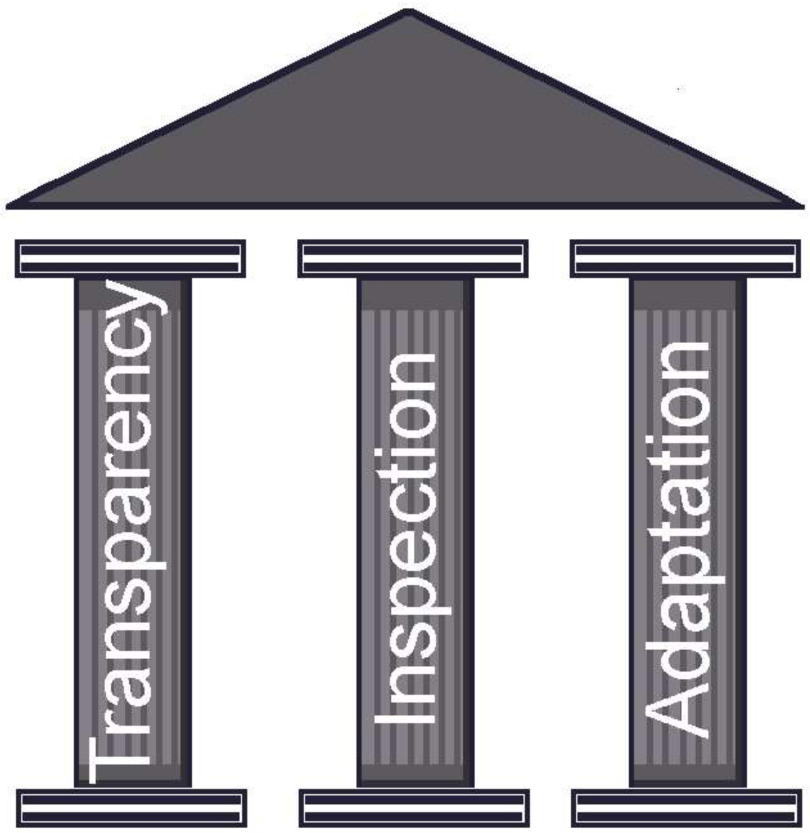
Chapter 2

Scrum Theory

There are three pillars that represent the empirical process control within Scrum theory, which are transparency, inspection and adaptation.

Scrum Pillars



Transparency

The Scrum artifacts are designed to be transparent for a good reason; to allow anyone with an interest in the product and Sprint to be able to analyse and give feedback. Every single person in the Scrum Team should be able to know **Why** they are doing what they are doing, **What** impediments there are and **Where** they are going.

Inspection

Scrum also incorporates checkpoints that give the opportunity for inspection of the process, product and anything else that can be improved. The events included in Scrum are suitable times for Inspection, where discussion and collaboration is encouraged (within the reasonable time boxes of course!).

Adaptation

Through Transparency and Inspection, people are able to learn, constantly, improve themselves and their contribution to the product. They are able to be more confident with forecasting their efforts and releases and more importantly, they learn from things that went wrong, to prevent them from happening again in the future.

The three pillars are the essence of Scrum, and as you continue reading, you will notice how everything revolves around these three pillars of **Empiricism**.

Scrum Values

- **Commitment** - Commitment to the Sprint Goal, to the work that the Team wants to get done, to the Team, to delivering value and to continuous improvement. Following through on what you wish to achieve helps with building trust with team members and stakeholders.
- **Focus** - Focusing on one thing at a time means delivering an outcome with higher quality. Scrum does not recommend having Development Team members working on different products, Sprints or teams concurrently.
- **Openness** - Impacts trust between members of the Scrum Team and also between the stakeholders and the organisation. It induces collaboration and creates transparency around challenges that are met during product development. It's important for the Scrum Team to be open and honest about their tasks, accomplishments, impediments and progress.
- **Courage** - Courage to speak up, expose impediments and blockers, to remain transparent, to communicate errors and to trust others.

