UNIVERSITATEA DIN BUCUREȘTI

VIRTUTE ET SAPIENTIA

tremend

A company of publicis sapient

Course 7
Agile planning & Team Management

500 Technology Fast 500 2019 EMEA WINNER Deloitte.

IMPACT STAR Award by Deloitte. Technology FAST 50 CENTRAL EUROPE 2020

Proud Member Of EBRD Exclusive Blue Ribbon Global Network Of Best Performing SMEs

Agenda

Agile definition

Scrum in review

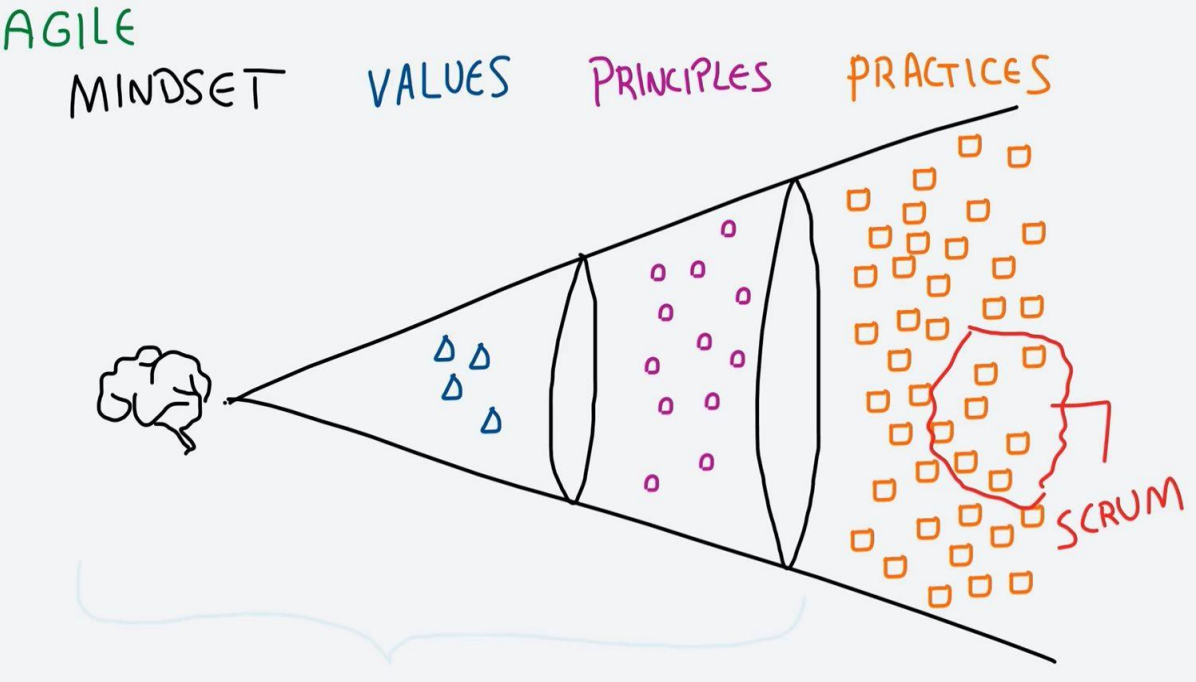
Estimation techniques

Improving estimation or getting back on track

21st's century buzz work is…

Agile

a MINDSET based on a set of values, on which principles and practices are applied, reviewed and updated.



Agile Manifesto: values ..

