```
<script type="text/javascript">
    window.addEventListener("load", function() {

        revealDiv = document.querySelector("body div.reveal")
        footer = document.getElementById("logo");
        revealDiv.appendChild(footer);
        clogo = document.getElementById("clogo");
        revealDiv.appendChild(clogo);

    } );
</script>
<style>
    #logo {
      position: absolute;
      bottom: 20px;
      right: 50px;
      width: 200px;
    }
    #clogo {
      position: absolute;
      top: 20px;
      left: 20px;
      width: 300px;
    }
</style>
<div id="logo">
Agile Scrum
</div>
<div id="clogo">
<class="imageblock column"><img src="images/logo-gh-consulting.png" alt="Greg
Hutchinson Consulting Inc" width="160" height="140">
</div>
```

Agile/Scrum

Where Theory Meets Practice

Greg Hutchinson

Introduction

Who am I?

- Software Developer since 1982
- 8085 Assembler, C, Smalltalk, Java, Ruby
- Computer aided self driven tractors (1985)
- Consultant for Credit Suisse and UBS banks in Zurich
- Reviewer for eXtreme Programming Explained (1996)
- Director of Agile Center of Excellence
- IT Principal

Scuba Diver

- Red Sea

- Caribbean
- Indian Ocean
- Philippines

Duplicate Bridge Player

And I really like Halloween

Format

- 10 min breaks ~45 mins. Please be back on time.
- The Miro board (Stories, etc.)

Introduce yourself

- Name / Current Job role
- Experience with Agile/scrum
- What are you hoping to get out of this training
- What you like to do in your spare time

High Level Agenda

- Agile Theory
- Scrum Framework
- Wrap up & What's Next

Goals

At the end of this training you will:

- Understand what it means to have an Agile mindset, by understanding its values and principles
- Have a basic understanding of the Scrum process, its roles, artifacts and ceremonies

Goals

And understand the following:

- It’s not about being agile for the sake of being agile. It’s about delighting customers.
- Work in progress (WIP) is a liability towards being agile.
- Agile teams aren’t afraid of REWORK.
- Agile Teams are about delivering business value sooner and more frequently. This is not necessarily the most **efficient** way to deliver.
- Agile is not an absolute term, rather it is a relative term
- Agile isn’t a thing we do, it’s a mindset.
- Agile delivery minimizes Risk.

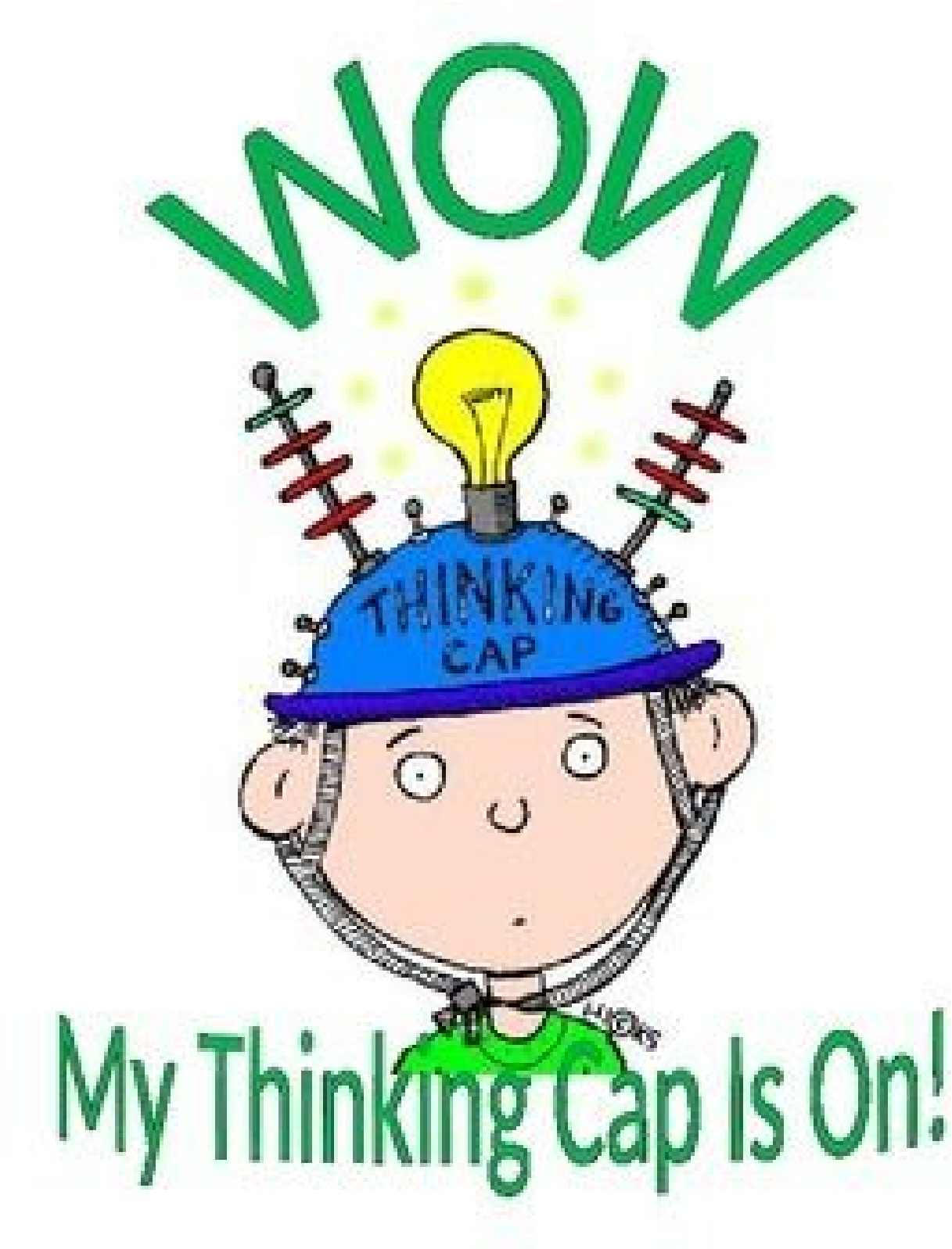
Working agreement

- Be present
- Turn cameras and microphones on.
- Ask questions

Make note of any A-ha’s

Warmup Exercise

- Your task is to identify the rule to the next number in a list of numbers that I will give you.
- You can ask me only one question - "Is n the next number in this list".
- However, you can ask this **any number** of times.
- When you think that you know the rule you can guess - but you only get 1 guess
- If you are right - you win.
- Here is the start of the list:
- 2, 4, 6, 8, ...



Other Comments

- "Agile Thinking" came out of the software development industry.
- I believe that they can be applied in many areas
- We all deliver value to the end-customer either directly or indirectly.

- You may hear me say - "Go Live" or "put it into production". These terms just mean - deliver value
- Team - Any team that is delivering value to a customer

Why do we want to be "Agile"?

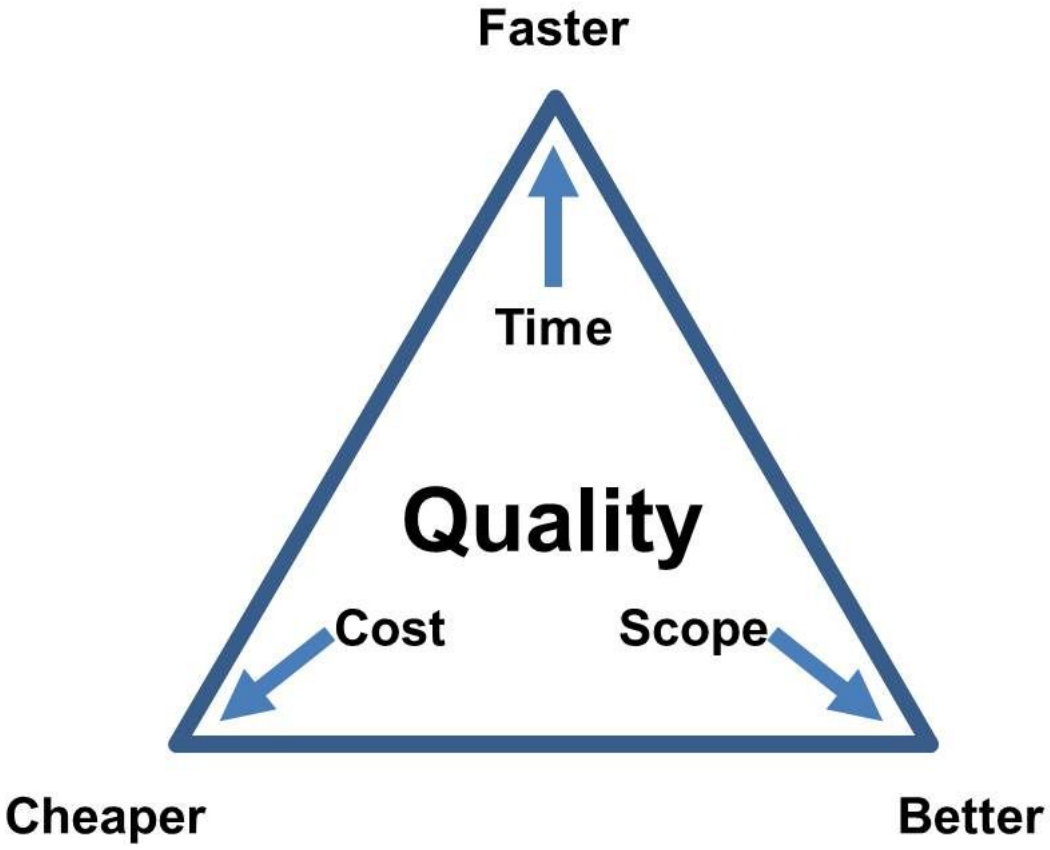
Before we can answer that, we need to answer this:

What was wrong with our traditional approach?

What are characteristics of a failed traditional software project?

- Late, Over-budget
- Large project / large number of people
- Big bang delivery
- Not what was expected/promised
- Risky - Large deliverables are riskier

Measuring the Success of a Traditional Software Project



But what is missing?

How Do We Measure the Success of an Agile Project

[smile-width.jpg] | *smile-width.jpg*

By the width of the customer's smile

What went wrong?

Traditional Approach

```
<iframe src="https://docs.google.com/presentation/d/e/2PACX-
1vS9smGZw51GJGOEOhH1R48qclmPb9gpFZ0MIHxVfpOlErRqdMtoV5cXrrt7NdqderCpluVP_wR2dj4l/embed
?start=false&loop=false&delayms=15000" frameborder="0" width="960" height="569"
allowfullscreen="true" mozallowfullscreen="true"
webkitallowfullscreen="true"></iframe>
```

Next Approach

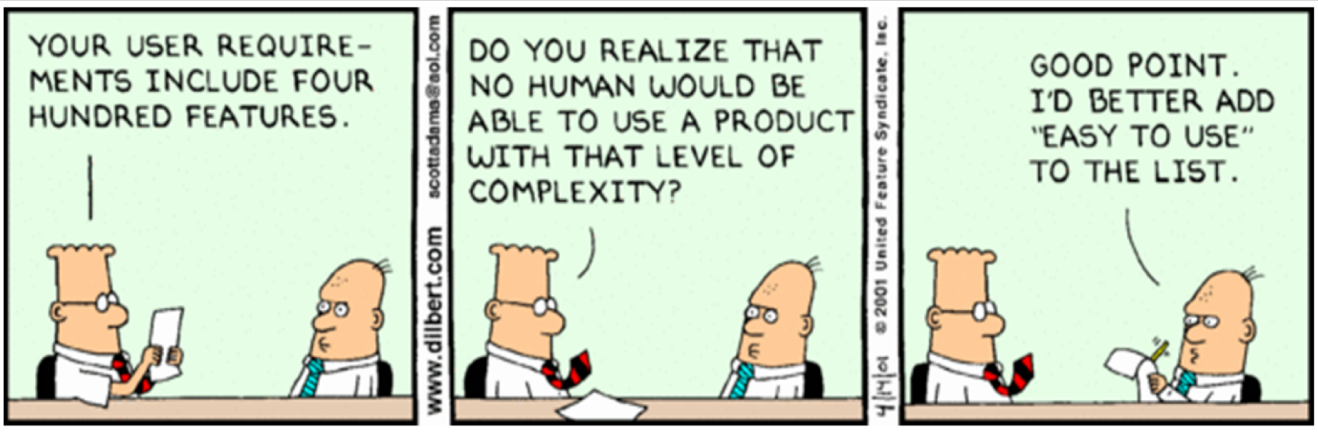
We thought it was a lack of analysis.

- Did more analysis
- Project took even longer
- Exact same results
- **Madness**

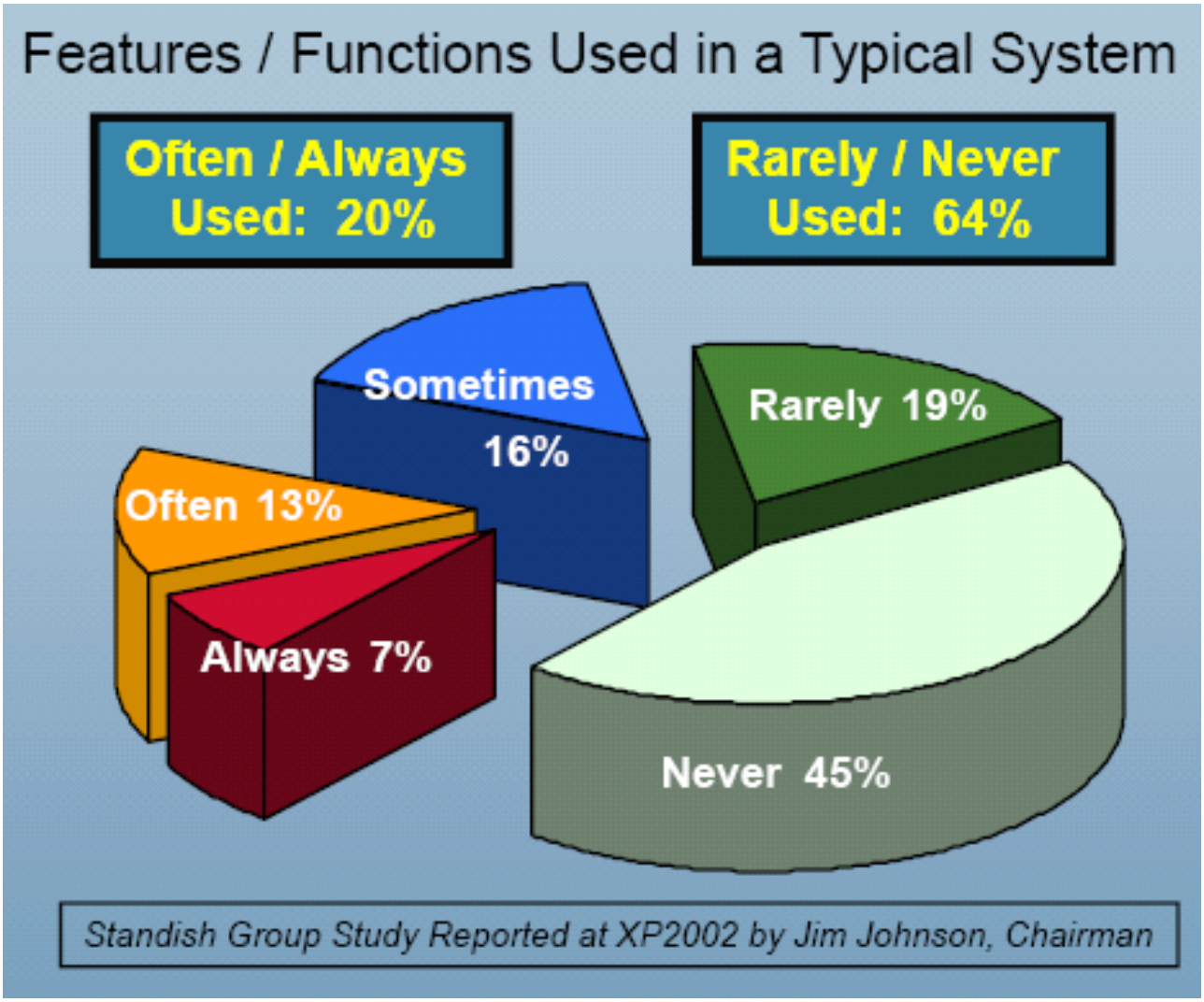
What did we notice?

- We had very unhappy customers.
- Planning and design phases were time consuming and added little value (as a deliverable) by themselves.
- It changed the customers' behaviour.

What did we notice?



Other Consequences

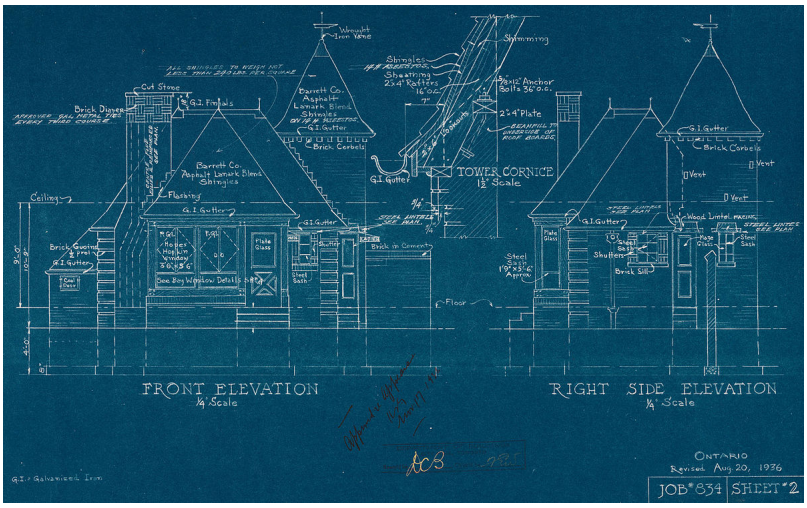


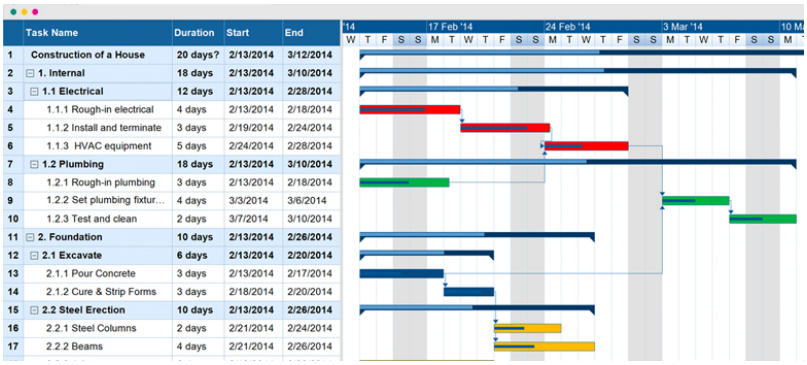
An Agile Approach

```
<iframe src="https://docs.google.com/presentation/d/e/2PACX-
1vRKRCrEvic1xMCkOHTx8t30TRIomLK81gLZ8rFcbAU24X7LG3AsEZ6fVGcnzkm79PIWewbXfYq2q-
8w/embed?start=false&loop=false&delayms=15000" frameborder="0" width="960"
height="569" allowfullscreen="true" mozallowfullscreen="true"
webkitallowfullscreen="true"></iframe>
```

Why have so many traditional software projects failed?

We managed them as if they were other construction projects (A Complicated Problem)





	Task Name	Duration	Start	End
1	Construction of a House	20 days?	2/13/2014	3/12/2014
2	1. Internal	18 days	2/13/2014	3/10/2014
3	1.1 Electrical	12 days	2/13/2014	2/28/2014
4	1.1.1 Rough-in electrical	4 days	2/13/2014	2/18/2014
5	1.1.2 Install and terminate	3 days	2/19/2014	2/24/2014
6	1.1.3 HVAC equipment	5 days	2/24/2014	2/28/2014
7	1.2 Plumbing	18 days	2/13/2014	3/10/2014
8	1.2.1 Rough-in plumbing	3 days	2/13/2014	2/18/2014
9	1.2.2 Set plumbing fixtur...	4 days	3/3/2014	3/6/2014
10	1.2.3 Test and clean	2 days	3/7/2014	3/10/2014
11	2. Foundation	10 days	2/13/2014	2/26/2014
12	2.1 Excavate	6 days	2/13/2014	2/20/2014
13	2.1.1 Pour Concrete	3 days	2/13/2014	2/17/2014
14	2.1.2 Cure & Strip Forms	3 days	2/18/2014	2/20/2014
15	2.2 Steel Erection	10 days	2/13/2014	2/26/2014
16	2.2.1 Steel Columns	2 days	2/21/2014	2/24/2014
17	2.2.2 Beams	4 days	2/21/2014	2/26/2014

Why do we want to be "Agile"?

Because it is really cool!!!

Why do we want to be "Agile"?

It’s all about "Delighting Customers"

Complicated Vs. Complex

The Cynefin Framework

It is a sense making framework to help:

- Decide what type of problem you are trying to solve
- Decide which approach you should take to solve the problem.

