

Course 4 - Agile Project Management

Inginerie Software

An universitar 2022 - 2023

Agenda

- *Project & Roles*
- *Estimations (sprint, release, project)*
- *Stakeholder management & communication in general*
- *Risk assessment and mitigation*
- *Monitoring and reporting*
- *Process improvements & practices*
- *Q&A*

Define the project

WHAT

A **project** can be defined as a temporary endeavour undertaken to accomplish a unique product, services or results.

The Following are the attributes of Project:

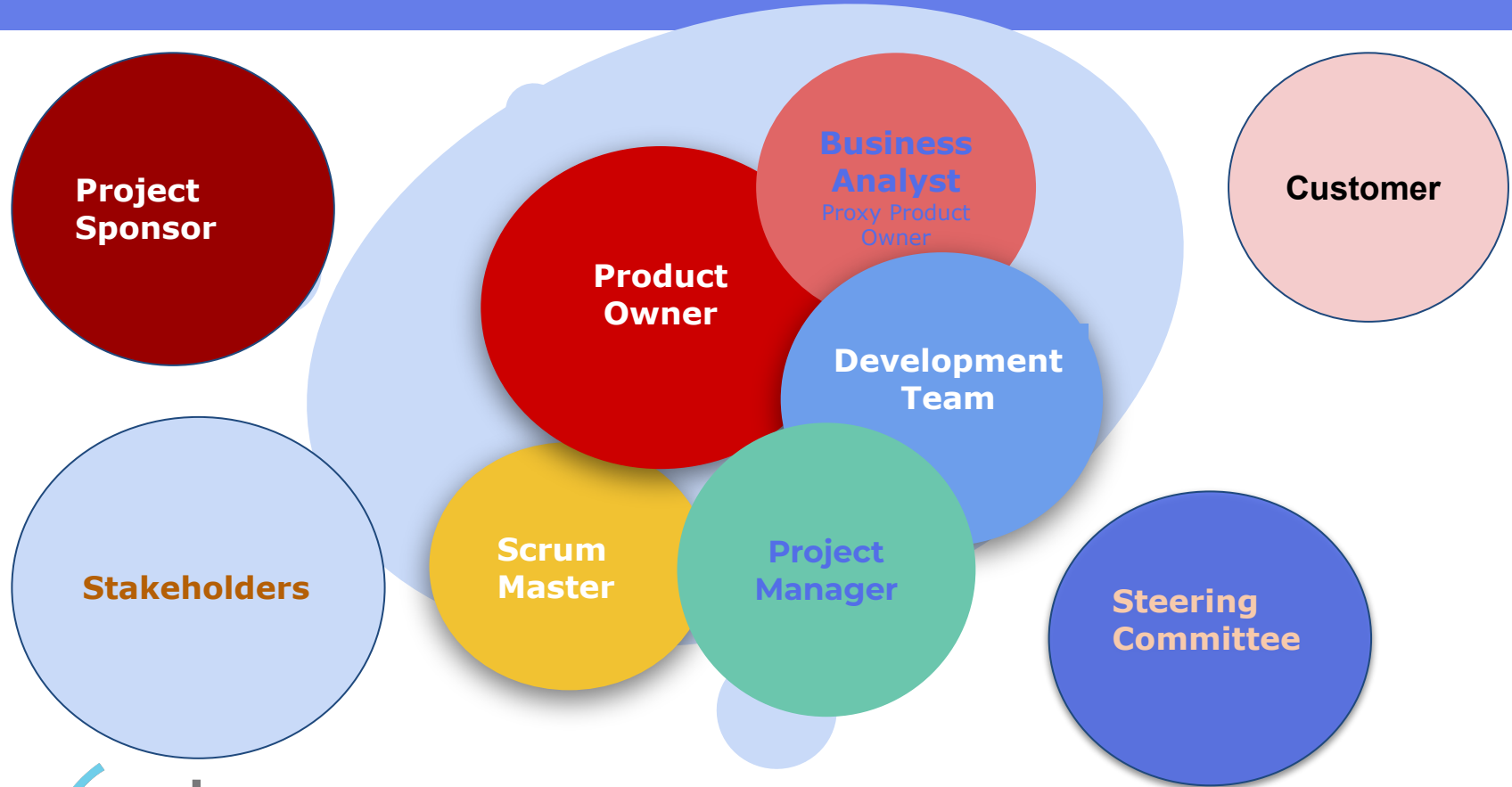
- a. Time frame**
- b. Purpose**
- c. Ownership**
- d. Resources**
- e. Roles**

HOW

Using formal, specific template to state the project characteristics

- Project Charter -> project objectives, scope, vision, team and their responsibilities and stakeholders, budget, timeline & milestones
- Roles and responsibilities - *RACI is deprecated in Agile*
- WBS - work breakdown -> Themes, Initiatives, Stories
- RAID Matrix - risk, assumptions, issues and dependencies
- Project Communications Plan
- Project Schedule -> **timeline with start dates, end dates and milestones** or Roadmap

Define the roles



Project roles - PM

The Project Manager

Is responsible for planning, organizing, and directing the completion of specific projects for an organization while ensuring these projects are on time, on budget, and within scope.

Project managers play the lead role in planning, executing, monitoring, controlling, and closing out projects. They are accountable for the entire project scope, the project team and resources, the project budget, and the success or failure of the project.



Estimation

Agile project estimations

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.

Velocity based estimation

In order to map a story points to a unit of time when a certain feature is done we can use the **Sprint Velocity** that can be measured over time.

Sprint velocity is the number of story points that can be completed during a sprint by a specific team.