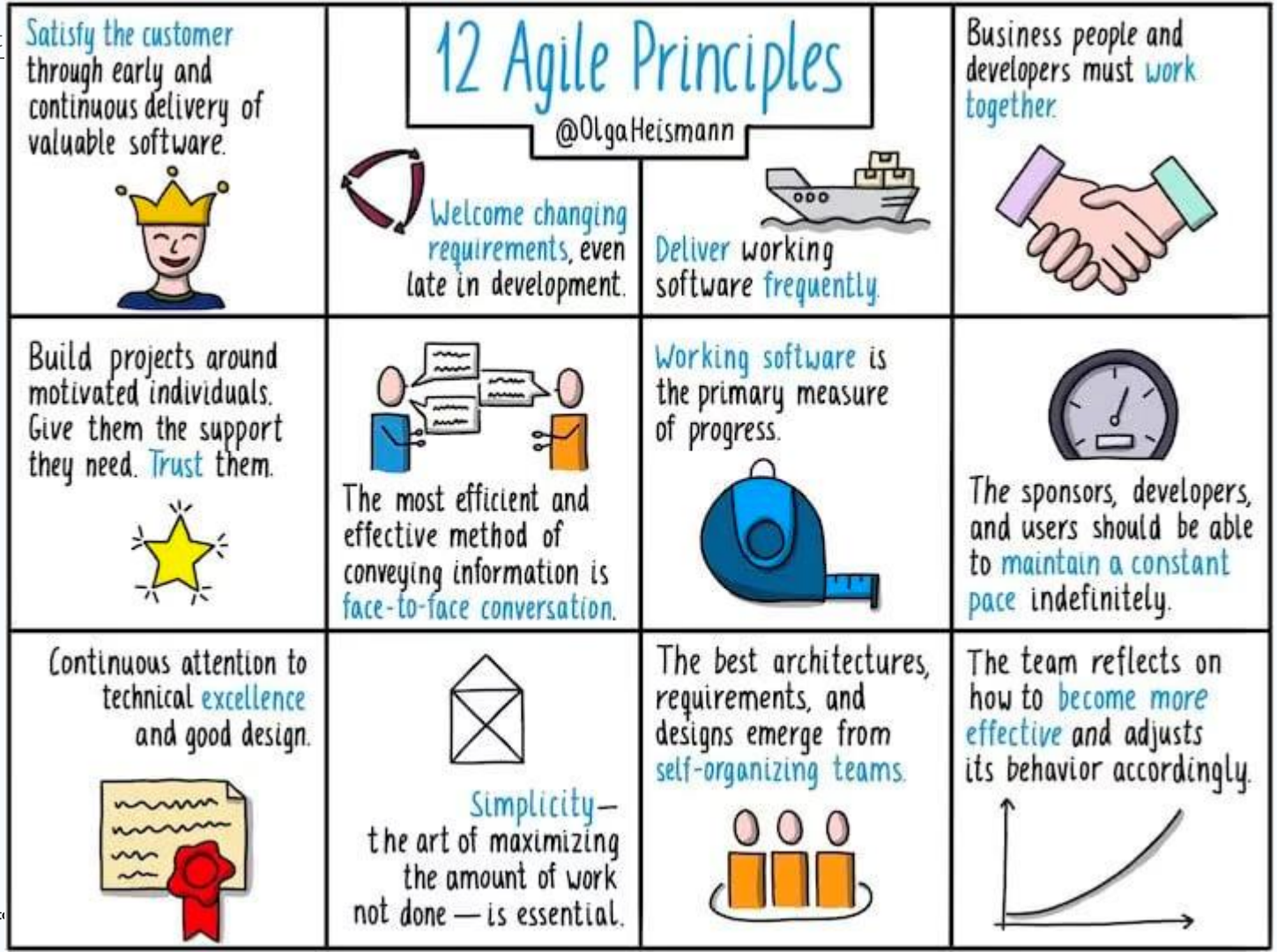
Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Agile principles

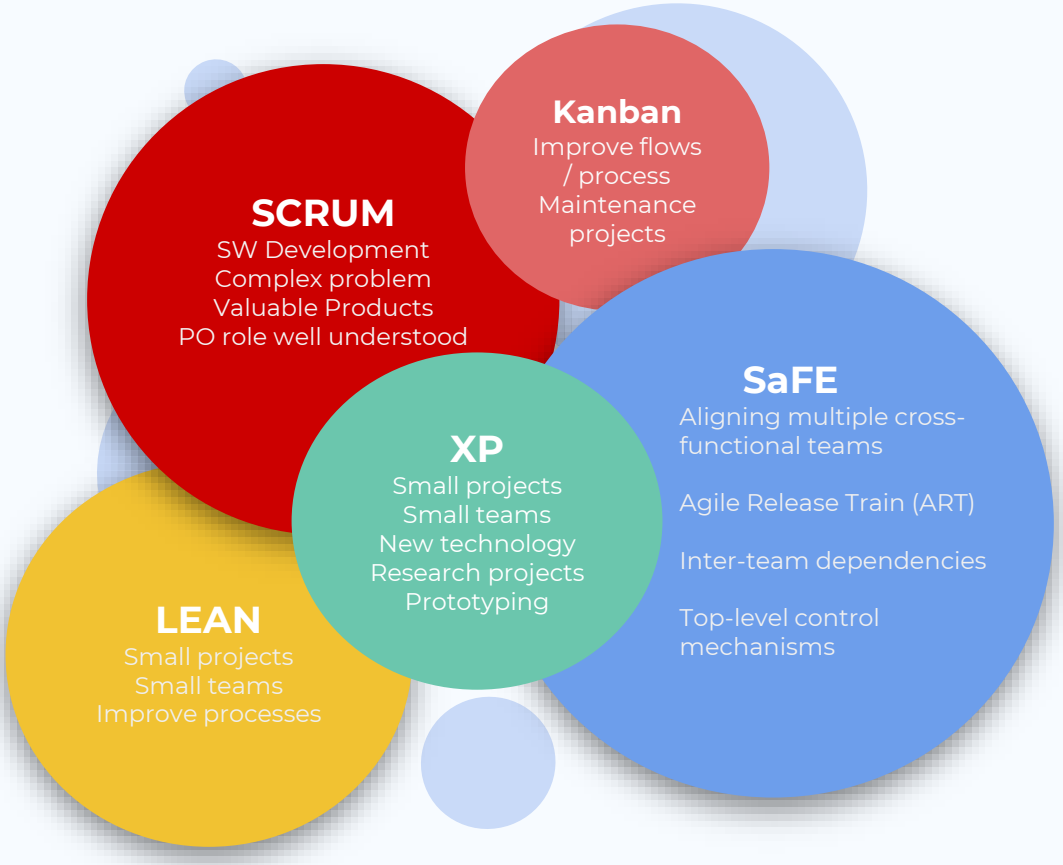


https://blog.logrocket.com/product-management/agile-manifest

Agile frameworks

An Agile framework

- is a specific approach to planning, managing, and executing work.
- Is one of many documented software-development approaches based on the philosophy articulated in the Agile Manifesto.
- incorporates elements of continuous planning, testing, integration, and other forms of continuous development.