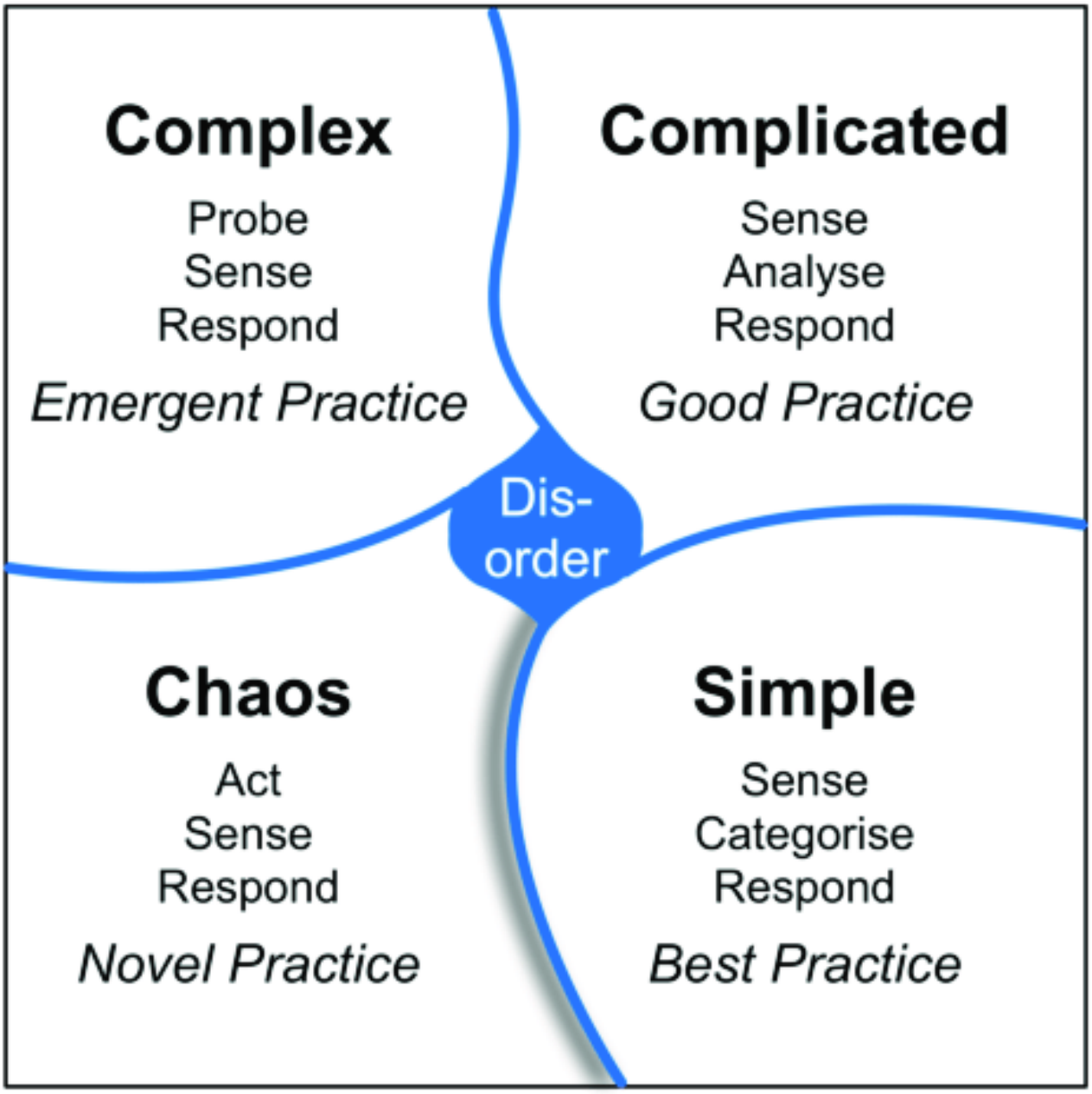
The Cynefin Framework

▶ https://www.youtube.com/watch?v=N7oz366X0-8 *(YouTube video)*

For the full video go to: https://www.youtube.com/watch?v=N7oz366X0-8

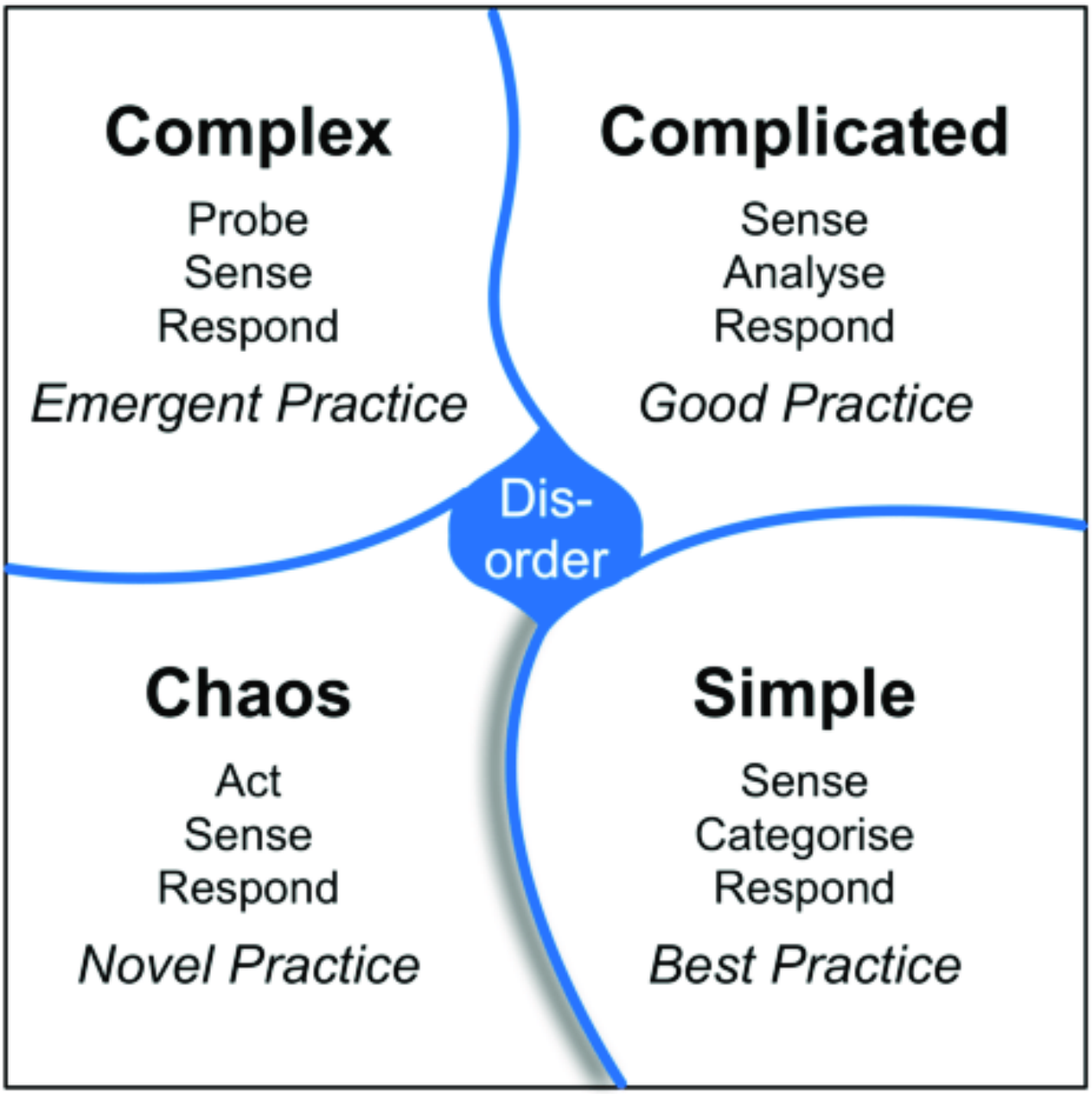
Ordered Systems

Simple and Complicated



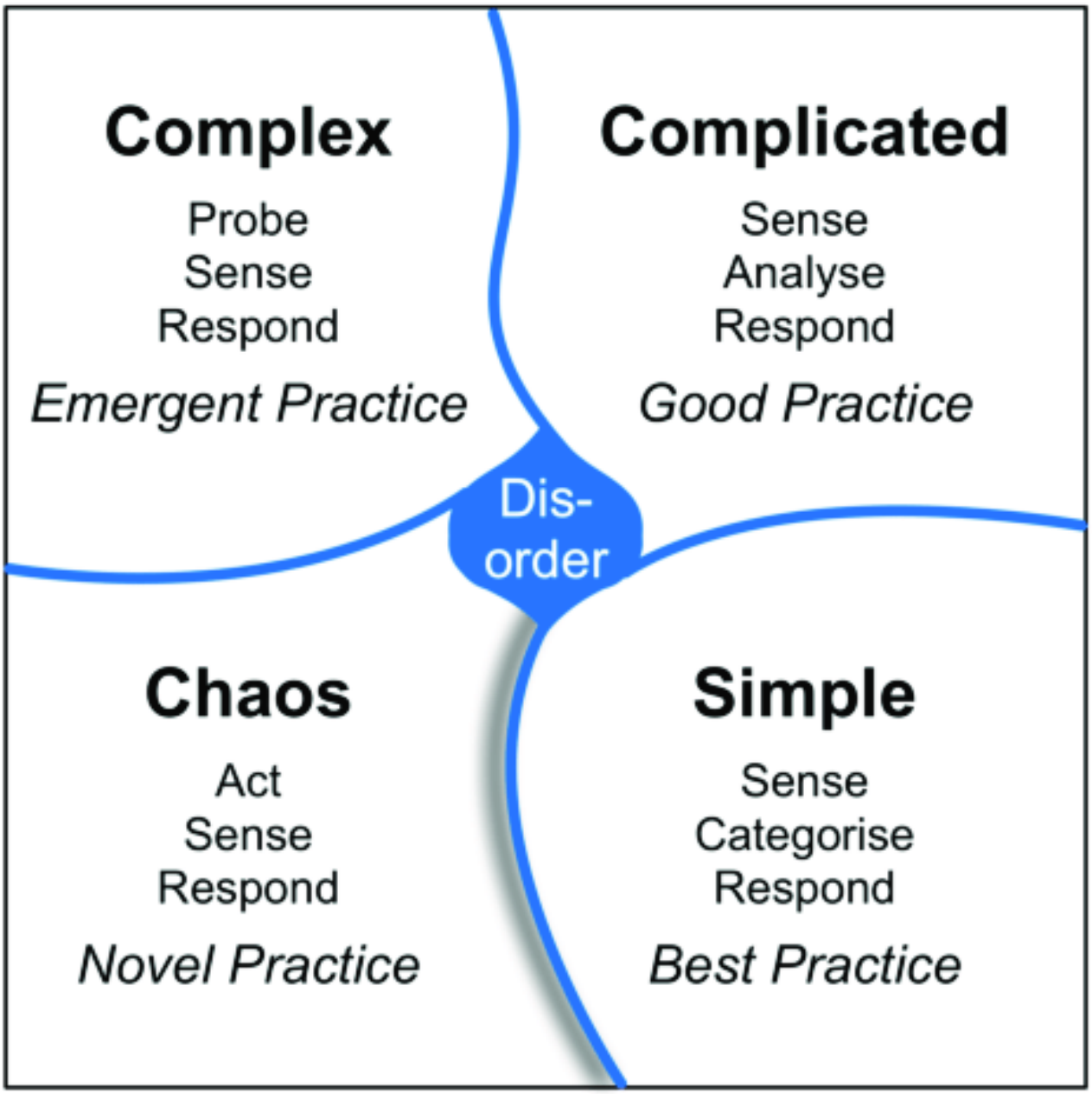
Simple Domain

- Relationship exists between cause and effect
- Observable in advance
- Obvious to everyone
- Best practice



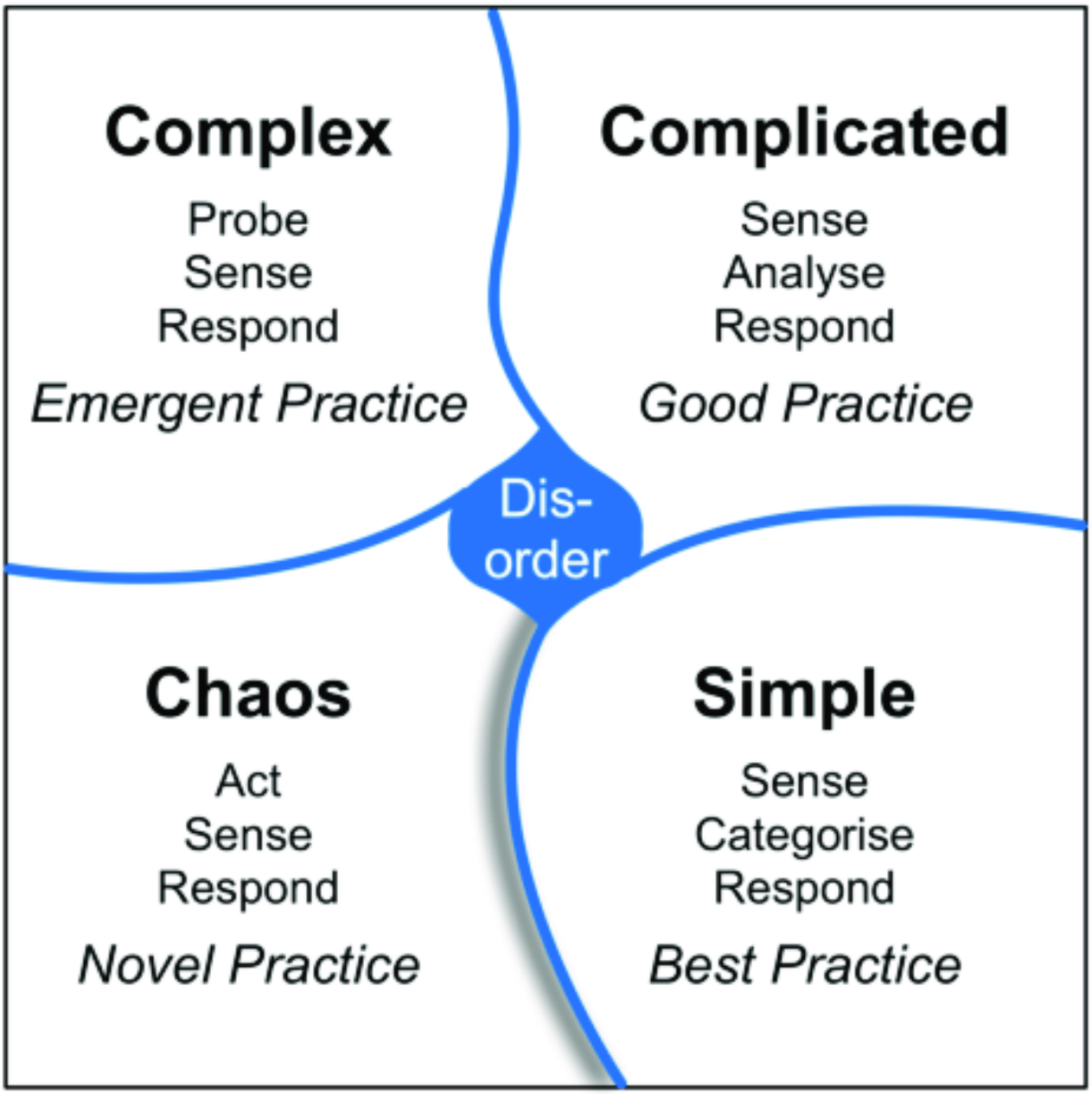
Complicated Domain

- Relationship exists between cause and effect
- Observable in advance
- However, needs Analysis or expertise
- Good practice



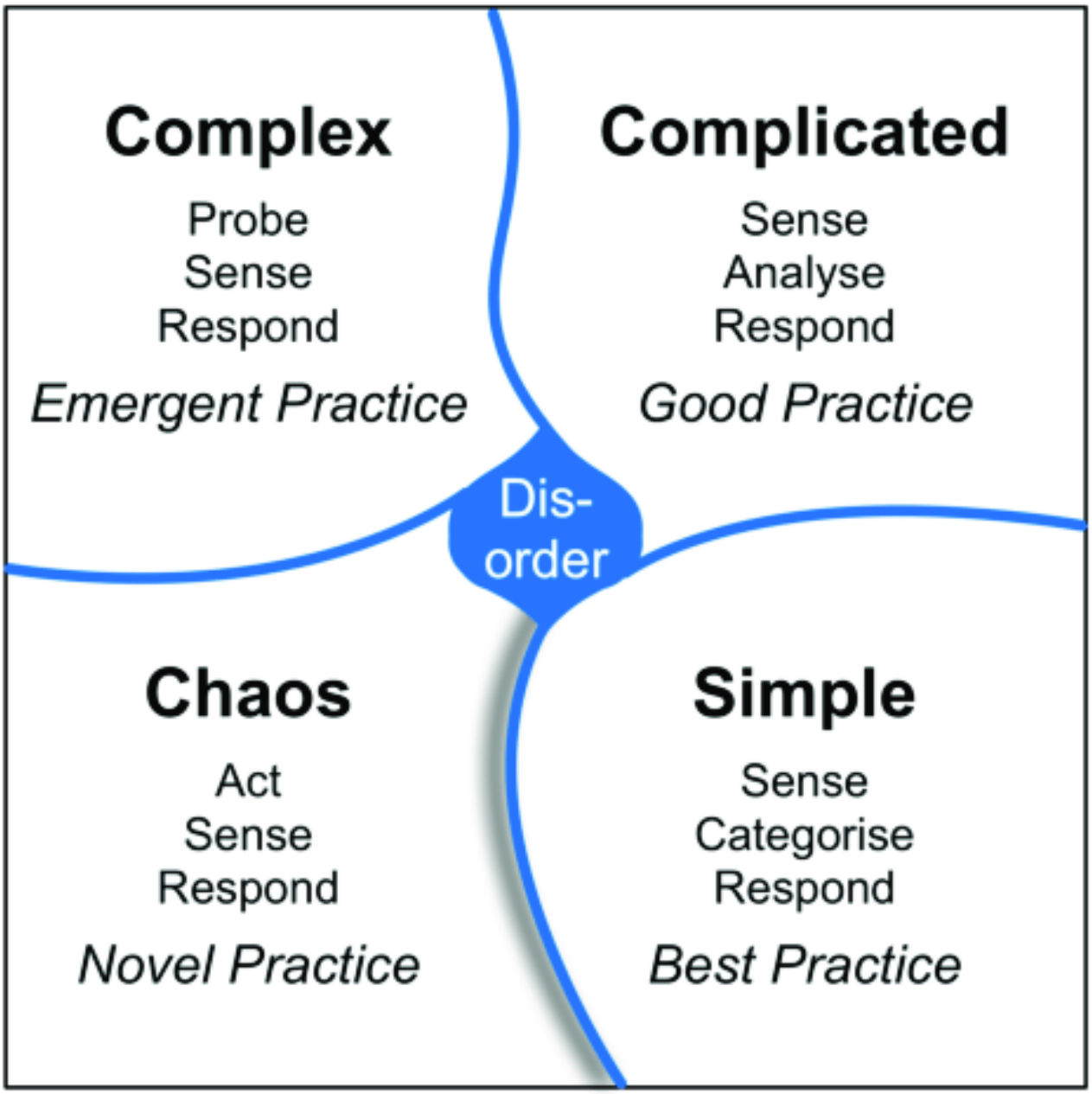
Unordered Domains

Complex and Chaotic



Complex Domain

- Relationship exists between cause and effect
- Only observable in hindsight
- Don’t use Fail-safe design rather safe-fail experiments
- Emergent practice



Distinguishing characteristics of these two problem domains

- Complicated problems - Cause and Effect is predictable in advance
- Complex problems - There is a relationship between cause and effect but it is only recognizable in hindsight

Different approach to solve these problems

- Complicated problems - Do some analysis, make a plan, execute the plan
- Complex problems - Have an approach, if it works continue to do it, if it doesn't, change something

Concrete Examples

Raising a child

Rocket to the Moon

Which is Complex? Complicated?

Expertise

Expertise can contribute but is neither necessary nor sufficient to assure success

High levels of expertise in a variety of fields are necessary for success

Formulas

Formulas have limited application

Formulas are critical and necessary

Experience

Raising one child provides experience but no assurance of success with the next

Sending one rocket increases assurance that the next will be OK

Relationships

Every child is unique and must be understood as an individual - relationships are important.

Rockets are similar in critical ways.

Outcome

Uncertainty of outcome remains

There is a high degree of certainty of outcome

Triangle Experiment

What kind of Problem did we just solve?

Complex? or Complicated?

And if I changed it?

Complex? or Complicated?

When to use which approach?

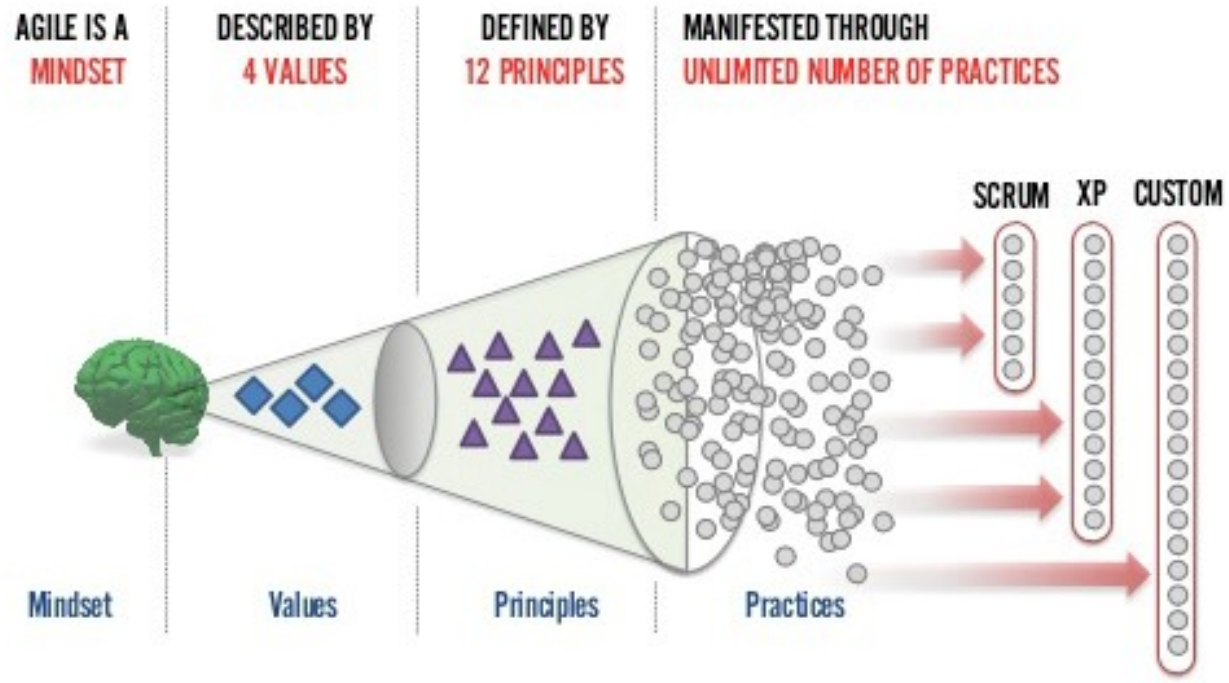
Relationships

Agile	Waterfall
Problem is Complex	Problem is Complicated

Problem is Complicated	Change/Rework is expensive
Solving a part of the problem will affect the rest of the solution	Something similar has been done before
High chance of target changing	Target is somewhat fixed

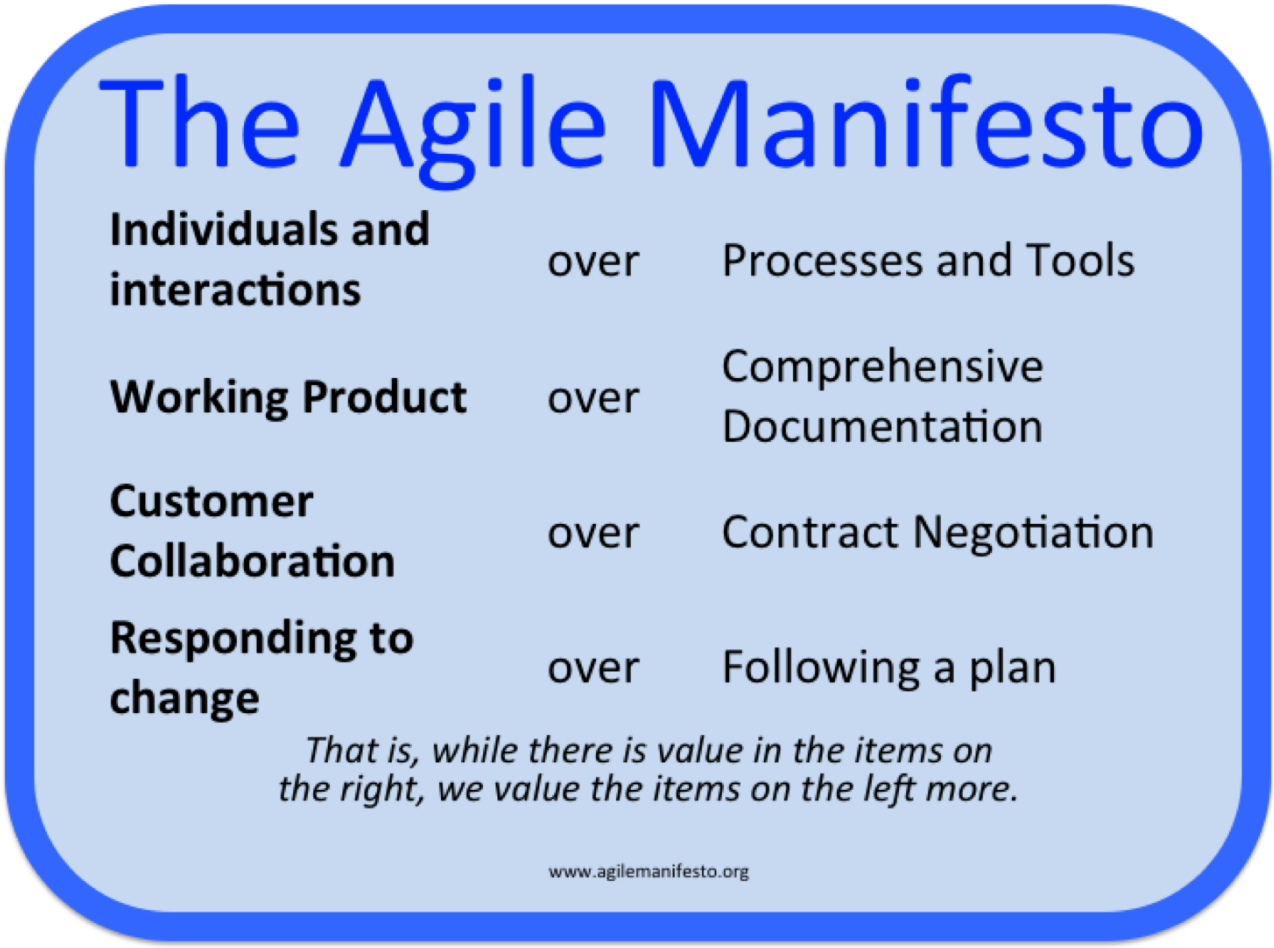
Agile Mindset

Mindset, Values, Principles, Practices



- Values - beliefs that govern the behavior of a person
- Principles - support and satisfy Values
- Practices, Tools & Processes - the actions we take