The future velocity can be predicted by analyzing the team's historical data (i.e., keeping track of how many story points were completed in a previous sprint).

At the end of each iteration, the team adds up effort estimates associated with user stories that were completed during that iteration.

Velocity per sprint = 30 points

Feature = 60 points

Feature = 2 sprints

This is a way to map the more abstract comparison of story points back to time, through team velocity.

T-Shirt size estimations

In T-Shirt sizing Agile estimation technique, the items are estimated in standard T-Shirt sizes: **XS**, **S**, **M**, **L**, and **XL**

Features T-Shirt size can be determined by:

- Number of cards (stories)
- Number of story points
- Number of sprints to deliver
- Days to deliver

Examples:

XS - 1-3 cards | 1-5 SP | < 1 Sprint | 1-5 Dev days

S - 3-10 cards | 8-40 SP | 1-2 Sprint | 7-20 days

M - 10-30 cards | 40-120 SP | 3-4 Sprint | 20-40 days

L - 30-50 cards | 40-120 SP | 5-6 Sprint | 40-60 days



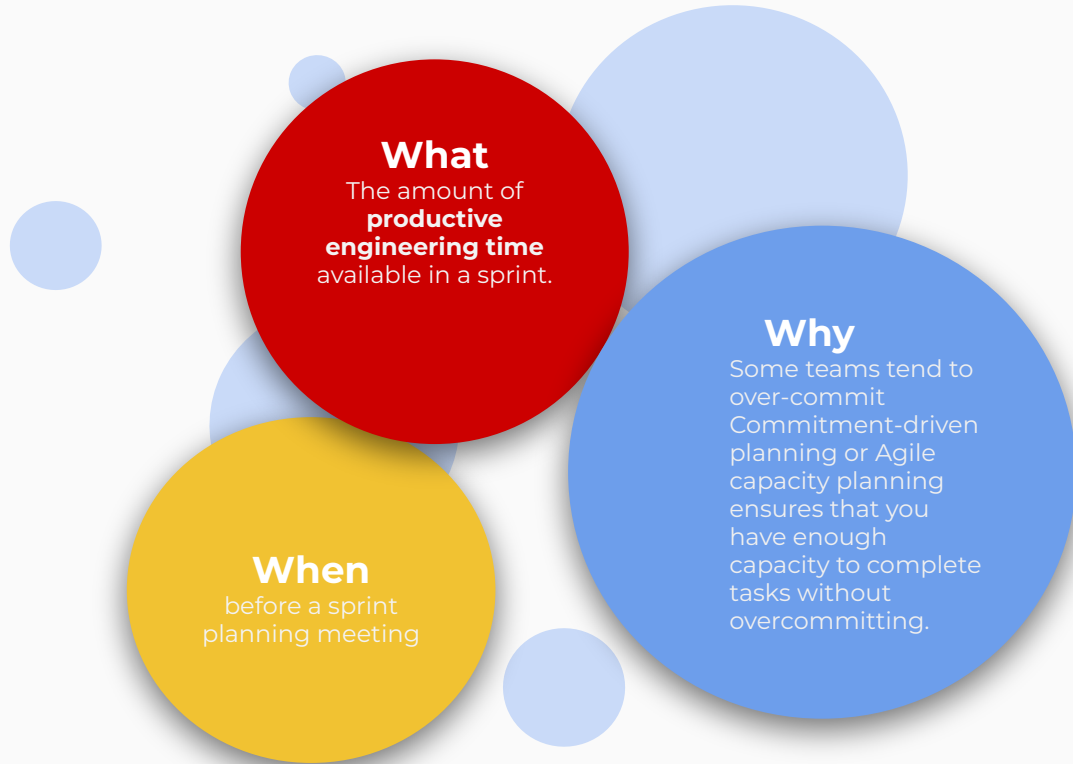
Backlog estimation

Technique	Description
Story Points/ Planning Poker	gamified technique that development teams use to guess the effort of project management tasks. <i>Hand out the cards => Read the story => Discuss=> Estimate and share => Work toward consensus</i>
T-Shirt Size	Relative Estimation Technique. Items are classified into t-shirt sizes: XS, S, M, L, XL. <i>Decide on sizes => Align on what each size means => Decide who assigns sizes => Assign t-shirt sizes to each initiative => Track t-shirt sizes => Use t-shirt sizing to gauge workload</i>
Affinity Mapping	each story is grouped according to similar complexity. Each group is then assigned a value, whether a size or a number, creating a scale. The scale is unique to the team as each team has different skill sets and is capable of achieving various things
Three-Point Method	Three-point Estimate (E) is based on the simple average and follows triangular distribution. $E = (O + M + L) / 3 \text{ or } (O+P+4M)/ 6$ <i>Story Breakdown => for each story find three values – most optimistic estimate (O), a most likely estimate (M), and a pessimistic estimate (L) => Calculate the Three-point Estimate for each story => Calculate the Three-point Estimate of the project => Calculate the Standard Deviation of the project</i>

Backlog estimation

Technique	Accuracy	Suited for	Description
Story Points/ Planning Poker	Ground	Stories	Estimating complexity Small number of cards Mutual understanding of size Late-stage estimation + Backlog Highly-prioritised
T-Shirt Size	Ground/ Forest	Stories/ Epics	Estimating complexity or cards Running rough estimations The team is new to Agile estimation Running early-stage estimations + large backlogs
Affinity Mapping	Forest/ Sky	Epics/ Initiatives	Estimating a long-term plan for a project Gaining mutual understanding in the team Running early-stage estimation in large backlogs
Three-Point Method/	Ground	Stories	Time or complexity: $(O+P+M)/3$ or $(O+P+4M)/6$ The team is new to Agile estimation Running later-stage estimations with highly prioritized backlogs

Capacity planning




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!



