Agile: The Bigger Picture

Dr. Leigh Griffin
Engineering Manager,
WIT Summer School
31 May 2017

My Goals for today!

- Introduce you to a broader view of Agile & Scrum
  - It's a big topic, you won't come out of this as experts
  - You will hopefully come out of this with ideas you can bring to your future career as a Software Engineer
  - You will hopefully understand reasons why we try to be Agile practitioners
  - You will hopefully spot things that you aren't doing correctly right now which will help you become a better Engineer
- More importantly, to have fun!

Exercise 1: Form an orderly line

How much do you know about Agile?

10 == Couldn’t be better; Agile all the time; living the dream!
1 == What’s an Agile?

You have 30 seconds to form an orderly line.

The end nearest the projector is the 1 side of the scale, the end nearest the door is the 10 side of the scale

Exercise 1: Form an orderly line

Reform the line based on how tall you are.

You have 30 seconds to form an orderly line.

The end nearest the projector is the small side of the scale, the end nearest the door is the tall side of the scale

Exercise 1: Form an orderly line

Reform the line based on shoe size.

You have 30 seconds to form an orderly line.

The end nearest the projector is the small side of the scale, the end nearest the door is the tall side of the scale

What did we learn?

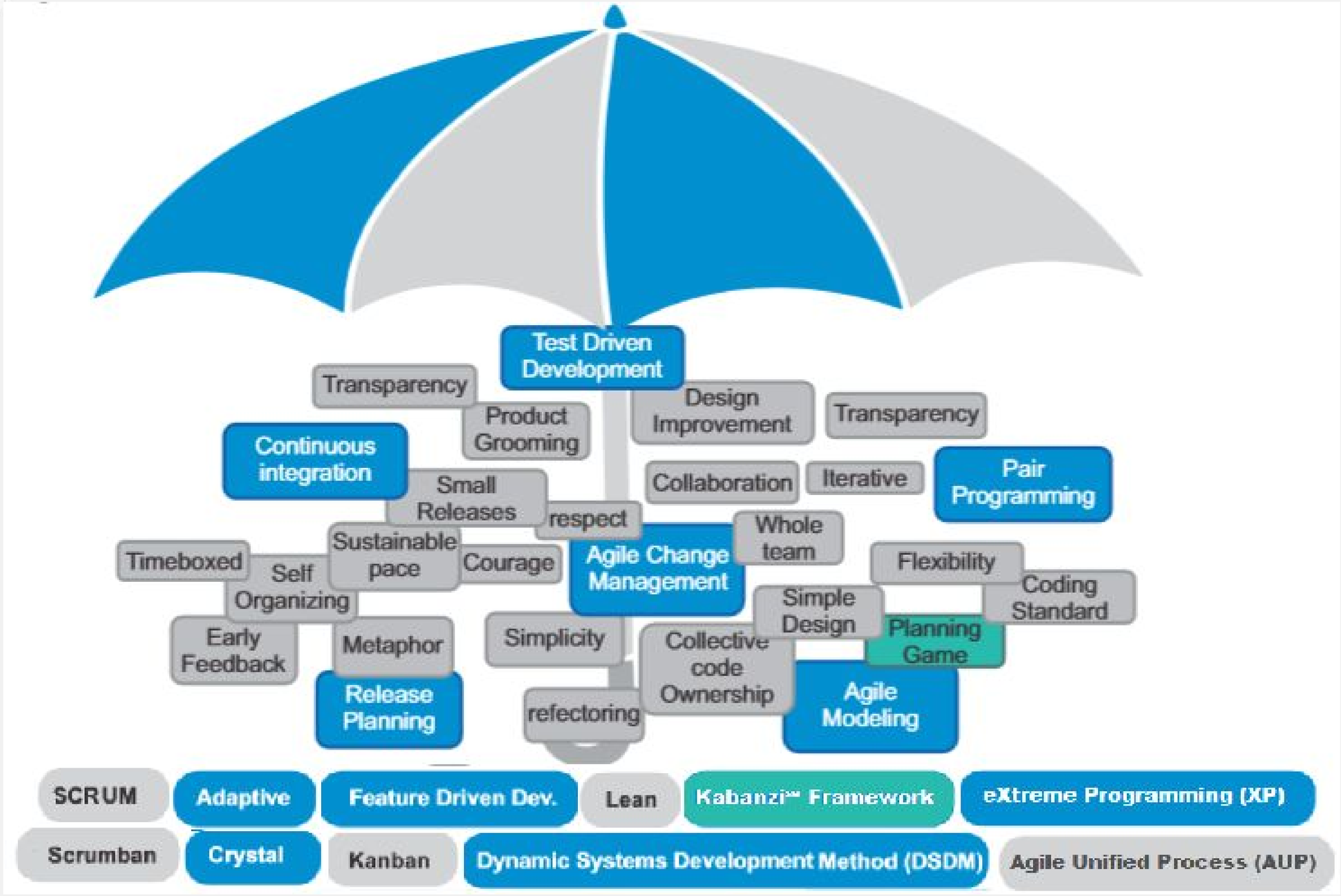
- The first exercise taught you about personal opinion, you formed your own thought and opinion and knew where you stood. **Individuals ideas and thoughts matter in Agile teams**
- The second exercise taught you about observation, you didn't need to communicate with your colleagues you could look and guess and fall in line silently. **Awareness of what others are doing is an important part of a functioning team.**
- The third exercise taught you about communication, you had to talk with your colleagues, listen and interpret. **Effective communication is the key to any good team**
- **These are not Agile principles but they are close! They are good team principles and Agile helps reinforce them**