Agile Manifesto - Values

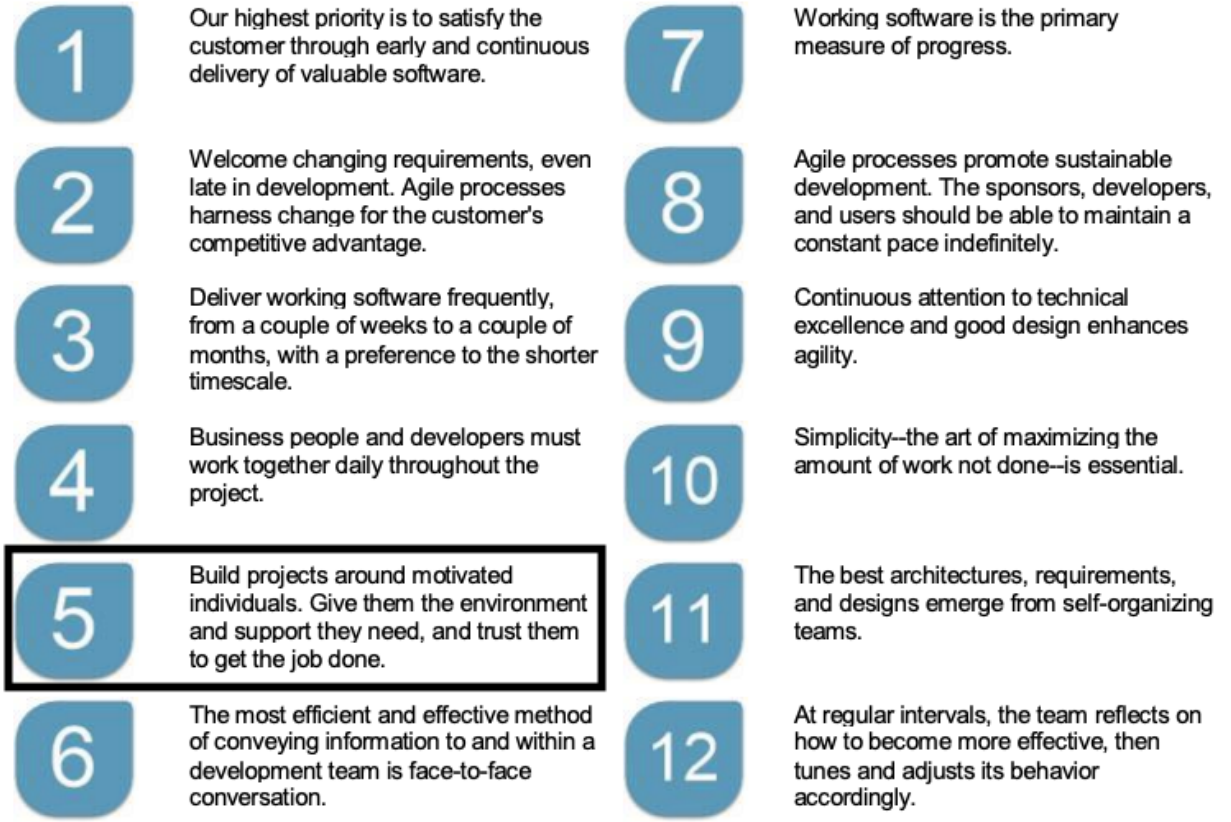


Agile Principles

Agile Principles

Agile Principles

Agile Principles

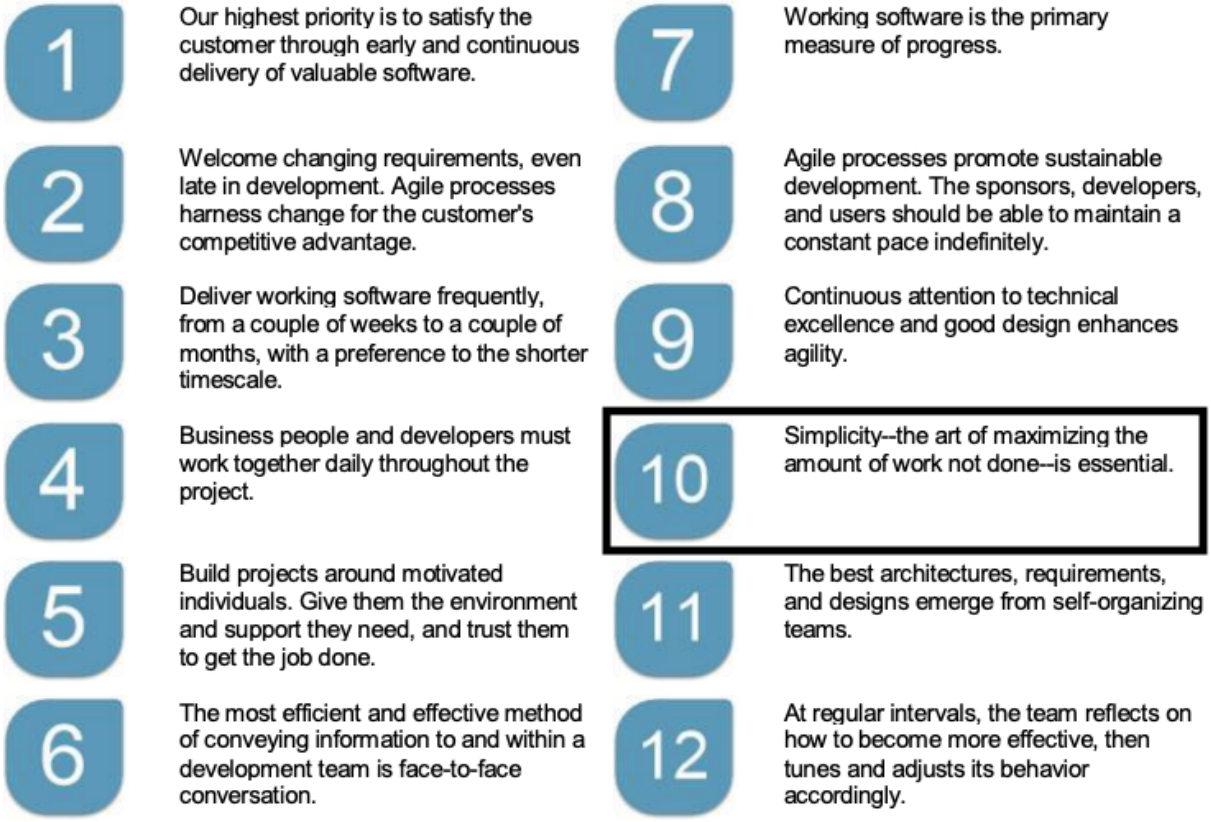


```
 
 
 
```

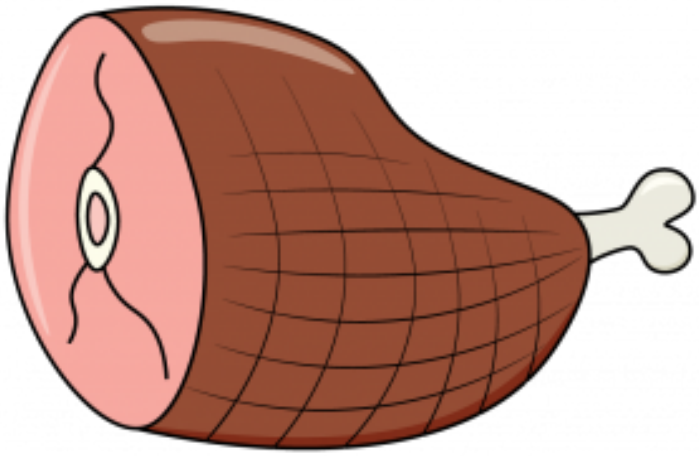
Agile Principles

- Automation is Key (Unit Test Cases)
- Software as an Asset Mentality
- Refactoring (Involving Conversations with the PO)
- Code Reviews (Pull requests)
- Continuous Integration (with servers)
- Distributed Version Control Systems (DVCS - Git)

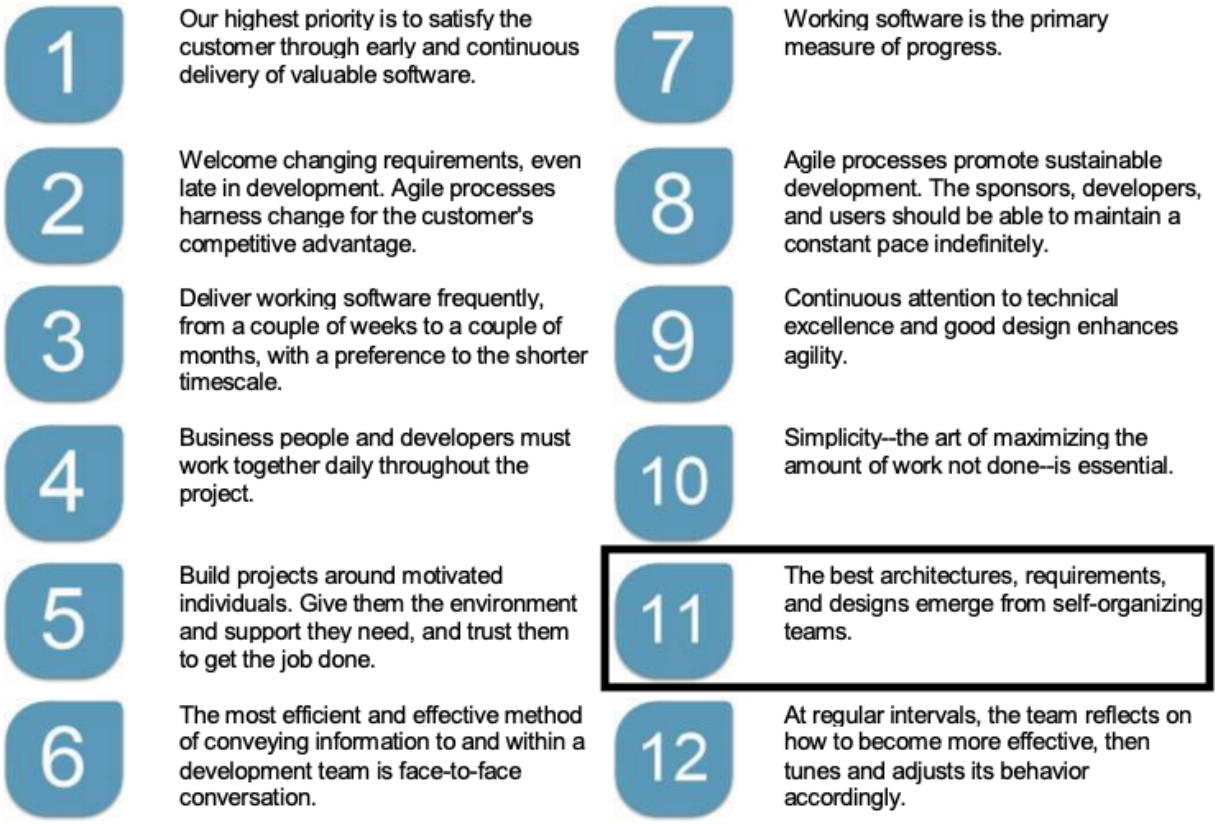
Agile Principles




```
 
 
 
```



Agile Principles



- Emergent Design (Stays real)
- Fits with the team is responsible for all statement

Agile Principles


```
 
 
 
```

Agile Principles - Experiment

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

```
 
 
 
```

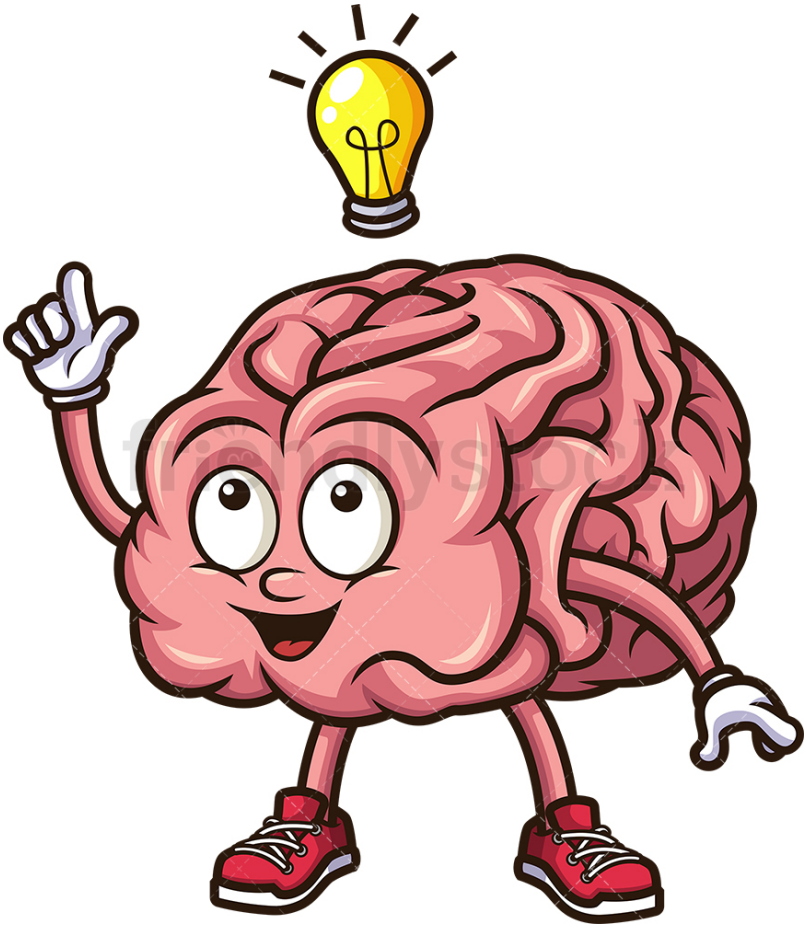
Agile Principles - Failure


```
 
 
 
```

Agile Principles - Learn


1
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4
Business people and developers must work together daily throughout the project.
5
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7
Working software is the primary measure of progress.
8
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9
Continuous attention to technical excellence and good design enhances agility.
10
Simplicity--the art of maximizing the amount of work not done--is essential.
11
The best architectures, requirements, and designs emerge from self-organizing teams.
12
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

```
 
 
 
```



Values

Agile		Waterfall
People come first		Use standardized processes
Respond to change		Get it right the first time
Early and frequent value delivery		On time on budget
Customer collaboration		Make early commitments

Principles

Agile		Waterfall
Frequent delivery		Plan the work
Simplicity		Deliver it all
Team is responsible		Single point of responsibility
Progress = value delivered		Measure on % complete
Effective over efficient		Efficient first
Have Slack		Maximize utilization

Practices Tools Processes

Agile		Waterfall
Sprint, Standup, Demo		Requirement documents
Refinement, Burndown		Weekly status updates
Product Owner, Scrum Master		Change requests, Approval Gates
Retrospective, Backlog		Early Sign Off, Test plans

Why do we want to be Agile?

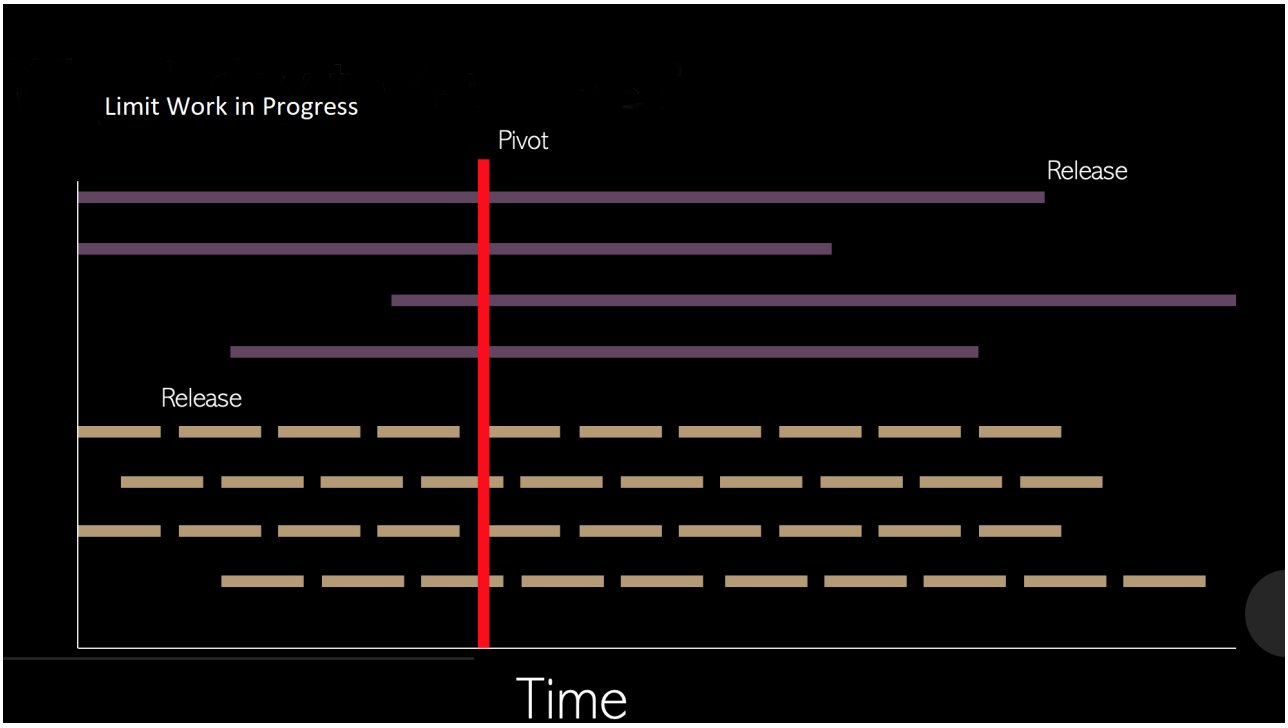
> The pace of change has never been this fast, yet it will never be this slow again

We can’t predict the future

What were they missing?


BLOCKBUSTER
VIDEO
®

How do we adapt quickly?



So … agile is not

- a specific way of delivering solutions
- a framework or process
- a methodology
- It's a way of **being**

Best Project Ever

What do we always notice?

- Common themes
- Team, Empowerment
- Fun
- Clear objectives
- Success
- Challenging/Learning
- More often on the left side of the Manifesto

Requirements vs Design

Requirement

Is **what** is required (Not the how)

Design

Is **how** a requirement will be implemented

Easy, right ???

Building the I-Car

The car must be able to transport at least 4 people

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Requirement</p>
</details>
```

Building the I-Car

The car must have a CD-player

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Design</p>
</details>
```

Building the I-Car

The car must have an alarm

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Design</p>
</details>
```

Building the I-Car

The car must have brakes

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Design</p>
</details>
```

Building the I-Car

The car must have four wheels

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Design</p>
</details>
```

Building the I-Car

The car must not be longer than 3 meters

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Design</p>
</details>
```

Building the I-Car

The car needs to be able to park in a small parking spot (3 x 2 meters)

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Requirement</p>
</details>
```

Building the I-Car

The car must be able to travel at least 500 km without having to stop (such as for fuel)

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Requirement</p>
</details>
```

Building the I-Car

The car must be green

```
<details>
  <summary>Requirement or Design?</summary>
  <p>Depends on Context</p>
  <p>Green - the color or ...?</p>
  <p>Green - Environmentally Friendly</p>
</details>
```

What might this new I-Car look like?

Why is this important?

- Team doesn't understand the why of the problem
- Stifles creativity
- Missed opportunities for better solutions

Summer Meadow

Scrum Overview

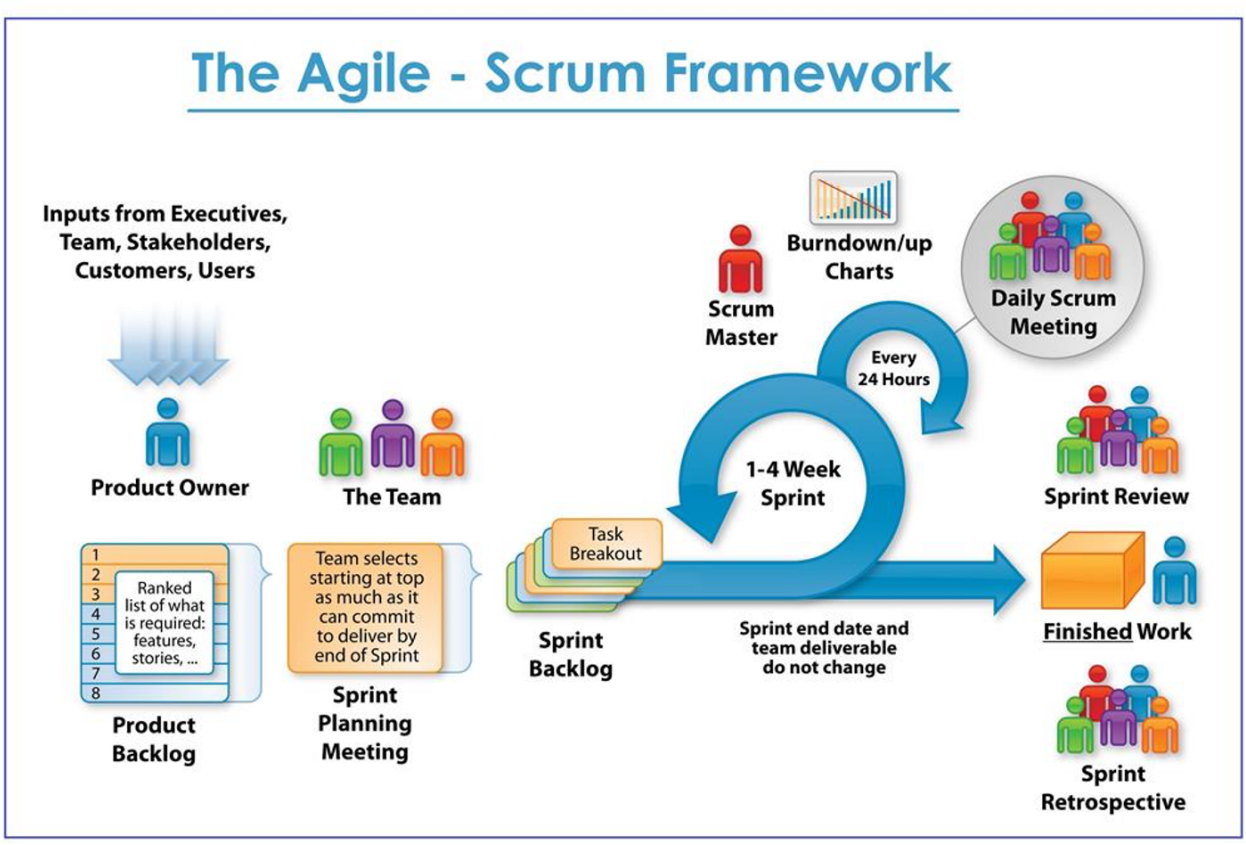
▶ https://www.youtube.com/watch?v=TRcReyRYIMg *(YouTube video)*

Scrum Theory

- Scrum is founded on empirical process control theory
- Scrum uses an iterative, incremental approach to optimize predictability and control risk.
- Empirical process control is upheld by 3 pillars:
- Transparency – the process must be visible to all
- Inspection – Scrum artifacts must be frequently inspected as well as the progress towards the Sprint goal
- Adaptation – if the process is deviating outside of acceptable limits, and the product will be unacceptable, the process must be adjusted.