I	independent
N	negotiable
V	valuable
E	estimable
S	small
T	testable

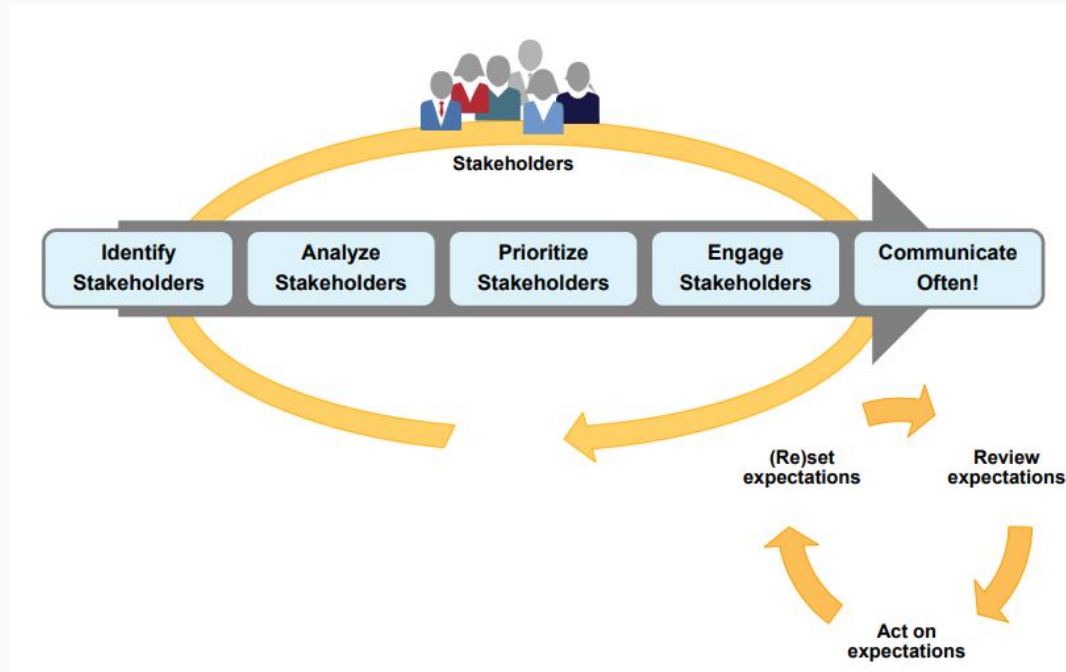
Use a Reference story

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

Stakeholder management & communication

The term **STAKEHOLDER** refers to those who have an interest and influence to impact the product, program, team or project.



Communication - ways to interact with stakeholders

SCRUM events

- Sprint **Review** - *visibility and opportunity for feedback*
- **Daily** Scrums - invited when required
- Stakeholder **Retrospectives** (*quarterly*)
- *Additional Feedback sessions*

Educational Initiatives

- **Dashboards** - **not reports!!!** - visual, *aggregate information* across teams
- **Release Notes** - tell a story, not list tickets
- Trained **Ambassadors** - liaison between Product and Eng.
- **Training** Classes

Regular Meetings

- Product **roadmap** [planning and review]
- **User story mapping**
- **Q&A sessions** in the form of **Lean coffee**
- Observing/ Operationally in Stakeholders Dep.

Risk assessment and mitigation

Risks identification

A **risk** event is something identified in advance that may or may not happen.

Risk factors:

- **Probability** → How likely the risk is to occur
- **Impact** → What's the impact or amount at stake
- **Timing** → When it is expected to occur
- **Frequency** → How often will risk events occur from the source

Risk identification

The process should be driven by PM but needs to implicate all team members so that they can take ownership for the identified risks.

Risk identification will target the following areas: Strategic | Operational | Contractual | Technical | Infrastructure | Resources

When?

- During project offer
- At the beginning of the project
- During project monitoring phase
- Agile methodology integration: **Daily / Refinement / Planning / Review / Special meeting type** - Risk assessment meeting

How?

There are many techniques identifying risks, but the most important one should be brainstorming. This is done with a group of people that knows the scope of the project or have experience in similar projects.



Risk assessment and mitigation

Risk management

The purpose of risk management is to identify potential problems before their occurrence so that we can prepare a plan that can be monitored across the project life cycle to reduce the probability of the risk happening, eliminate or transfer the risk.

Risk management is a continuous process that focuses on identifying and assessing the risks to the project and managing those risks to minimize the impact on the project.

Where to look for risks

- Project background
- Initiation documents
- Human Resources
- Scope, time, cost plans
- Quality & Procurement
- Communication & Stakeholders