The Agile Manifesto

- Individuals and interactions over processes and tools
- Working deliverables over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
- There is value in the items on the right, but we place more value on the items on the left
- http://www.agilemanifesto.org/ (2001)

Agile has many implementations

redhat.


Test Driven Development
Transparency
Product Grooming
Design Improvement
Transparency
Continuous integration
Small Releases
Collaboration
Iterative
Pair Programming
respect
Sustainable pace
Whole team
Timeboxed
Self Organizing
Courage
Agile Change Management
Flexibility
Coding Standard
Simple Design
Early Feedback
Metaphor
Simplicity
Collective code Ownership
Planning Game
Release Planning
refectoring
Agile Modeling
SCRUM
Adaptive
Feature Driven Dev.
Lean
Kabanzi™ Framework
eXtreme Programming (XP)
Scrumban
Crystal
Kanban
Dynamic Systems Development Method (DSDM)
Agile Unified Process (AUP)

The many faces of Agile : LEAN

- Eliminate Waste
  - Promotes efficiency and money saving
- Amplify Learning
- Deliver as Fast as Possible
- Decide as Late as Possible
  - Keeping options open, every voice and opinion matters
- Empower the Team
- Build integrity in
- See the whole

The many faces of Agile : Kanban

- Japanese for Signboard
  - Very visual as a result
- Does not prescribe roles or process steps
  - Respects the current titles and processes
  - Gains support and confidence, reduces fear of change
- Incremental, evolutionary change
  - Continuous small changes
- Limits work in progress
- Improve Collaboratively
  - Work as a team, very empirical and data driven

The many faces of Agile : Scrum

- Highest Business Value in the Shortest Time Possible
- Delivers value in short, repeatable processes
- Business sets the priority
  - With help from the team
- Role and Ceremony Driven
  - Roles help the process flow and unblock the team
  - Ceremonies focus the team and improve collaboration
- Teams self organise around work
  - No micro management
  - Anybody can work on anything
- Team succeeds and fails together