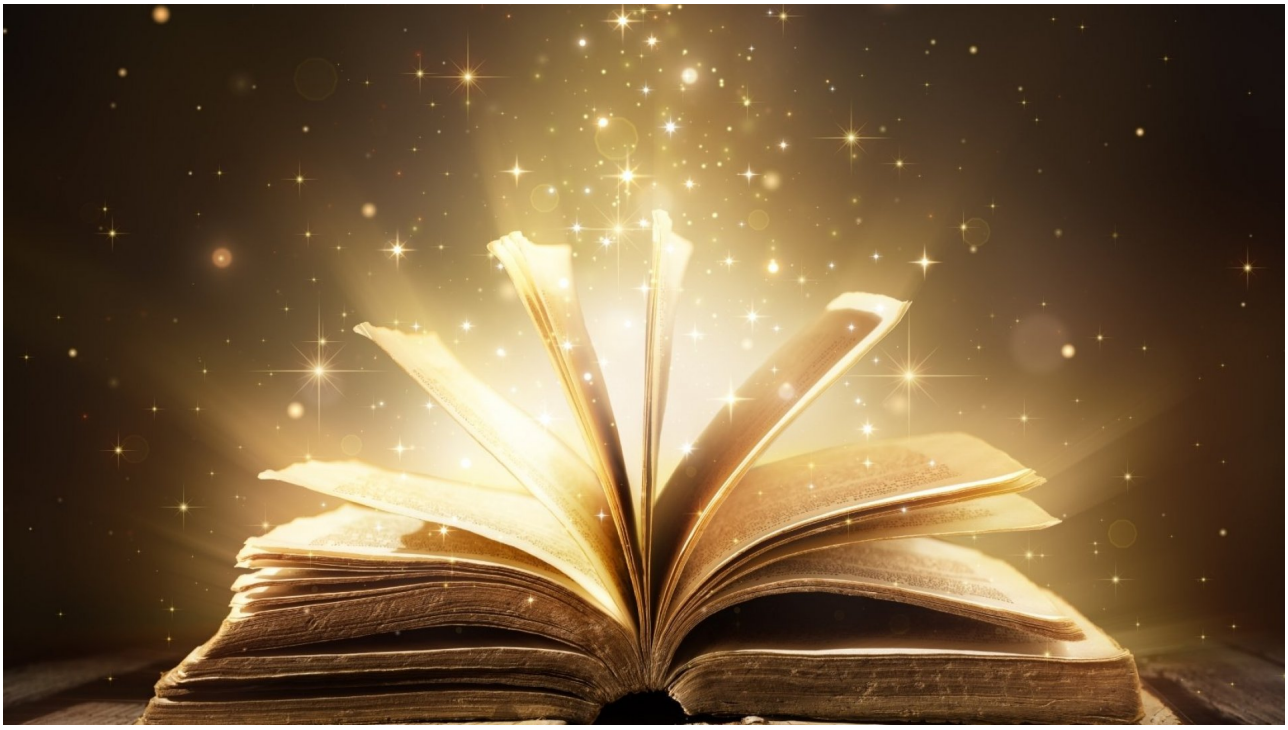
Scrum is Easy

(and yet hard)

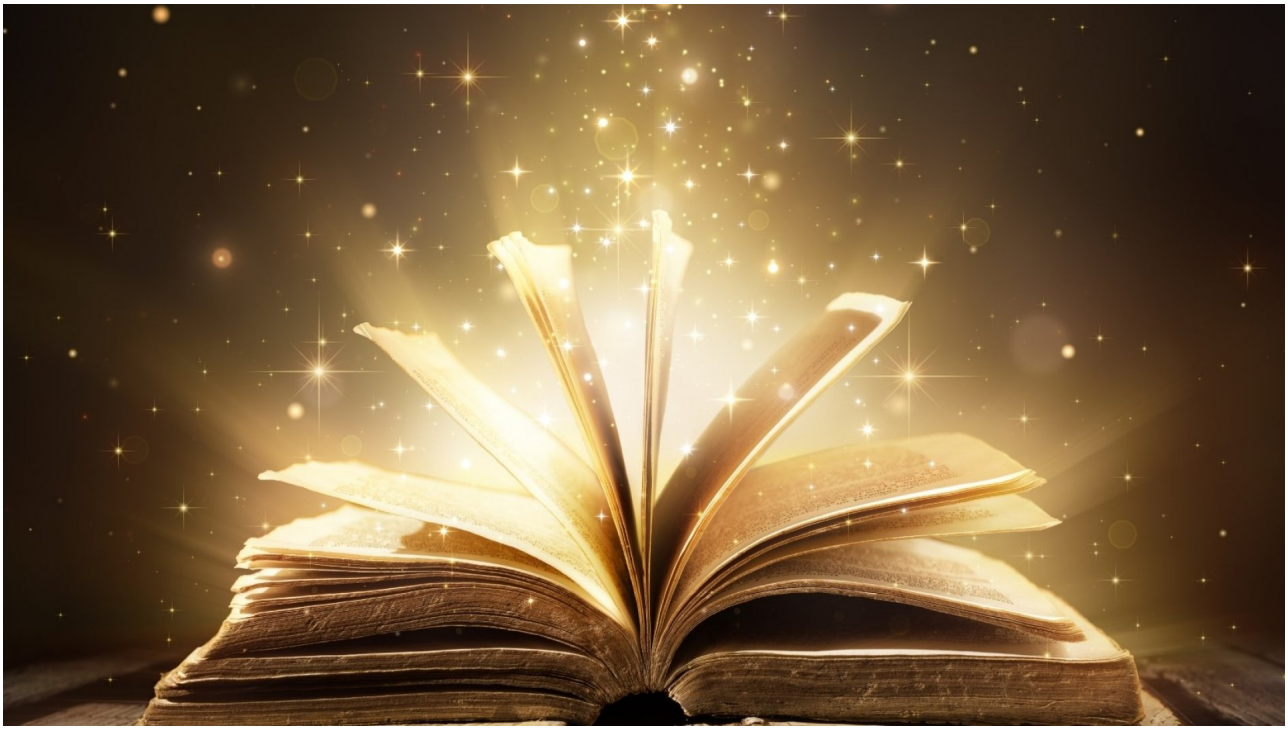


Other Examples???

User Story



Definition

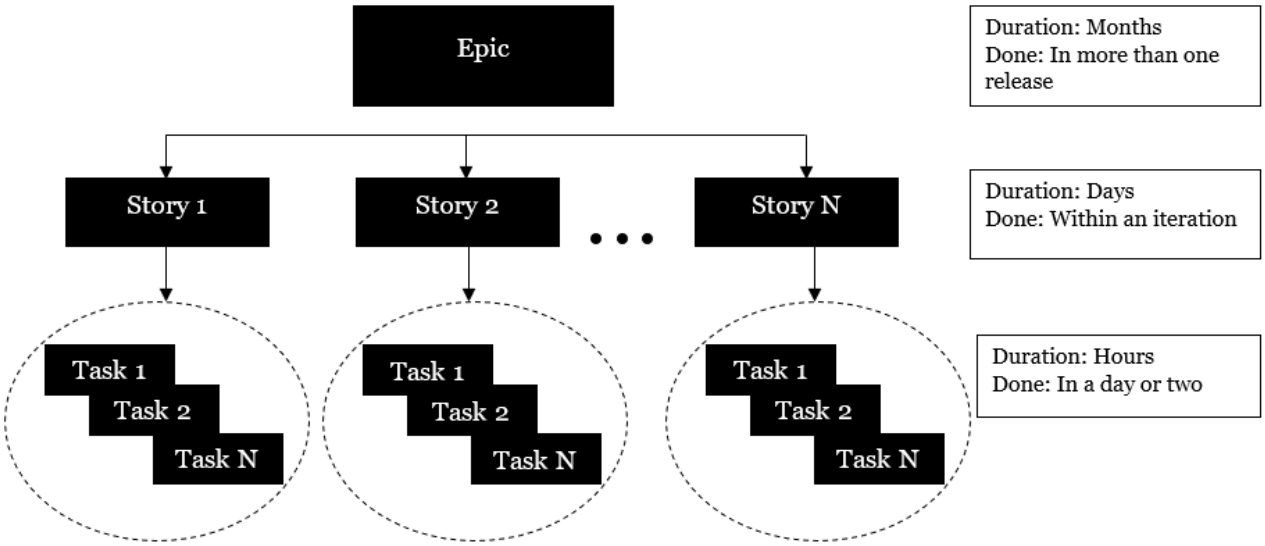


- Are the requirements that go into the product Backlog
- Are recognizable as providing value by your customer
- Are deliverable by themselves.
- Exist in the Product backlog
- Exist in the Sprint backlog
- Consist of other tasks

Worded often as such:

As a [Role]
I want to [be able to]
So that I can [justification]

Epics, Stories, Tasks



- Epics are just a grouping of Stories
- Story splitting is discussed later
- Tasks can have other tasks

Story points

- Estimate of how "Big" a story is (not how long will it take)
- Are relative estimate of the "Teams" effort
- not any individual's effort

Story points

- Loosely follows the Fibanocci sequence
- Fibanocci - 0,1,1,2,3,5,8,13,21,34,55,89
- Story Pts - 0,1/2,1,2,3,5,8,13,20,40,60,100

Story points

- As stories get large so does the range of error
- Cone of uncertaintity
- Are relative to each other
- Difficult to explain in theory
- But works in practice
- And does take practice

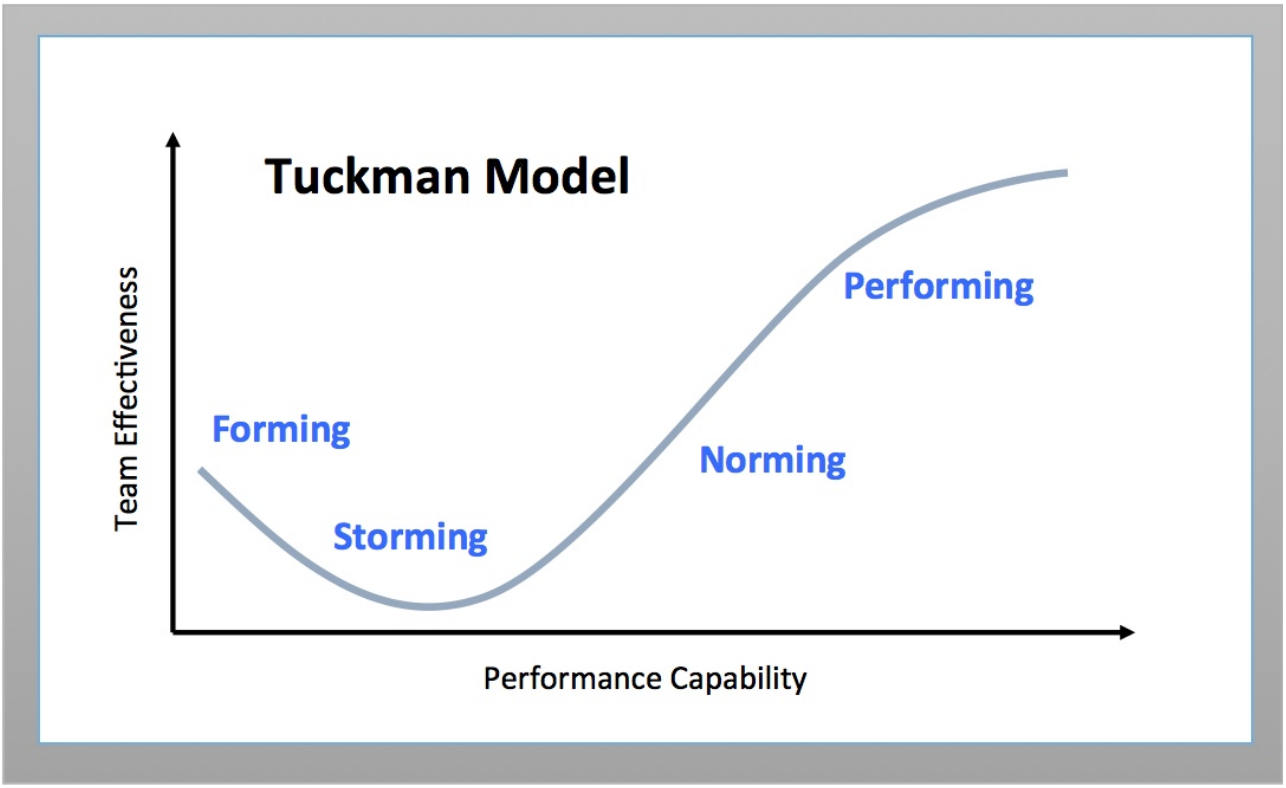
Jira Demo

Delivery Team

Attributes

- Goals are clear
- Focus
- Small team
- Specialist in some things
  - But not all are snipers
- Generalists in others
  - But all can shoot
- Self-organizing (Autonomous)
- Does not disband

Group Development



- Learning about each other
- Challenging each other
- Working with each other
- Working as one…

Delivery Team Responsibilities

- Align the team around the **same** goal. (similar goals aren't good enough)
- Responsible to deliver a:
  - secure
  - understood
  - quality
  - …
  - Solution

Delivery Team Responsibilities

Work closely with the Product Owner to:

- discuss solutions
- suggest alternatives
- estimate stories
- design/development/testing

- translation/implementation

Everything from Start to Finish

Delivery Team Responsibilities

Assume that you have 10 things to finish.
Are these 2 statements equal?

- 100% of 8 things
- 80% of 10 things

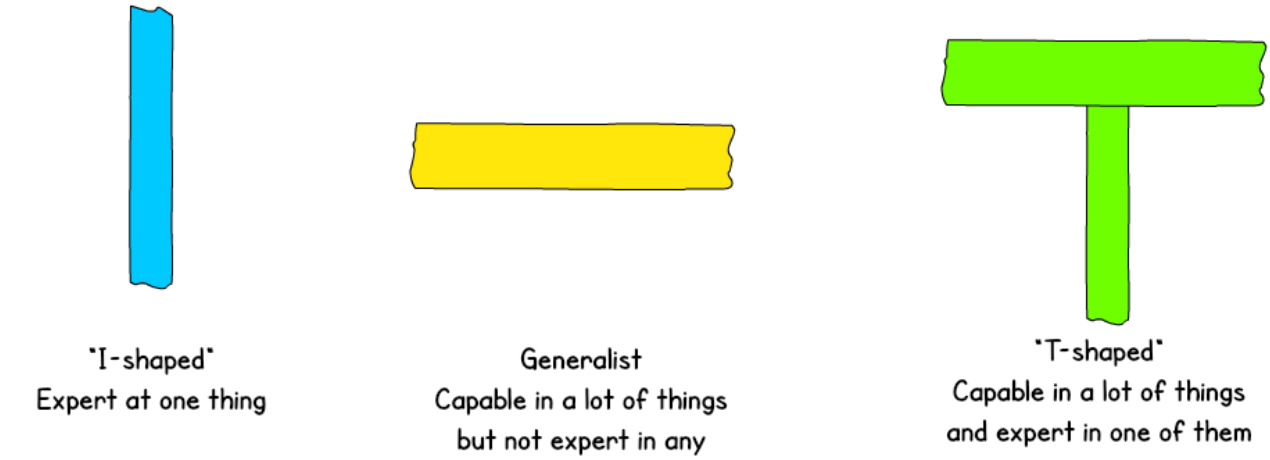
Focus on FINISHING work

But how?

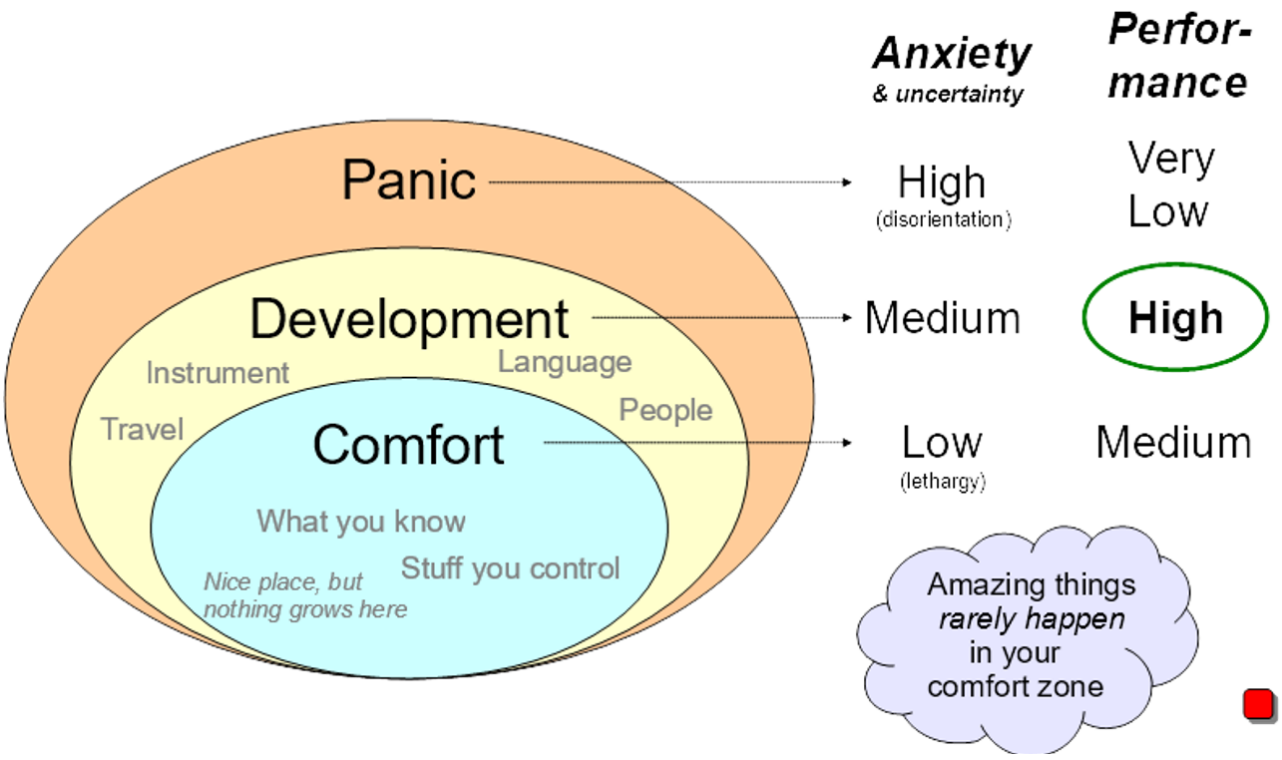
Swarming

- Put more than one person on a story
- Only possible if you break a story down into tasks.
- And possibly break those tasks down into sub-tasks

T-Shaped Skills



Acquiring T-Shaped skills



Summary of T-Shaped skills

Everybody can do everything!!!!

- No - But we want that very few things can be done by only one person

Specialization Issues

- Creates dependencies between tasks

- Creates handoffs
- Dependencies on individuals
- Prioritization by skill and not ROI

Utilization

- Focus should be on effectiveness
- Slack time is important

Slack Time

> Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency
>
> — Tom Demarco

Effective vs Efficient Video

(https://fccfac.sharepoint.com/sites/agilecentreofexcellence/Shared%20Documents/Blogs/Effective%20vs%20Efficient%20blog%20post.mp4?csf=1&e=rS96fh)

Product Owner

Makes sure that the right things get done!

- Sets the vision for the product
- The Product Owner will prioritize based on business value first. Once the team adds in the cost, she will prioritize based on ROI.

Product Ownership in a Nutshell.

▶ https://www.youtube.com/watch?v=502ILHjX9EE *(YouTube video)*

Other Considerations

ROI Calculation

ROI = BusinessValue / Cost

For the Mathematicians in the crowd
As Cost goes down, ROI goes up
(As Cost → 0 then ROI → ∞)

And conversely
As Cost goes up, ROI goes down
(As Cost → ∞ then ROI → 0)

Product Owner Other Responsibilities

- Maintains and grooms the product backlog
- Make sure the team understands what the criteria is for deciding between stories
- Runs the refinement
- Runs the planning session
- Work with Stakeholders (50%)

Celebrity Ordering

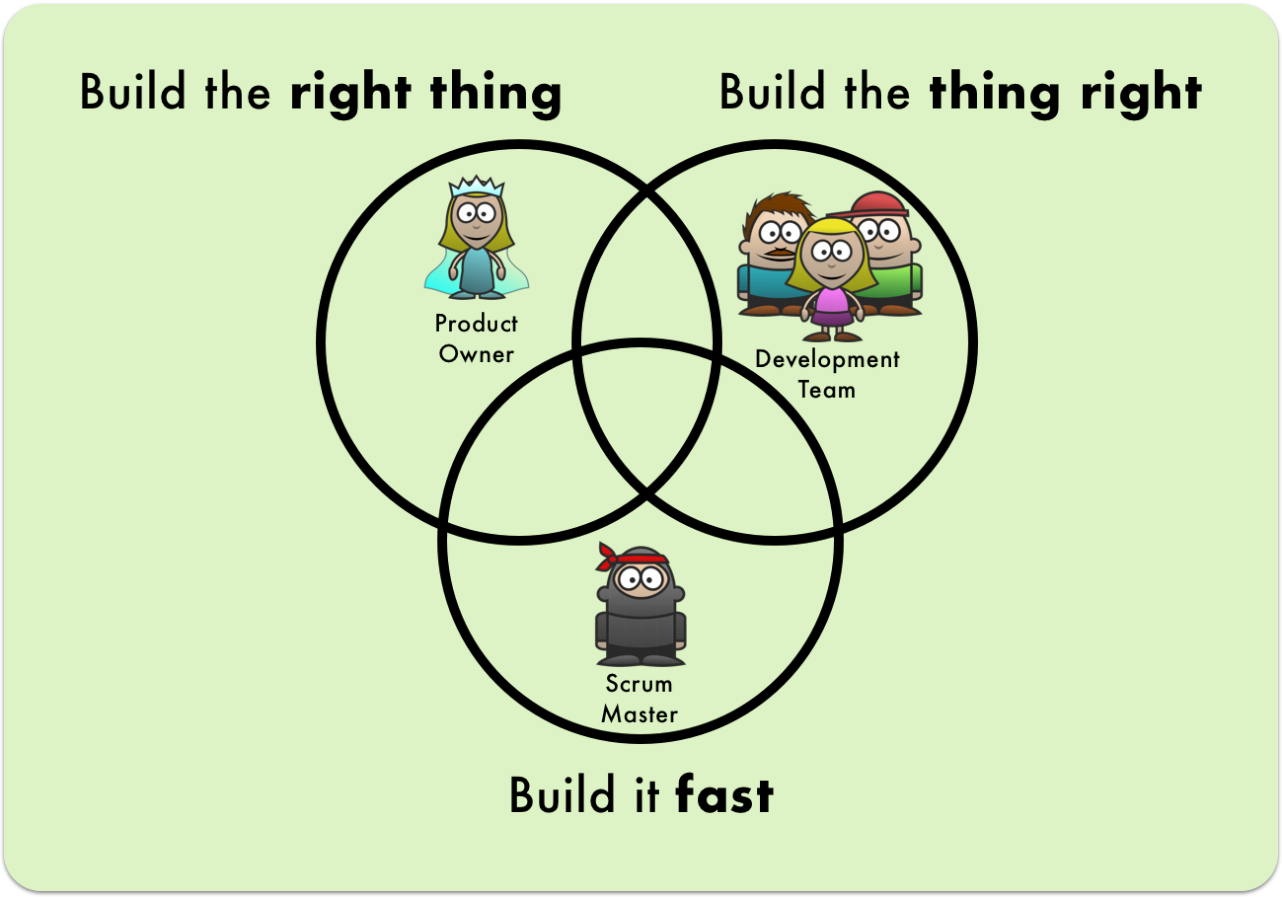
Save the drowning passengers

Celebrity Ordering

Now - Pick a product owner.

Product owner should decide on the criteria for saving people and articulate that to the team

Scrum Master



Scrum Master Responsibilities

- Removes impediments !!!
- Reinforces agile principles
- Help promote that things are done right!
- Monitors team health
- Facilitates

A True Servant Leader

Three Project, Three experiments

Project 1 - 1, 2, 3, 4, 5, . . . 30

Project 2 - a, b, c, d, e, . . . z

Project 3 - 2, 4, 6, 8, 10, ...56

Iteration 1

- Complete Project 1, Project 2 and then 3 consecutively.
- Capture how long it took you to do all of them.

Iteration 2

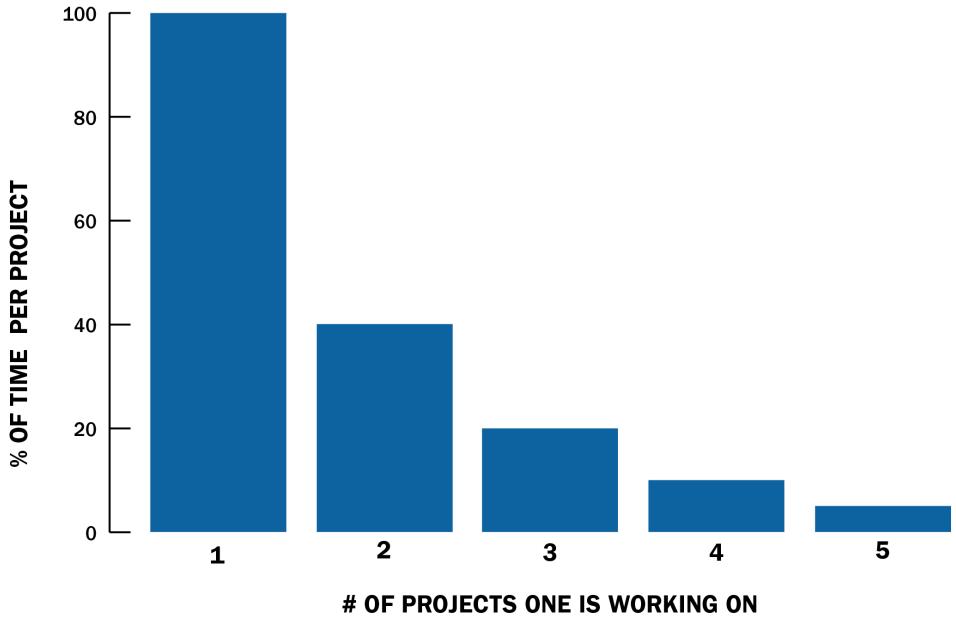
Business conditions have changed and your priority will now be dictated in real time.