Strategies for risks tracking and mitigation

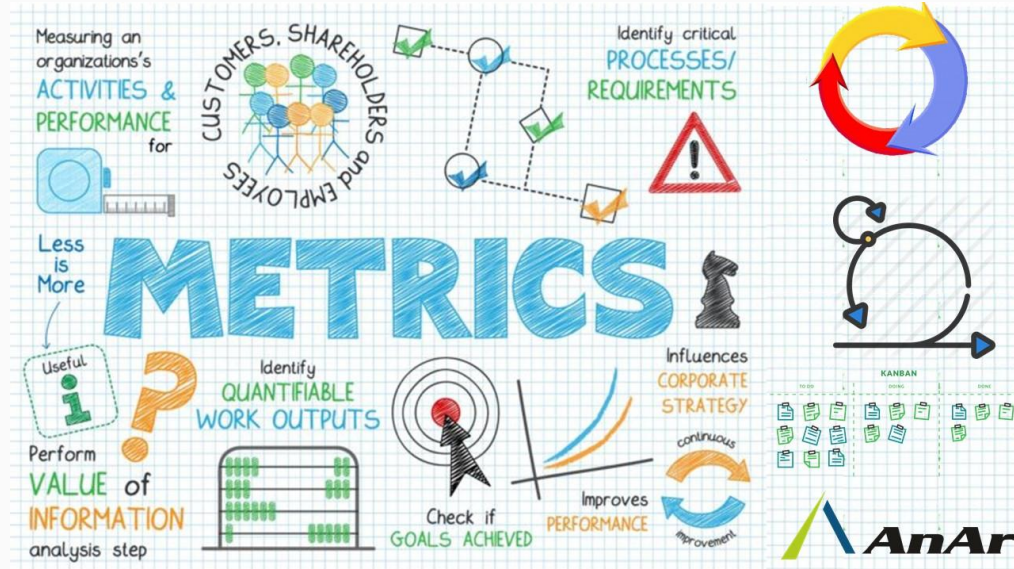
- Accept
- Avoid
- Transfer
- Mitigate

Risk category	Risk description	Risk effect	Risk probability	Risk impact	Risk Score	Risk owner	Risk response technique
Infrastructure	APIs from client and stage environment may not be ready on time	Impact on Time	30% Low probability	0.1 Very low	0,03 Low	PO, PM	Mitigate
Resources	Developer allocations are not complete, team is unbalanced	Impact on Time and Budget	50% Mid probability	0.7 High	0,35 Medium	PM, AM, EM	Mitigate
Contractual	We have a payment plan of once a month with penalties if we do not deliver as promised	Impact on Budget	30% Low probability	0.7 High	0,21 Medium	PM, TTL	Mitigate
Operational	Design is provided by Client with text that needs modifications	Impact on Time / Budget / Scope	70% High probability	0.7 High	0,49 High	PO, PM, Design Lead	Mitigate

Monitoring and reporting

Agile Metrics

Agile metrics are an essential component of the development process. For companies or teams that work on the agile framework, agile metrics help in assessing software quality.



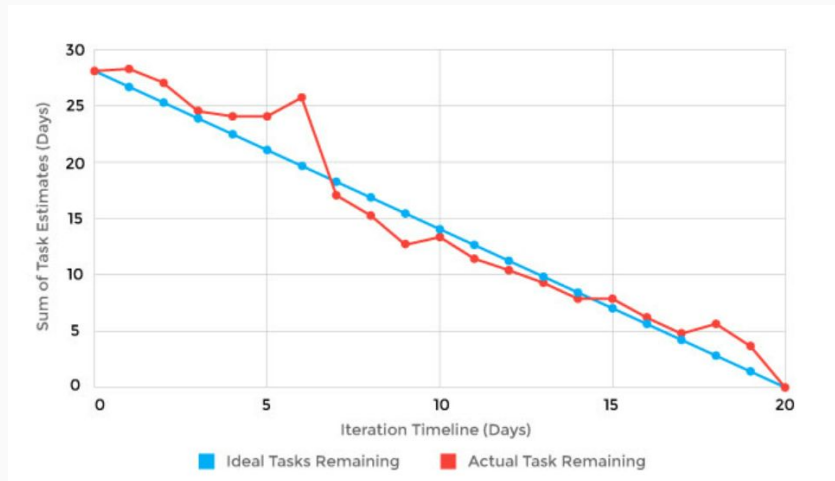
Agile Metrics - Sprint Burndown

Agile Scrum teams organize their processes into sprints. Since a sprint is time-bound, it's important to track task progress frequently.

A sprint burndown report is for tracking the completion of different tasks during a sprint. Time and work left to complete are the two main parameters of measurement in this case.

The X-axis refers to the time. The Y-axis represents the work left. The unit of measurement is hours or story points.

The team forecasts the workload at the beginning of a sprint. The target is to complete the workload by the end of the sprint.



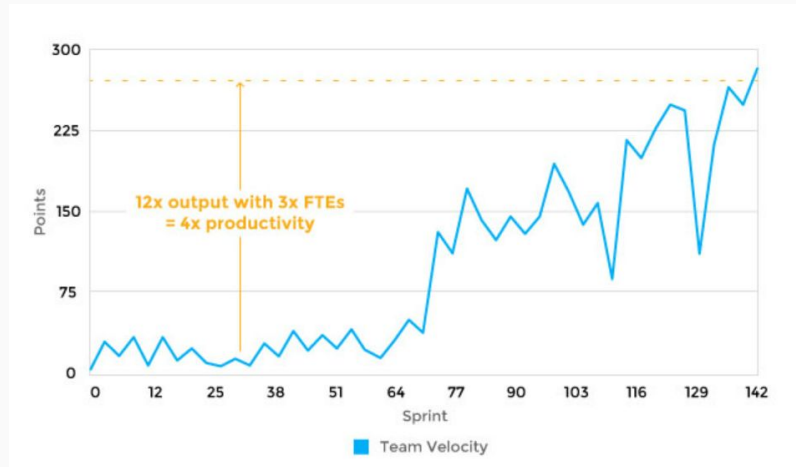
Agile Metrics - Velocity

Velocity measures the average work a team does during a sprint.

The report, in this case, contains several **iterations**. The accuracy of the forecast depends on the number of iterations. The more iterations, the more precise the forecast.

The unit of measurement is **hours** or **story points**. Velocity also determines the ability of a team to work through backlogs. As time passes, velocity tends to **evolve**.

To ensure consistent **performance**, it's important to track velocity. If the velocity declines, it's a sign that the team needs to fix something.