Scrum in review

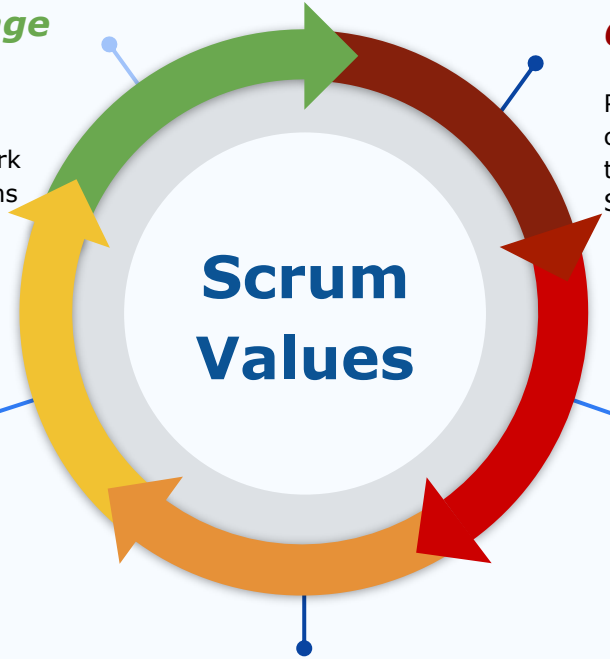
Values

Scrum in a nutshell

Roles

Events

SCRUM Values



Courage

Scrum Team has courage to do the right thing and work on though problems

Commitment

People personally commit to achieving the goals of the Scrum Team

Respect

Scrum Team members respect each other to be capable, independent people

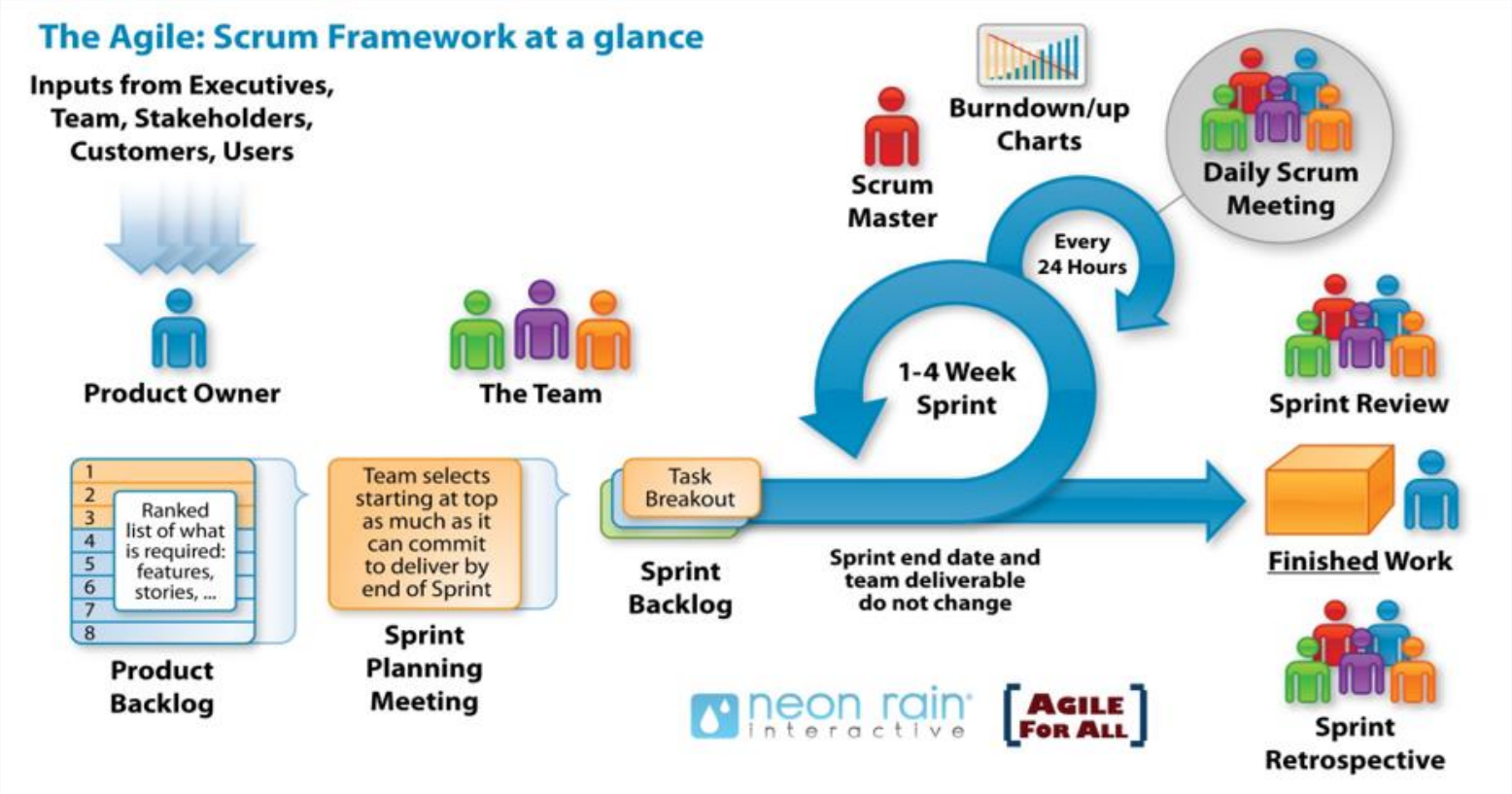
Focus

Everyone focus on the work of the Sprint and the goals of the Scrum Team

Openness

The Scrum team and its stakeholders agree to be opened about the work and challenges with performing the work