- Begin with Project 1 and switch to Project 2 when you hear "switch"
- Pick up where you left off as you cycle through the story.
- Capture how long it to you to do all of them.

Iteration 3

- Complete Project 1, Project 2 and then 3 consecutively.
- There is a new goal that ensure that no one is ever idle.
- Capture how long it to you to do all of them.

Debrief

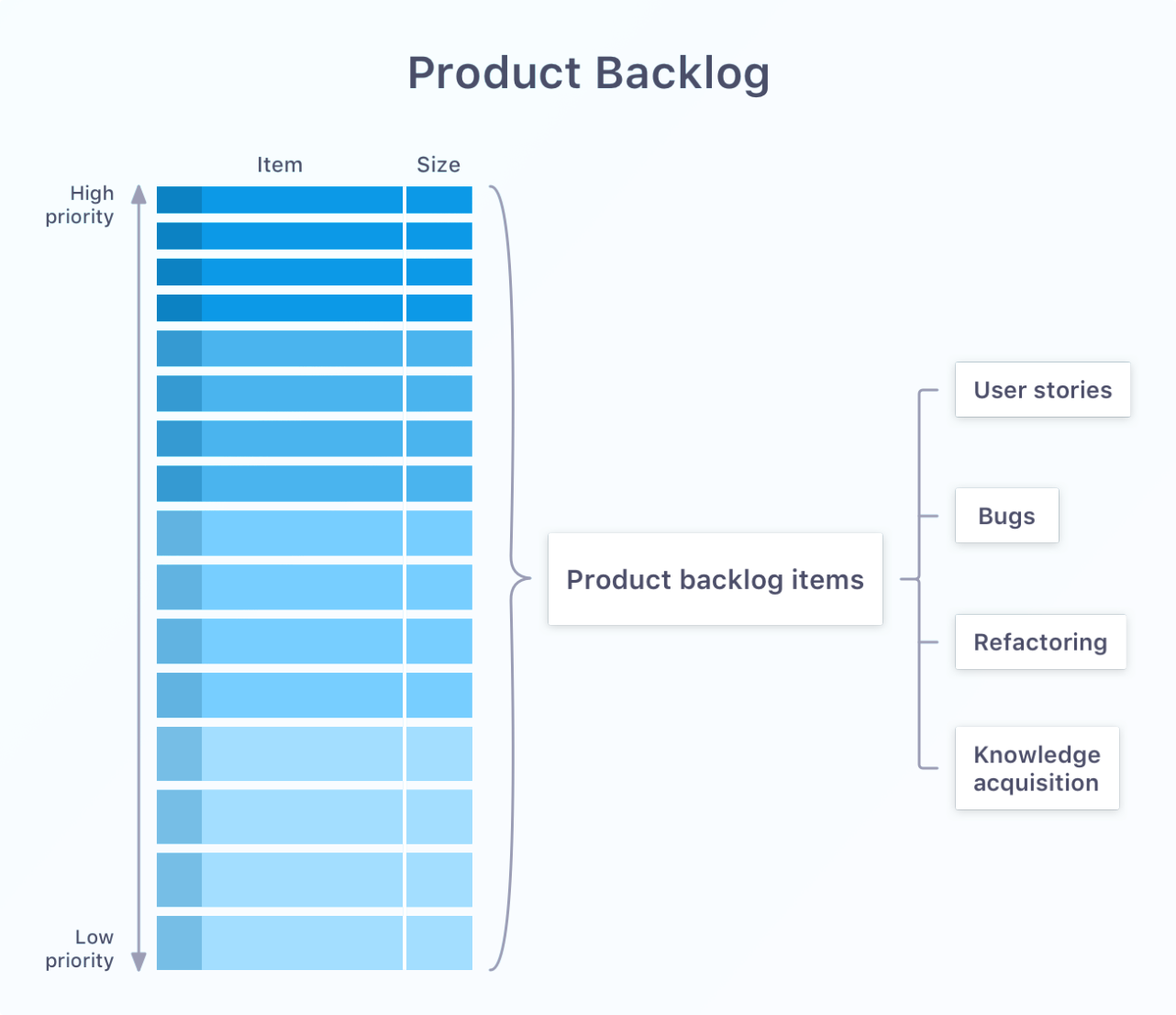


Based on information from *Quality software Management: Systems Thinking*, p284

Product Backlog

What is the Product Backlog?

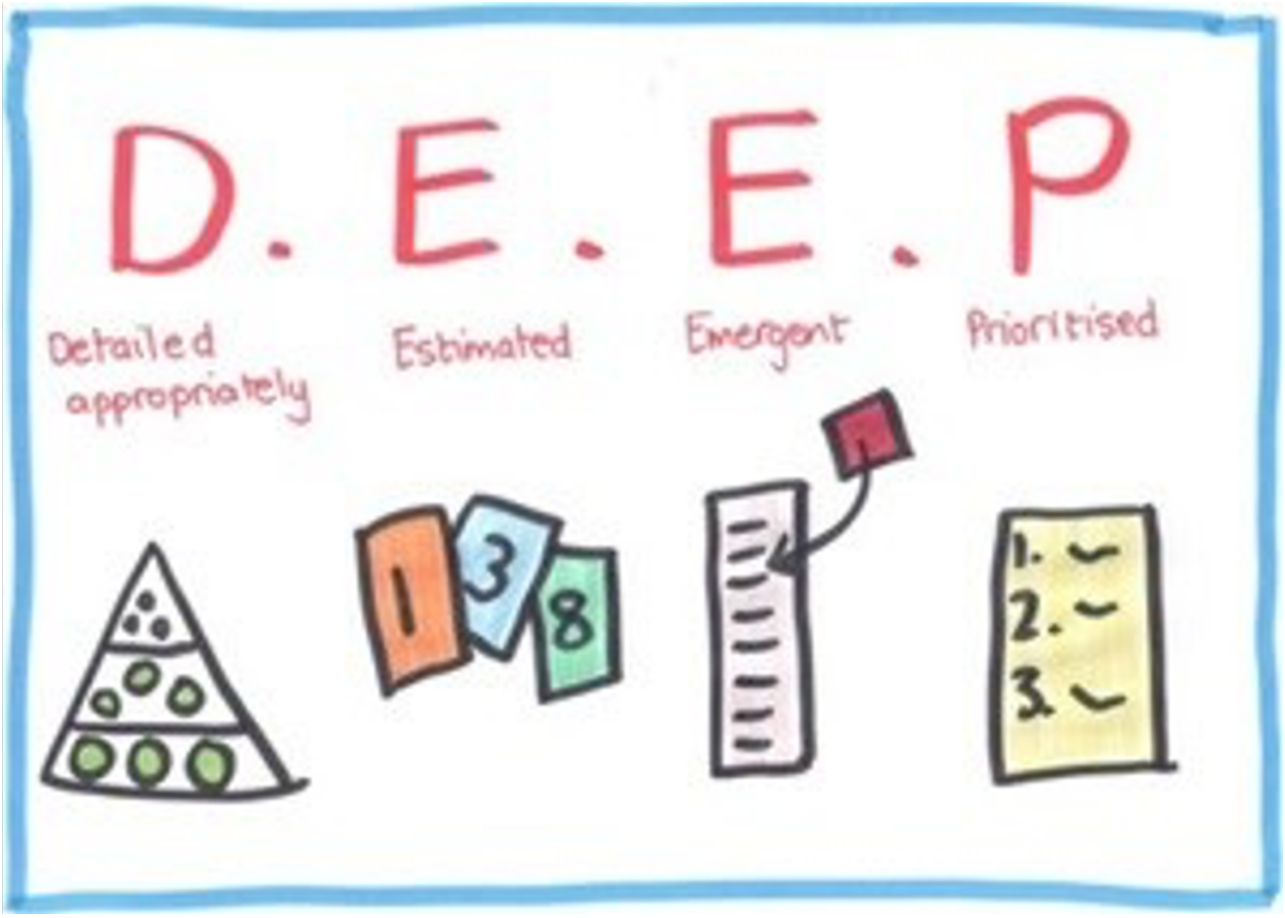
- Requirements
- Prioritized
- Provides Transparency to stakeholders



Refinement meeting

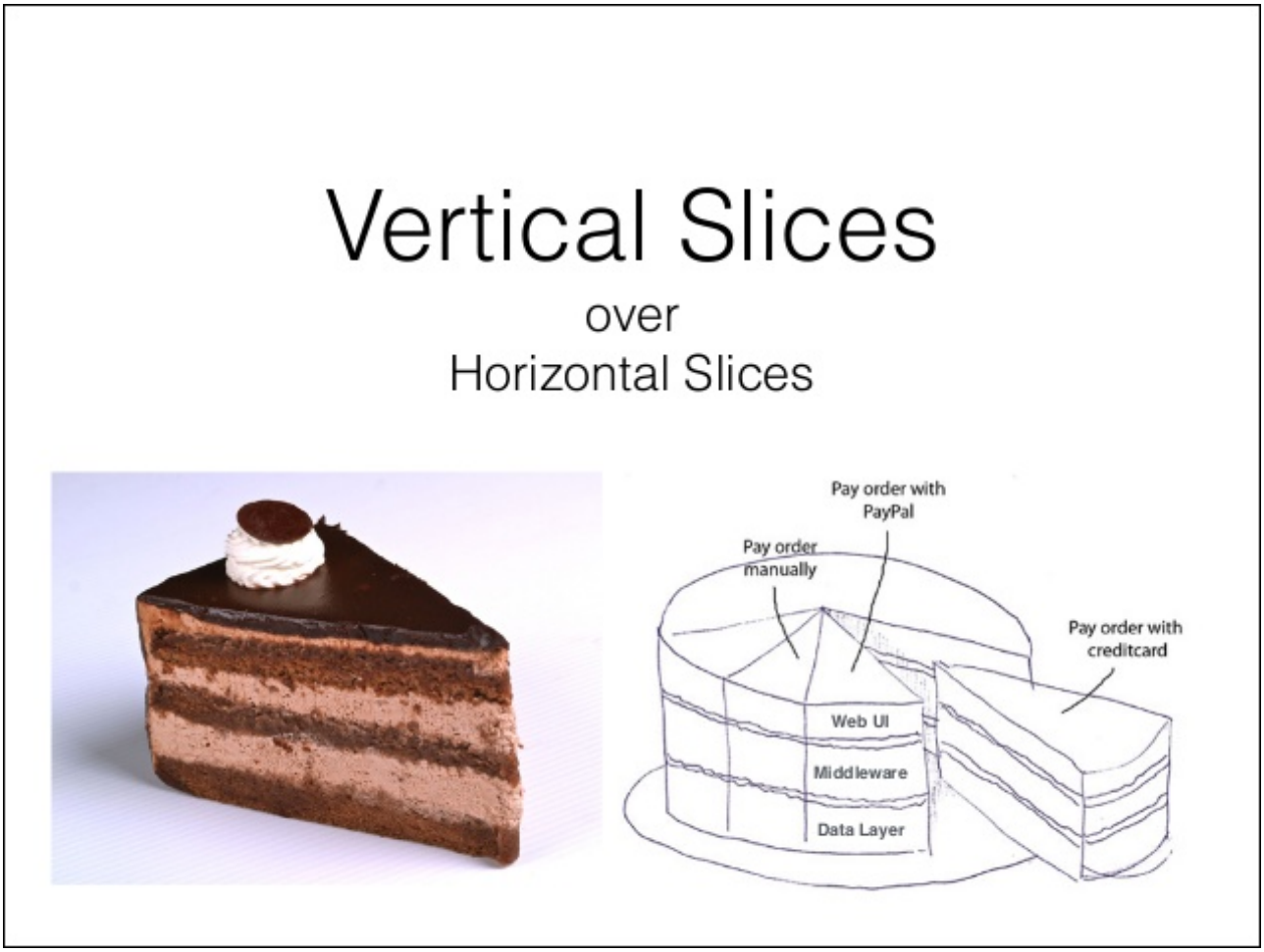
- What – the act of adding detail, estimates, and order to items in the Product Backlog
- Why – help the Product Owner get the Backlog ready for the next sprint planning meeting
- When – Near the end of the sprint (usually 1-2 hours) - maybe longer.
- Who – Product Owner, Scrum Master, Dev Team, SMEs (if needed)
- Lead by the Product Owner

What Does a Good Product Backlog Look Like?

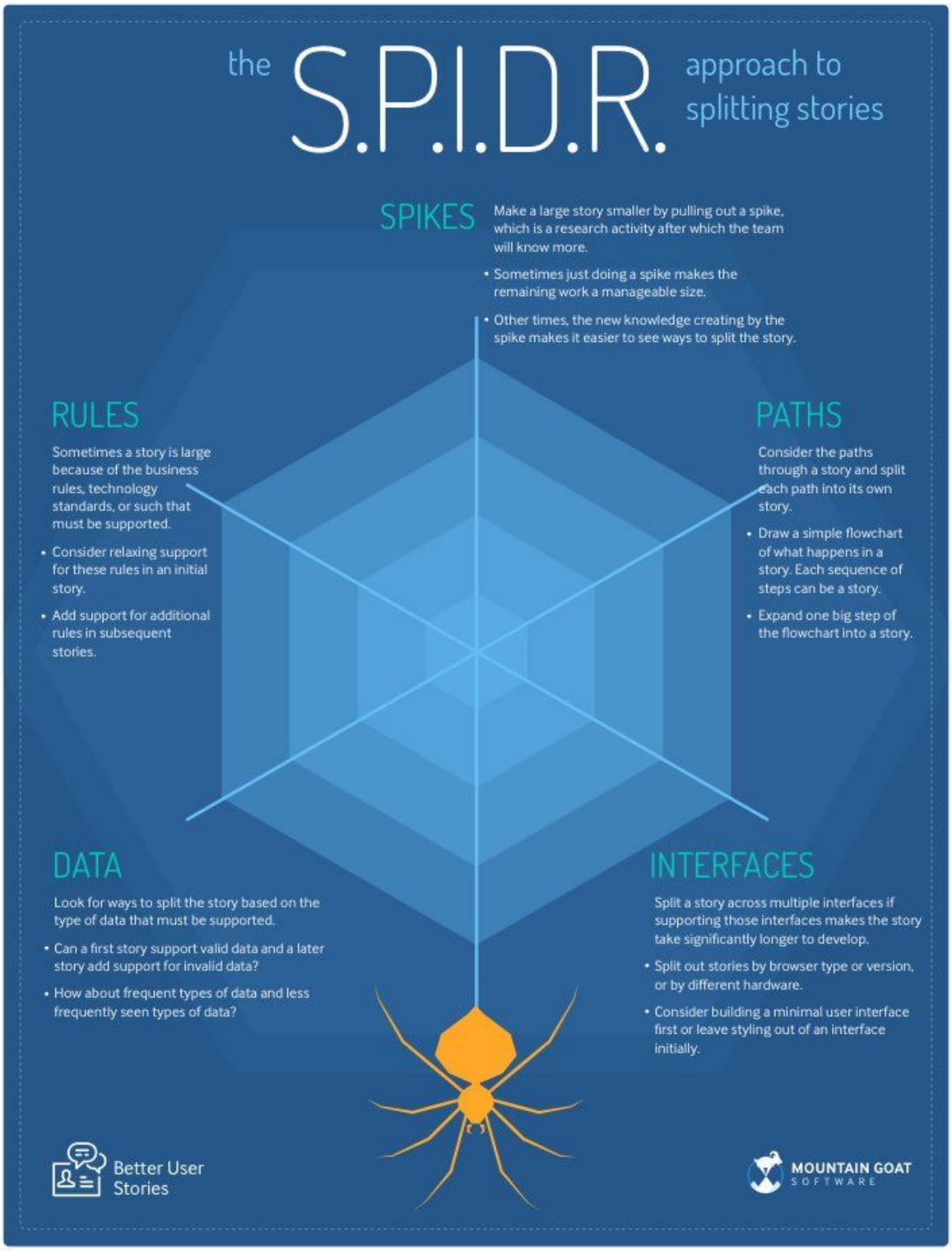


Story Splitting

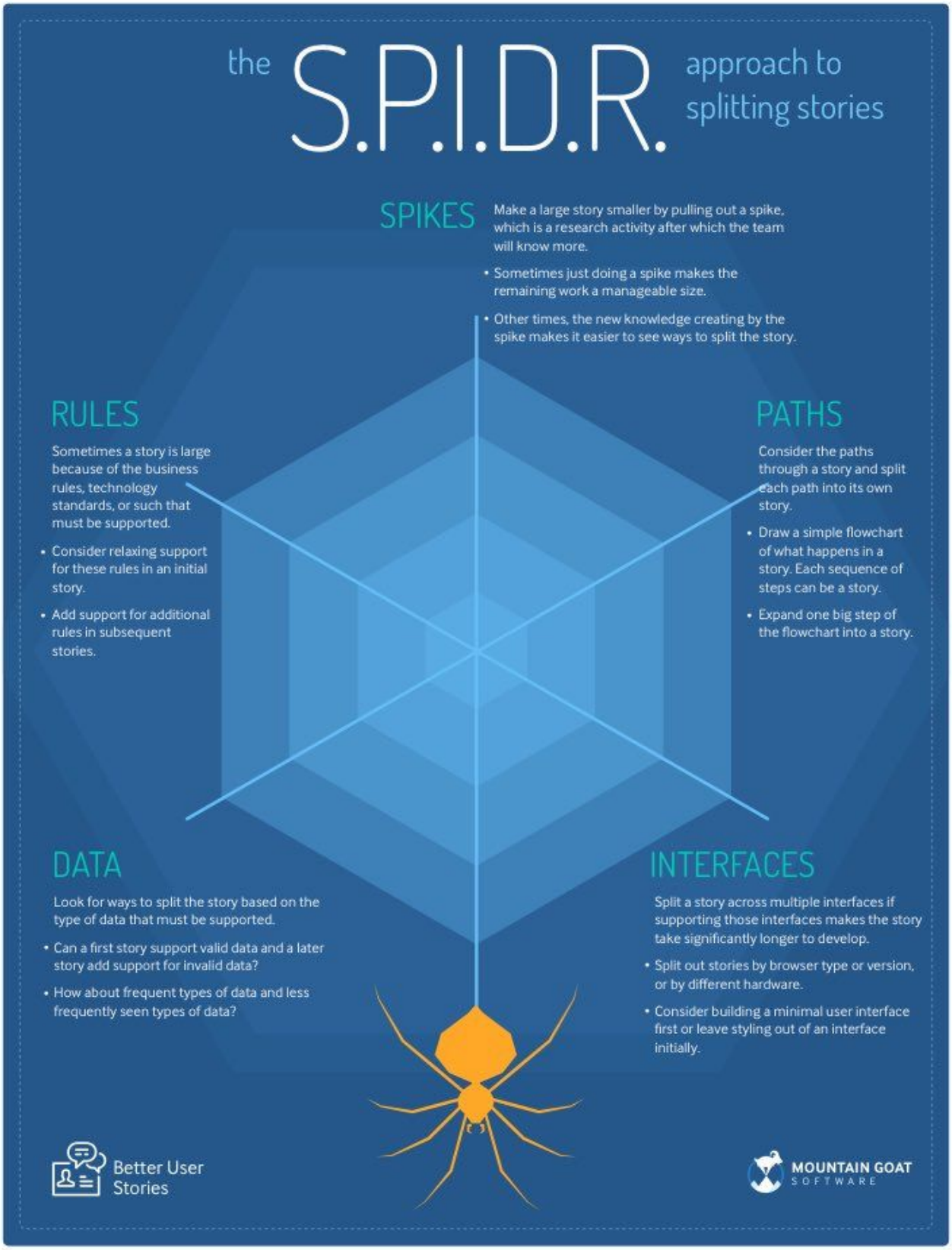
Try and slice vertically



Tactics for Splitting

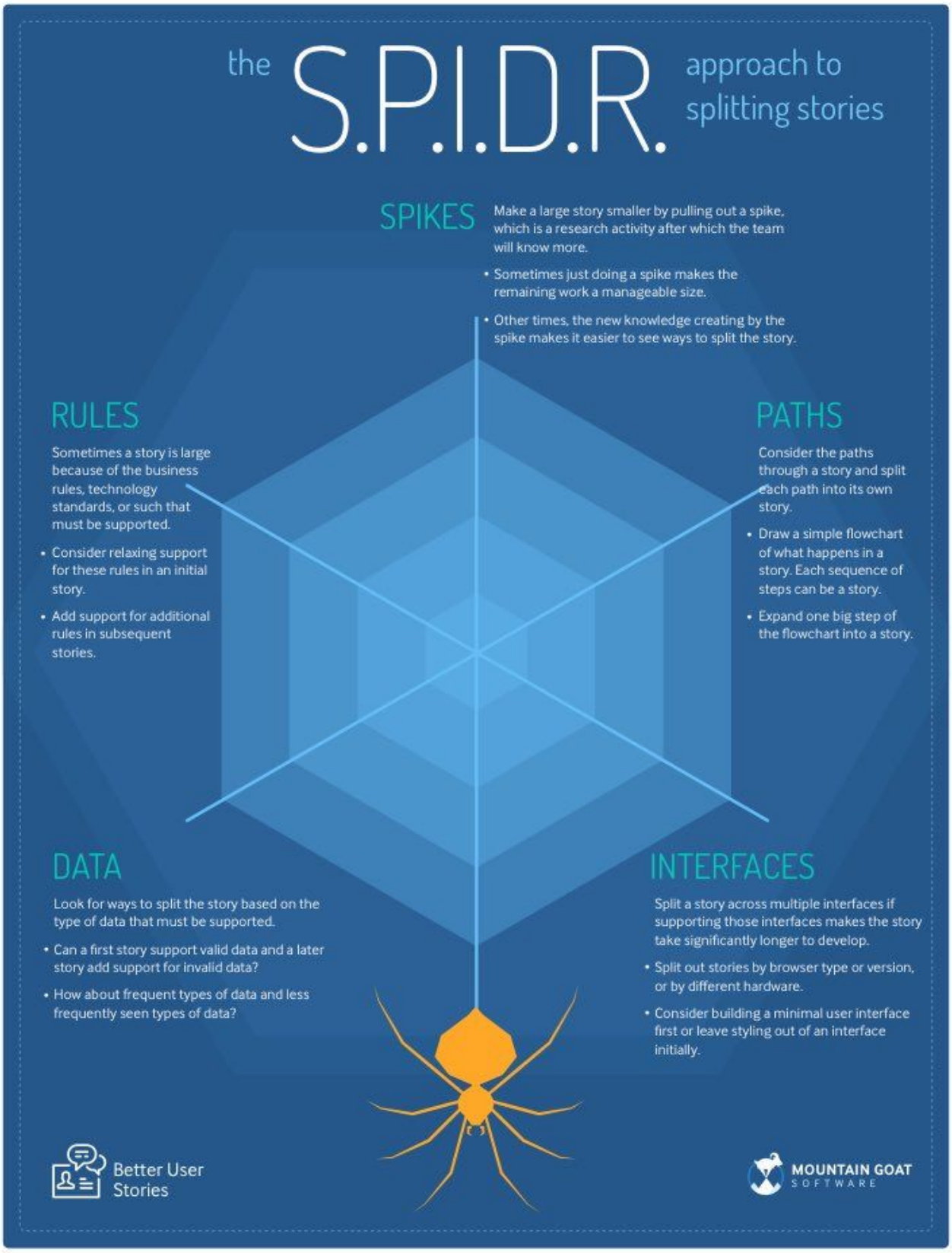

the S.P.I.D.R. approach to splitting stories
SPIKES
Make a large story smaller by pulling out a spike, which is a research activity after which the team will know more.
• Sometimes just doing a spike makes the remaining work a manageable size.
• Other times, the new knowledge creating by the spike makes it easier to see ways to split the story.
RULES
Sometimes a story is large because of the business rules, technology standards, or such that must be supported.
• Consider relaxing support for these rules in an initial story.
• Add support for additional rules in subsequent stories.
PATHS
Consider the paths through a story and split each path into its own story.
• Draw a simple flowchart of what happens in a story. Each sequence of steps can be a story.
• Expand one big step of the flowchart into a story.
DATA
Look for ways to split the story based on the type of data that must be supported.
• Can a first story support valid data and a later story add support for invalid data?
• How about frequent types of data and less frequently seen types of data?
INTERFACES
Split a story across multiple interfaces if supporting those interfaces makes the story take significantly longer to develop.
• Split out stories by browser type or version, or by different hardware.
• Consider building a minimal user interface first or leave styling out of an interface initially.
Better User Stories
MOUNTAIN GOAT SOFTWARE