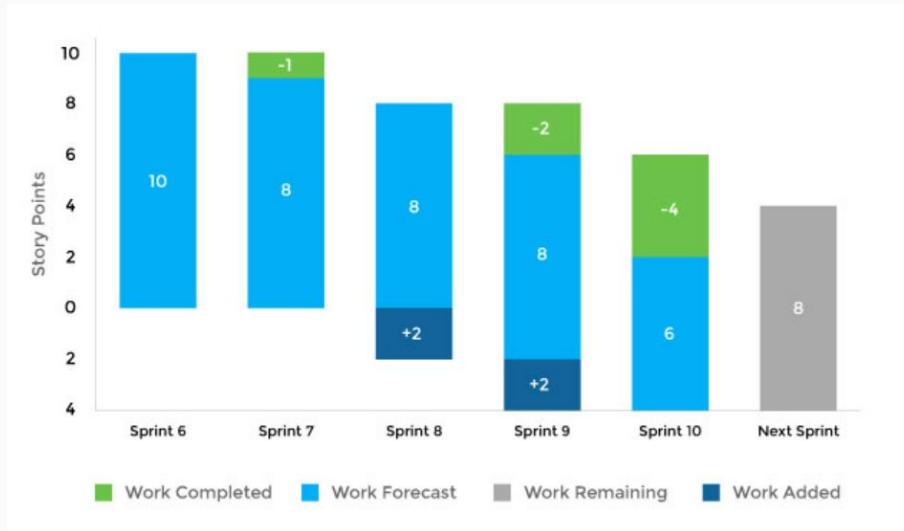
Agile Metrics - Epic and Release Burndown

Unlike a sprint burndown, **epic and release burndown** focus on the bigger picture.

They track progress over a **large work body**. There are many epics and versions of work in a sprint. So, it's important to track their progress as well as each sprint.

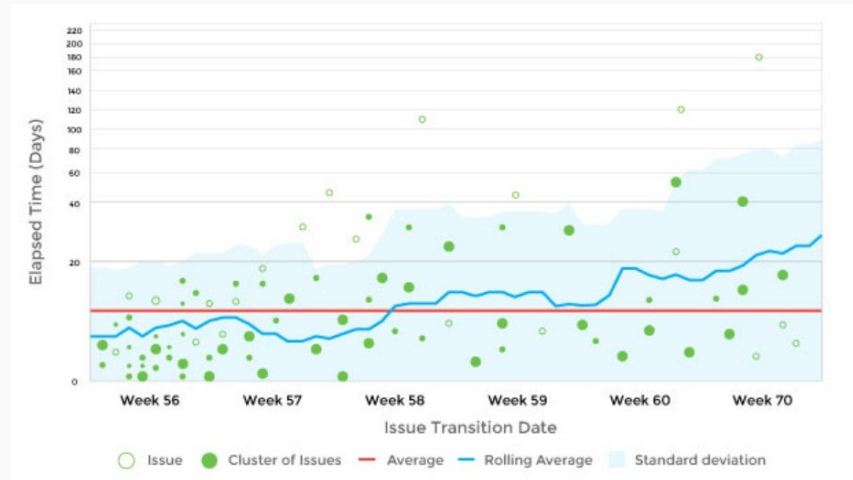
The entire team has to be aware of workflow in the epic and version.

Epic and release burndown charts make that possible.



Agile Metrics - Control Chart

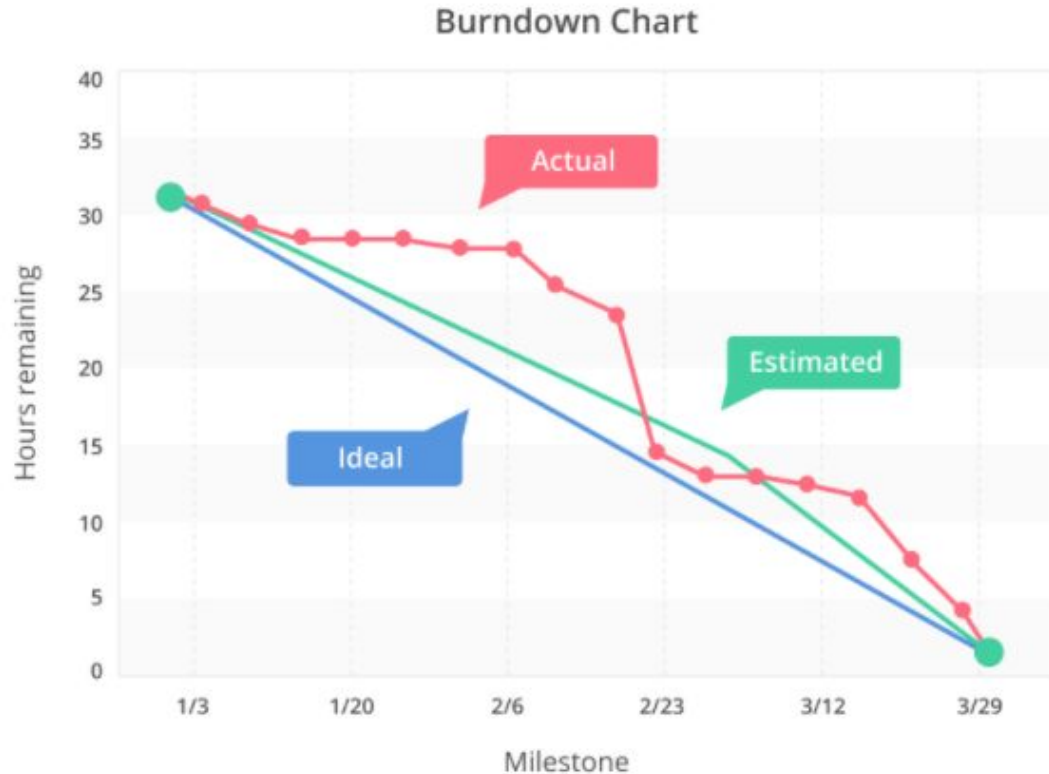
Control charts focus on the time duration from the “in progress” to “complete” status of tasks. Their purpose is to check the cycle time of a single issue. When teams measure cycle times, they improve the flexibility of their processes. For instance, in the case of changes, you can discern the results instantly. As a result, team members can make the necessary adjustments. In general, a short and consistent cycle time is the target to achieve in every sprint.



