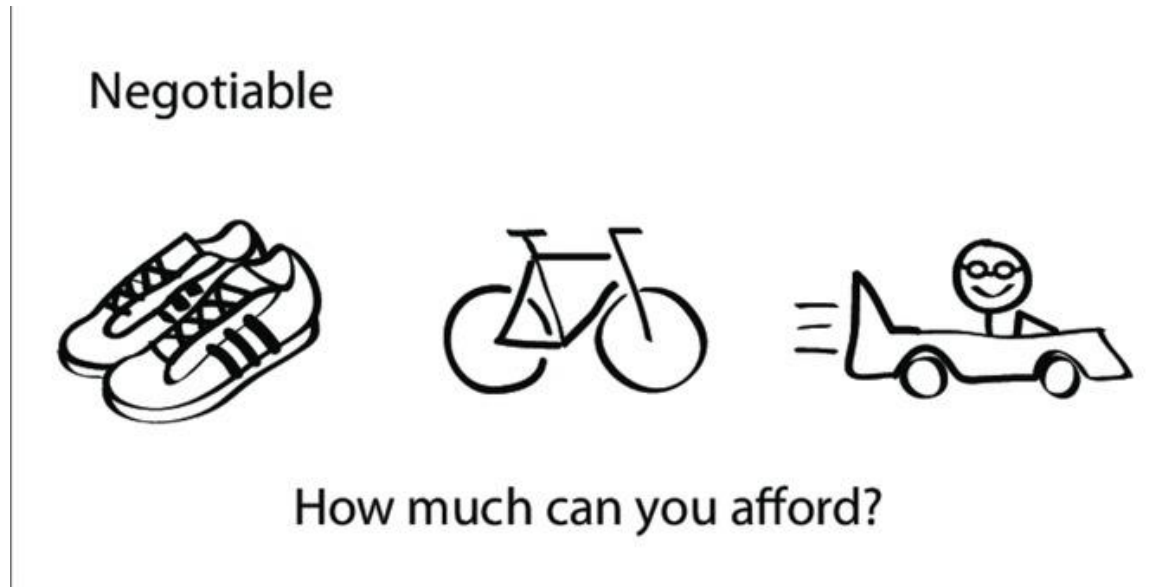
- Bill Wake came up with the **INVEST** acronym for good user story.
- Good user stories also have the following characteristics:
- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimatable
- **S**mall
- **T**estable

Independent – INVEST acronym



- Things change on projects. What was really important last week can suddenly becomes not so important this week.
- If all of our stories are intertwined and dependent on one another, making trade-offs becomes hard.
- **Characteristic of a really good user story is one that goes from end to end—or as we like to call it, “slices the cake.”**
- Slicing our stories from end to end and gathering them by feature enables us to treat the vast majority of **our stories as independent and be flexible on scope when necessary.**

Negotiable – INVEST acronym



- There are always multiple ways to deliver any given story.
- Negotiable stories are nice because they give us the wiggle room we **sometimes need to work within our budgets**