SCRUM in a nutshell



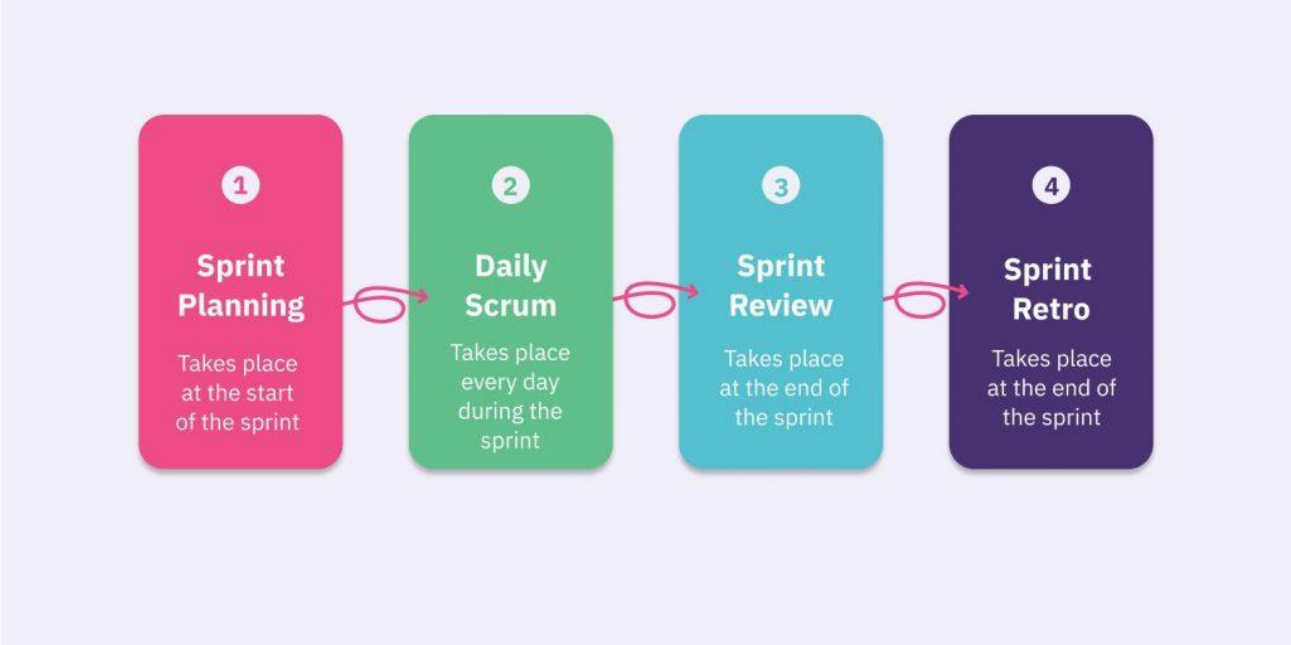
SCRUM roles

Product Owner
Responsible for maximizing the value of the product

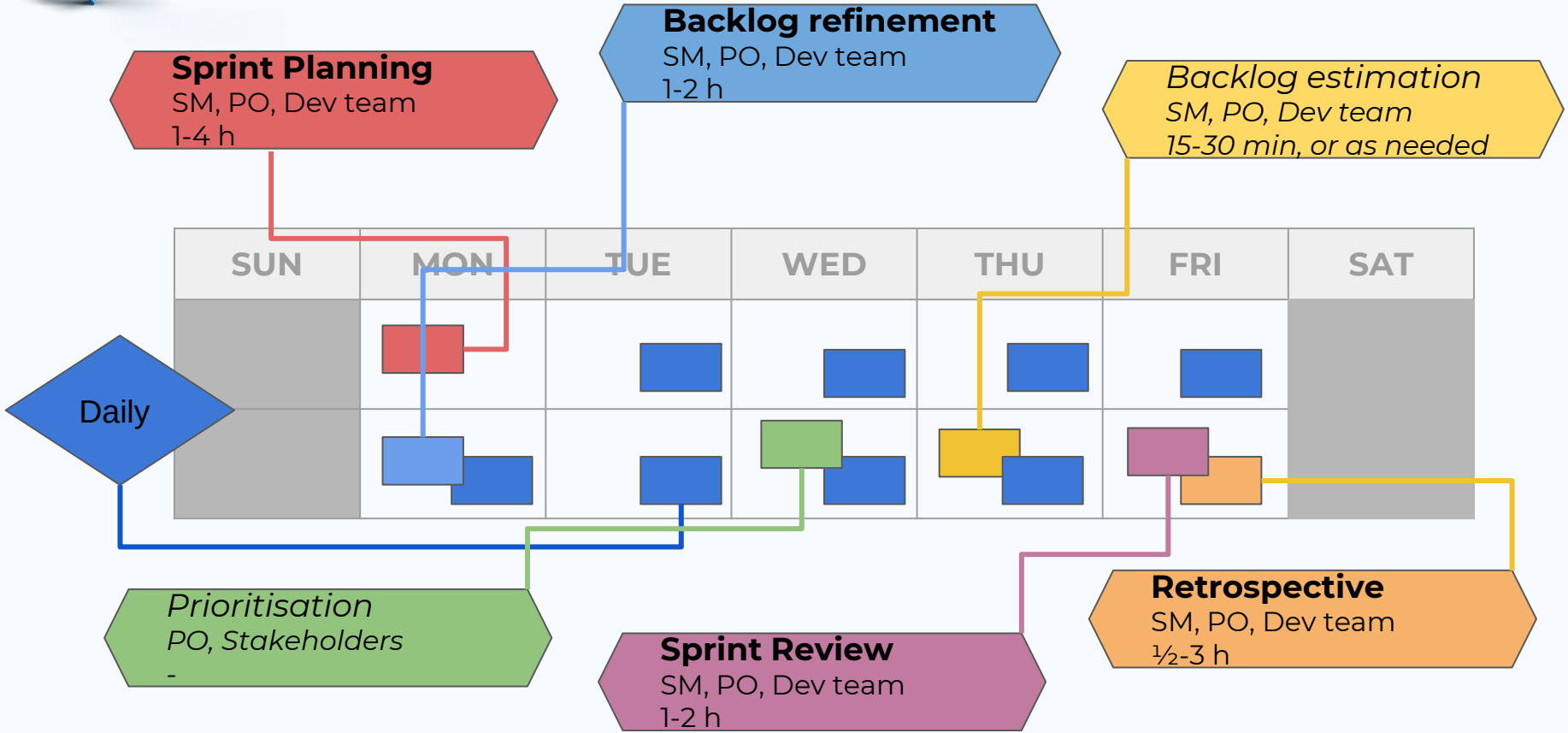
Development Team
Self organized
Size guide: 6 +/-3
Team accountability
Autonomous
Cross-functional
Empowered

Scrum Master
Responsible for maximizing the value of Development Team

SCRUM events



SCRUM events (extended)



Refinement

Backlog refinement is an ongoing process in which the Product Owner and the Development Team collaborate to ensure that items on the Product Backlog:

- are understood the same way by all involved (shared understanding),
- have a size estimate for the (relative) complexity and effort of their implementation, and
- are ordered according to their priority in terms of business value and effort required.

Step 1: Decide if the item is added to Backlog

Step 2: Define and clarify the item

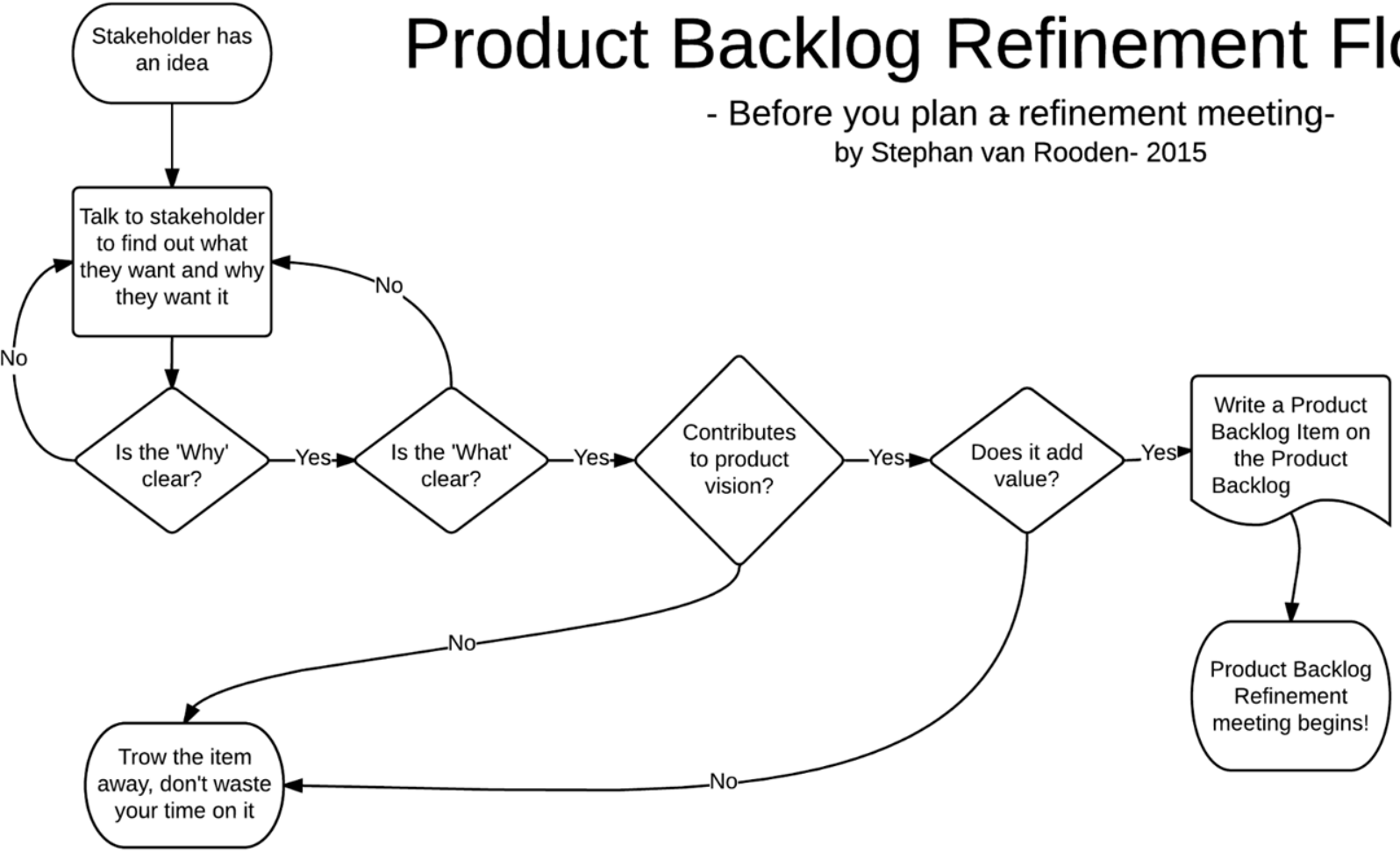
- Spikes
- Define, Slice, Acceptance Criteria