Scrum Artefacts - *Tracking progress*

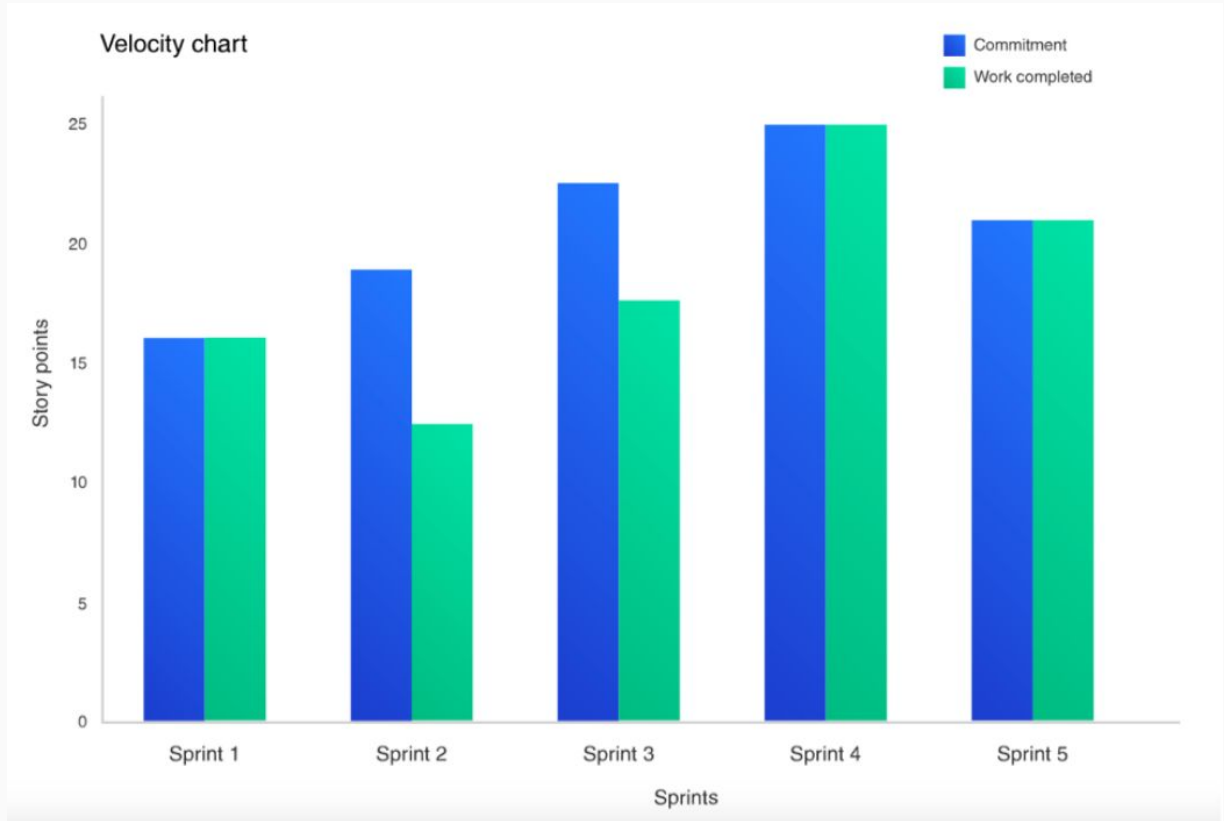
Burndown chart

- tracks the total work remaining and projects the likelihood of achieving the sprint goal, helping the team to manage its progress and respond accordingly.

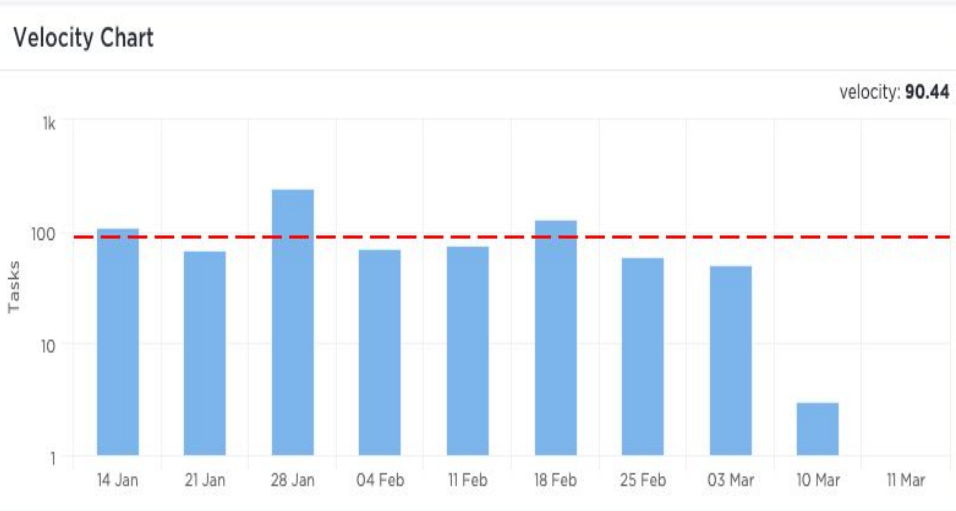


Scrum Artefacts - *Tracking progress*

Velocity chart - track 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.

