# Preparing to Iterate/Sprint

**Week 4**

# Taking Stock

The schedule for the course

Where are we now?

The Assessment Schedule

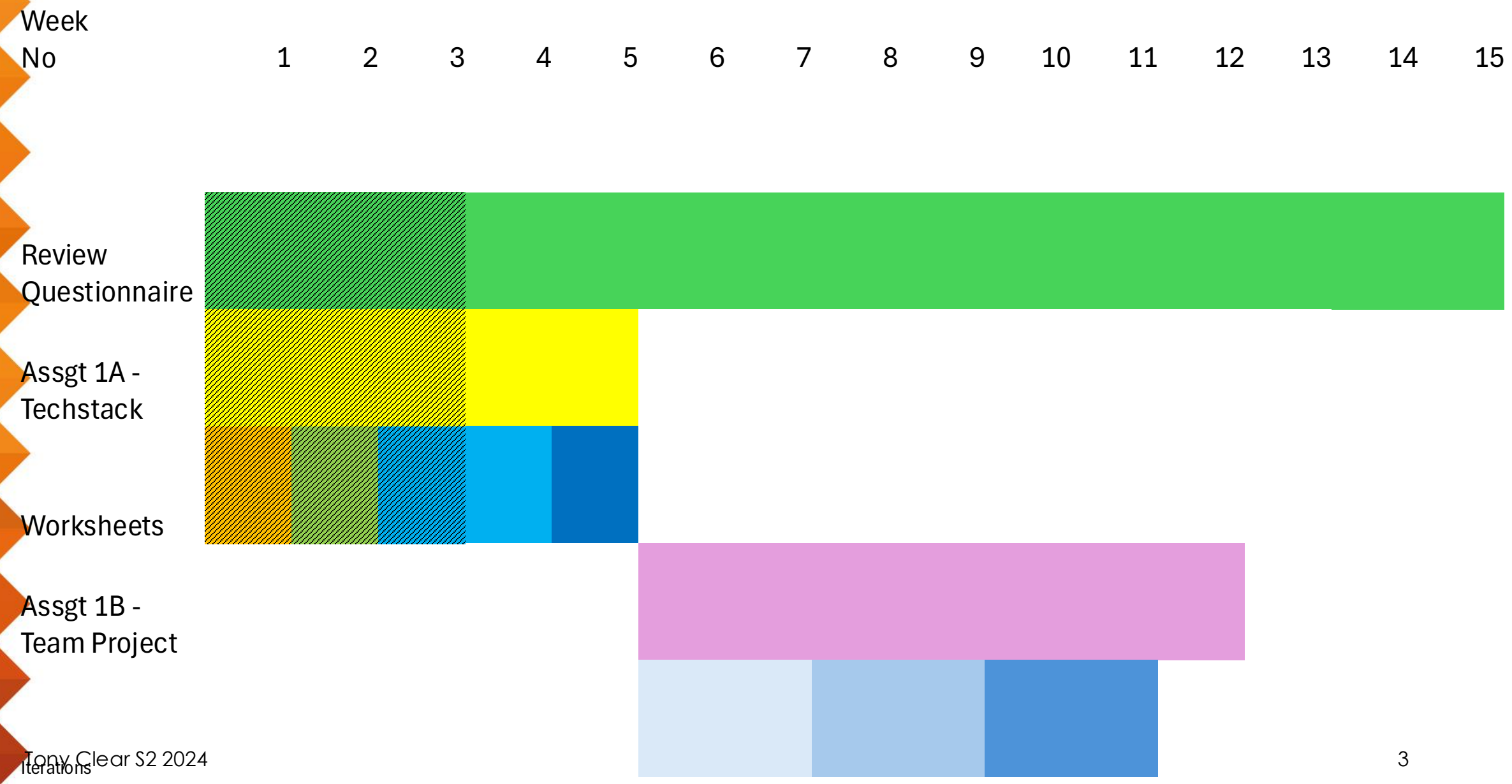
Progress – feedback, any issues?

Overview - what's coming up?

The Lecture Schedule

How does it relate to the assessment?

# Taking Stock



# Assignments Drive your Learning

**Ass 1A preparing for Software Development** (20%)
(*Individual*)

- Set up the tools an individual needs to support coding, good code craft, version control and unit testing
- Set up the tools needed to collaborate with a team to achieve product goals together

Sharing code – integrate code, review code,
Setup the tools needed to work with the selected Tech Stack (front-end/backend)
Set up tools to assure quality of product
Set up tools to deploy the product to the cloud
Set up tools to monitor and alert issues post deploy
Learn how to use the tools
Learn how to use the Tech Stack
Understand the product goals -> Product Backlog
Sprint 1 Goals -> Sprint Backlog

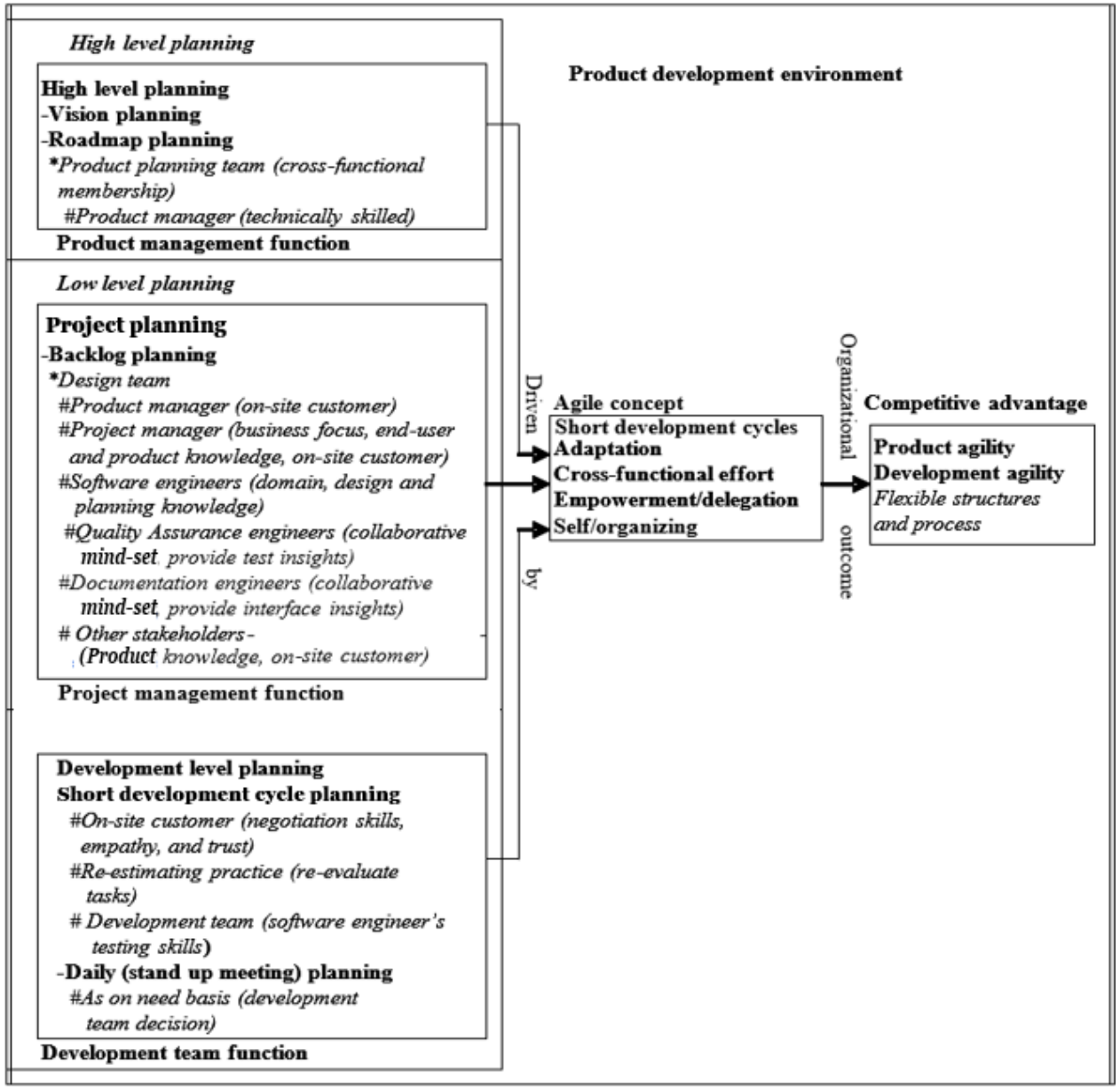
**Submission in Tutorials weeks 1-5 (sign off by TA)**
Evidence portfolio and demo

**Ass1B Full SDLC full stack product Dev (50%)**
(*small team - 4 Including QA*)
**Capability building by Developing a Product in a small team**
Apply a new tech stack and tool set
Practice DevOps and Scrum WoW
Collaborate with a Product Owner and team
Three sprints to learn fast – fast feedback
**Submit – reviews weeks 7,9,11 (tutorials)**
Capability and learning Portfolio with Evidence
Product increments

Sprint 1 weeks 5 and 6
Sprint 2 weeks 7 and 8
Sprint 3 weeks 9 and 10

**Ass 2 Knowledge Check (30%)**
(*Individual, online questions*)
A set of questions about scenarios to confirm you have understood main language and principles
**Sometime in Revision weeks (Faculty schedules)**

# Agile Planning - Levels




Lal, R., & Clear, T. (2021). Three Levels of Agile Planning in a Software Vendor Environment. In *Australasian Conference on Information Systems* (pp. 1-12). https://aisel.aisnet.org/acis2021/48/

# Quick Recap on the CISE custom process for developing software

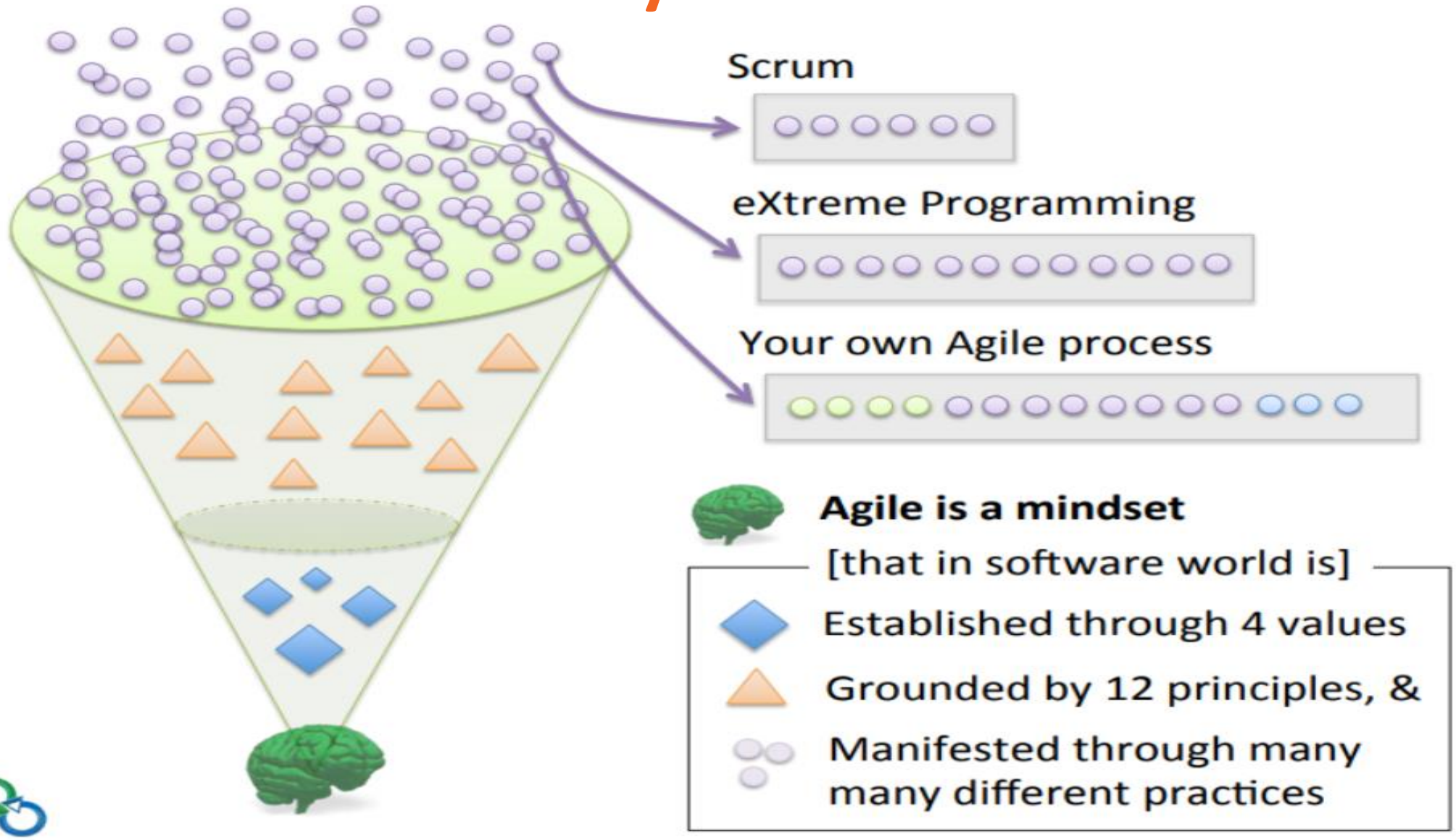
**R3.** Every project needs a slightly different methodology, based on those people characteristics, the project's specific priorities, and the technologies being used. This result indicates that **a team's methodology should be personalized to the team during the project** and may even change during the project.

**R6.** All the above suggests a repeating cycle of behavior to use on projects.

1. The members establish conventions for their interactions — a base methodology — at the start of the project. This can be likened to them "programming" themselves.
2. They then perform their jobs in the normal scurry of project life, often getting too caught up to reflect on how they are doing.
3. They schedule regular periods of reflection in which they reconsider and adjust their working conventions.

Cockburn, A. (2003). *People and Methodologies in Software Development* [Doctoral Dissertation, University of Oslo]. Oslo. Retrieved 8/03/2022 from https://www.researchgate.net/profile/AlistairCockburn/publication/253582591_People_and_Methodologies_in_Software_Development/links/56d434b208ae2ea08cf8e076/People-and-Methodologies-in-Software-Development.pdf

# The Agile Manifesto – many WoW



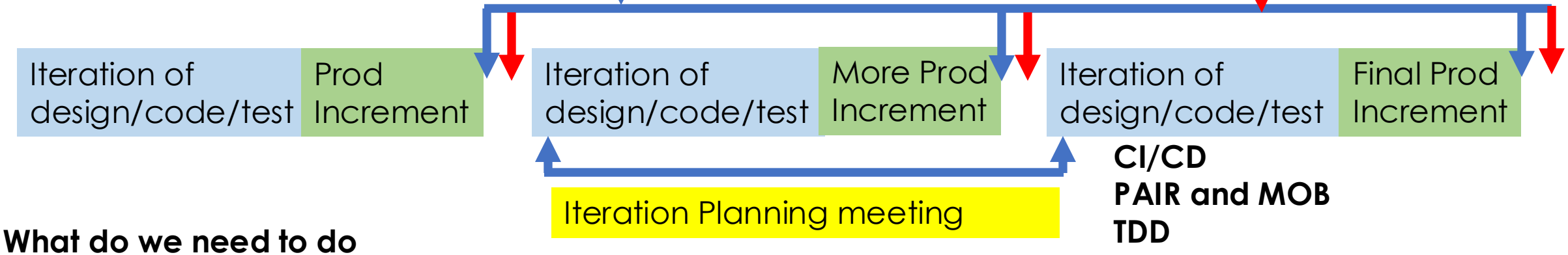

Webinar by Ahmed Sidky – CEO of ICAgile course
https://www.softed.com/assets/Uploads/Resources/Agile/The-Agile-Mindset-Ahmed-Sidky.pdf

**How will we get feedback on product from users/client?**

Regular review of product increment

**How will we keep improving our process**

Regular review of team process




**CI/CD**
**PAIR and MOB**
**TDD**

**What do we need to do before we start coding?**

**How will we decide what is in each Iteration?**

-Initial product backlog and story map (some uncertainty)
-User stories with Acceptance criteria
-Detail understanding and design of features for next iteration only
-Architecture and tech stack and deployment
-Dev Environment set up
-Plan for iteration 1
  Goal and Iteration Backlog

Regular team meeting during iteration…
Is there anything stopping us from reaching the goal?

**How will we coordinate work with each other and keep on the same page?**

**How will we manage changes to requirements?**

**How will we manage risks?**

**How will we assure quality?**

# User stories will be how we document user requirements

**WHY?**

**User stories will be the unit of work for**

**What about system requirements and features?**

Understanding user needs

Splitting up work

Designing product features

Testing product features

Organising the order of doing work

Planning iterations
- Estimation
- -hypothesis

Monitoring work

# Lifecycle of User Stories

Discover user needs

Product Goal

Many sources, techniques

IDENTIFY USER TYPES

Write high level user Stories (EPIC Stories)

Keep these on Monitoring board – Product Backlog

Keep these on User Story Map – product overview

Break into smaller user stories and include Acceptance criteria, success criteria, DOD

https://www.youtube.com/watch?v=Hq9O7mnUNM4

Decide on the order to work on

First product hypothesis to test

Two sets – ordered near the top, unordered below them

Capture in Product Backlog and User Story Map

Decide on which will be in the next iteration

(Starting from top of PB)

Translate into design and code

Get detailed SHARED understanding from PO

Estimate how many user stories team can do in one iteration (team capacity)

Monitor progress and adjust

How to manage change

Estimate the size of each user story and add user stories until team capacity is reached

Skills we need?

# Discovering User Stories (user requirements)

**Big upfront effort**

plan-driven control based on high-confidence (certain), **long-term** predictions

**Iterative and incremental effort**

Frequent opportunity for changes based on empiricism and short learning loops and **short-term** certainty and long-term uncertainty.

**User Story Workshops** – product stakeholders, PO, BA, Dev, Tester etc
Stakeholders write them and group (and agree on order or in/out)?

**Interviewing** users or the PO – get placeholders for future conversations about needs, changes to the current situation, and some detail for user needs that are of most value to be worked on first

**Observation, surveys, impact mapping, customer journey mapping……**

# INVEST – a quick checklist to check the usefulness (quality) of user stories

The INVEST checklist comes from INVEST in Good Stories by Bill Wake. It's an acronym with six important quality characteristics for user stories:

Independent — Can the story stand alone by itself ?

Negotiable — Can this story be changed or removed without impact to everything else?

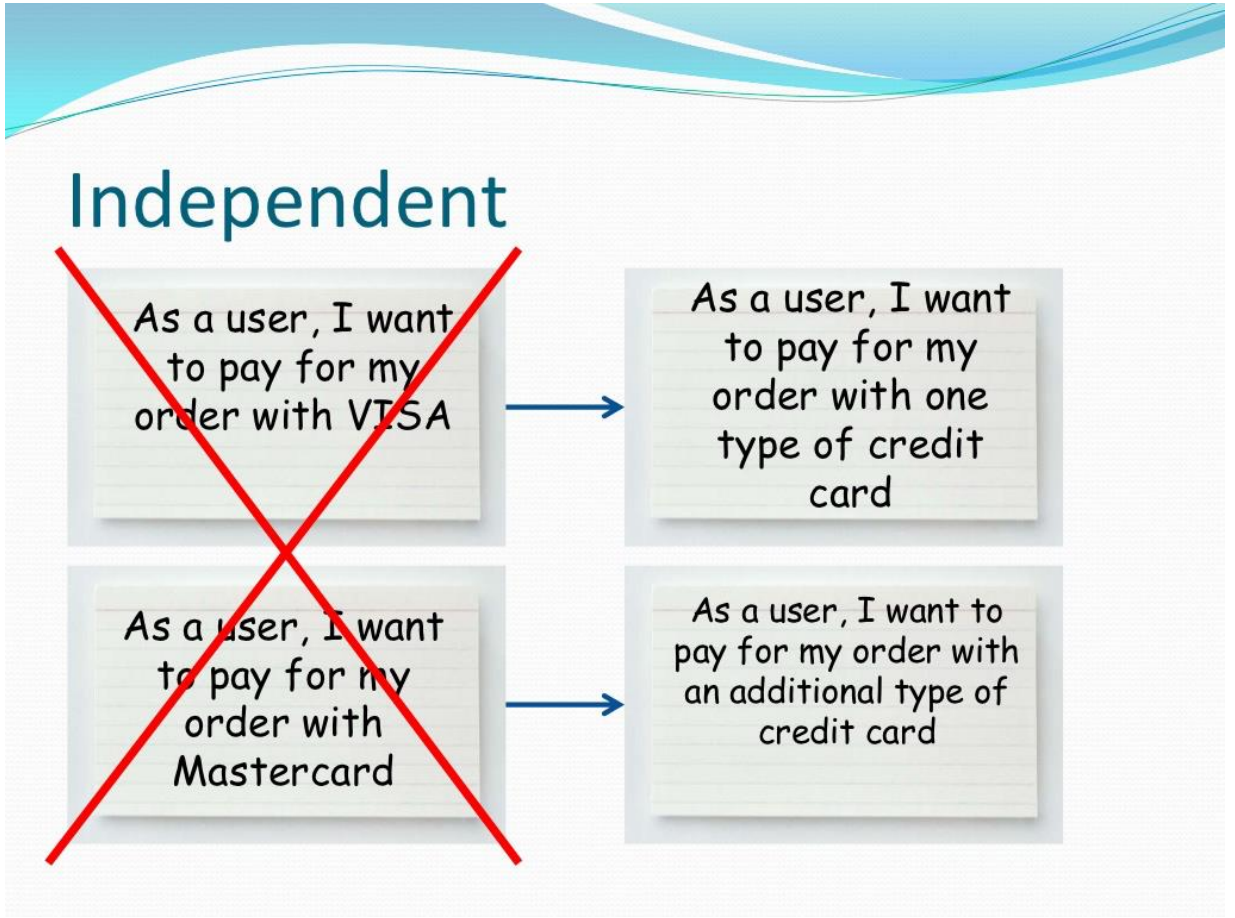
Valuable — Does this story have value to the end user?

Estimable — Can you estimate the size of the story?

Small —Is it small enough?

Testable — Can this story be tested and verified?

# INVEST


Independent
As a user, I want to pay for my order with VISA
As a user, I want to pay for my order with one type of credit card
As a user, I want to pay for my order with Mastercard
As a user, I want to pay for my order with an additional type of credit card

# Negotiable – not a contract!

As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express. Consider Discover

As a user, I want to pay for my order with my credit card

Note: Accept Visa, MasterCard and American Express. Consider Discover. On purchases over £100, ask for card ID number from back of card. The system can tell what type of card it is from the 1st two digits of the card number. The system can store a card number for future use. Collect the expiration month and date of the card

# Estimable

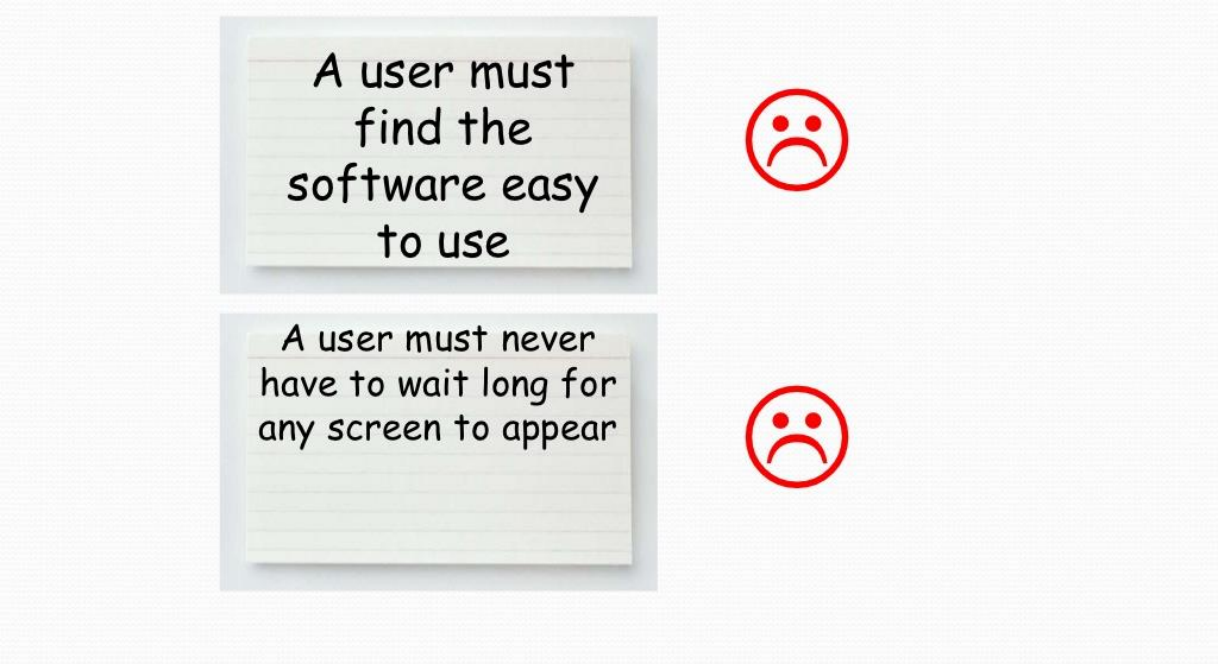
A job seeker can modify a CV already uploaded

- It is important for developers to be able to estimate or at least take a guess at the size of a story

**3 common failures**

1. Developers lack technical knowledge
2. Developers lack domain knowledge
3. Story is too big

# Testable


A user must find the software easy to use
A user must never have to wait long for any screen to appear

# Definition of Done

What satisfaction criteria apply to every user story so that the story is "out of my life"

May be different for different teams/sprints

IDEAS?

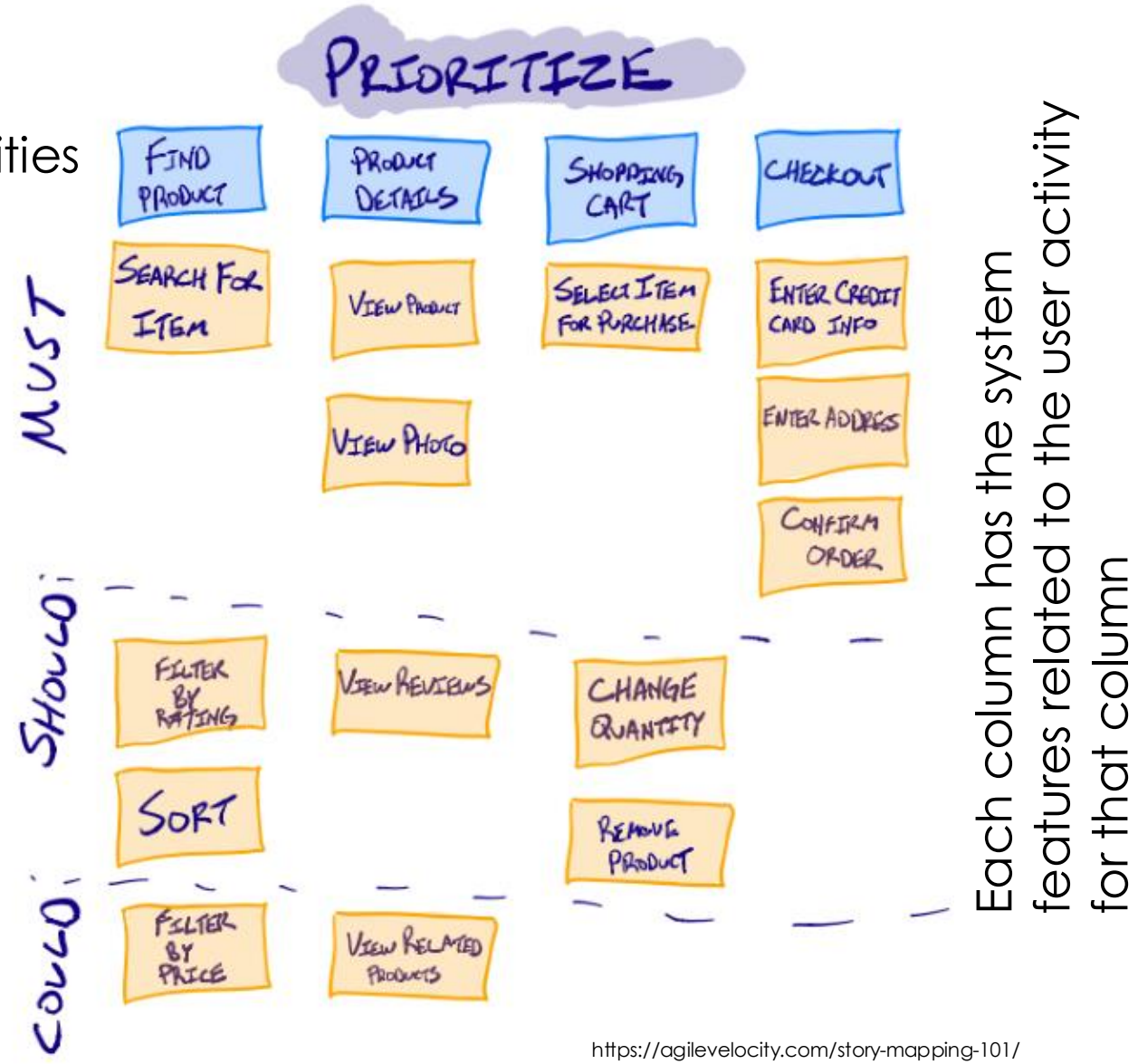
All unit, integration and acceptance tests are passed
Code is submitted to the repository and reviewed
The feature has been deployed on the test/staging/UAT/production environment
Documentation is completed and uploaded to the wiki
UAT is completed
CAB has signed it off

# Using a story map to slice out a delivery plan or sprint plan


https://www.scrumalliance.org/system/slides/118/original/christianhossa_specificationbyexamplewithgherkin.pdf?1349824954

# The anatomy of a Story Map

Columns relate to user activities



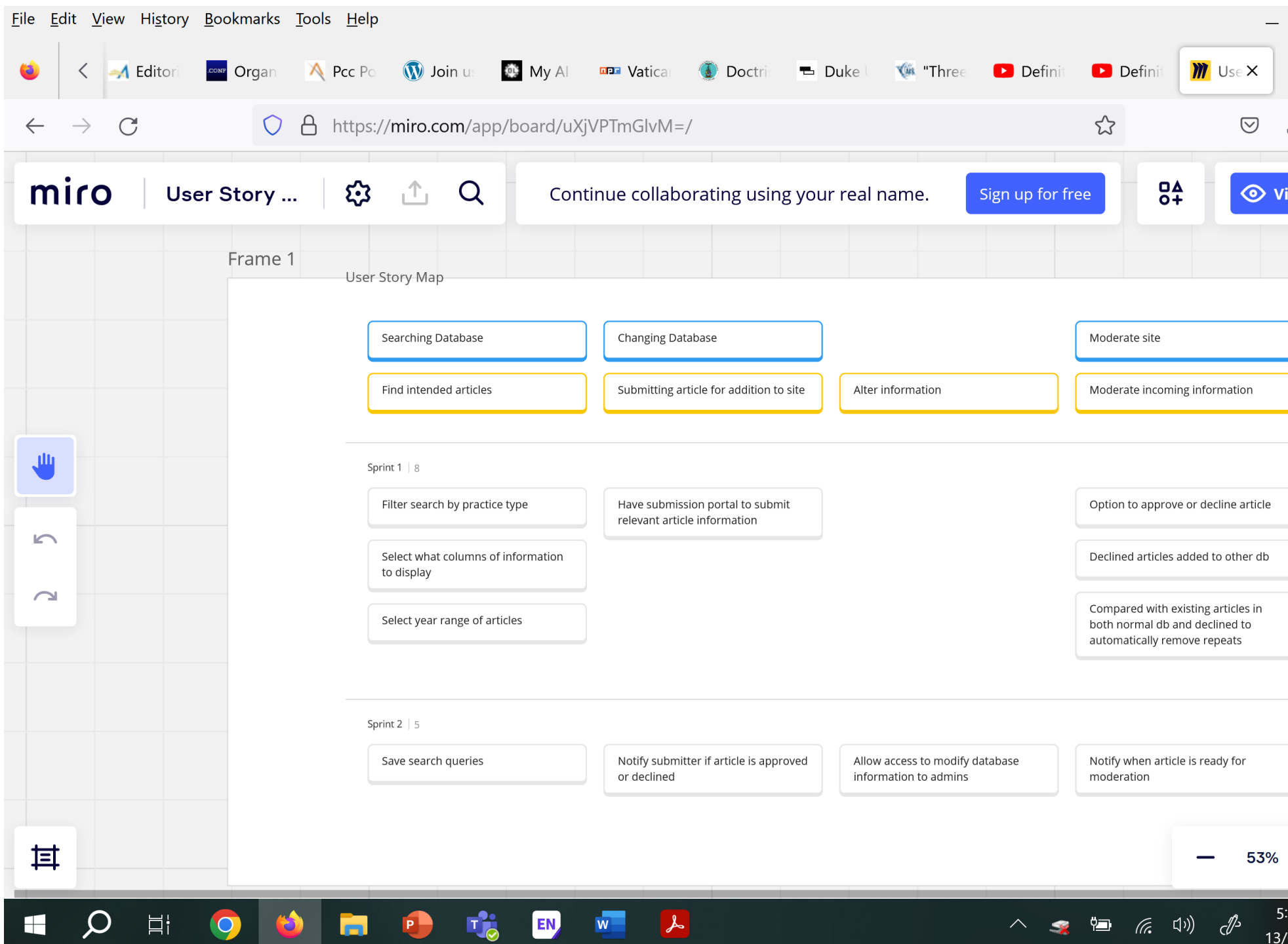

Horizonal (slices) groups are priority
Importance or frequency of use

Can create other horizontal slices
Of features to represent the scope of delivery cycles or MVP or sprint cycles

https://agilevelocity.com/story-mapping-101/

# Story maps and sprint plans – Miro a useful tool

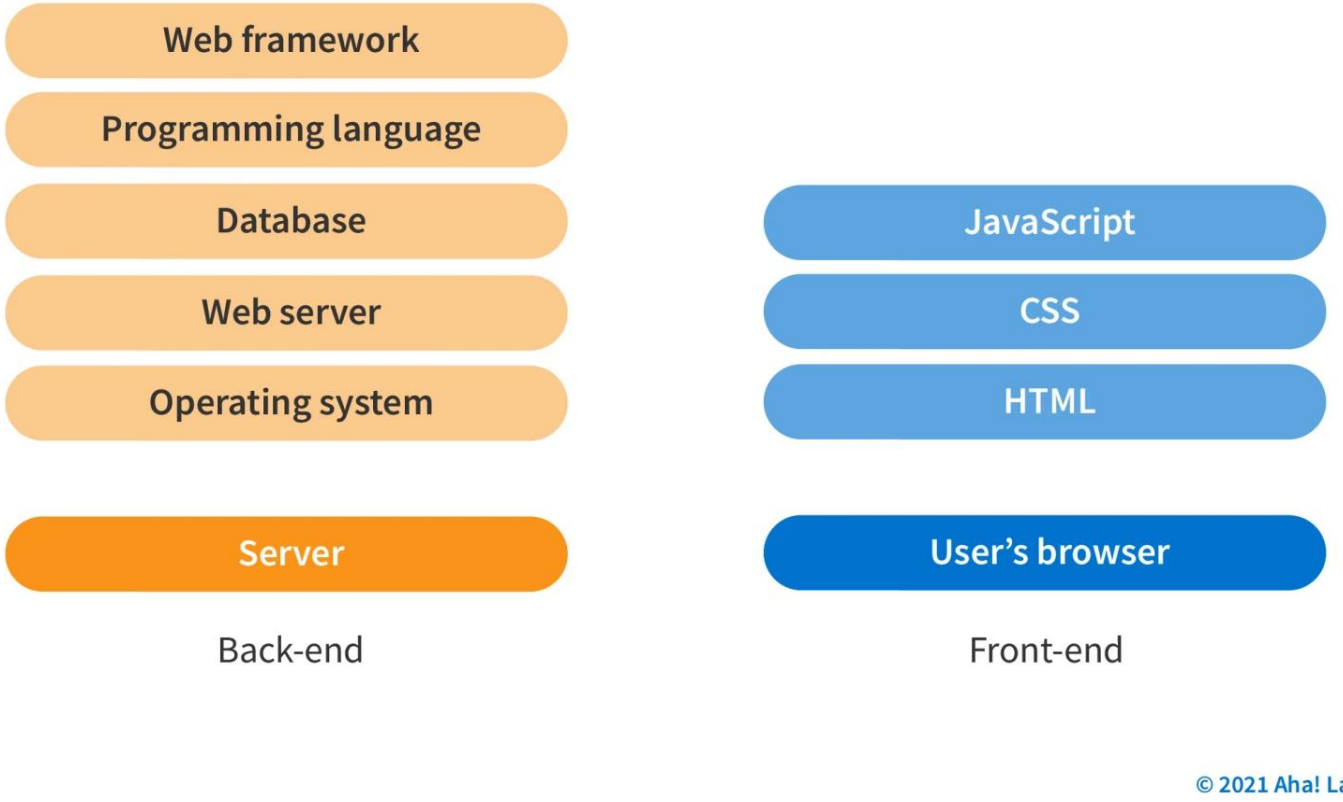
https://miro.com/index/?utm_source=public_board



# Technology stacks and Roadmapping strategy

https://www.aha.io/roadmapping/guide/it-strategy/technology-stack

https://www.aha.io/roadmaps/overview



# User Story Success Criteria

*acceptance criteria, Acceptance

A list of "rules" or criteria or tests or behaviours that should be met if the story is to be successful

Behaviour Driven Development (BDD)
Acceptance Test Driven Development (ATDD)

As a purchaser I want to be able to pay online by credit card for convenience

***Possible Success criteria?***

| | Initial situation | Event | New situation (output) |
|---|---|---|---|
| **Common template** | **Given** <???????> | **When** <???????> | **Then** <???????> |

# Pre-sprint – Big picture planning schedule when we know the least

3-month – Big room planning

6-week blocks – ShapeUp

Roadmap

Release plan

Big focus on delivering **Value** to users through
Agreeing on and refining

**Product goal**(s) and

**Sprint goals**

We value responding to change over following a plan

In our case – 3 short sprints
Roadmap = 3 x 9 day sprints (dates) with 3 sprint goals
Release plan = deploy increment every sprint

# Pre-sprint – Selecting what to work on for first sprint

Some estimate of the **size** of a user story

Some estimate of how many user stories the team can do in a sprint (**team's capacity** or velocity)

Keep selecting sized user stories from the top of the PB until the team's capacity is reached (or almost)

The sum of all user story sizes = or < team's capacity

Need to measure story size and team capacity in the **same units**

**Size** based on complexity, familiarity/novelty, effort

Function Point Analysis. Cocomo

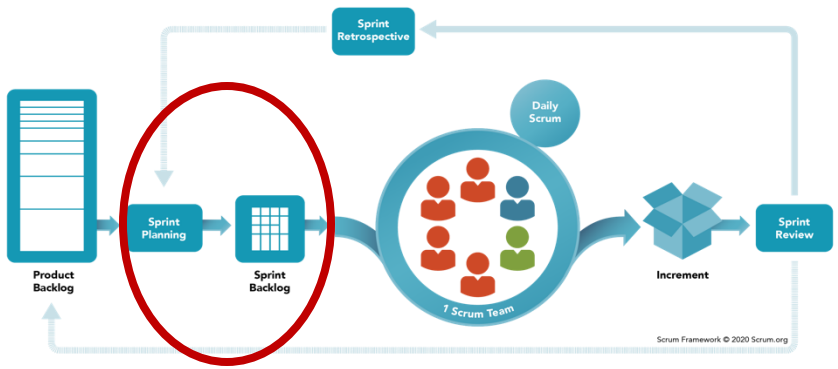
**Absolute estimation**
Hours/time

**Relative estimation** – need a baseline user story with defined size to compare to
Story Points
T-shirt sizes

NO NEED TO BE PRECISE!

Make all user stories around the same small size
Measure team sprint velocity in user stories

If deliver small changes and deploy quickly then no need to estimate size ( read about #noestimates movement)





https://www.scrum.org/resources

# Using team diversity to get a good estimate of a user story “size”

# Software Project Estimation - Issues

**Being able to predict** is a hallmark of any meaningful engineering discipline and software engineering is no exception.

Researchers have been exploring prediction systems for areas such as cost, schedule and defect-proneness for more than 40 years. …**empirical evaluation has not led to consistent or easy to interpret results**.

This matters because it is hard to know what advice to offer practitioners who are — or who ought to be — the major beneficiaries of software engineering research.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology, 54*(8), 820-827.

# Software Project Estimation - Examples

Three examples of inconsistent systematic review findings are:

- Jorgensen [13] reviewed 15 studies **comparing model-based to expert-based estimation**.
  - Five of those studies found in favour of expert-based methods,
  - five found no difference, and
  - five found in favour of model-based estimation.
- Mair and Shepperd [25] **compared regression to analogy methods for effort estimation** and similarly found conflicting evidence. From a total of 20 empirical studies,
  - seven favoured regression,
  - four were indifferent and
  - nine favoured analogy.
- Kitchenham et al. [21] found seven relevant empirical studies for the question **is it better to predict using local, as opposed to cross-company, data**.
  - Three studies reported it made no significant difference,
  - whilst four found it was better.

Shepperd, M., & MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Information and Software Technology, 54*(8), 820-827.

# Software Project Estimation in ASD?

How big is the software to be developed?

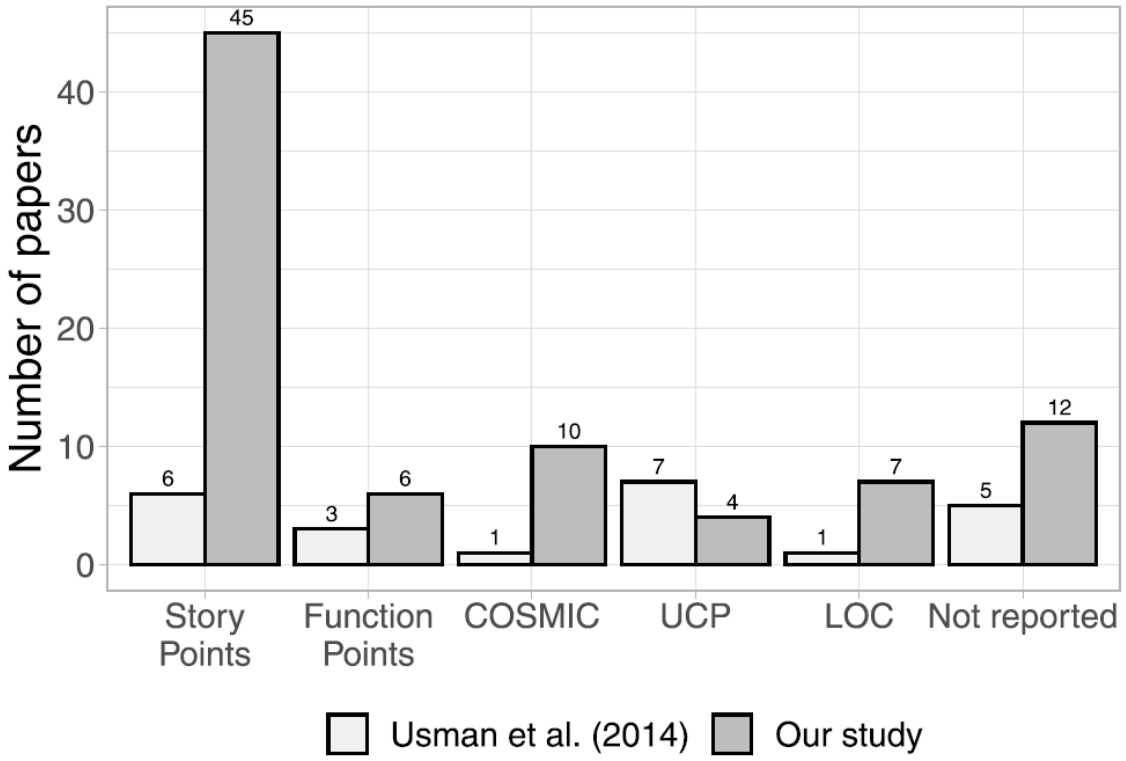



**FIGURE 5. Size metrics: Comparison of our results with those of Usman *et al.* (2014).**

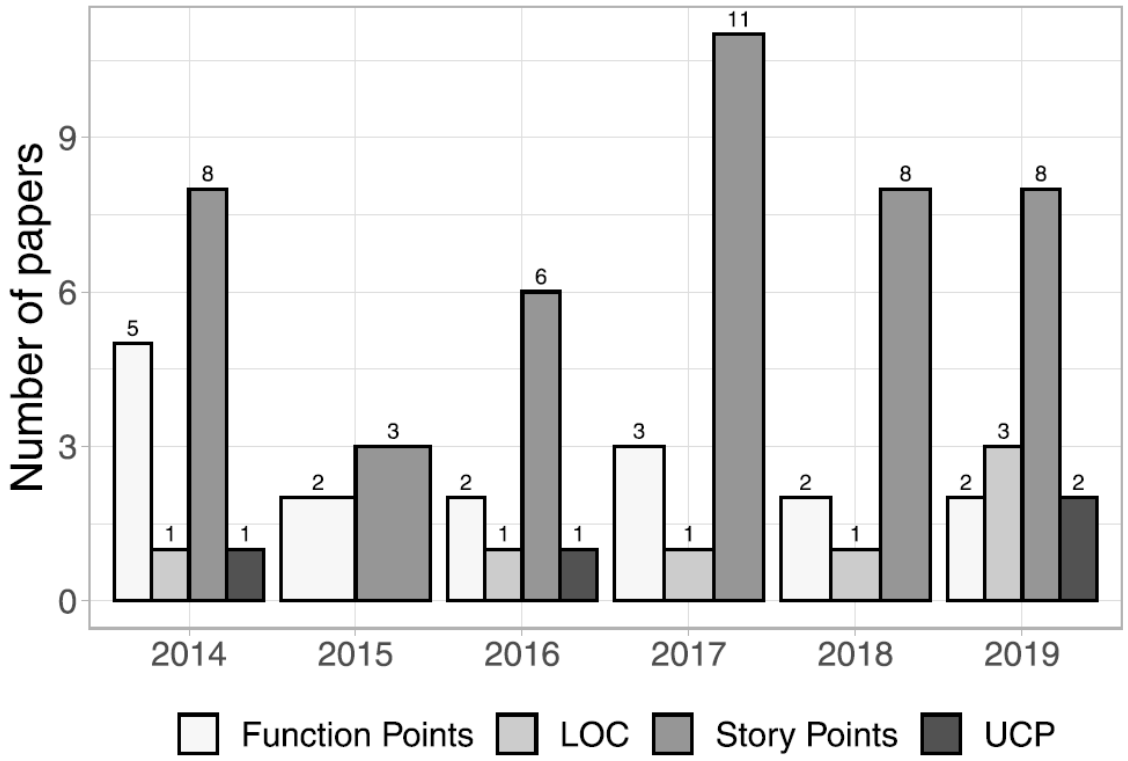



**FIGURE 6. Size metrics used in the papers per year of publication.**

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*, *8*, 166768-166800. https://doi.org/10.1109/ACCESS.2020.3021664

# Software Size Estimation – Function Points as one technique?

## 1 A SHORT INTRODUCTION TO FUNCTION POINT ANALYSIS

FP (Function Points) is the most widespread functional type metrics which is suitable for quantifying a software application. It is based on 5 user identifiable logical "functions" which are divided into 2 data function types and 3 transactional function types (Table 1). For a given software application, each of these elements is quantified and weighted, counting its characteristic elements, like file references or logical fields.

| | Low | Average | High |
|---|---|---|---|
| ILF (Internal Logical File) | 7 | 10 | 15 |
| EIF (External Interface File) | 5 | 7 | 10 |
| EI (External Input) | 3 | 4 | 6 |
| EO (External Output) | 4 | 5 | 7 |
| EQ (External inQuiry) | 3 | 4 | 6 |

**Table 1. Complexity weights with corresponding number of UFP.**

The resulting numbers (Unadjusted FP) are grouped into Added, Changed, or Deleted functions sets, and combined with the Value Adjustment Factor (VAF) to obtain the final number of FP. A distinct final formula is used for each count type: Application, Development Project, or Enhancement Project.

Meli, R., & Santillo, L. (1999). Function point estimation methods: A comparative overview. FESMA,

# Software Project Estimation in ASD - Mobile Apps?

How can we determine how big is the software to be developed – mobile apps?

…the techniques most commonly used for mobile apps were

- **Function Size Measurement and**
- **Expert Judgment.**

planning and development of **mobile applications differs from other traditional software applications** characteristics of the mobile environment…;

- high autonomy requirements,
- market competition,
- and many other constraints.

With regard to the **size metrics and cost drivers**, the results showed that

- the **number of screens**
- **and type of supported platform for smartphones**

**most common factors used** to measure the estimation prediction.

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*, *8*, 166768-166800. https://doi.org/10.1109/ACCESS.2020.3021664

# Software Project Estimation in ASD – some conclusions?

…**a combination of data-based methods (using project historical data and context-specific methods) with expert-based methods may be promising** in improving accuracy levels.

.. been a decrease in the use of general-purpose size metrics (e.g., UCP) and **increase in the use of size metrics that take agile methods into account**.

In this respect, **Story Points was the metric most frequently used to determine the size of the software.**

Usman *et al*. [60] also found that **Story Points is the size metric most frequently used by agile teams.**

However, some authors suggest that:

- Story Points should be estimated collectively … to reach a team consensus so as to
  - alleviate the chances of over-optimism and
  - reduce the issues
  - of anchoring and
  - strong personalities

Fernández-Diego, M., Méndez, E. R., González-Ladrón-De-Guevara, F., Abrahão, S., & Insfran, E. (2020). An Update on Effort Estimation in Agile Software Development: A Systematic Literature Review. *IEEE Access*, *8*, 166768-166800. https://doi.org/10.1109/ACCESS.2020.3021664

# Working with the Client: Useful Questions to Ask

- What business problem are you trying to solve?
- What's the motivation for solving this problem?
- What would a highly successful solution do for you?
- How can we judge the success of the solution?
- Which business activities and events should be included in the solution? Which should not?
- Can you think of any unexpected or adverse consequences that the new system could cause?
- What's a successful solution worth
- Who are the individuals or groups that could influence this project or be influenced by it?
- Are there any related projects or systems that could influence this one or that this project could affect?

# Pre-sprint – User Requirements_v1 User Stories part 1

Significant user type with characteristics that distinguish from other user types that may affect the design. NOT "User"

The new capability the user wants so they can reach smaller goal (outcome) In "business" language & NO solution details

What is the goal (desired outcome) of this new user capability?

**As a** <???> **I want to be able to** <???????> **so that** <?????????>

As the PO I want to have lots of articles in the evidence repository

As a practitioner, I want lots of evidence to be available so the evidence is convincing

I want some way to be able to keep adding articles with evidence for analysis to add evidence to the repository

I want people to be able to add new evidence so the repo gets bigger and I leverage crowd sourcing

As a **researcher** I want to be able to **recommend articles to include in the evidence repository** so that **the evidence available keeps expanding**

We should check the article is about SE with evidence (relevance)

We should check the article is not already in the repo (avoid duplicates)

We should check the quality of the evidence in the article (quality)

The submitter should be informed/thanked if the article they submitted is accepted or not (and why not)

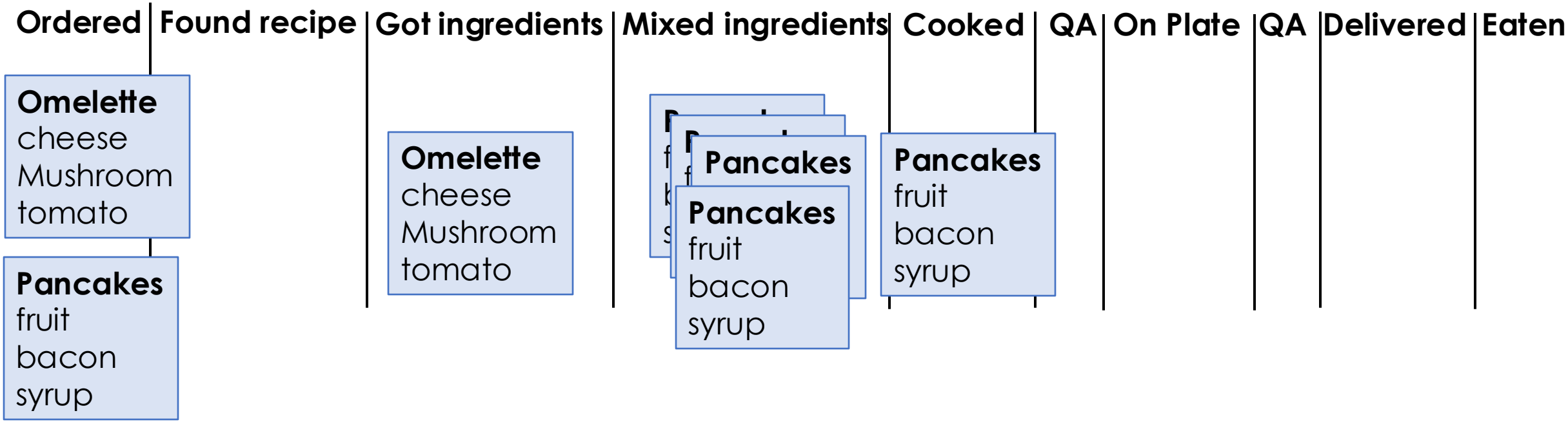
**Placeholders for conversations.**

**Estimating story points??**

# Monitoring progress during a Sprint and overall

Decide on a set of status labels for user stories- these are the columns in a work board

Target is 10 minutes after order



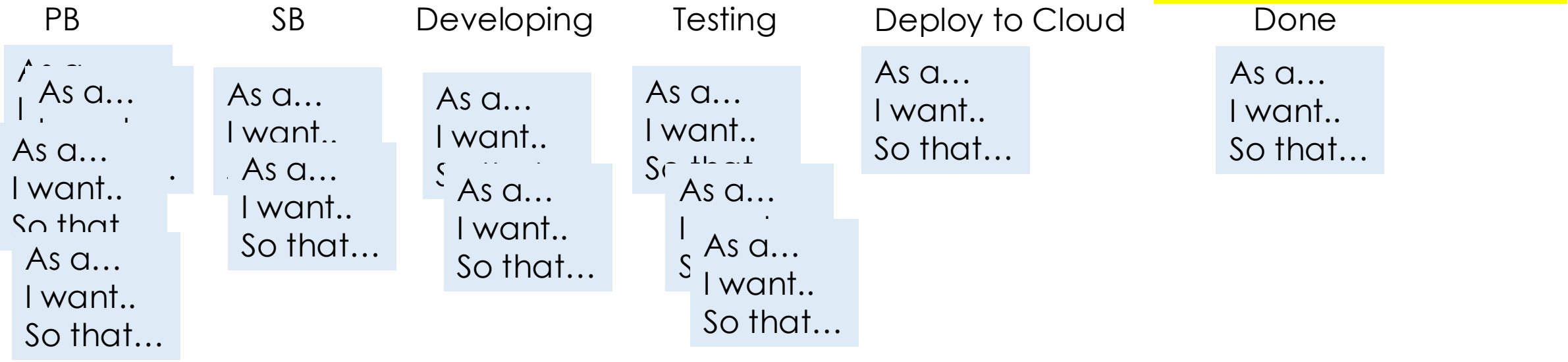
# User Story Board

**Physical** Boards
**Electronic** Boards
-Trello
-Asana
-Jira
-Azure DevOps

Advantages/Disadvantages?

**States of User Stories**

**Definition of Done?**



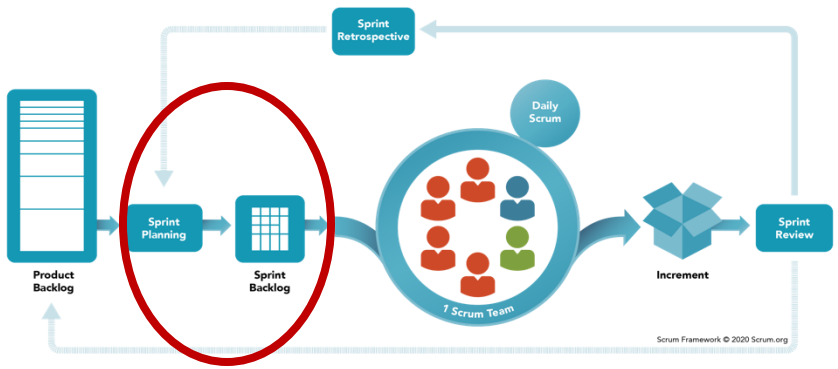
# Pre-sprint – Design, tech stack, architecture Non-functional (quality) Requirements

Layered architecture

Front-end NEXT.js

Back-end Nest.js on Node.js

Database. MOngoDB





https://www.scrum.org/resources

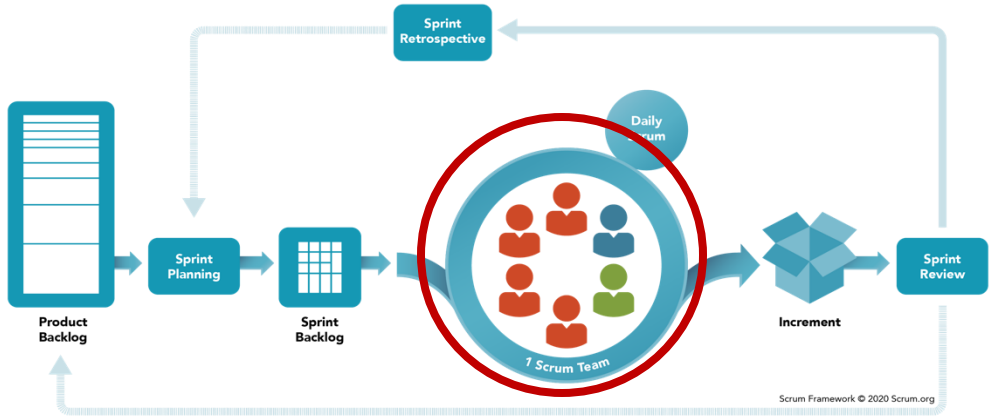
# Working as an Individual in a team - workflow

Continuous Integration workflow

Coding standards, code craft, code quality

Non-functional requirements

NOT this should be easy to use and perform well



# Team Collaboration

**How to be a high Performing team (or at least a team that does not want to kill each other!) (revisit later)**

Build trust and share expectations and values ***What would happen if NOT true?***

Coordinate work, expectations and goals frequently **Scrum meeting**

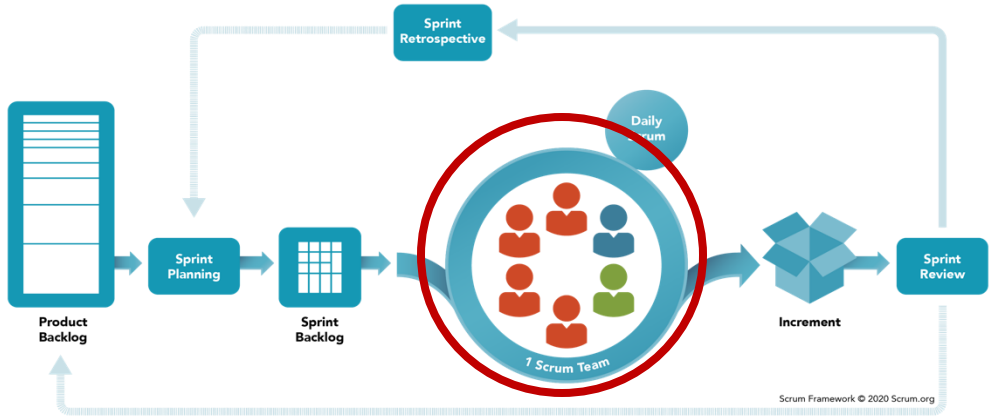
Reflect on how the team works together and learn from previous work to improve

Feel safe together

Team ownership/success trumps individual

Break work up – integrate work Cont. Integration

Check work and goals frequently with Users/PO




Work together on the same work – **mob programming**
**Solve problems together, share understanding, learn from each other**


https://www.scrum.org/resources

# Risk planning (revisit this later)

**Risk** – **chance** of not meeting expected quality or goals

Describe quality and goals!

HIGH IMPACT?

Risk of making something that does not meet expectations of PO

Risk of making something that does not get used

Risk of learning new tech taking longer than predicted

***What else?***

Risk that the product is not robust

Risk code is not easy to change

Risk that someone with critical knowledge is not available to team

# Expectations for Sprint 1

Dev environment and tool pipeline set up for each team member including single repo in GitHub

Product Backlog — Epic stories, detail for priority stories

User Story Map

Sprint Backlog — Based on some way of estimating story sizes and team velocity
Try Planning poker

Continuous integration — Frequent build (every few days

Testing automated to some extent

Mob programming tried

Releasable Product Increment — Deployed to cloud

# Comparative Agility Assessment

At the highest level, the CA approach assesses agility on seven *dimensions*:

- Teamwork;
- Requirements;
- Planning;
- Technical Practices;
- Quality;
- Culture; and
- Knowledge Creating.

Each dimension is made up of three to six *characteristics* for a total of 32 characteristics.

Williams, L., Rubin, K., & Cohn, M. (2010). Driving Process Improvement via Comparative Agility Assessment. In *2010 Agile Conference* (pp. 3-10). https://doi.org/10.1109/AGILE.2010.12



# Questions and Comments….