Valuable – INVEST acronym

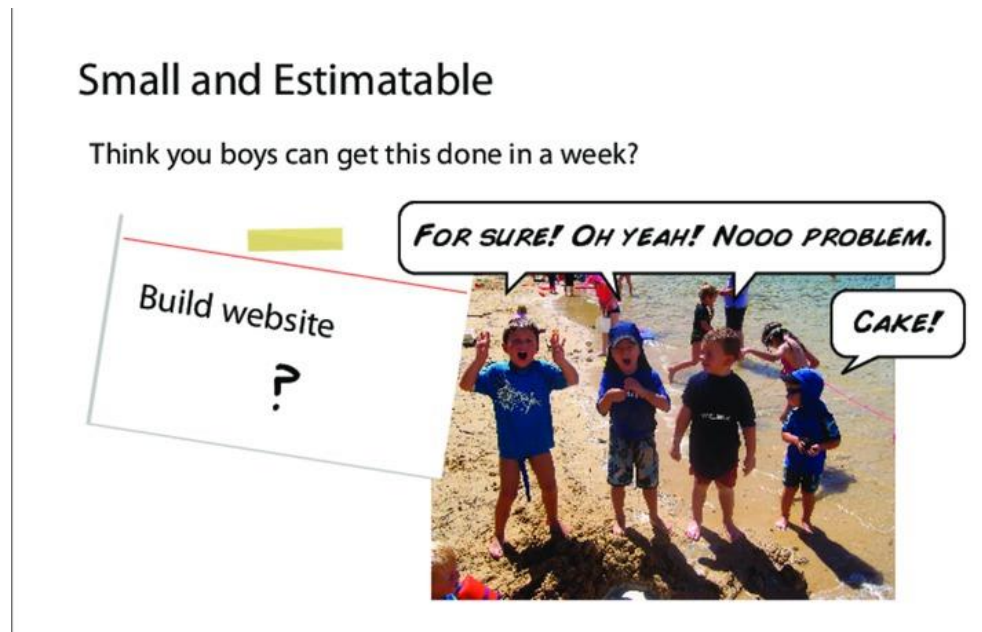


- The important element of a good user story is that it's something of value to our customers.
- What's valuable? Something they would pay for.
- **User stories have to make sense to business.** That's why we always try to write them in simple terms that they understand and stay away from any technical mumbo jumbo.

Estimatable & Small – INVEST acronym



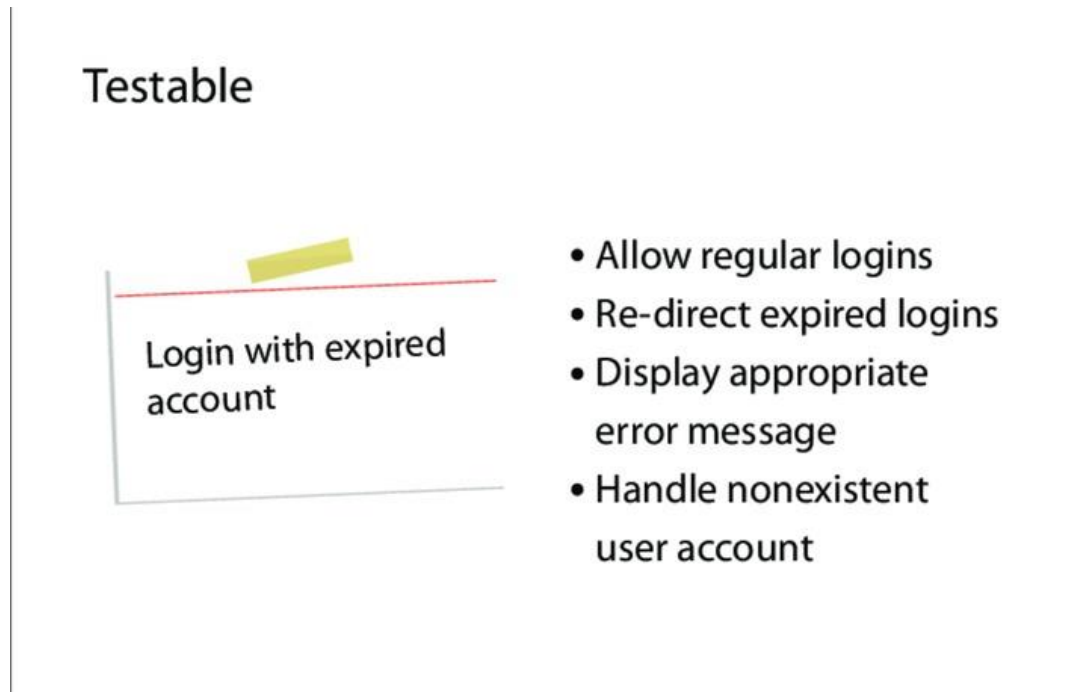
- How do we know a story will fit in within the time frames we have?
 - By **making our stories small** (one to five days)
 - We can ensure they can fit into our one- to two-week iterations, which will enable us to **estimate more confidently**.



Testable – INVEST acronym



- We like our stories to be testable (as opposed to detestable) because **we like to know when something is working.**
- By writing tests against our user stories, we give the development team a stake in the ground and a way of knowing when they are done.



User stories vs Documentation for gathering requirements:



User stories



- Lean, accurate, just-in-time
- Encourage face-to-face communication
- Simplified planning
- Cheap, fast, easy to create
- Never out-of-date
- Based on latest learnings
- Enable real-time feedback
- Avoid false sense of accuracy
- Allow for team-based collaboration and innovation

Specifications & requirement docs



- Heavy, inaccurate, out-of-date
- Encourage guesswork (false assumptions)
- Complex planning
- Expensive, slow, hard to create
- Always out-of-date
- Based on little or no learning
- Disable real-time feedback
- Promote false sense of accuracy
- Discourage open collaboration and innovation