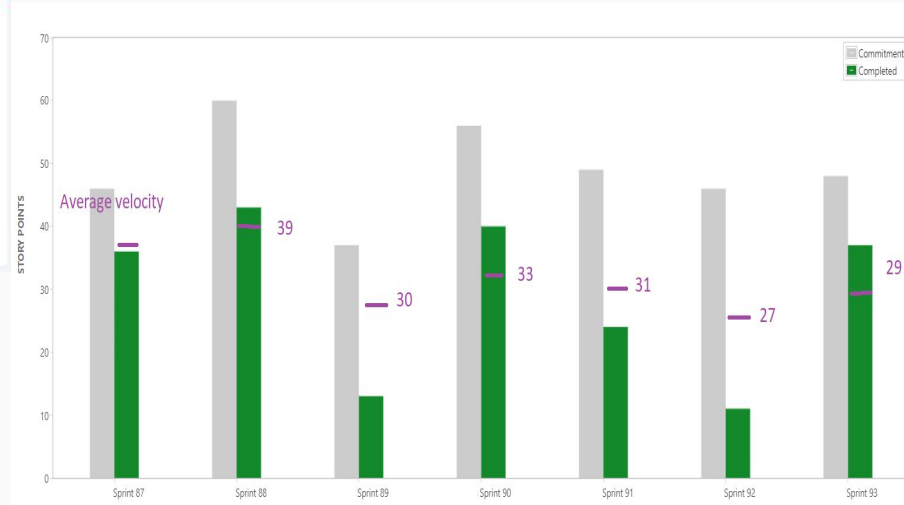


Reading velocity reports

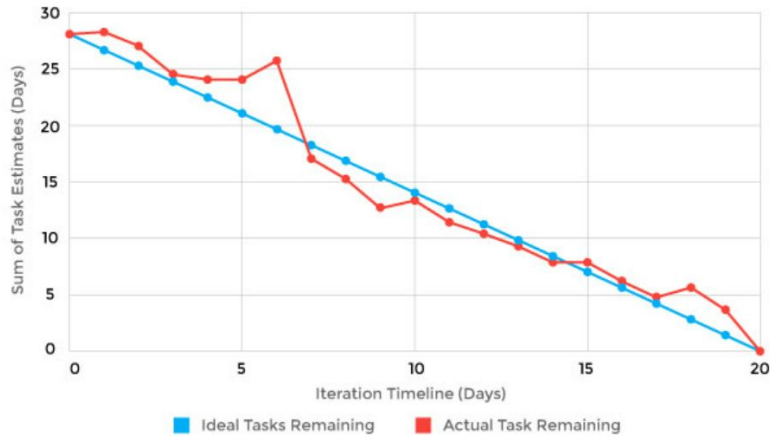


Velocity may actually be **relatively stable** in a successful and **well-established team** as the amount of raw effort available for each Sprint remains constant

Newly formed teams will have a velocity graph that may **seems chaotic**, but a good metric here is to look at the **average to see if it increases**, if not analyse the cause



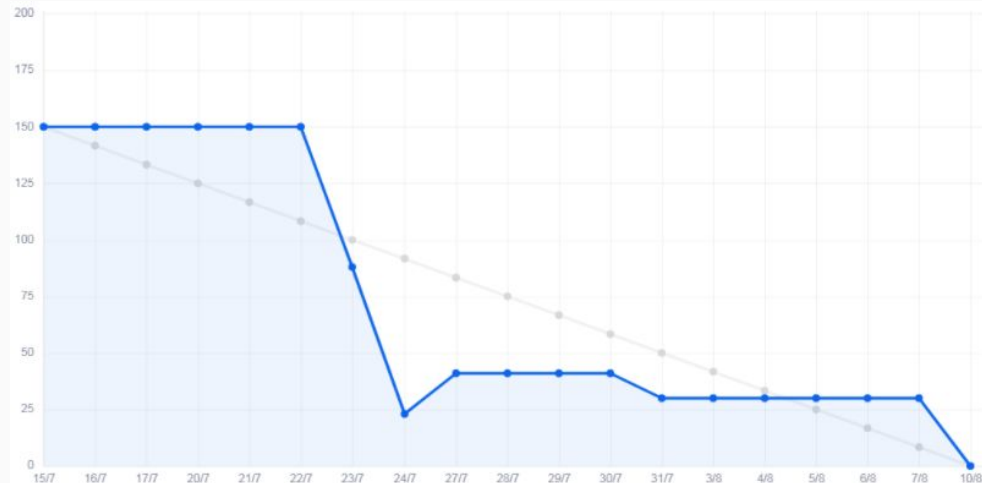
Reading burn-down charts



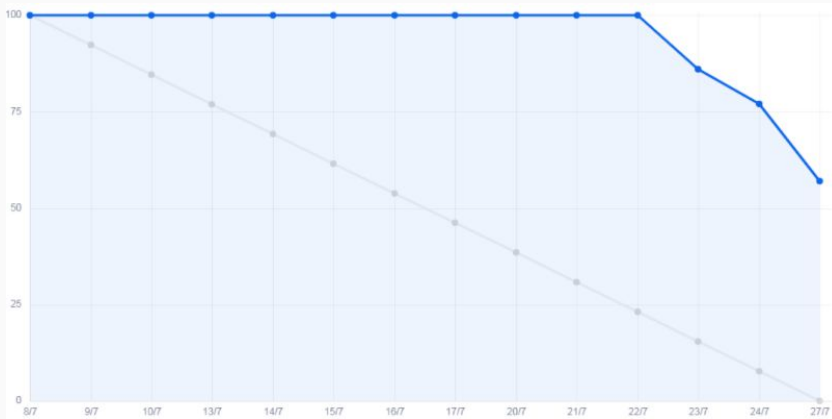
How to Read Burndown Chart

1. **On track:** the two lines are close together
2. **Behind schedule:** progress line is above the guideline => team should complete more tasks
3. **Ahead of the schedule:** progress line under the guidelines => team will reach target before the end of the sprint