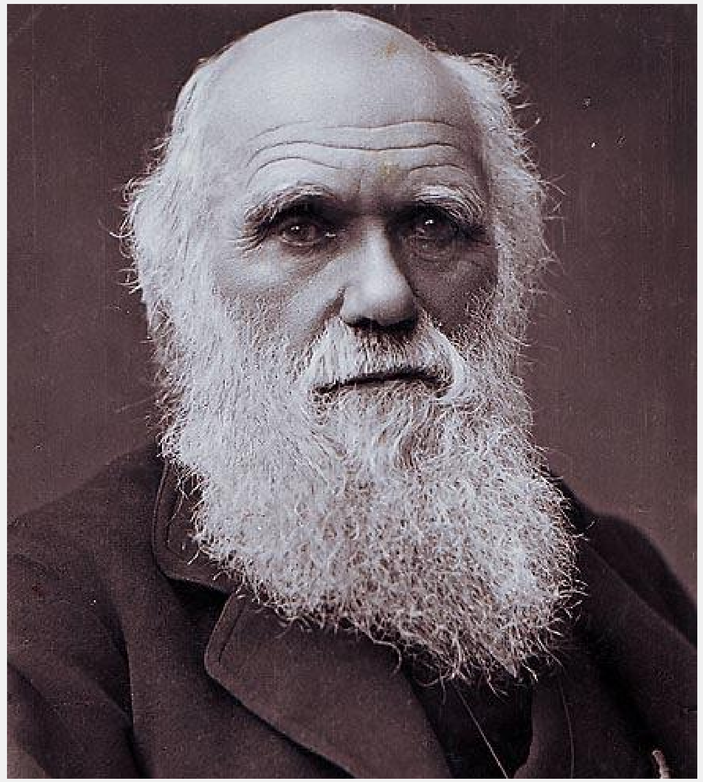
The many faces of Agile : What they Share

- Planning
  - Focused on small areas at a time
  - Delivering real value
- Discovery and Empowerment
  - Every voice in the team matters
  - No one person dominates what we do as an Agile Team
- Delivery
  - Short term, frequent deliveries
  - Long term, a more accurate delivery


A ~~Scrum~~ Agile Approach

What is Agile?

"It is not the strongest of the species that survives, nor the most intelligent that survives.
It is the one that is most adaptable to change" -- Misquote attributed to Charles Darwin (who is NOT a Software Engineer)



- Agile originally comes from Software Development.
- *We are uncovering better ways of developing software by doing it and helping others do it.*
- It was a response to how the industry was moving
- It was a response to how the customers perception of consuming technology was changing
- It is part of an Umbrella of other process improvement methodologies

A product and team vision

- At it’s core Agile Teams work towards a vision around a specific goal
- Our paying customer can help set the vision for what they want from the product or service
- Our company sets its own Vision, Red Hat is very strong on vision and principles
- Agile teams can set their own vision
  - To be the best? Most efficient? Reliable?
- As team members, we should be defining a Vision for our teams
  - It has to align with the wider stakeholder visions
  - Bring this back to your teams at your next Retro

A vision becomes a Backlog

- Visions are really implemented at a project level
  - Some piece of software we are developing
  - Some future goal for the platform which we build towards
- Every project has some form of requirements
  - A feature set
  - A goal or objective
  - Tying the requirements tangibly to our Teams Vision is an important step, it can inflate the requirements but for the better
- A Backlog can thus be defined as a “Queue of work”

A Backlog becomes a queue of Work

- The Backlog represents all of the steps that need to be taken by the team
- The team focus on a 1 at a time principle
  - More efficient
  - Focuses hugely on collaboration
  - Breaks down silos of responsibility
  - Stops the misperception that you can work 50:50 on two things
- The queue of work becomes an evolvable priority queue
  - Things change every day in our industry
  - We need to be reactive in a short time frame
  - We need to plan for short cycles

A queue of work becomes a Sprint

- Short cycles in an Agile context are called a Sprint
- A Sprint is a timeboxed effort with an agreed thematic goal
  - Minimum of 1 week duration
  - Maximum of 30 day duration
  - Our teams settled on 2 week Sprints initially
    - Just moved to 3 weeks to trial it out
- The goal of the Sprint is to produce something of value for someone
  - Very subjective what that value is to various stakeholders
  - As a team, we should communicate and agree on the goal before we start
    - Do NOT allow a Sprint start if you are honestly unsure of the goal

A Sprint needs Planning

- Knowing what our next feature to focus on allows us to plan what work will be required to get this “done”

- As a team we invested time into our “Definition of Done”
  - We are done with this work when….peer reviewed? Deployed? tested? looks good?
  - As Scrum Team members, you should encourage your team to define or enforce when work is Done
  - It sets a quality benchmark, improves team communication and confidence