Extracting user stories for an example project,

Big Wave Surf Shop – Website Development

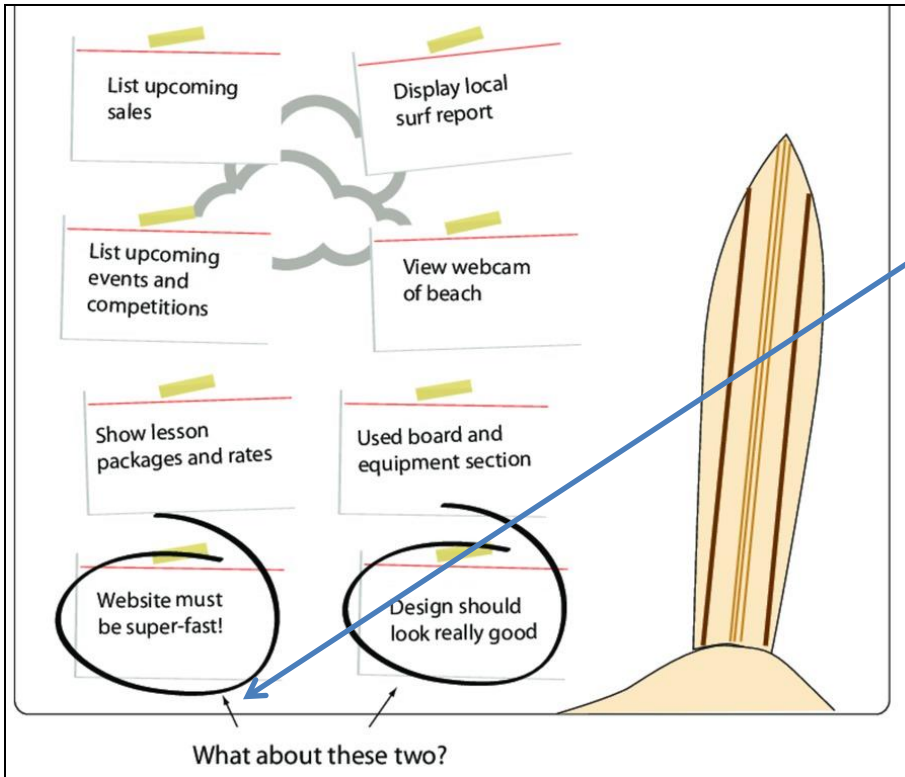
Example: Big Wave Surf Shop

– Website Development



1. I want the website to be a place for the local scene. Somewhere the kids can come and check out upcoming **events—surf competitions, lessons, things like that.**
2. I need a place to **sell merchandise.** Boards, wet suits, clothes, videos, and things like that. Website have to be easy to use and look really good.
3. I've always wanted **a webcam pointing at the beach.** This way, you don't have to come down to check out the conditions.
 - You can just open your laptop, go to the website, and see whether it's worth getting up. This also means the website has to be fast.

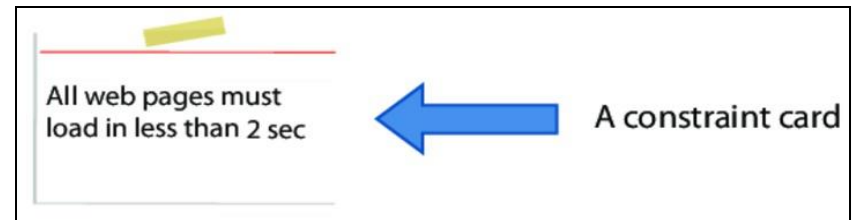
Extracting User Stories



- **Stories like these, we call constraints.**
- They aren't your typical user stories that we can deliver in a week.
- But they are important because they describe characteristics our customers would like to see in their software.

Sometimes, we'll be able to translate these into testable stories. For example, we could rewrite The website must be super-fast! like this:

Constraints are important, Capture them on **different colored cards**. Make sure everyone on the team is aware of them, and test for them periodically while you're developing your software.



The User Story Template



As a <type of user>
I want <some goal>
so that <some reason>.
Value/Benefit



who is this story for
what they want to do
why they want to do it

For example, using the template, some of our stories for Big Wave surf shop might look like this:



As a surfer who likes to sleep,
I want to check local surf conditions via a webcam
so that I don't have to get out of bed if there is no swell.

As a land-locked Canadian hockey player,
I want to sign up for some adrenaline-pumping lessons
so I can feel the thrill of going "over the falls."

As a grommet looking for the latest surf wear,
I want to see all the latest board shorts and designs
so that I can look stylin' for the Sheilas this summer.