work done by an experienced team. The team can **complete tasks on time** and **reach the sprint goal**. What is more, it **can adapt to the backlog**.



Reading burn-down charts



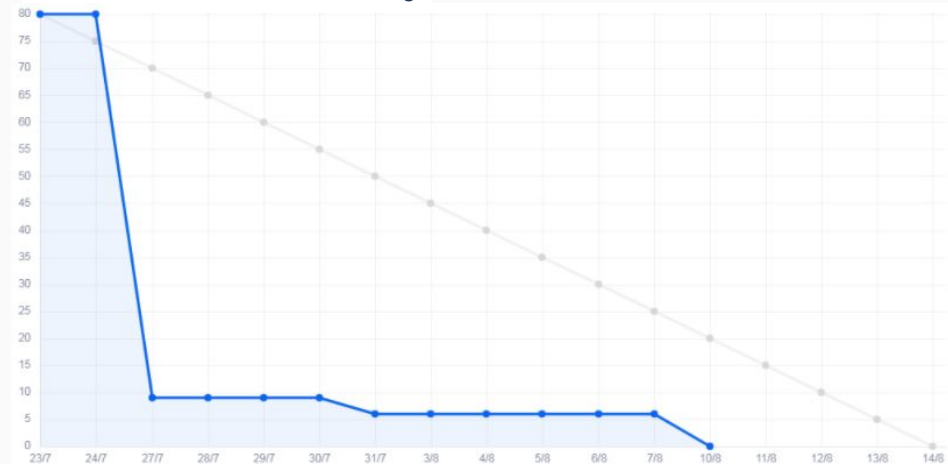
“Too-late” team

- team did not finish the task.
- failed to adjust their work and there are user stories waiting to be finished
- they should be further split and moved to the next sprint.

“Too-early” team

- team finishes faster than expected
- team might be able to do more user stories
- user stories may be overestimated

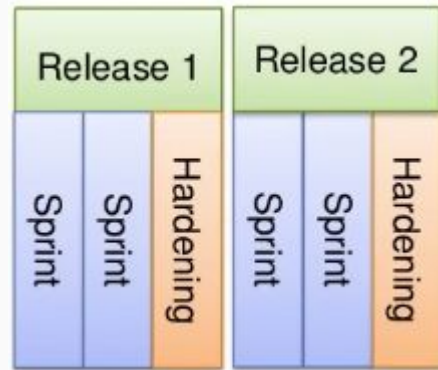
This reveals that team velocity was not estimated reasonably.



Release preparation

What is a release

- release is a **change or set of changes** that is created to be **delivered to the production environment**. There is a release process that makes this possible, and a release package that contains the code and artifacts needed to make the desired change.
- For each project, it's important to ensure everyone understands what is meant by 'deploy' and 'release'. Often, people prepare a release, deploy it to production and then release it to customers (some process frameworks, such as SAFe, specifically decouple deploy from release for this reason).



A hardening sprint is **used by Quality teams to finalise a product for release**. This often requires project team to stop working on new features and all available resources are involved in stabilising the release. *Hardening sprint length may vary from dev sprint length.*

It is not part of any Agile frameworks.

Good practices to sustain Agile teams development

Processes

- Collaborate with the customer - **Refinement** and **Review**
- Work together daily - **Daily, Pair programming, Code-review**
- Build projects around motivated individuals - team **empowerment**
- Convey information **face-to-face**
- Form self-organizing teams - let the **team drive the process**, not the other way around, **good design practices**
- Reflect on how teams can become more effective - **Retrospective, continuous technical improvements**

Practices

- Continuous **integration**
- **Automation**
- Well known **branching policies**
- **Light release documentation**
- **Test-driven development**
- **Code refactoring**
- **Simple code design**
- **Pair-programming**
- **A common codebase and a single coding standard.**

Main advantages

- In Agile Project Management you **deliver projects faster**
- It fosters an environment of constant **collaboration, teamwork** and focused on **execution**
- You have **direct input** from your customer throughout the whole project. He can start using it even after the first iteration
- You constantly **reflect for improvement** during Retrospectives
- Agile Project Management shifts from traditional long documentation processes and long meetings to **agile, lean processes, practices and rituals** such as the daily standup
- In Agile Project Management **roles blur**, meaning everyone can participate on different tasks regardless of whether it's their "official" role or not. *For example, not only testers test, but others too (if the teams decides that)*
- Agile Project Management allows you to focus on value and the minimum required working solution known as the **MVP** (Minimum Viable Product). So you apply a basic/yet really important concept all the time: keep it simple!!! *Don't build a Ferrari if all you need is a skateboard!*
- **Visual boards and reports** allows everyone to stay on the same page and be able to understand the speed of execution, areas of improvement and more

Fastest growing tech company in Romania by Deloitte Fast
50

Course 4 - Agile Project Management

Q&A