- Knowing how much work is involved to get something Done allows us to enable a Planning Session
  - Our definition of Done, combined with the complexity of the work at hand
  - Means we can break a Feature request into workable User Stories

Planning needs Stories

- A User Story is used to interpret requirements and is customer focused
- As a User, I want <Something> to allow me some <Benefit>
- This helps capture the needs, wants and desires related to the Feature
- As a team we can break this down into actionable tasks
  - Tasks should adhere to the teams definition of done
  - Tasks should be small and independent of each other
  - Allowing multiple team members progress them accordingly
  - Who works on what is up to the team -- we self organise
- As a team we assign a Story Point to each User Story

Stories need Estimates

- Story Points are imaginary numbers :)
- They are a made up scale that the team uses to guess at the Complexity of the task at hand.
- We do not use time because of the natural bias we have for it
- We follow a loose Fibonnaci scale instead
  - 0,1,2,3,5,8,13,21,34,50,100
  - As a team we set a baseline, if we know what a 1, 2 and 3 are
  - We can guess the complexity in a compare and contrast manner
- As time goes on the team gets very strong at this, leading to accurate guesstimates
- Our estimates feed into our Velocity

Velocity leads to stronger Planning

- Each Sprint the team take on X number of Story Points
- At the end of the Sprint, the number of tasks “Done” are added up
  - 99% Done =/= Done
- That becomes the team's pace or velocity
- We can count up the remaining Story Points on the backlog
- Total Story Points / Team Velocity == Our completion date
- The team is said to burn down (or burn up) the Backlog
  - Great sense of focus and accomplishment

Strong Planning Leads to Successful Sprints

- After each Sprint the team will hopefully have something of value
- They can demo this to the team
  - Great chance to self learn within the team
  - Great chance to share work with the wider team
- Demos lead to much more feedback and allow a chance of direction
  - Reality Vs a Planning Document often differ wildly
  - A chance to reflect and refocus if necessary
- Successful or not, the Team learned something

Consecutive Sprints lead to a new way of working

- After each Sprint we hold a retrospective
- A critical analysis of how we worked
- At it’s simplest
  - What worked well
  - What didn’t work so well
- The team self manage this process
- End result is a team flavoured version of Agile that could be very unique
  - But very powerful