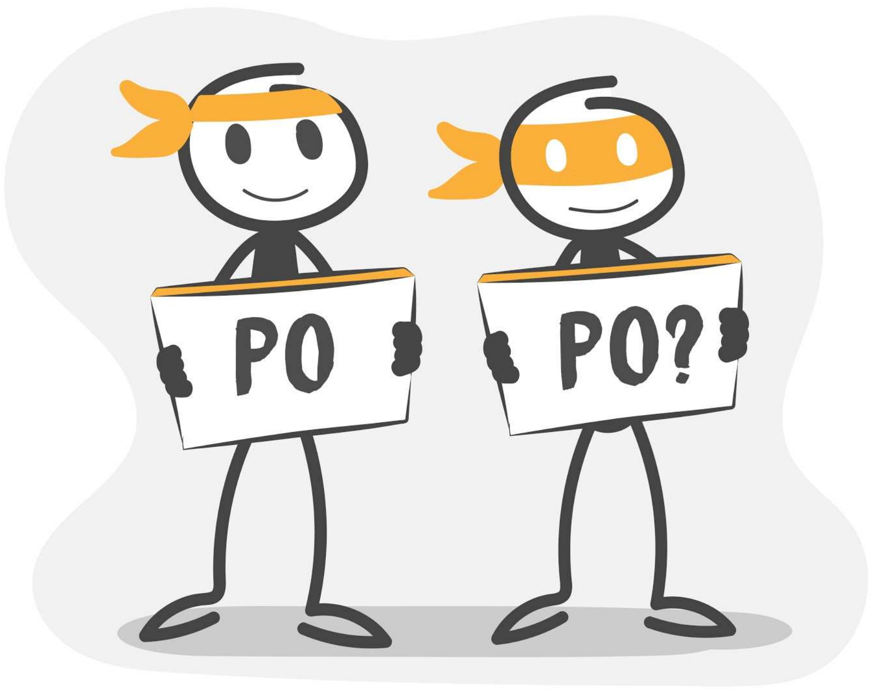
- **Respect** - For example, respecting and following up on the decisions that were taken as a team, respecting their estimations without pushing them to work long hours and burning out. Respecting colleagues and each other affects employee behaviour as a whole, builds a solid team culture and also improves employee happiness and success.

These values are not just needed in the context of Scrum, but in any functioning organisation.

Chapter 5

The Product Owner

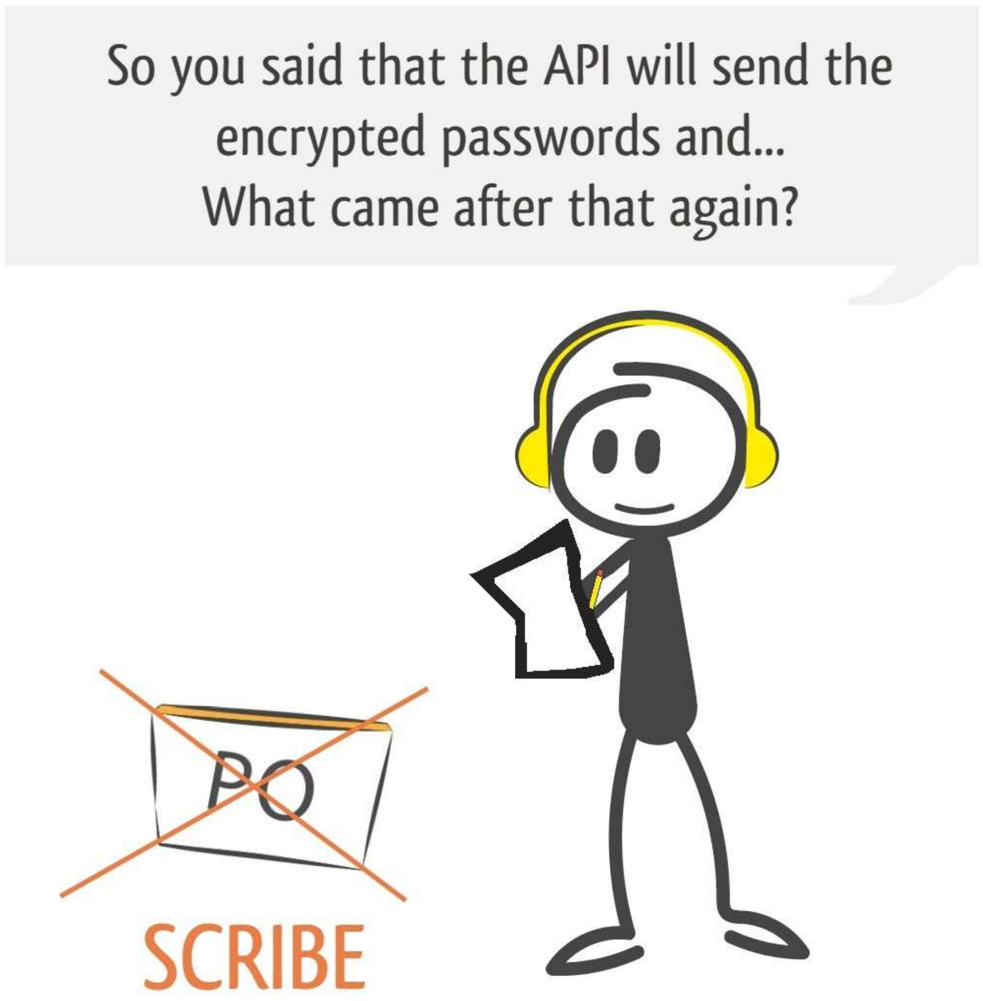
A commonly disputed topic is the role and responsibilities of a PO in Scrum. In order to remove any misconceptions that you may be familiar with and disassociate them from the Scrum definition, it's important to define a PO that isn't fit for purpose (i.e. the imposter PO).