Step 3: Refinement meeting with the team to validate DOD

Product Backlog Refinement Flowchart

- Before you plan a refinement meeting-

by Stephan van Rooden- 2015

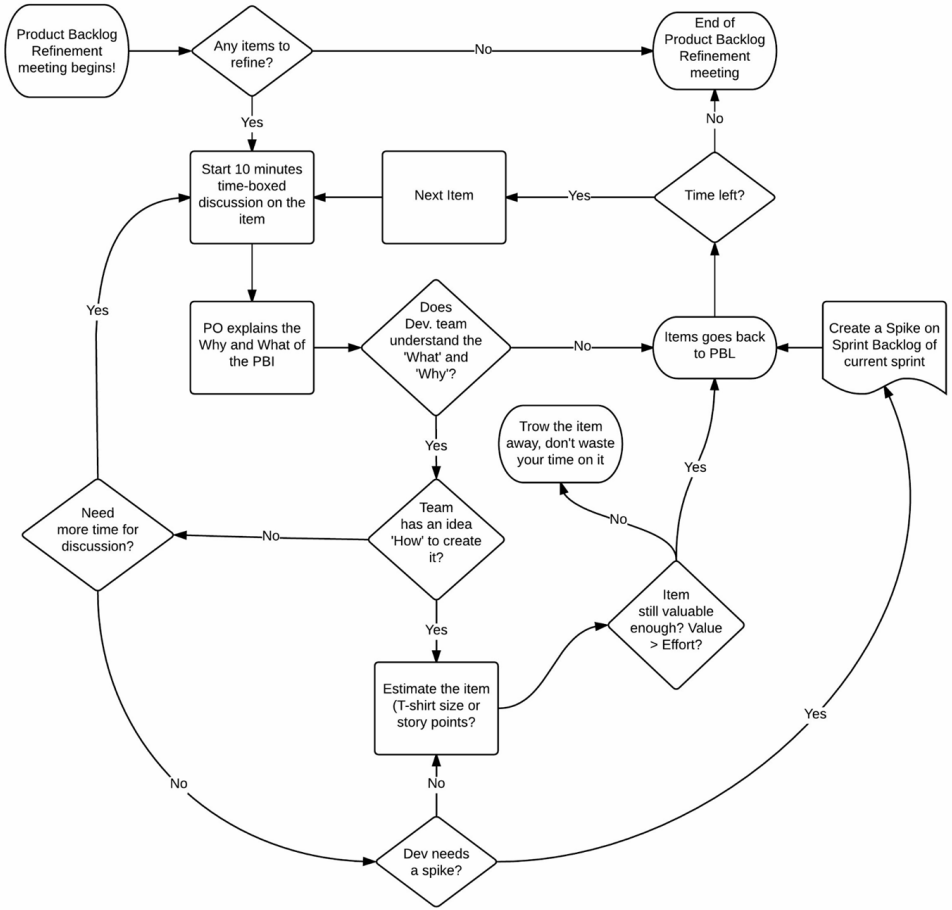


Refinement

Product Backlog Refinement Flowchart

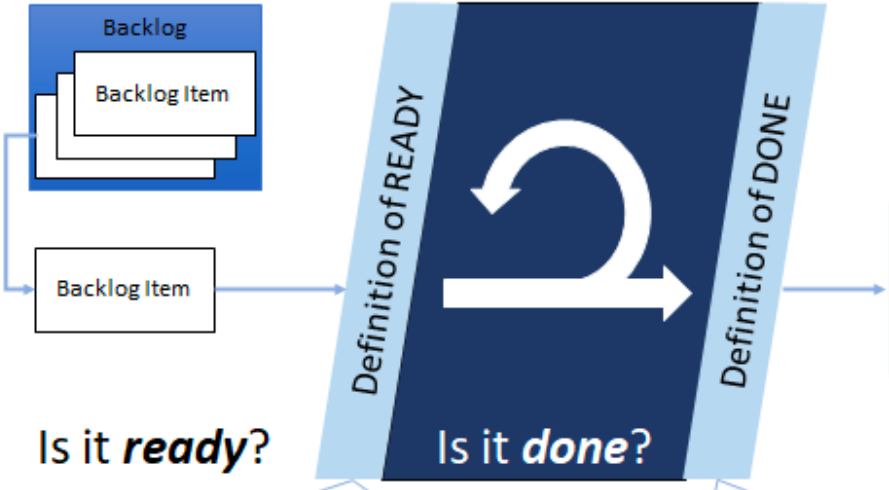
- How to do Product Backlog Refinement -

by Stephan van Rooden- 2015



DOR & DOD

Quality Built-In by implementing DoR & DoD



Definition of Ready (DoR) & **Definition of Done (DoD)** as *foundation* of our *agile* way of working