Predictive

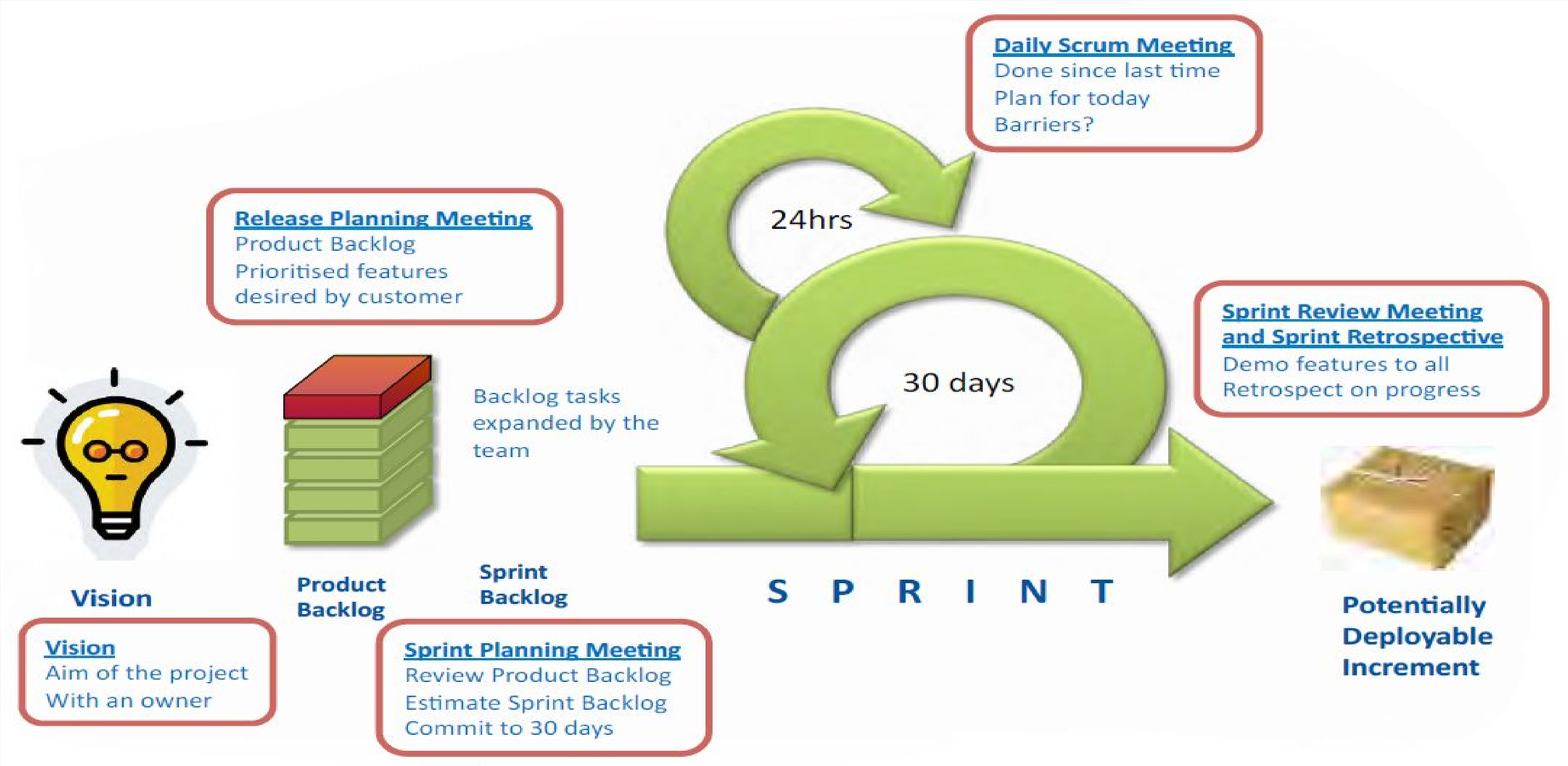
Aim Aim Fire!

Vs

Empirical

Fire Aim Aim!

Mini High Level Summary



Agile as a Team

redhat.

At the heart of Agile is the team

- At every opportunity we want to empower our team members
- The right idea always wins -- the meritocracy that Red Hat values
- We actively avoid telling our teams HOW they should do something
- Why bother hiring intelligent people to tell them how to do their job?

The team succeeds and fails together

- Celebrate good ideas within the team
- Empower the team to come up with better ways of doing things
- Be open to criticism, it should be constructive and designed to help the team
- Sometimes we can allow the team to fail, identify why the failed and let them identify how to recover from it and not encounter it again
  - Obviously within reason!!!

The team self organises and self polices

- We always make sure the team knows WHAT it is working on and WHY
- However, the team self organises to ensure that work gets Done. The team always know when an item is Done
- We have a Definition of Done within the team, which the team polices and self monitors and self enforces
  - Is it done when it is peer reviewed?
  - Is it done when it is deployed / released / shared?
  - It is done when the right people know about it?
- This ensures a quality focus within the team and the team drive this

As a team member you need to enable this

- Your support and actions will help make or break the team
  - Everybody counts in a team
- Be an enabler to the team, not a blocker
  - Proactively use your skillset
  - Help unblock colleagues
  - Help bring blockers to others attention, don’t sit silently
- Be supportive and change your default response to be more considerate
  - This is hard, we are a negative species by design!

Exercise: Negative & Positive Responses

- Pair up with the person next to you
- Your job is to plan the Christmas Party for the team for next year
- One person takes the role of manager, the other takes the role of planner
- Managers:
  - Every answer must start with “Yes, but….”
- Do this for 60 seconds…..GO
- Switch roles, now the Manager must start every answer with “Yes, and….”

Exercise: Thoughts?

- Very difficult when you are being stonewalled by someone
- Increases the frustration over time
- You probably want to stop coming up with ideas at that point right?
- The “yes, and” is not necessarily a confirmation
  - The “yes” is often forgotten because the conversation has pivoted
  - “And have you thought of this?”
- Bring that style into your Pull Request reviews
  - Socratic questioning to get more information and rationale

A quick focus on Story Points

redhat.