S - Spike



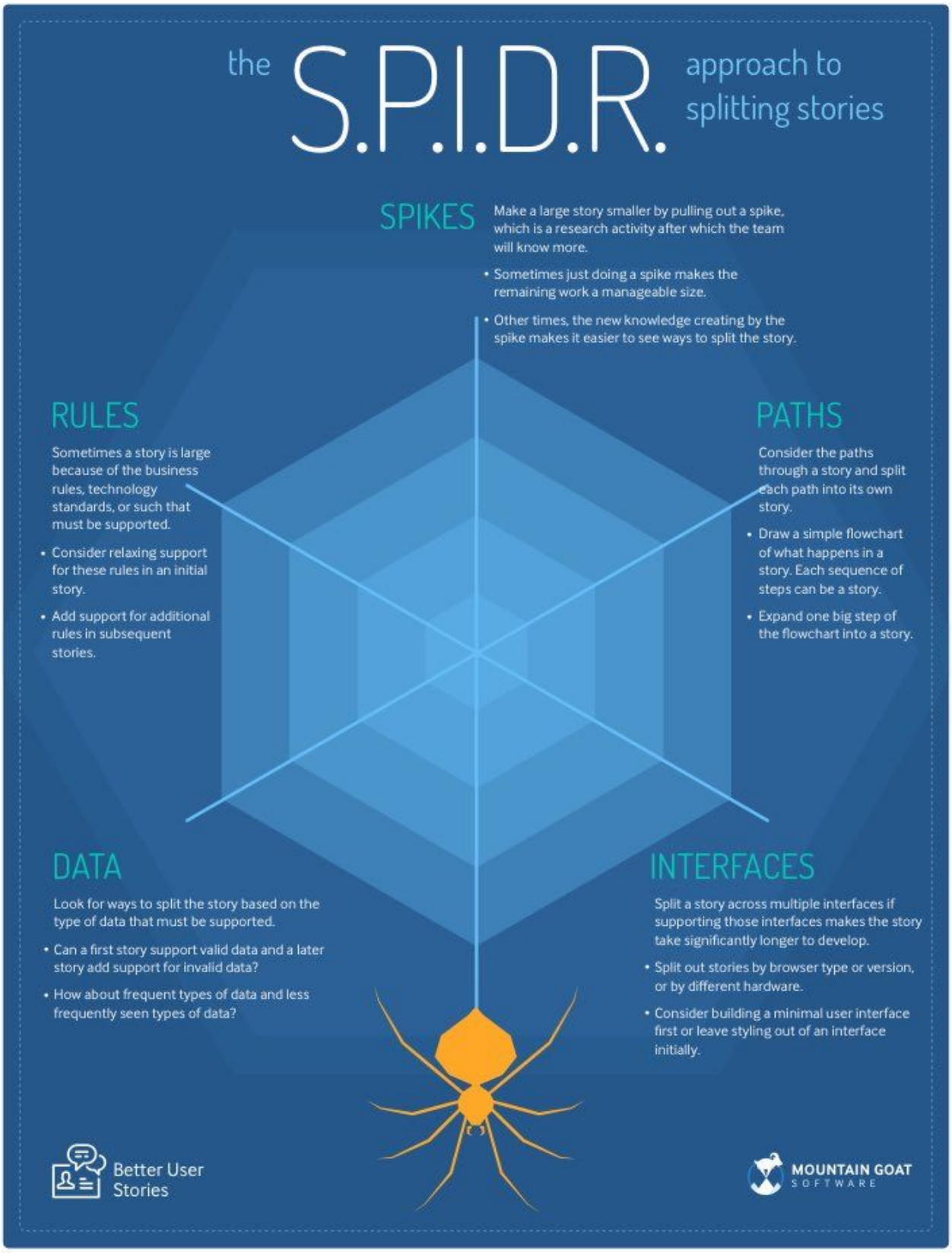
Story is very big
Gain a better understanding of the problem and/or solution

P - Path



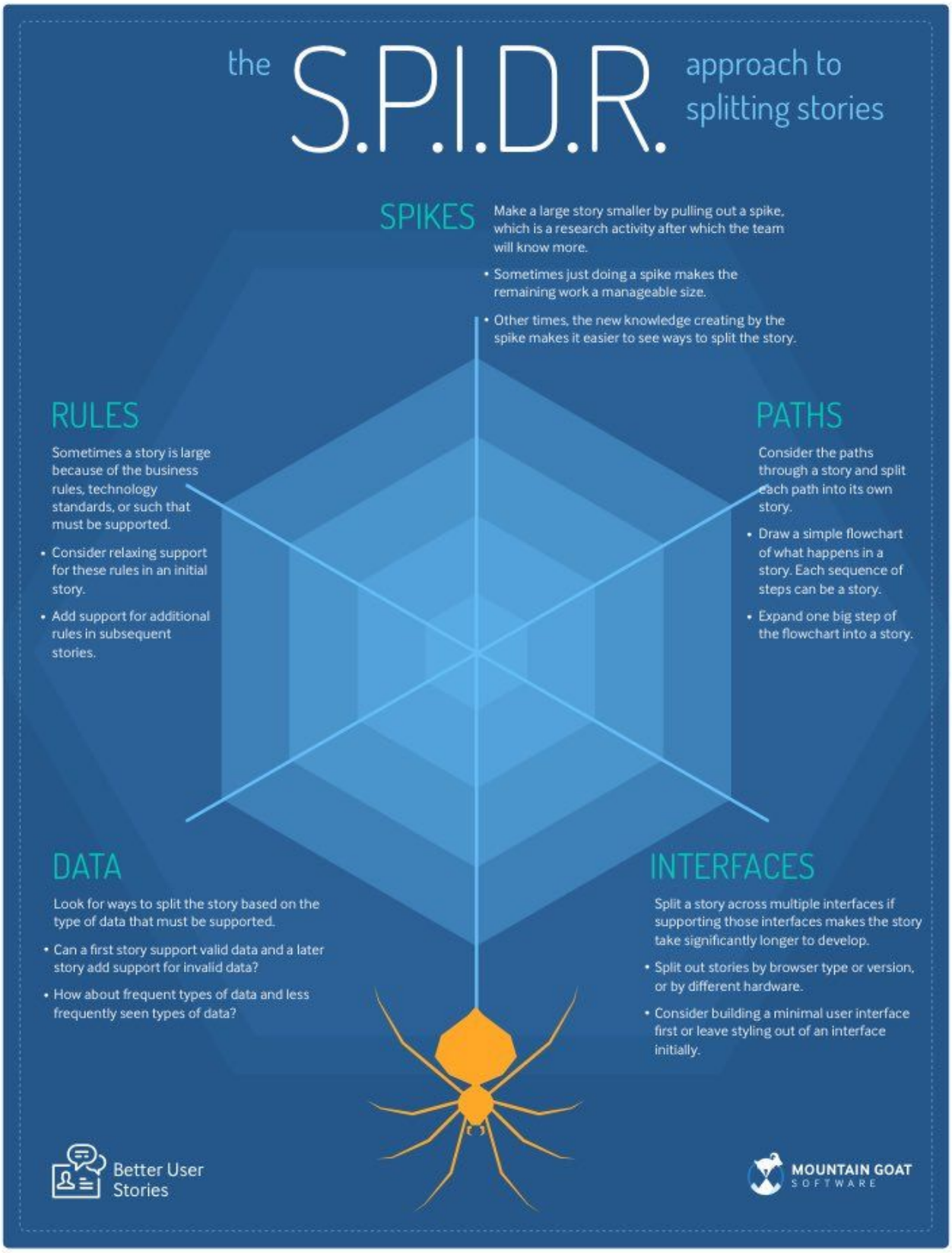
Shopping cart with two methods of payments

I - Interface



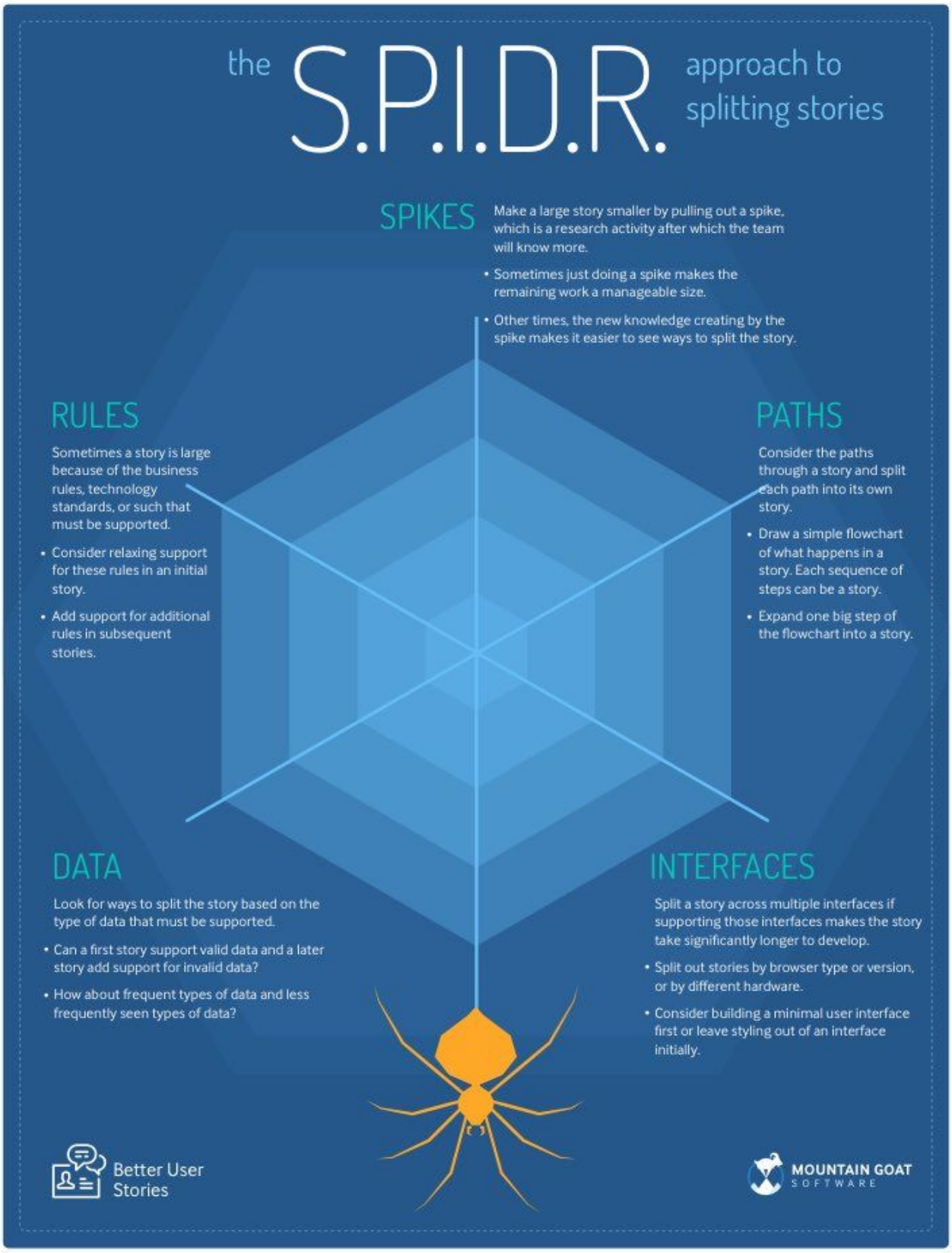
Application is available for multiple browsers

D - Data



Two types of customer
Individual
Company

R - Rules



Relax the Rules

Tour de sprint

The Assignment

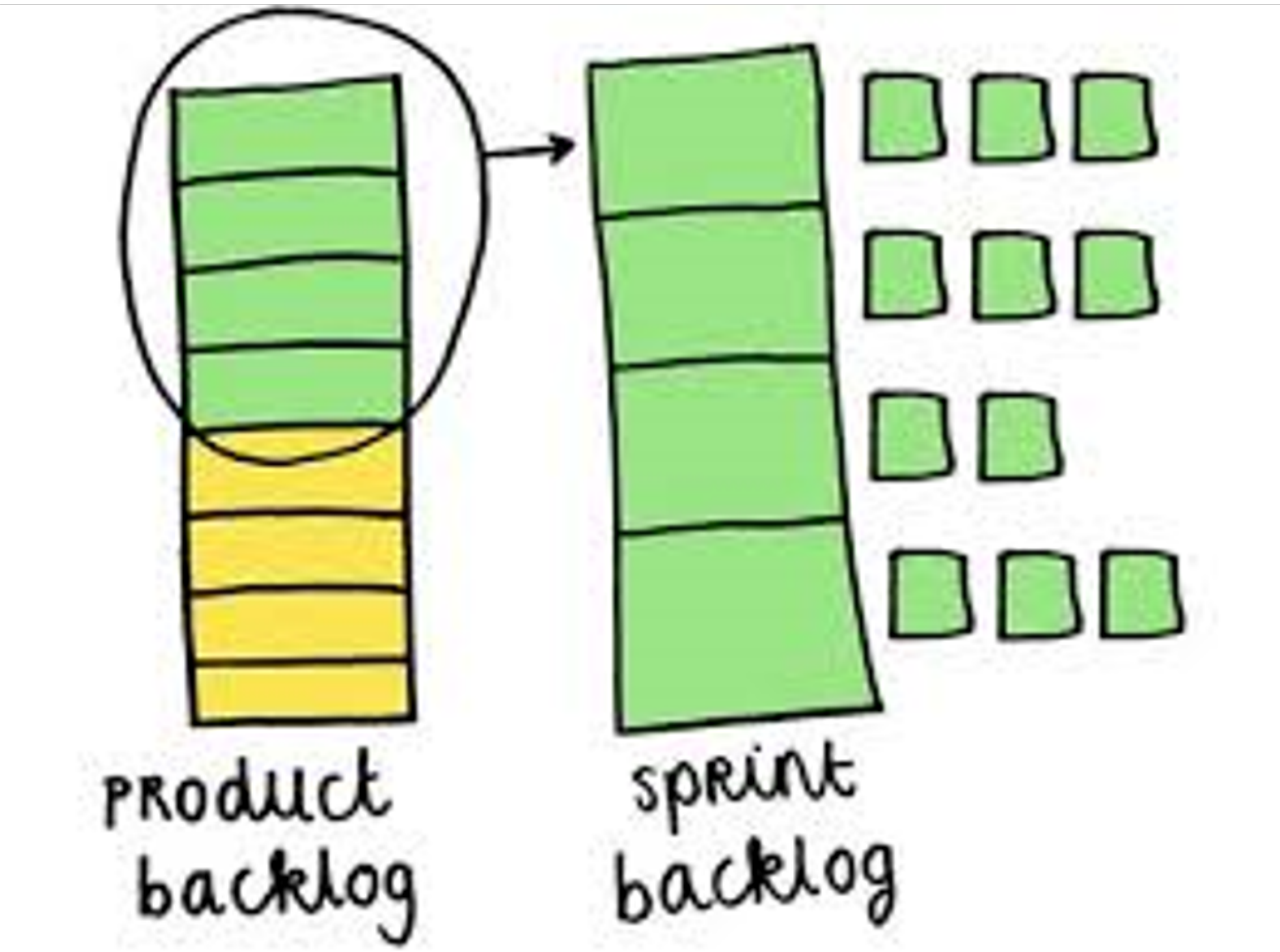
The Attractors travel Agency wants to offer new travel destinations as a brochure so that we can grow our business. Our customers are from Regina, Canada and Frankfurt, Germany. You are the Scrum Team tasked with completing this work. Attractors Travel Agency is your customer You will work in 2 sprints to complete the work. At the end of each sprint you will demo your completed stories. You will also talk about the work planned for your upcoming sprints. Each Sprint is 20 mins. Use the provided template to fill in the information that you research

Those details include:

Flights, Accommodations, Transportation, Food & Attractions, etc.

Sprint Planning

What is the Sprint Backlog?



- Subset of Stories from the Product Backlog
- Tasks within each Story
- What the team plans to finish in the Sprint

When and who is involved?

- Sprint planning occurs on Day 1 of the Sprint
- Team determines velocity
- Product Owner determines stories

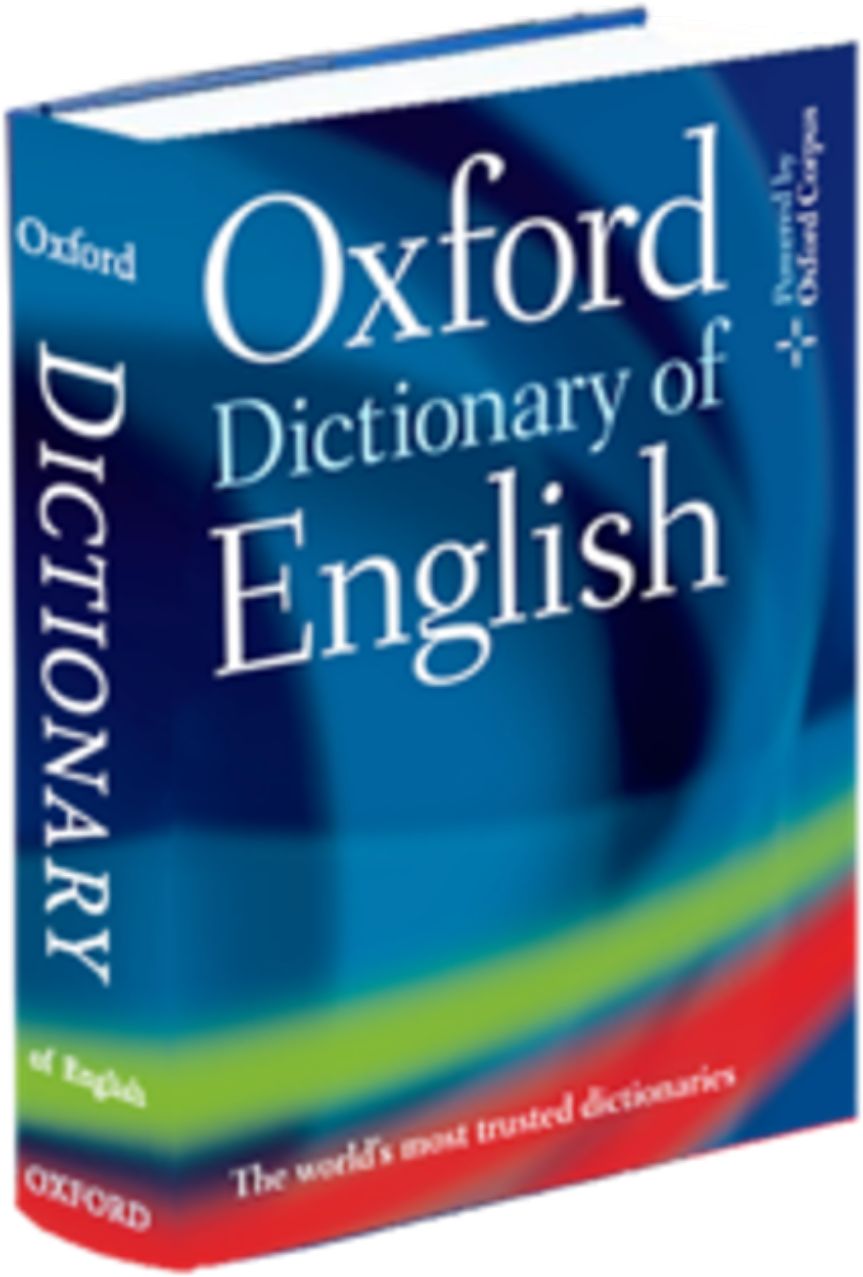
What does velocity mean in Scrum?

Velocity = Number of Story Points on average that a team has historically **completed** during a Sprint

Sprint

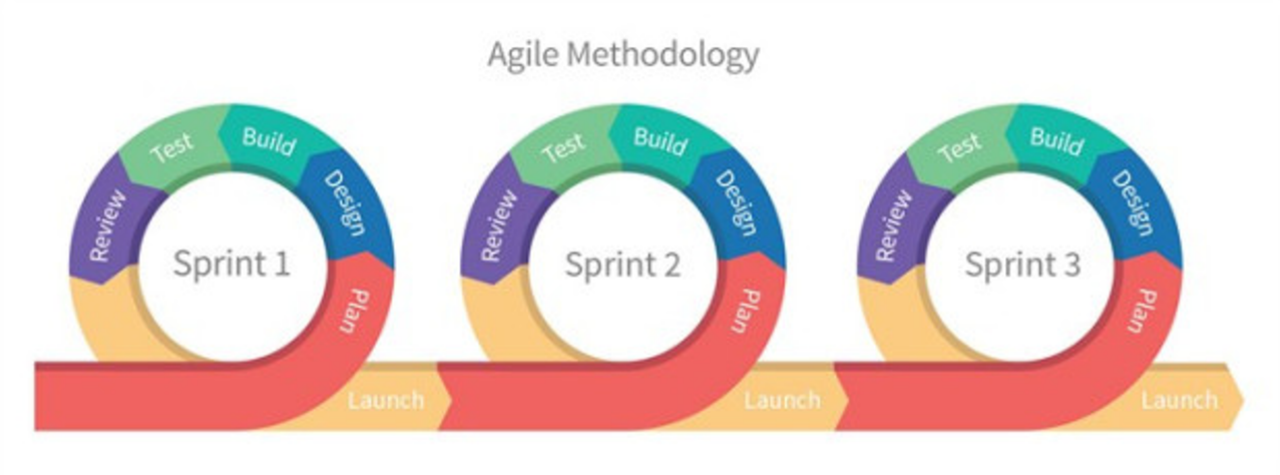
Word of the Day

- Word: schadenfreude
- Pronounced: shaw-den-froy-day
- Meaning: to take pleasure in someone else’s misfortune



What is a Sprint?

1 Day to 4 Weeks



Sprint Rules

- No “non-delivery team” changes
- Consistent sprint lengths
- Team chooses how to complete stories (by priority)

Why such a short timeframe?

- Highest priority (ROI)
- Reduce risk
- Risk grows exponentially

Risk Graph

```
set xlabel "Number of weeks" font "Helvetica,20"
set ylabel "Number of Bugs" font "Helvetica,20"
set key on bottom
plot [0:6][0:110] x**2.5 title 'Risk grows exponentially!'
```

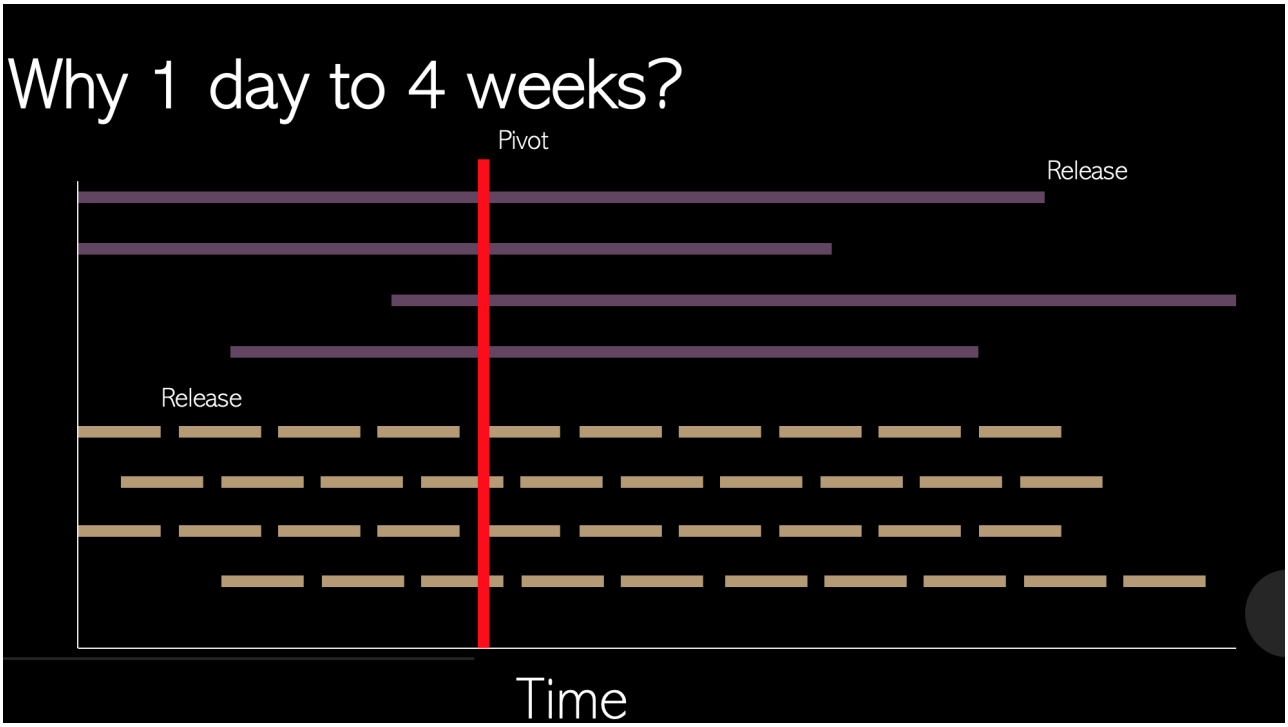
Other considerations that grow exponentially

- OCM
- Training
- Problem resolution time
- Bugs, wrong features

Test - Word of the day?

- Word: schadenfreude
- Pronounced: shaw-den-froy-day
- Meaning: to take pleasure in someone else's misfortune
- We spent 30 seconds to learn this word.
- We should expect to learn 100 words in 50 minutes.
- Could you achieve that?

Why 1 to 4 weeks



What can we conclude?

Work in progress (WIP) is a **liability** to being agile.

Daily Scrum

What is it?

It is an opportunity to get together and communicate with the rest of your team.

What is it?

- Daily ~15 minute huddle
- Not an update to the ScrumMaster
- Same time & same place

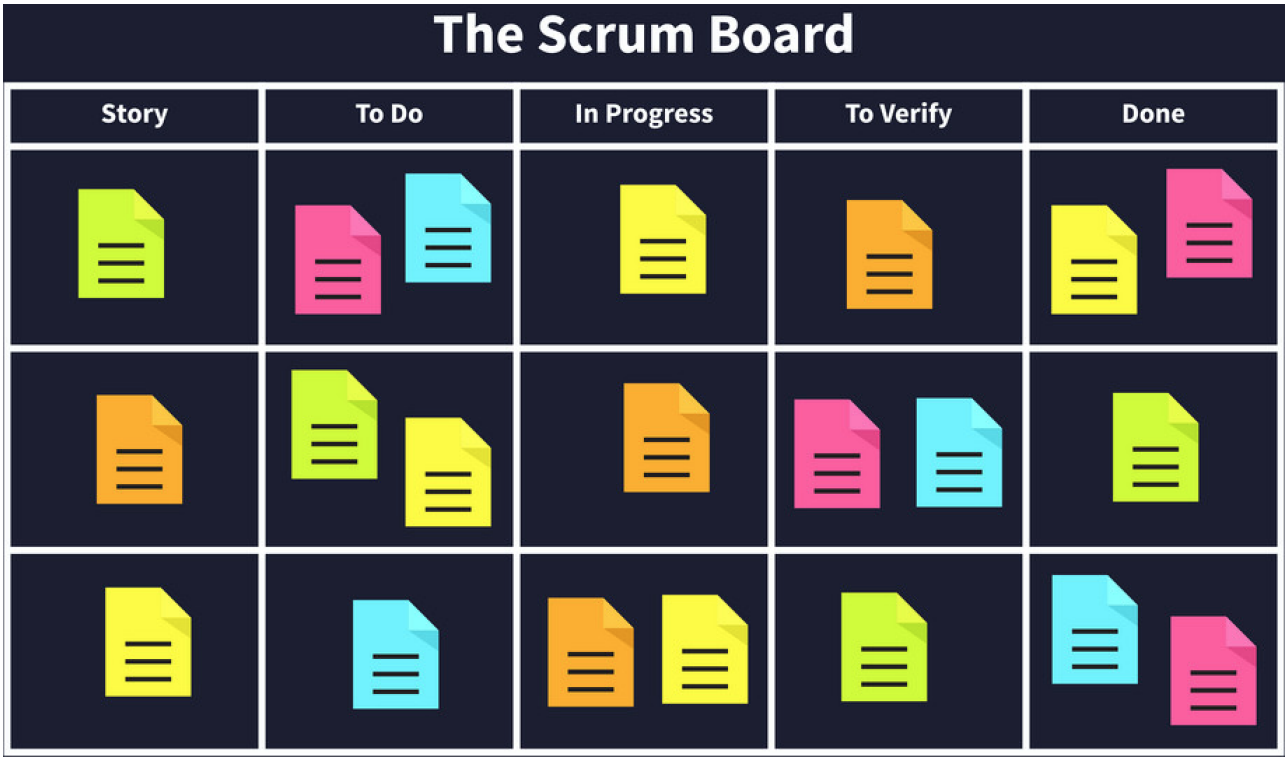
- Same questions
- What did you work on yesterday?
- What will you work on today?
- What is getting in your way?
- How long is it going to take?

Who does it?

- Scrum Master (Lead)
- Delivery Team
- Product Owner (if invited)

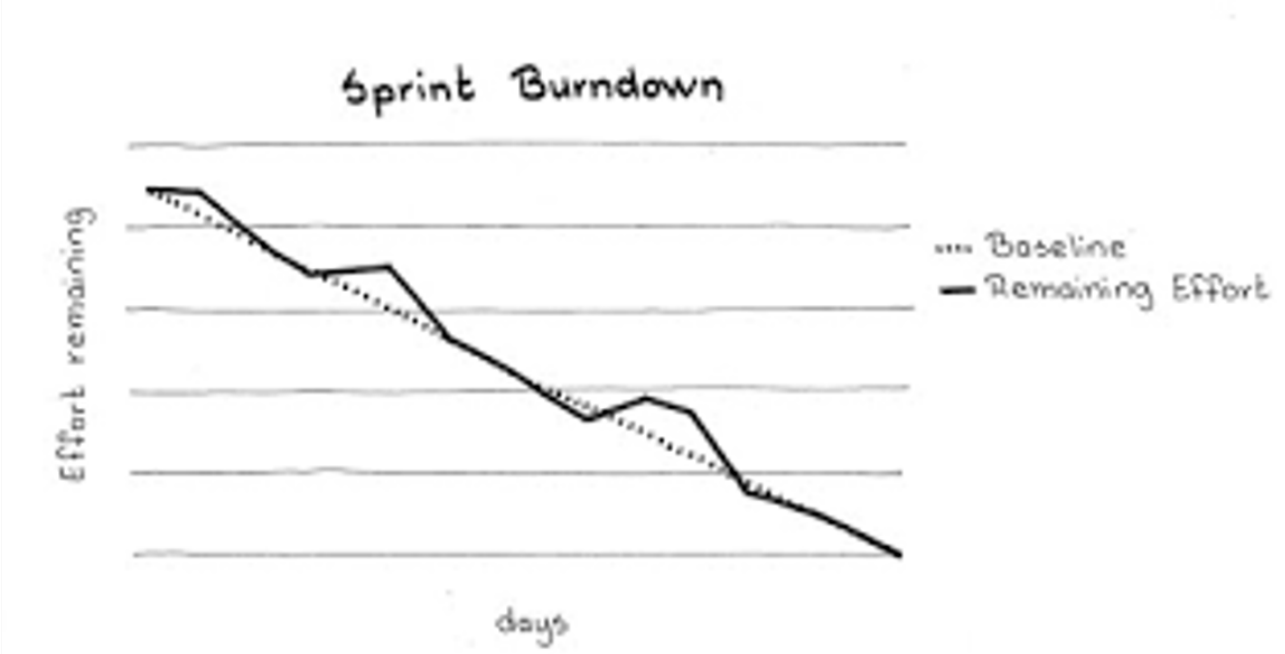
What is a Scrum Board?

- Creates visibility
- Tracks progress
- Keeps focus

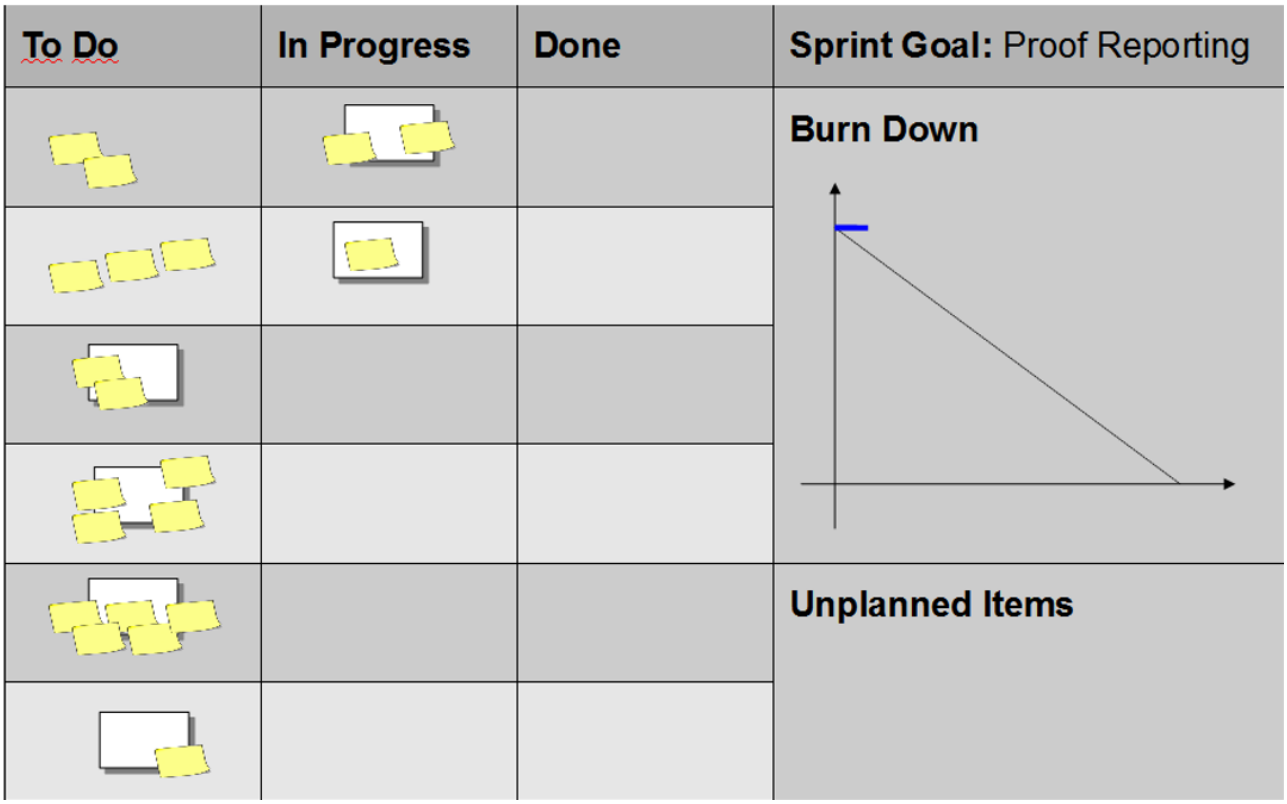


What is a Burndown Chart?

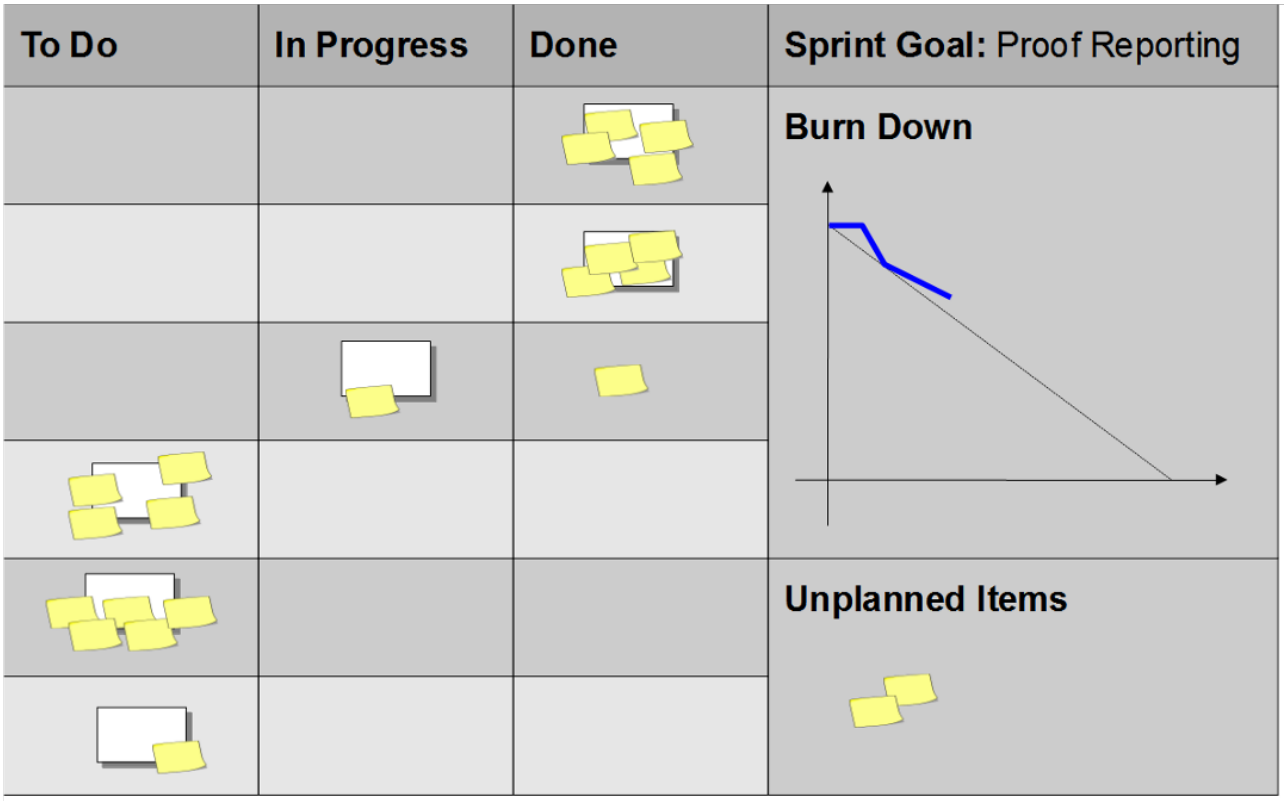
- Tracks progress toward the sprint goal
- The jury is out on this one



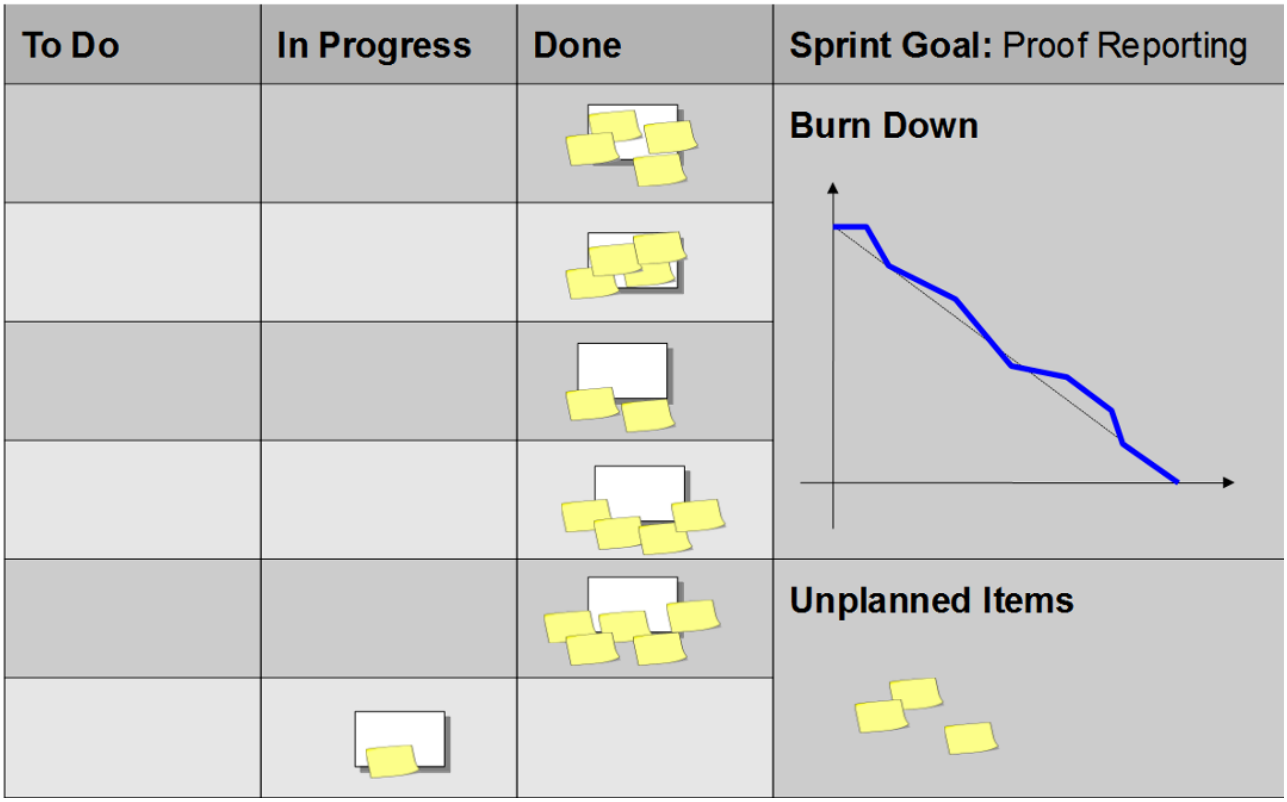
Sprint Board - Beginning



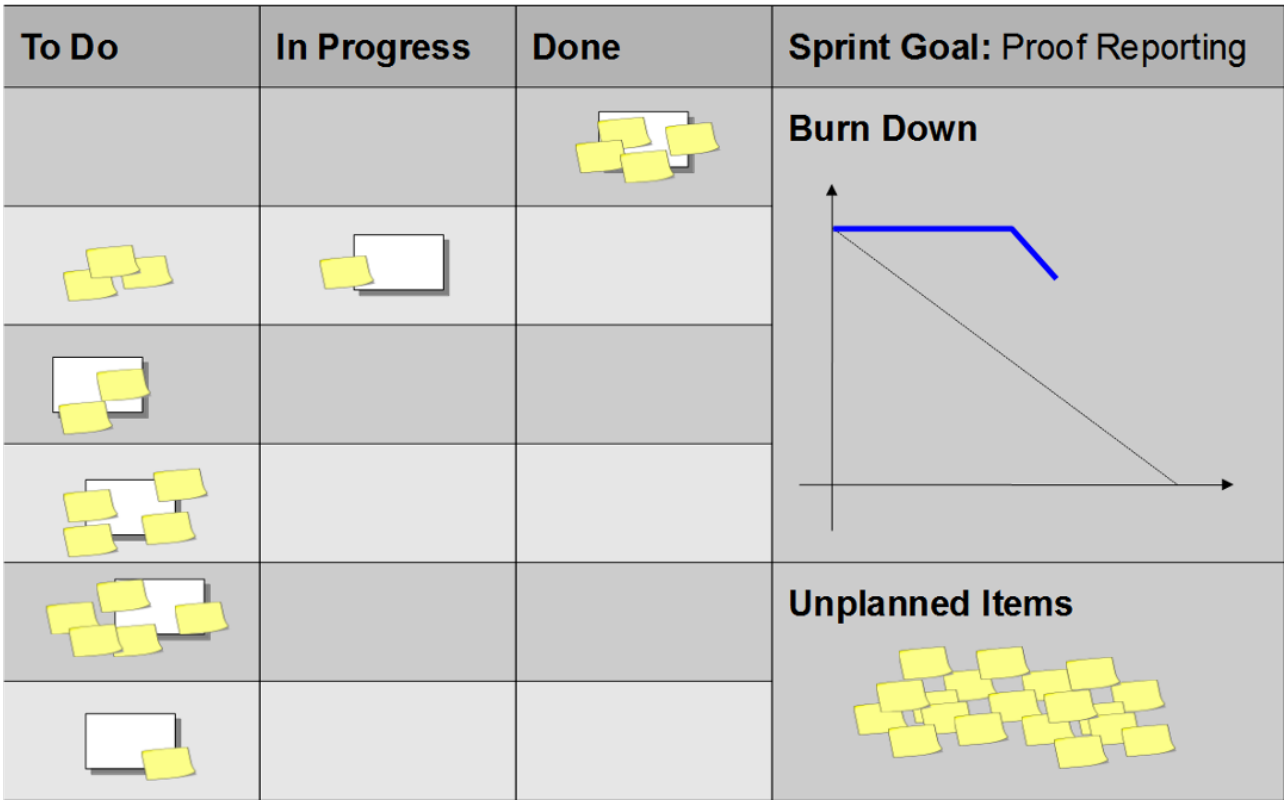
Sprint Board - During



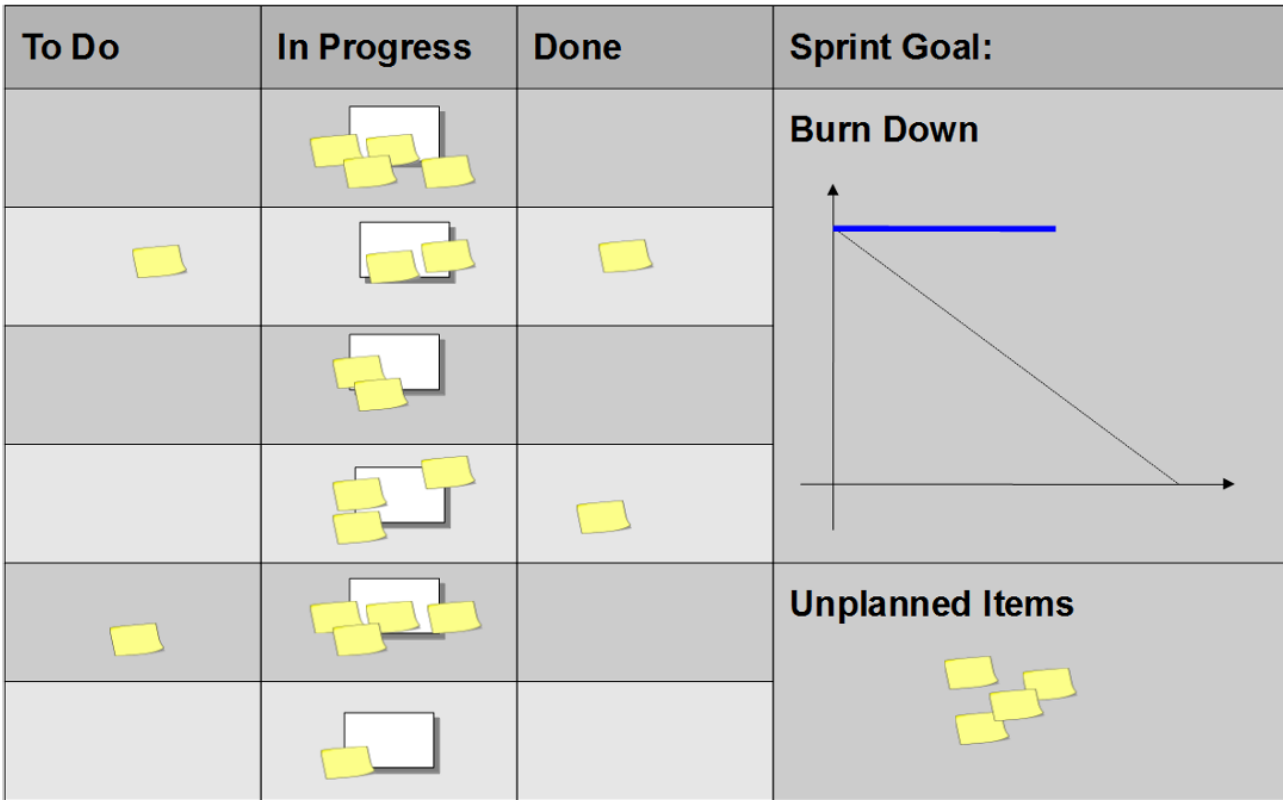
Sprint Board - End



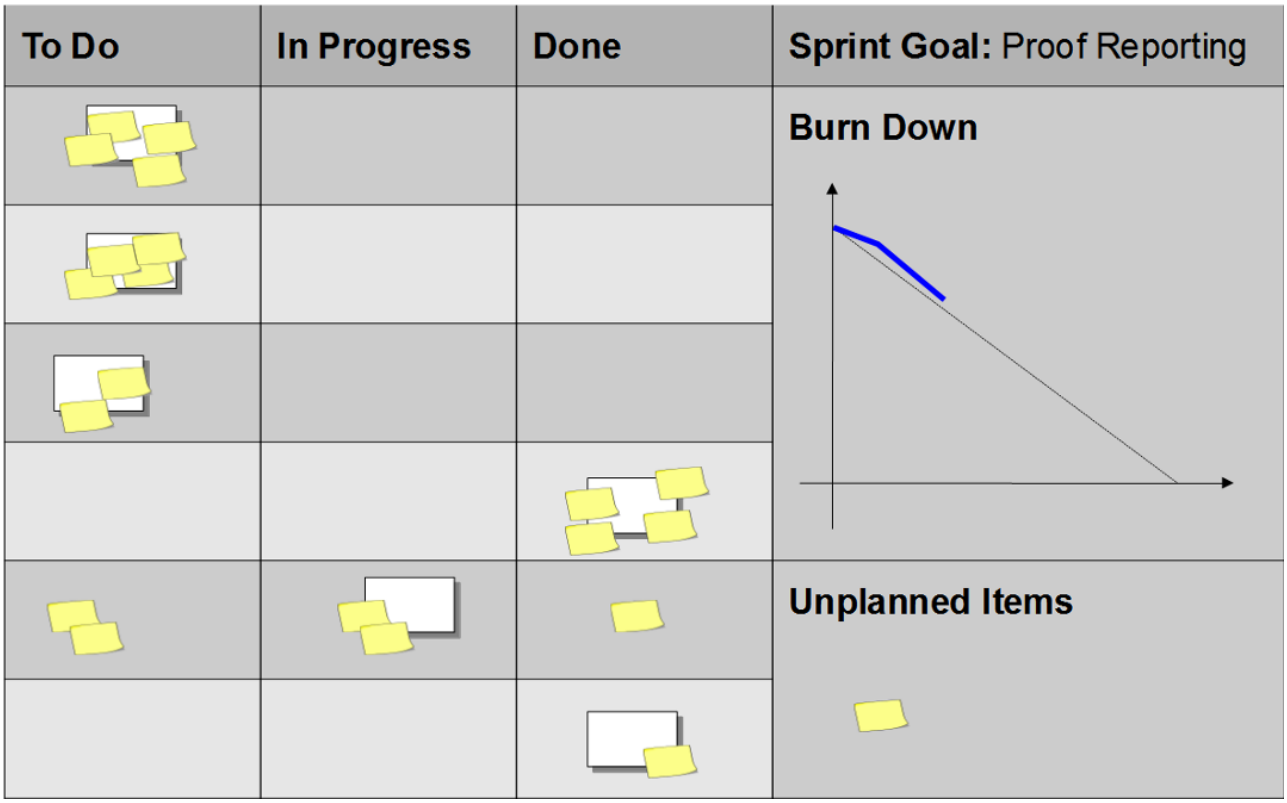
Sprint Board - Pitfall #1



Sprint Board - Pitfall #2



Sprint Board - Pitfall #3



Sprint Review

The Review

What's included?