The Agile Samurai by Jonathan Rasmusson Published by Pragmatic Bookshelf, 2010

Extracting user stories for an example project,

Big Wave Surf Shop – Website Development

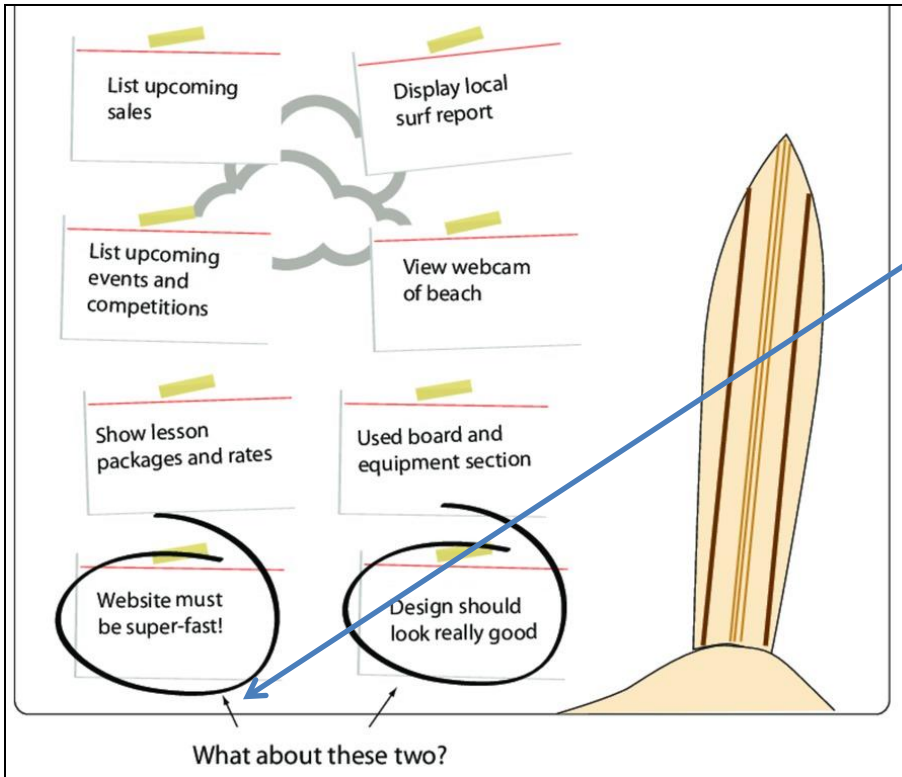
Example: Big Wave Surf Shop

– Website Development



1. I want the website to be a place for the local scene. Somewhere the kids can come and check out upcoming **events—surf competitions, lessons, things like that.**
2. I need a place to **sell merchandise.** Boards, wet suits, clothes, videos, and things like that. Website have to be easy to use and look really good.
3. I've always wanted **a webcam pointing at the beach.** This way, you don't have to come down to check out the conditions.
 - You can just open your laptop, go to the website, and see whether it's worth getting up. This also means the website has to be fast.

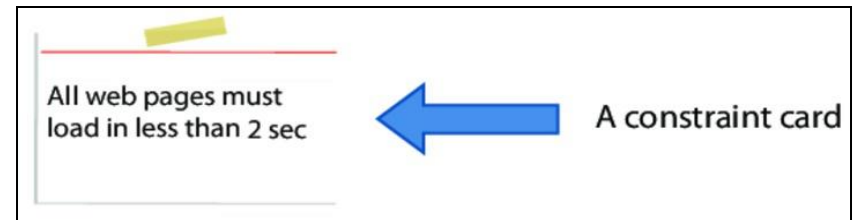
Extracting User Stories



- **Stories like these, we call constraints.**
- They aren't your typical user stories that we can deliver in a week.
- But they are important because they describe characteristics our customers would like to see in their software.

Sometimes, we'll be able to translate these into testable stories. For example, we could rewrite The website must be super-fast! like this:

Constraints are important, Capture them on **different colored cards**. Make sure everyone on the team is aware of them, and test for them periodically while you're developing your software.



The User Story Template



As a <type of user>
I want <some goal>
so that <some reason>.
Value/Benefit



who is this story for
what they want to do
why they want to do it

For example, using the template, some of our stories for Big Wave surf shop might look like this:



As a surfer who likes to sleep,
I want to check local surf conditions via a webcam
so that I don't have to get out of bed if there is no swell.

As a land-locked Canadian hockey player,
I want to sign up for some adrenaline-pumping lessons
so I can feel the thrill of going "over the falls."

As a grommet looking for the latest surf wear,
I want to see all the latest board shorts and designs
so that I can look stylin' for the Sheilas this summer.

The Agile Samurai by Jonathan Rasmusson Published by Pragmatic Bookshelf, 2010