Why we should be using User Stories

- “Customer” Focused
  - Real Paying Customers
  - Operations
  - Consultancy
  - Engineering
  - Any other stakeholder who might want to get involved

- They express a need, want or desire as well as a situation, motivation and outcome

- They help focus WHAT we are doing and WHY we are doing it


GOOD USER STORIES
INDEPENDENT
NEGOTIABLE
VALUABLE
ESTIMA(TA)BLE
SMALL (SIZED APPROPRIATELY)
TESTABLE

INVEST - Independent

- Where possible it should be:
  - Free of dependencies on other stories
  - Self Contained
  - Capable of being completed in any order

- Watch out for:
  - Blocked dependencies on someone else external to the team
  - Writing Stories as a flow of one task into the next

I**N**VEST - **N**egotiable

- Avoid too much detail initially
- Allow room for improvisation on implementation i.e. the HOW
- Changes can be made on backlog refinement / sprint planning
- Watch out for:
  - Going too deep on requirements
  - Describing how the work is done instead of what the end result looks like

IN**V**EST - **V**aluable

- Why are we doing this again?
- What are we doing exactly?
- Product Owner / Customers need to set that
- Watch out for:
  - Writing from the perspective of the person doing the work
  - Describing work instead of an outcome

INVEST - Estimable

- Clearly Understood by all stakeholders
- Contains enough detail to allow an estimate emerge
- Watch out for:
  - Stories that are too large -- split them!
  - Missing important details or context e.g. from the Definition of Done perspective

INVEST - Small

- Needs to be completed in one sprint iteration

- Watch out for:
  - If it was contentious to estimate it is a good indicator that it is possibly too big
  - If it is difficult to understand it is probably too big

INVEST - Testable

- Testing is a first class citizen we need to be able to validate this
- Have we outlined acceptance criteria?
- How will we know that we are done?
- Watch out for:
  - Lack of obvious evidence to say it is done
  - Intangible work that can only be described and not measured

Story Points

- Our imaginary number scale is important, it helps size our work
- It allows us speak the same language across teams
- Consistent sizing means we can plan with reasonable accuracy
- We base our scale off of known values
- We guess on a ticket based on the information there
  - Always ask for clarity on the ticket
  - Don’t take for granted and certainly don’t assume

Story Point Exercise

- Go back to your Christmas Party teams

- What are the Relative weights of the following zoo animals 1,2,3,5,8,13,21,50 is our scale
- Giraffe
- Bear
- Kangaroo
- Rhinoceros
- Lion
- Gorilla
- Hippopotamus
- Tiger

Story Point Exercise

- Go back to your party teams

- What are the Relative weights of the following zoo animals: 1,2,3,5,8,13,21,50 is our scale
- Giraffe
- Bear
- Kangaroo
- Rhinoceros
- Lion - 2
- Gorilla
- Hippopotamus
- Tiger

Story Point Exercise

- Go back to your party teams

- What are the Relative weights of the following zoo animals 1,2,3,5,8,13,21,50 is our scale
- Giraffe - 5
- Bear - 2
- Kangaroo - 1
- Rhinoceros - 13
- Lion - 2
- Gorilla - 1
- Hippopotamus - 13
- Tiger - 2

Story Points: Final Thoughts

- Just because a team historically knows what a 2 is ; don’t feel that you have to accept it without you knowing what it really is
  - My Lion was an African female lion
  - My Hippo could have been a baby hippo

- Ask the relevant questions to the Product Team, the Technical Lead, your manager, your lecturer or your colleagues
  - Don’t assume that a request is straightforward
  - We are a multi lingual company, translations and interpretation mistakes can happen
  - We also have a huge differential in skill and domain expertise

Agile & Scrum is an iceberg

- Formal, paid for training for:
  - ScrumMaster -- 2 days
  - Product Owner -- 2 days
  - Scrum Team Developer -- 3 days
- Real word experience takes months and years to accumulate
- I can only scratch the surface in ~2 hours
- Any and all questions based on anything I went through?


Question Time

redhat.

lgriffin@redhat.com

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos