Agile

Agenda:

- Agile Background
- Terminologies
- Scrum Components
  - Roles, Artifacts and Events

Agile Background

Definition, Methodologies, Manifesto, Principles

AGILE DEFINED

To **move quickly** and **easily; flexible**

Ability to **create** and **respond to change**

Deal with and **succeed in** an *uncertain* and *turbulent environment*

QE-360

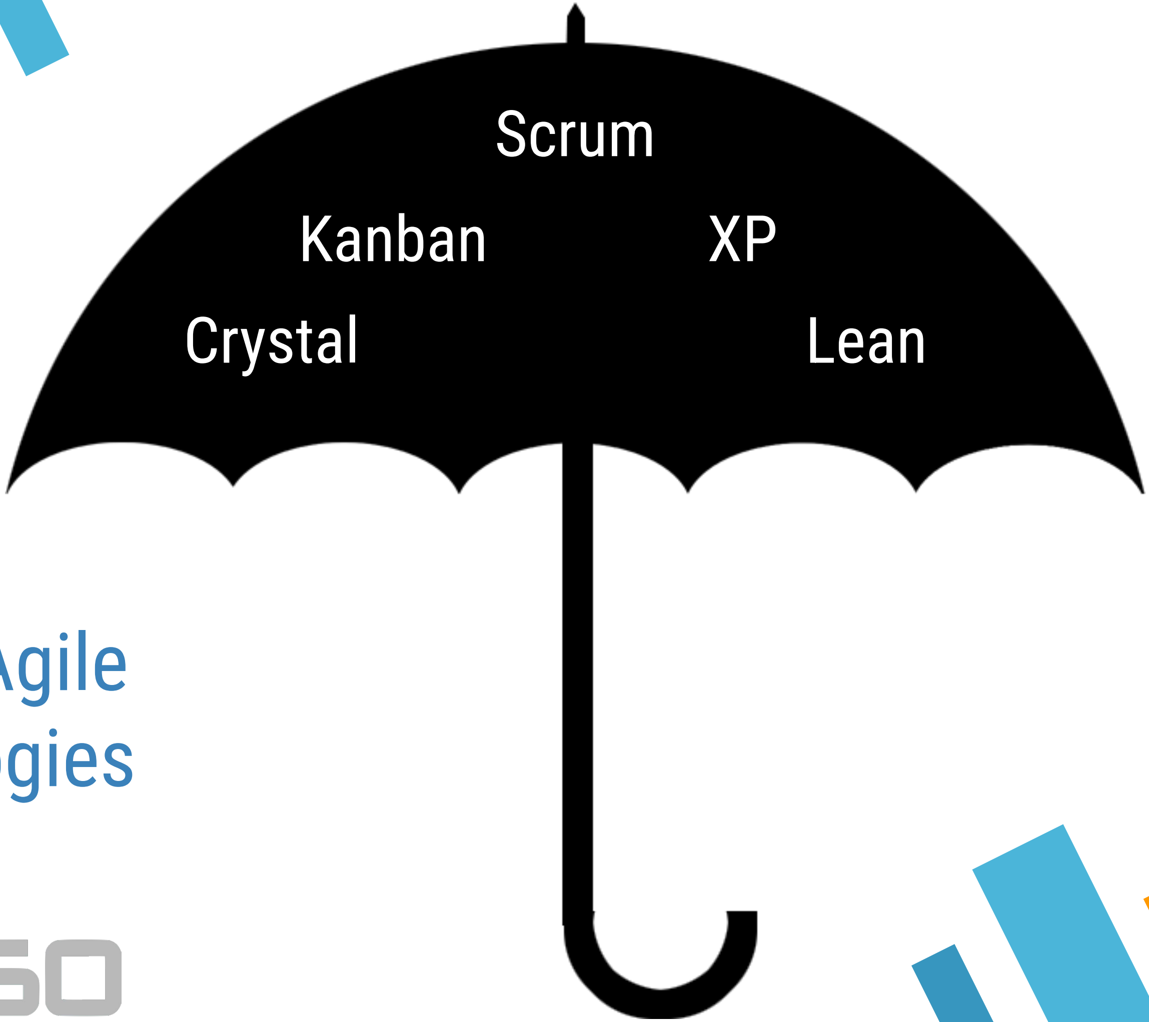
AGILE

Ability to **react well** to **changes** despite **uncertainties**

AGILE SOFTWARE DEVELOPMENT

A **MINDSET** formed by the values of the Agile Manifesto and 12 Principles

Agile Methodologies



Common Agile Methodologies

Agile Methodologies emphasized:

- **Close collaboration** - dev and biz
- **Frequent delivery** of business **value**
- **Tight, <u>self-organizing</u>** teams
- **Smart ways** to *craft, confirm and deliver code*
- **Inevitable requirements churn** *is not a crisis*

Agile Manifesto

Manifesto Authors

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
Robert C. Martin
Steve Mellor

Dave Thomas
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Ken Schwaber
Jeff Sutherland

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions

over processes and tools

Working software

over comprehensive documentation

Customer collaboration

over contract negotiation

Responding to change

over following a plan

That is, *while there is value* in the items on the *right*, we **value the items on the left more**.

12 Principles of Agile Software

We follow these principles:

Our <u>**highest priority**</u> is to **satisfy the customer** through **early and continuous delivery** of valuable software.

1

Welcome changing requirements, even late in development. Agile processes harness change for the **customer's competitive advantage**.

2

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

3

Business people and developers must **work together daily** throughout the project.

4

Build projects around **motivated** individuals.

Give them the **environment** and **support they need**, and **trust them** to get the job done.

5

The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

6

Working software is the primary measure of progress.

7

Agile processes promote **sustainable development**.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

8

<u>**Continuous**</u> attention to **technical excellence** and **good design** enhances agility.

9

Simplicity -- the art of **maximizing the amount of work not done** -- is essential.

The best architectures, requirements, and designs emerge from **self-organizing** teams.

At regular intervals, the **team reflects on how to become more effective**, then **tunes and adjusts** its behavior accordingly.

AGILE SOFTWARE DEVELOPMENT

A **MINDSET** formed by the values of the Agile Manifesto and 12 Principles

AGILE SOFTWARE DEVELOPMENT

To provide **guidance** on how to ***create*** and ***respond*** to *change* and ***deal*** with *uncertainty*

Terminologies

Scrum

A simple framework for effective team collaboration on complex projects

Simple to understand but difficult to master

Sprint

A timeboxed period (usually 2 weeks) when work is completed and made ready for review

Timebox

Approach that consists of **stopping work** when the *time limit is reached*

All scrum meetings are timeboxed

User Stories

Work to be done

- **Single piece of functionality that can be developed in a single sprint**

Basic unit of <u>communication</u>, <u>planning</u>, and <u>negotiation</u>

Epic

A **large user story** that CANNOT be delivered in a single sprint

Should be dissected into user stories

Should NOT be taken as is in the Sprint Backlog

Product Backlog

A prioritized to-do list (features) for the team

Definition of Done

Criteria wherein everyone is in agreement that something is ready to be released

Definition of Done Example

- Developed
- Unit Tested
- Code Comments
- Code Commit Messages
- Passed Code Review
- Passed Testing in Staging
- Passed demo to PMO, Tech Ops and PO

Definition of Done

- Code Commit Messages
- Passed Testing in Staging
- Critical to Medium bugs are closed
- Passed demo to PO / stakeholders

Definition of Ready

Criteria wherein everyone is in agreement that a User Story needs to meet before it’s ready for inclusion in the work of next sprint.

Scrum Theory

EMPIRICAL PROCESS CONTROL

- knowledge comes from experience
- Making decisions based on what is known

ITERATIVE, INCREMENTAL APPROACH

- repetitive rounds with increase in scope until the desired output is achieved

3 Pillars of Empirical Process Control

TRANSPARENCY

- Visibility to those responsible for the outcome
- Requires common understanding

INSPECTION

- Inspect artifacts and progress towards sprint goal

ADAPTION

- Inspector determines process deviation, adjustment must be made

Scrum is NOT

- Gantt charts and strict steps
- Knowing everything ahead of time

Why Scrum?

- Higher productivity
- Better-quality products
- Reduced time to market
- Improved stakeholder satisfaction
- Better team dynamics
- Happier employees

SCRUM COMPONENTS

I. TEAM ROLES

1. PRODUCT OWNER

PRODUCT OWNER

Responsible for RETURN OF INVESTMENT

Makes DECISIONS

Focuses on the WHAT Than the how

VOICE of the CUSTOMER

PO and the Product Backlog

- PO manages the Product Backlog (PB)
  - Decides what goes in the PB
  - Prioritizes PB
- PO ensures that there sufficient detail and that the team understands them

Common PO Pitfalls

- Unavailability of PO to make timely decisions
- Can’t make decisions on his own
- Spread too thinly
- PO working on too many products
- PO that is not co-located with team and too far to meet clients
- Several POs in one project

2. SCRUM MASTER

2. SCRUM MASTER

Facilitates PROCESS

COACH of the Scrum Team

Helps REMOVE OBSTACLES

CHECK AND BALANCE
of the team

ScrumMaster and the Scrum Process

- Arranges and facilitates team’s meetings and ensures they’re attended
- ScrumMaster has no authority over the team but can make process changes
  - Format of retro, daily stand up, or length of sprints
  - Remind team of the process
- Can’t set goals for the team but ensures everyone agrees to the goals they’ve set

ScrumMaster and the Organization

- Helps in adoption of Scrum
- Plans of Scrum implementations
- Shares awareness of what the Scrum team is doing
- Protects Scrum team from impediments
  - E.g. Moving team members to different projects
  - Having too many projects for a Scrum Team / Member

ScrumMaster and the Organization

Improves interactions between Scrum team and the organization to maximize the team's productivity

ScrumMaster and the Product Owner

- Is the Product Backlog ready for Grooming?
- Makes sure the Scrum team understands the need for clear and concise Product Backlog Items
- Reviews the clarity and size of the user stories, and no dependencies from other user stories within the same or future sprints

Common ScrumMaster Pitfalls

- Multi-tasking ScrumMaster
  - After contributing to the work, there's no more time to carry out their main task
- Doesn't know how to remind/correct wrong practices
- Is new to Scrum yet assigned to Scrum
- PM turned ScrumMaster
- Can't read the tension between team
- Can't protect team from constant work overload

3. SCRUM TEAM

3. SCRUM TEAM

CROSS-FUNCTIONAL

ATTEMPTS TO BUILD a "potentially" shippable product

COLLABORATES

SELF-ORGANIZES

Scrum Team in Practice

- Interacts a lot during sprint events
  - Dynamics established
  - **Self-organization** builds up
  - Improves delivery rate over time
  - Lessens mistake
- Should not be too large
  - Everyone should be physically and mentally “present”
- If the need arises, Scrum of Scrum

Common Scrum Team Pitfalls

- Lack of interaction, participation and attendance in meetings, unprepared in meetings
- Discarding Demos / Retrospectives
- Cherry-picking of tasks/user stories
- Team doesn’t digest their data
  - Agile Board, Burndown chart, Sprint health, Bugs
- Lack of communication / interaction
- Geographically distributed teams
- Demotivated members

1. Everyone is part of one big team
2. Each ball must have air-time
3. Each ball must be touched at least once
4. Balls cannot be passed to your direct neighbor to your immediate left or right
5. Each ball must return to the same person who introduced it into the system

1. Initial planning is 2 minutes - think of the fastest way to deliver
2. Exactly after the timebox of the planning, you are given 1 minute to start the Ball Flow game
3. After 1 minute execution, reassess to improve the time of the delivery
4. 3 rounds

Exercise: Who Is It?

Identify The Role

II. ARTIFACTS

1. PRODUCT BACKLOG

PRODUCT BACKLOG

A. Epics
B. User Stories
C. Bugs
D. Production Incidents
E. Spikes

Epics

Large User Stories that can’t be delivered within a single sprint

Epics

- Keep track of large, loosely defined ideas in the backlog
- Placeholder for a vague idea until things become clearer
- Are not for estimation but should be broken down to user stories

Epic Examples

1. Post on your wall
2. Comment on a post
3. Update profile picture

Epic Info Needed

1. **Epic Name** - one word that best describes the epic
2. **Summary** - statement to describe the Epic
3. **Description** - placeholder for any details you may come up with for the Epic, potential stories
4. **Component/s** - modules that will be impacted by the epic

Exercise: Create Epics

Create epics based on your product

USER STORY

User Stories

Work to be done

- **Single piece of functionality that can be developed in a single sprint**

Basic unit of <u>communication</u>, <u>planning</u>, and <u>negotiation</u>

Who IS THE ONLY PERSON who can write user stories?

PO, ScrumMaster and Scrum Team Members

But the bulk of the User Stories in the Product Backlog is the responsibility of the?

PRODUCT OWNER

A User Story is READY if it INVESTs

INVEST

Independent (of all others)
Negotiable (not a contract for a specific feature)
Valuable (adds value to the customers/business)
Estimable (clear enough that effort needed can be estimated)
Small (that you can fit several in a sprint)
Testable (can be validated)

Exercise: User Stories

User Story Format

- User Story Name
- User Story Narrative
- Acceptance Criteria

Story Narrative Format

```
As a <user>
I can <functionality>
so that <benefit>
```

Story Narrative Examples

```
As a <chef>
I want a <knife holder>
so that I can <store & access knives>
```

Story Narrative Examples

```
As an <FA>
I want to <record videos>
so that I can <have proof of the stocks>
```

Acceptance Criteria

Conditions to be satisfied for the user story to be deemed complete

Acceptance Criteria Examples

- The record button is at the chat screen
- The record button is displayed as pencil icon as a disguise
- Recording is saved on the device

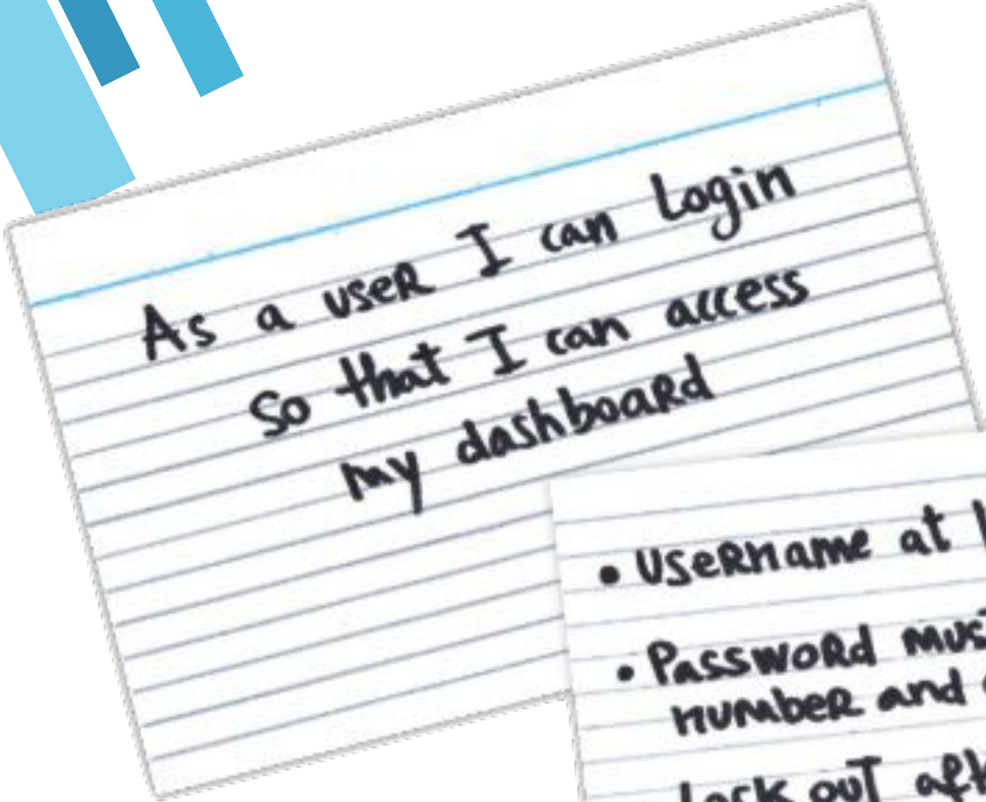
Record View for FA

```
As an <FA>
I want to <record videos>
so that I can <have proof of the stocks>
```

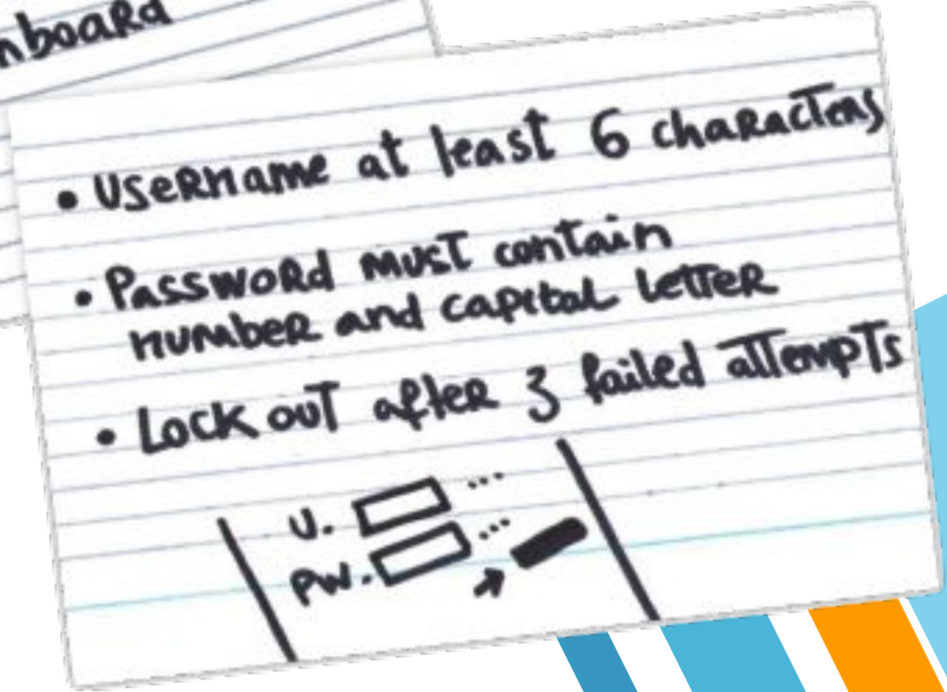
- The record button is at the chat screen
- The record button is displayed as pencil icon as a disguise
- Recording is saved on the device

No editing, no renaming of the recording yet, but that can be another story...
BUT does it INVESTs?

How much details?



As a user I can login
So that I can access
my dashboard



- Username at least 6 characters
- Password must contain number and capital letter
- Lock out after 3 failed attempts

https://manifesto.co.uk/agile-concepts-user-stories/

- Usually written in index cards / post-its
- With minimum amount of detail necessary to encapsulate the value of the feature
- Specifications that arise during the conversation should be in the Acceptance Criteria - written at the back

DISSECTING EPICS INTO STORIES

Dissecting Epics

Done on epics so that it can be user stories and be included in sprint/s

Frosting - front-end

Mango / Cashew - Security

Wafer - Database

Chiffon - API

Exercise: Splitting Epics and create User Stories

Slice one of the following FB Features and create them into User Stories:

1. Post on your wall
2. Comment on a post
3. Update profile picture

2. SPRINT BACKLOG

SPRINT BACKLOG

A subset of the product backlog that team **targets** to deliver in the sprint

SPRINT BACKLOG EXAMPLE

- Critical / High Sev. Prod. Incidents
- User Stories
- Bugs
- Urgent Feature Requests

SPRINT BACKLOG

- Taken from the top of the Product Backlog
- Targets for the sprint/s
- Team agreed to take in their sprint
- Owned by the Scrum Team*
- Has tasks - representing the HOW

3. INCREMENT

A.K.A. Potentially Shippable Increment (PSI)

INCREMENT

- Sum of all the Product Backlog items completed and all previous sprints
- Done according to team's DoD

III. SPRINT EVENTS

During the Sprint:

- ***No changes*** are made that would **<u>endanger</u>** the Sprint Goal
- **Quality** goals **<u>do not decrease</u>**
- **Scope** may be **<u>clarified and re-negotiated</u>** between the Product and the Scrum Team

Cancelling the Sprint:

- Sprint can be cancelled before the time-box is over
- Only **PO can cancel** or he/she is influenced by Stakeholders, Scrum Team or Scrum Master
- Cancel when goal becomes **obsolete**

1. PRODUCT BACKLOG REFINEMENT / GROOMING

PRODUCT BACKLOG REFINEMENT/ GROOMING

1. Clarification of requirements
2. Decomposition of large PBIs
3. Estimation of effort
4. Re-ordering PBIs if needed

Clarification of Requirements

- What are the acceptance criteria of the story?
- **Grooming** is the time for you to **THINK, ASK, SUGGEST** and **NEGOTIATE**
- Bring up any potential **RISK, DEPENDENCIES**

Splitting An Epic Into User Stories

Slice Horizontally

Cut into different components grouped by the type of work

- UI
- Backend
- Front-end

x Silos team into their specializations

x Waiting due to dependencies within the sprint and won’t be delivered

Slice Vertically

Slice through a user story so that it’s smaller but with value

- ✓ Provides value to the user
- ✓ Is testable
- ✓ Delivered faster

Approaches In Slicing User Stories

- Users
- Data
- Process
- Acceptance Criteria
- Performance
- Interface

Users

- Is the feature intended for several types of user?
- Can they have their own user story, starting with the most valuable?
  - US 1 : FAs can save videos
  - US 2 : Admin can extract all videos in one go

Data

- Does the feature handle several types of data? Can we restrict the data to a less complex initial story?
  - US 1 : Get the no. of product on the shelf
  - US 2 : Get the ratio of product of the shelf and the ones that are easily sold-out

Process

- Does it describe a process / workflow?
- Can we just deliver the first and last steps as a story? Steps in the middle to be additional stories?
- Or “Happy Path” first and alternative, exceptions, edge cases later?
    - US 1 : Filter report by date calendar picker
    - US 1 : Filter report by copy+pasting a date

Acceptance Criteria

- Can you leave out some acceptance criteria and make them a separate story?
  - US 1: Typing in existing Username and matching password allows the user to login
  - US 2: User is able to login if he previously logged in and checked “Remember me”

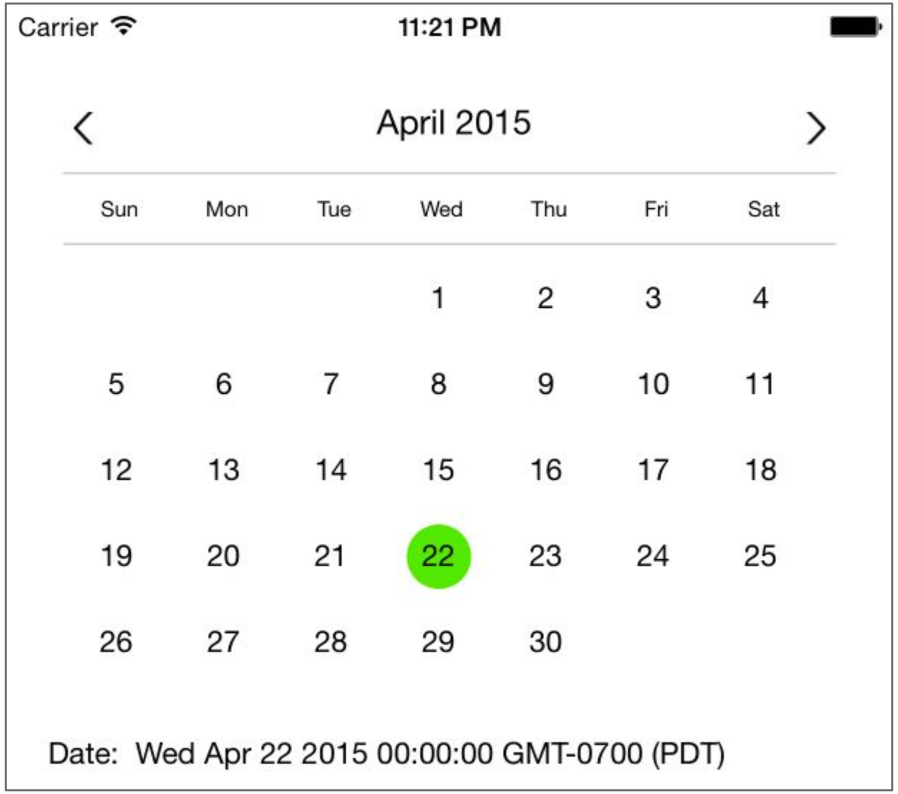
Performance

- Can the non-functional be a story but deferred from the functional story?
  - US 1 : Generate xyz report
  - US 2 : Make Generate xyz report done in 30 seconds

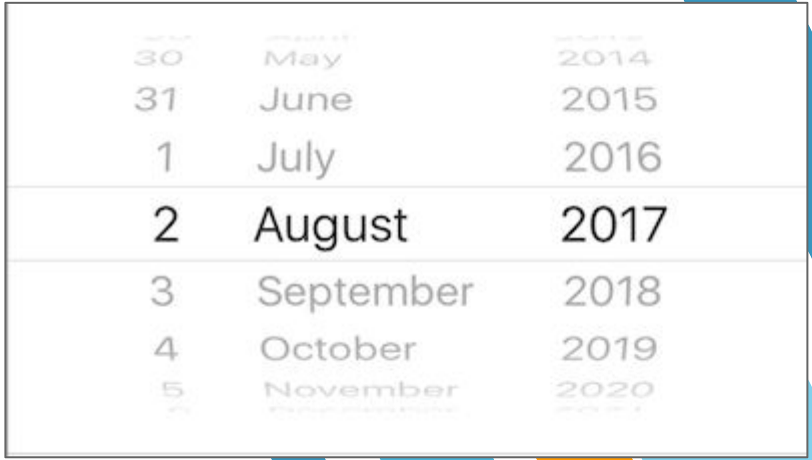
Interface

- Does is have a complex UI? Can we make it simple first?
- Does data from one page bring value? And integration to another page to follow?
  - US 1 : Birthdate to be a calendar picker
  - US 2 : Birthdate to be an inline date picker

Interface



Date: Wed Apr 22 2015 00:00:00 GMT-0700 (PDT)



User Story Estimation

- Estimate the **COMPLEXITY** of the User Story
- User Story is **CLEAR, no grey areas**
- Should have a **TEAM CONSENSUS**

User Story Reference

1. Find a User Story that will be your 3-story points story
2. Compare another user story and ask if it’s more complex or simpler that your 3-story point story

Estimation

- Identifying the user story’s complexity level
- T-shirt sizes (XS, S, M, L, XL) or
- Fibonacci Sequence (1, 2, 3, 5, 8, and so on)

  Fibonacci sequence adds the last two numbers in the sequence to get the next number.
- All scrum team members has a voice

QE-360

1. A team estimates together.
2. Product owners and Scrum masters should estimate with team.
3. Votes : Team members > PO or Scrum Master
4. When the product owner says an estimate should be higher, that’s odd. You are probably missing requirements.
5. When a team member says an estimate should be lower, that’s odd too. There are probably built-in functions you don’t know about. Discuss those.

Velocity

Measure of the rate at which work is completed per unit of time. Using Scrum, velocity is typically measured as the sum of the size estimates of the product backlog items that are completed in a sprint.

Velocity

Velocity measures output (the size of what was delivered), not outcome (the value of what was delivered).

Rules to estimate FAST

If the vote shows everyone is in agreement on size, you are done.

If the size is one Fibonacci category off (say 5 vs. 8), just pick the higher one.

If the estimate votes are two categories off (say 3 vs. 8), go with the middle number (5)

If the estimate is three or more categories off (say 3 vs. 13) proceed as with planning poker, and have the low and high advocates discuss why they are on such vastly different pages. Everyone else can chime in. Once the discussion is done, revote.

4

Work quickly; don’t try to be perfect. Close fast estimates are far better than slow, “more exact” ones.

5

Estimation: Story Points vs Time Estimates

User Story = Story Points

Fibonacci Sequence: 0, 1, 2, 3, 5, 8, 13, and so on

Sub-tasks = Task time estimates

30m, 45m, 1h, 2h, etc

Re-order PBIs when

- There were new user stories created during splitting of Epics
- Dependencies were identified
- There are gray areas for a User Story

Sprint Grooming:

Involved: PO, Scrum Team, ScrumMaster
Duration: 2 hours per 2-week sprint

PO discusses the stories, re-order
Scrum Team: clarifies, estimates complexity
ScrumMaster: assists team, facilitates estimation, process

2. SPRINT PLANNING

SPRINT PLANNING

Negotiation of sprint commitment
Phase 1 - The WHAT (User Stories) - PO
Phase 2 - The HOW (Tasks) - Scrum Team

Sub-Task Examples

1. API 1 Devt.
2. Web Devt.
3. API & FE Integration
4. * Test Creation
5. * Test Execution
6. * Demo

* Generic

Each sub-task requires one ticket

Sub-Task's Content:

Sub-task Name:
Description:
Time Estimate:

Sprint Planning:

Involved: PO, Scrum Team, ScrumMaster
Duration: Max of 4 hours per 2-week sprint

PO: **sets** the priority
Scrum Team: **commits** to what can be delivered in the sprint based on velocity, estimates time
SM: ensures consensus by the end of the meeting

Delivering the Sprint Backlog is your **Sprint Goal**…

So DON’T commit to something you **know you can’t deliver**

Exercise: Estimating Tasks

1. Create sub-tasks for the user stories your group has created
2. Estimate the sub-tasks

Sprint Goal

- Is a **one or two sentence that encapsulation** of what the team has agreed to achieve during the sprint
- Sprint goal is set by the **PO** and should encapsulate the bulk of work in the sprint *without requiring all stories to be completed for the goal to be reached*

3. DAILY SCRUM MEETING

A.K.A. Daily Stand Up Meeting

DAILY SCRUM MEETING

Shortest yet most important meeting
Done at the beginning of the working day
Around the Scrum Team's task board

It's not an update to the PO or SM, but a SYNCHRONIZATION between the whole team

Opportunity to SELF-ORGANIZE

DSM Format

1. Yesterday’s achievements
2. Today’s plan
    - If you have task/s that is a dependency of someone, tell them when to expect it
3. Blockers/impediments
    - Talk to only relevant people, take it offline
4. Are we **CONFIDENT IN MEETING OUR TARGET?**

TO DO	IN PROGRESS	FOR REVIEW	FOR TESTING	FOR DEMO	DONE

Daily Scrum Meeting

Involved: PO, Scrum Team, ScrumMaster
Duration: Max of 15 mins

Scrum Team: **team synchronizes**
PO: **clarifies** ambiguities, addresses
SM: helps **remove obstacles / impediments**

Exercise: Daily Stand-up Meeting

With your subtasks, put them in the different swimlanes and have a DSM

4. SPRINT REVIEW MEETING / DEMO

SPRINT REVIEW / DEMO

1. Keeps the stakeholders up to speed with progress
2. Gives immediate feedback
3. Amends the Product Backlog with new user stories

Demo Format

1. Done at the end of the sprint
2. If there’s multiple teams working on the project
   - Consider running their demo together
   - Removes redundancy of work
   - Facilitates sharing and collaboration
   - Can help address challenges met by the other team

Demo Format cont'd.

3. Mention the sprint goal
4. Demo the stories relevant to the sprint goal
   - Stories organized into a logical narrative sequence
   - Nominate a member to demo a story
   - Share the tasks, acceptance criteria and show the implementation

Preps for a Demo

1. No slide decks needed
2. Only demo stories that meet the DoD
3. Day before the Demo, team agrees on the
   - Order of stories and who demos it
   - Identify setup requirements

Finally...

1. Demo is not a sign-off
2. PO has already seen the stories before demo
3. Feedbacks turned User Stories are created
    - *It should not change what is considered done.*

Sprint Review / Demo:

Involved: PO, Scrum Team, SM, Stakeholders
Duration: Max of 2 hours per 2-week sprint

Stakeholder : **gives feedback**
PO : translates feedback to **User Stories**
Scrum Team: **demos**
SM: **facilitates** meeting

Exercise: Sprint Demo

Conduct a demo of one of the ff:

1. Post on your wall
2. Comment on a post
3. Update profile picture

5. SPRINT RETROSPECTIVE

SPRINT RETROSPECTIVE

- Agile way to get better in each iteration.
- An opportunity for the team to discuss **successes** and **failures**
- Did they meet their **sprint goal?**
- How can the **team improve next time?**
- Usually done right after the sprint demo.
- Speak freely and candidly about people, interactions, processes, impediments, tools

Retrospective Format

- Improvements team agreed to take on last sprint
- What went well?
- What went wrong?
- Did we meet our sprint goal?
- What are the 3 improvements are we committing to?

Retrospective Meeting

Involved: PO, Scrum Team, ScrumMaster
Duration: Max of 1 hour per 2-week sprint

PO: Listening / analyzing
Scrum Team: Gives input / explanation
SM: **asks** members for inputs/explanation

Retrospective Pitfalls

- Team focusing on the negative aspects
  - Counter-productive and would not build self-organizing teams
  - Blaming others
- No critical improvement item agreed upon by team
- Not learning from past retrospectives

Exercise: Sprint Retrospective

Conduct a retro mentioning the ff:

- Improvements team agreed to take on last sprint
- What went well?
- What went wrong?
- Did we meet our sprint goal?
- What are the 3 improvements are we committing to?

References

https://www.agilealliance.org

https://www.scrum.org

https://manifesto.co.uk/

https://www.scrum.org

https://blog.scottlogic.com/

https://www.kbp.media/