Quality criteria for ***definition*** of
- User **story**
- Business **epic**
- Product (release) **theme**

Quality criteria for ***delivery*** of ***product increment***

DoR & DoD:
- checklists that **must be completed *before*** product backlog items are considered to be in the respective state
- Applies to ***all*** user stories!

Estimation techniques

What do we estimate

Velocity planning

Capacity planning

How to estimate

Estimation

Scrum teams use project estimating to identify **how much work can be done** in a particular sprint.
In software development, an **estimate**, consists of a quantified evaluation of the effort which is necessary to carry out a given development task; this is most often express in terms of duration.

Agile estimations are essential for:

- Making teams accountable for deliverables
- Inducing discipline across the Agile team
- Predicting the approximate time it will take to finish a project
- Enabling better sprint management
- Improving team productivity

Project estimations come into play at the beginning and end of each sprint. They help teams determine what they can get done at the beginning of the sprint, but also show how accurate those initial estimates were at the end.

Capacity planning

What
The amount of **productive engineering time** available in a sprint.

Why
Some teams tend to over-commit Commitment-driven planning or Agile capacity planning ensures that you have enough capacity to complete tasks without overcommitting.

When
before a sprint planning meeting

Steps to calculate capacity-based sprint planning:

1. Calculate **Team Member Availability**: productive engineering hours available.
2. Calculate **Sprint Duration**: number of days allocated to each sprint.
3. Calculate **Standard Hours per Day**: number of hours worked each day.
4. Consider Other **Availability Factors**: holidays, vacations, shutdowns, and other factors that impact work hours during the planning process.
5. Identify **Other Work**: other projects or priorities that will take engineers away from sprint work.
6. Calculate the **Focus Factor**: actual percentage of each day that the team can focus on the sprint goals without interruption.

Capacity planning

	Sprint Capacity Planner														
	Max Limit	Team Hours				Sprint Days									
Team Members	5			Member	Allocation	1	2	3	4	5	6	7	8	9	10
Sprint Length (Days)	10			Person 1	100%	8	8	8	8	8	8	8	8	8	8
Hours per day	8			Person 2	100%	8	8	8	8	8	8	8	8	8	8
Focus Factor	80%			Person 3	100%	8	8	8	8	8	8	8	8	8	8
Daily stand-ups (Hours/Sprint)	0.25	12.5		Person 4	75%	0	6	6	6	6	6	6	6	6	6
Planning, Review, Retrospective, Refinement (Hours/Sprint)	4.5	22.5		Person 5	50%	4	4	4	4	4	4	4	4	4	4
Other meetings (Hours/Sprint)	2	10		Person 6		0	0	0	0	0	0	0	0	0	0
Total Capacity (Hours)	334														
Total Meeting Time (Hours)	45														
Focus Time (Hours)	57.8													Total Team Time (Hours)	334
Plan Capacity (Hours / Days)	231	28.9													

Velocity planning

What
The amount of **work [Story Points] the team can deliver** in a Sprint

Why
Team's performance decides commitment

Encourages Self-organization within team members.

Team can challenge themselves.

When
Each sprint after sprint closure

The steps in velocity-driven sprint planning are :

1. Determine the **team's target velocity**.
2. Identify the **Sprint Goal**.
3. Select **number of product backlog items** (normally stories) equal to that velocity, required to achieve the **GOAL**
4. ***Split stories*** *into tasks, if necessary*
5. ***Estimate tasks***

Velocity planning - how to estimate

Scales for story points

- the **power of two** (1, 2, 4, 8, 16, 32, etc.)
- **Fibonacci-like scale** ([0.5], 1, 2, 3, 5, 8, 13, 20, 40, 100). *(These are not really the Fibonacci numbers, but are a bit similar)*.

It doesn't really matter, which scale you use, the point is, that the **scale shouldn´t be linear**.

Follow the INVEST guidelines for good user stories!

Use a Reference story

1. Choose a common, medium-sized task: **very common**, most **members already worked** similar tasks, everyone **understands the work required**
2. **Estimate** the chosen task
3. Estimate all the rest of the task relative to the one above, considering: three factors: **Complexity**, **Uncertainty**, **Repetition**

Story points **do not reflect the size** of a task

Improving estimation or getting back on track

- Use velocity as a guide
- Use reference stories as a benchmark
- Use retrospectives as a learning
- Use calibration as a validation
- Use refinement as a preparation

Estimating stories

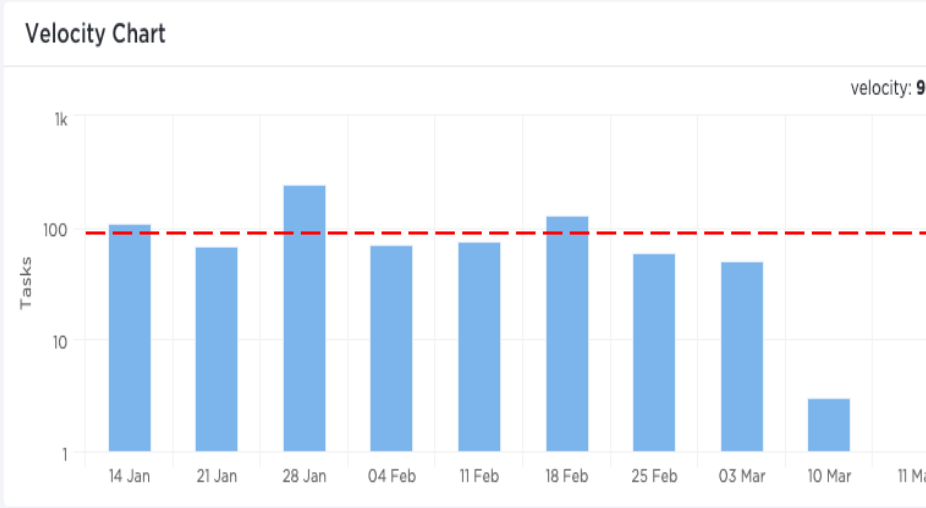
Estimation is a **key skill** for Scrum teams, as it helps them plan, prioritize, and deliver value to their customers.

However, estimation is also challenging, as it involves dealing with **uncertainty**, **complexity**, and **variability**.

.

Use velocity as a guide

Velocity is the **average amount of work** a team can complete in a sprint, measured in story points or other units.
It is **not a goal, a commitment, or a measure of productivity, but a guide** to help the team forecast how much work they can take on in the next sprint.



Velocity chart - tracks the amount of work completed from sprint to sprint, helping to determine team's velocity and estimate the work the team can realistically achieve in future sprints.

Use reference stories as a benchmark

Reference stories are stories that the team has already **completed and agreed on their size and complexity**. They can be used as a benchmark to compare and estimate new stories, by using a technique called relative sizing.

Relative sizing is based on the idea that it is easier to **compare two items** than to assign a precise value to one. Reference stories get **frequently updated based on feedback** from the team and the product owner, such as new insights, learnings, or refinements.

Reference stories examples	**1 SP**	**2 SP**	**3 SP**	**5 SP**
Front End	1: Display content in a field 2: Remove label from a screen	1: Change 1 step in a flow 2: Display and download	1: Display, Delete and Download files 3. Edit form	
Back End	Add properties or values to an existing list of values	1: Add property to an exiting entity / flow 2: Extend metadata configuration	1: New API for.. 2. Create and view entity ...	Upload completed flow to storage
FE + BE		1: Display user message at a certain step determined in BE 2: Add property and display in FE	Create entity and display to user in an existing page	

Use retrospectives as a learning

Retrospectives help the team use **historical data and feedback** to identify **patterns, gaps, or biases** in their estimation, as well as to experiment with new techniques, tools, or approaches.
For example, the team can use retrospectives to review their velocity, their reference stories, their planning poker sessions, and their sprint outcomes, and to agree on action items to improve their estimation accuracy and reliability.

Examples of estimation learnings in retros:

- **Split** stories larger than 8 SP using enablers
- Account **dependencies** in estimation
- Account for **testing complexity**
- Update reference story
- Always review during estimation **negative scenarios and error handling**
- Account for documentation update, test suite updates, regression testing, if required
- Define / update **Definition of Ready**
- Have a common understanding of when a story is considered done - **Definition of Done**

Use calibration as a validation

Calibration is a process of **validating and adjusting your estimates** based on the actual results of your work.

Calibration helps the team use historical data and feedback to compare their estimates with their actuals, and to learn from their deviations.

For example, the team can use calibration to track:

- their **estimate accuracy** ratio (the ratio of actual hours to estimated hours)
- their **estimate error** rate (the percentage of stories that were overestimated or underestimated), and
- their **estimate variance** (the difference between the highest and the lowest estimate for a story).

The team can also use calibration to **adjust their estimates for future sprints**, based on their learnings and feedback.

Use refinement as a preparation

Refinement is a continuous activity where the team and the product owner collaborate to **clarify, split, prioritize, and estimate** the product backlog items.
Refinement helps the team use historical data and feedback to prepare for the sprint planning, by ensuring that the backlog items are ready, valuable, estimable, small, and testable.

The **Definition of Ready** defines the *criteria that a specific user story has to meet **before*** being considered for estimation or inclusion into a sprint.

The **Definition of Done** for the Scrum Team and it is used to *assess when work is complete* on the product Increment. In short, DoD is a ***shared understanding within the Scrum Team on what it takes to make your Product Increment releasable.***

Definition of Ready for a User Stor

- User Story defined
- User Story Acceptance Criteria defined
- User Story dependencies identified
- User Story sized by Delivery Team
- Scrum Team accepts UE artefacts
- Performance criteria identified, where appropriate
- Person who will accept the User Story is identified
- Team has a good idea what it will mean to Demo the User Story

Definition of Done

The below examples might be included in a User Story DoD:

- Unit tests passed
- Code reviewed
- Acceptance criteria met
- Functional Tests passed
- Non-Functional requirements met
- Product Owner accepts the User Story