- Demo of Potentially Shippable Product Increment
- Discussion of unfinished work
- Plan for next sprint

Note:

- True transparency
- Early and frequent feedback
- Usually conducted at the end of the sprint

Demo of the increment: The demo is a largest part of the sprint review. Done – feedback Stakeholder language – problems solving and how we solved The demo is the truest form of status and therefore a great form of governance. (Watermelon reporting) Talk about what else was worked on (and maybe remains unfinished) Discuss upcoming iterations Know your audience … Don't kill ppl with technical details. Focus on the value When: At the end of each sprint

What is a Potentially Shippable Product Increment?

- All tasks are done for all stories

Who attends?

- Product Owner lead
- Delivery Team & Scrum Master
- Stakeholders / Customers
- Anyone else who is interested

What's the Advantage?

- Early feedback
- Able to react to unforeseen changes
- Team motivation
- Measure actual progress
- Reduce risks

Retrospective

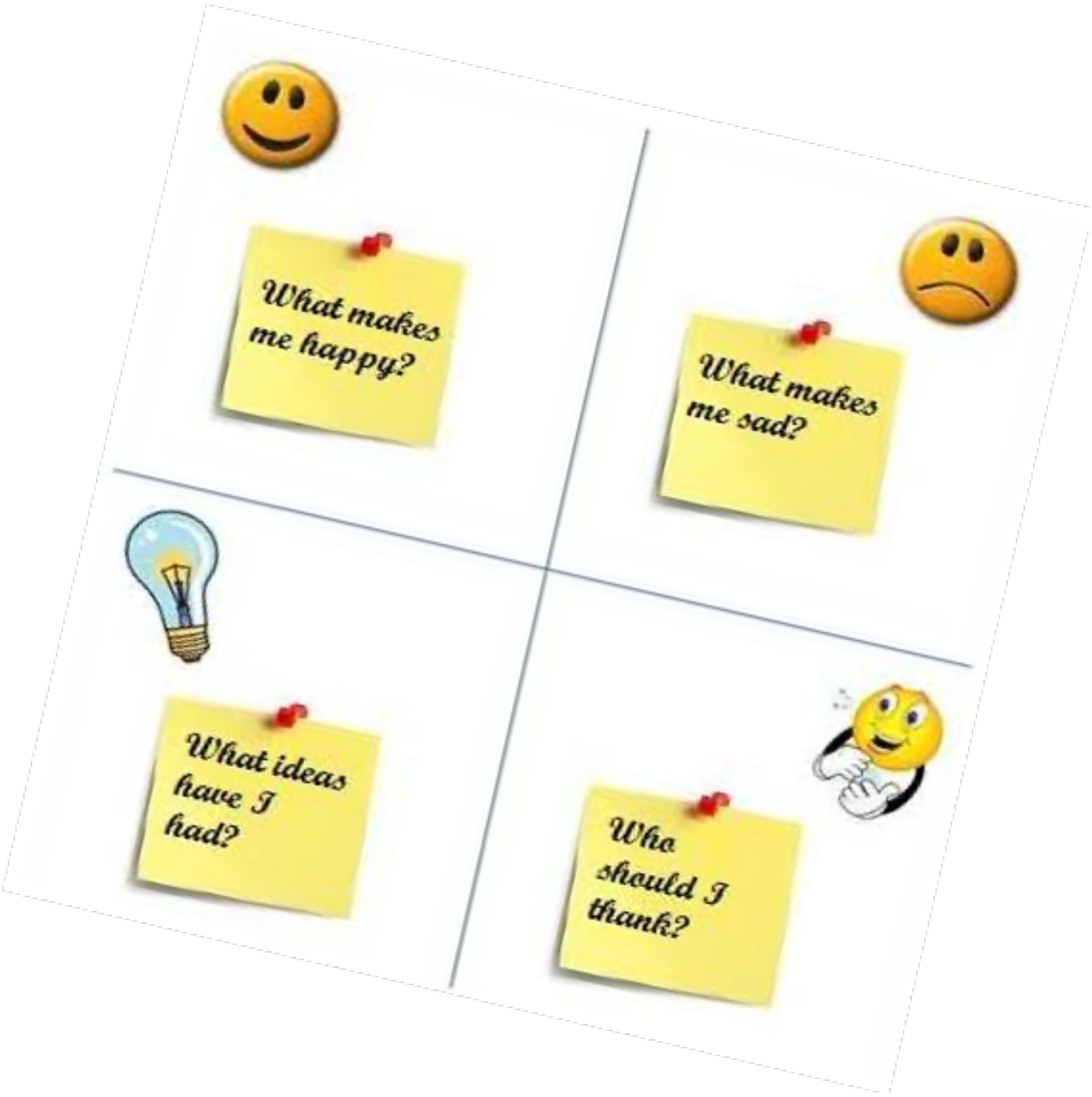
Allocate Time for Continuous Improvement

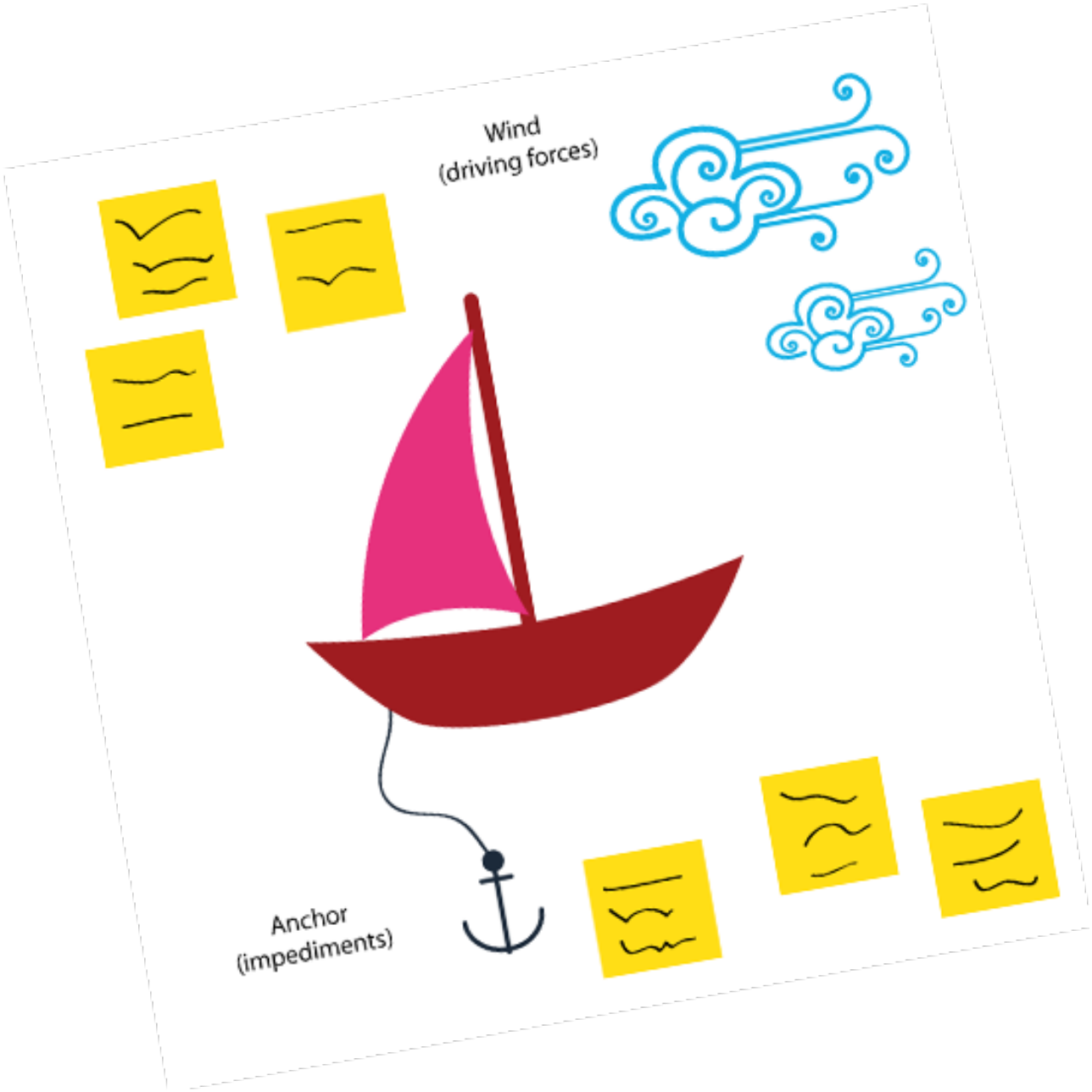
- Occurs after the Sprint Review and prior to the next Sprint Planning.
- Scrum master facilitates
- The goal is to identify one improvement in the team’s process that they can try.
- Review any experiments from last sprint

Reflect on the Past Sprint

- What went well - (What should we keep doing)
- What did not
- What can we do to improve

Collect Data

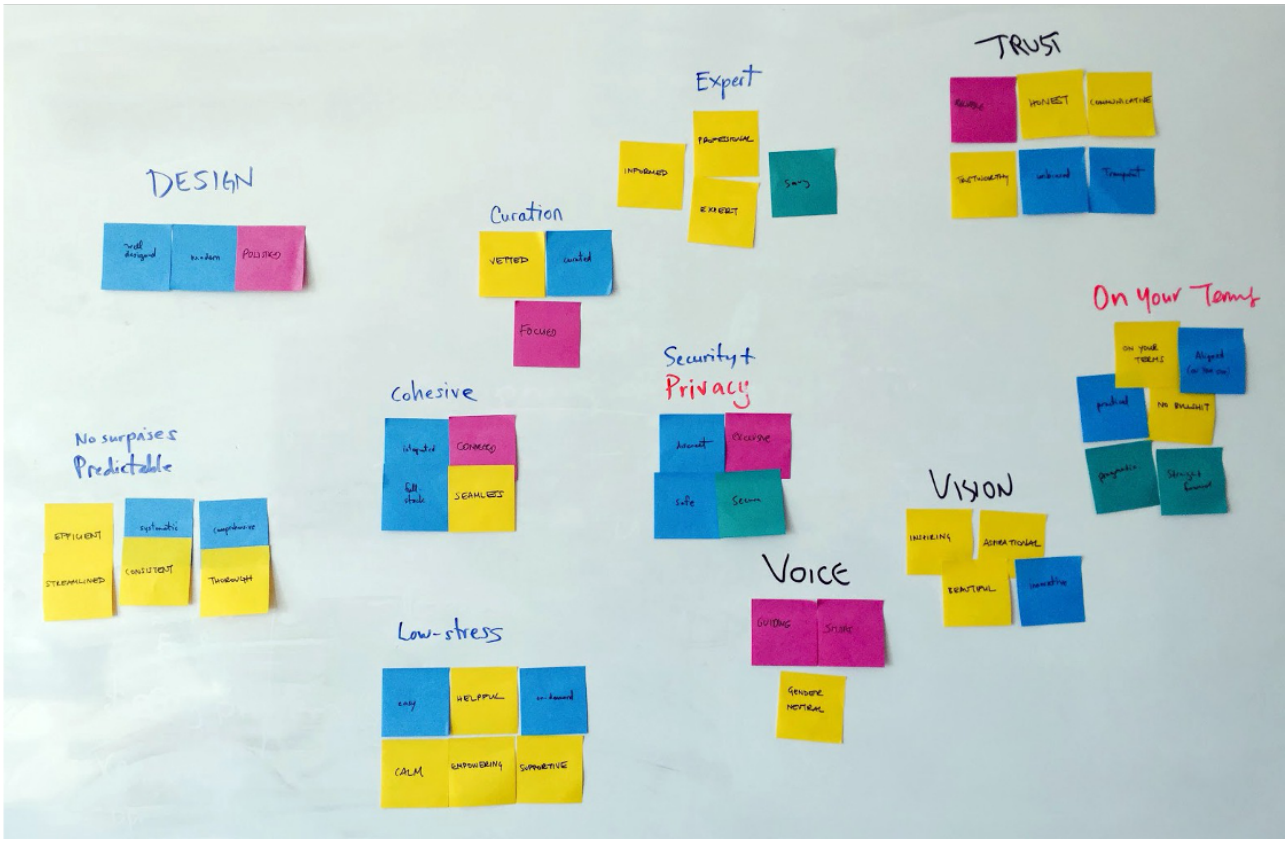

What makes me happy?
What makes me sad?
What ideas have I had?
Who should I thank?



Templates for Running a Retrospective

https://www.atlassian.com/blog/jira-software/5-fun-sprint-retrospective-ideas-templates

Theme and Vote



Design an Experiment

Hypothesis:
By including a translation person on the team, we will reduce the amount of items that are "almost completed".

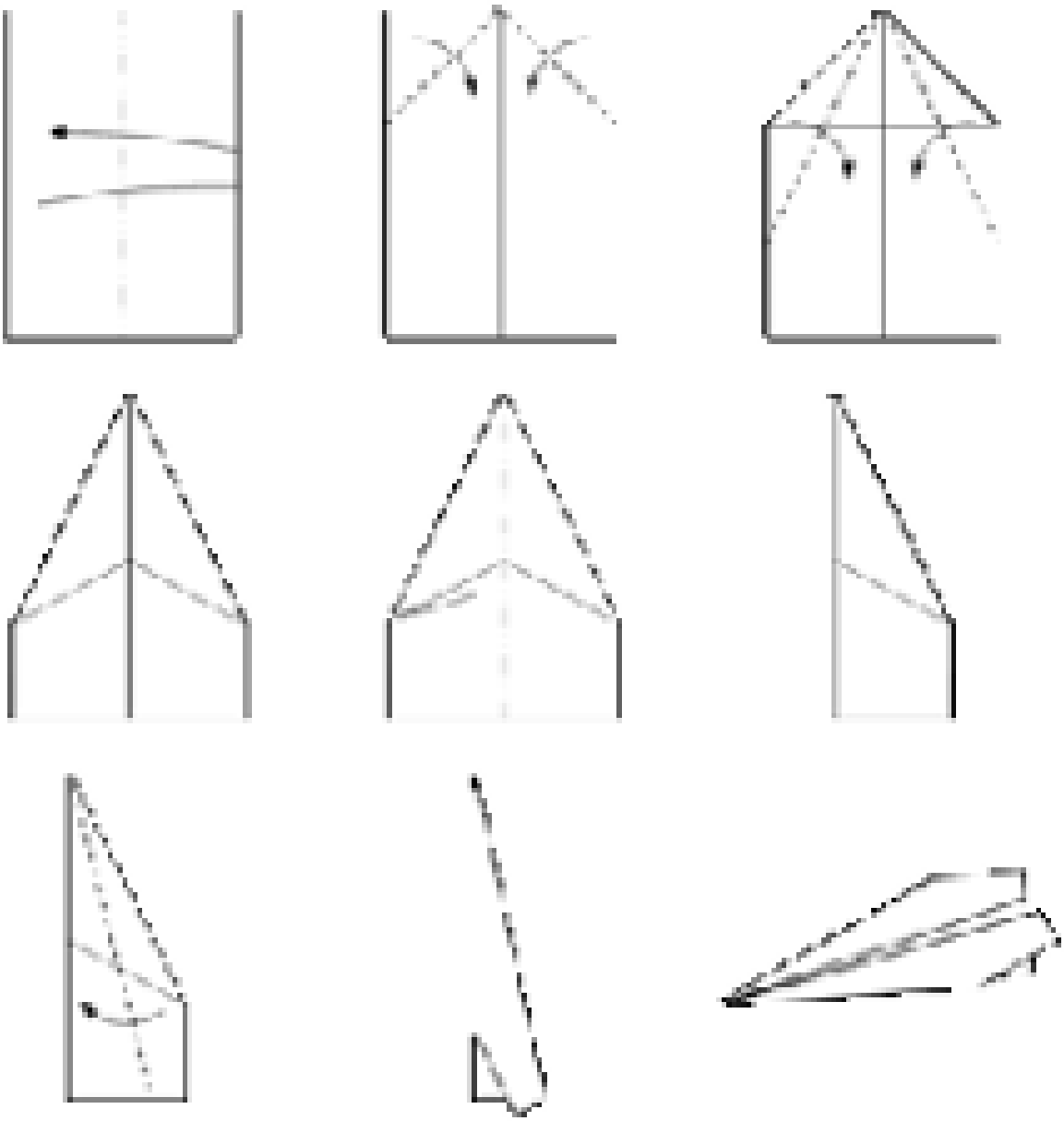
Retrospective Ground Rules

- Everyone participates & pays attention
- Be honest, constructive, professional, and respectful (don’t get personal or finger point)
- Focus on what’s relevant for the team
- Be positive and enthusiastic – you’re given the opportunity to improve!

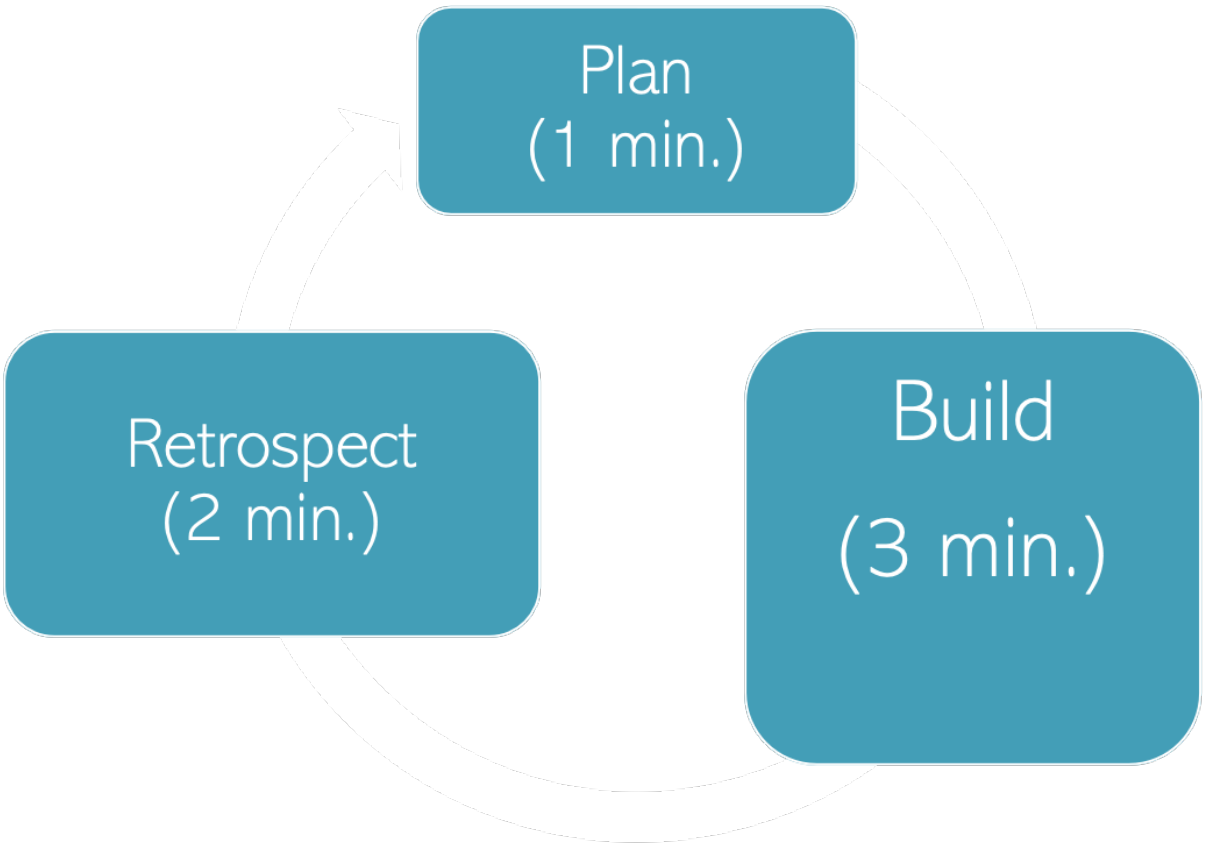
Building Paper Airplanes

Rules and Constraints

- ¼ of a sheet of paper (paper must be cut in 4)
- Only one fold at a time per team member
- Only the Product Owner tests
- Airplane must fly 3 meters by aerodynamic lift
- Airplanes may only be tested once
- Report results at the end of each iteration
- Airplanes built
- Airplanes tested successfully
- Work in Progress (Airplanes not completed)
- Partially built planes must be discarded after each iteration



The Sprint



Activity key learnings: Taking time to retrospect allows us to inspect and adapt our processes frequently Small incremental changes over time add up to big results Making small changes allows us to fail and recover quickly

Debrief questions: What did you change after the 1st and 2nd rounds, did you focus on process or design? Were you happy with the results? Not many companies would encourage teams to take an hour every 2 weeks to review their processes – we're very lucky at FCC that this practice is supported and encouraged – take advantage of it.

Marshmallow Challenge

Marshmallow Challenge

▶ https://www.youtube.com/watch?v=H0_yKBitO8M *(YouTube video)*

Wrap-up

Review?

- It's not about being agile for the sake of being agile. It's about delighting customers.
- Work in progress (WIP) is a liability towards being agile.
- Agile teams aren't afraid of REWORK.
- Agile Teams are about delivering business value sooner and more frequently. This is not necessarily the most **efficient** way to deliver.
- Agile is not an absolute term, rather it is a relative term
- Agile isn't a thing we do, it's a mindset.
- Agile delivery minimizes Risk.

Any A-ha's or parking lot?

Feedback exercise

And Finally …