A PO does not manage teams

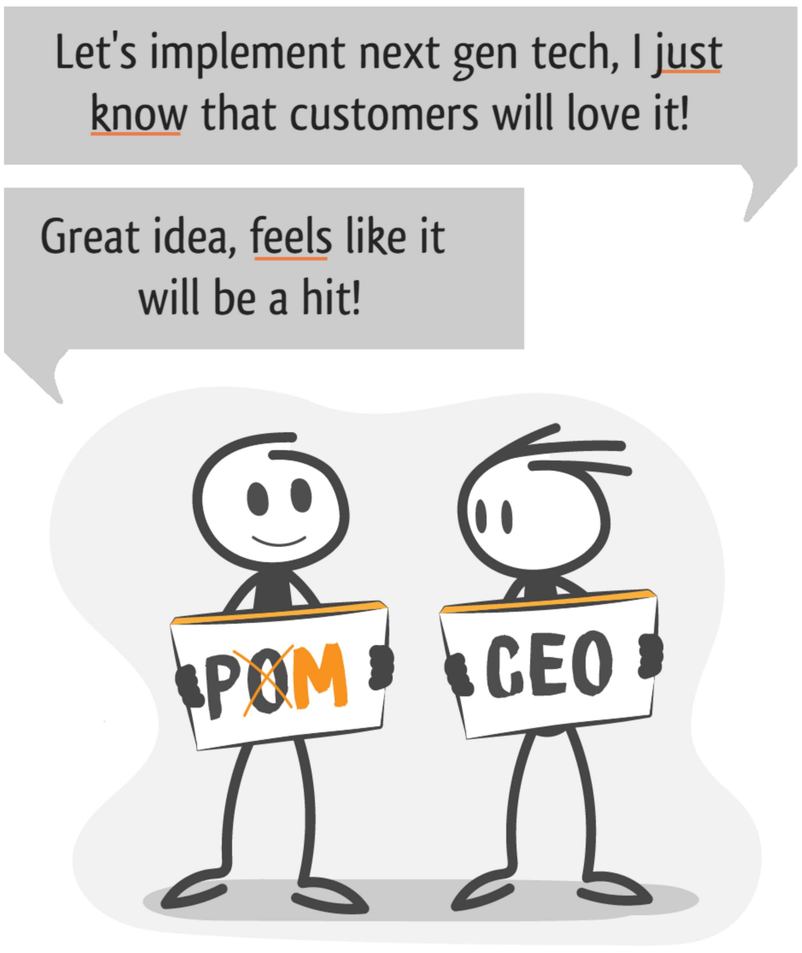
In Scrum, the Development Teams are self-organising - which means that they, alone, are held responsible for deciding how the work gets done, estimating development effort and moving the Product Backlog items to their Sprint Backlog. Essentially, the PO manages the Product and prioritises the features of the product through the Product Backlog.

A PO is neither a scribe nor a proxy


So you said that the API will send the encrypted passwords and...
What came after that again?
PO
SCRIBE

What is a Scribe PO?

Someone that spends the majority of their time writing requirements, user stories and product documentation, without taking responsibility for the content and its formation. PO responsibilities for


Let's implement next gen tech, I just know that customers will love it!
Great idea, feels like it will be a hit!
P~~O~~M
CEO

Note that the stakeholder does not question the CEO for empirical evidence to support this suggestion.

Thoughts and Predicaments

The CEO suggested shifting our focus toward a particular feature of the product - but Sales have barely been affected. What value did this feature really bring? What data was used to support the CEO's decision?

If the Team's velocity has only been going higher in the previous sprints - then why are we losing market share?

Think a little about these questions and ask yourself whether you can think of any similar situations that you have come across in your organisation.

Digging Deeper

Release Planning

Planning the release is all about strategising and negotiating product delivery. However, the higher the project uncertainty, the higher the probability of having an inaccurate release date. Before concentrating on the method used for release forecasting, consider the following factors before planning a release:

Development Uncertainty - As discussed before, in the *Cone of Uncertainty,* uncertainty greatly impacts the planning process. In earlier Sprints, tools, processes, technologies to be used and functionalities may still have gray areas. This means that dependencies, specialised skills, license/tooling costs, infrastructure and architecture decisions still need to be taken. The outcome of these decisions will influence the entire product timeline. Initially, focusing on a Minimum Viable Product as an early release will also test these key decisions.

Estimation Errors - A release plan takes estimates into consideration, but it should also take estimation errors. Depending on the number of sprints the team has already run in the past, data for the estimate vs the actual amount of effort spent on a feature, will give an indication of the error rate of estimations. The more data available (the more Sprints this specific Development Team has finished together), the more reliable the error rate of estimations. Knowing that, if the error rate has typically been +/- 20% of estimations in previous Sprints, this buffer can also be applied to the release date.

Integration - Functionalities sometimes need to undergo an integration process before being usable to the market. Examples include integrating with third party APIs or even integrating with a client's platform. Integration is a common dependency to having a shippable product and can act as a bottleneck. Examine the following questions if integration is needed:

- Is all the integration data available to the Scrum Team? Are endpoints, test cases, data and so on available in early Sprints?
- Could changes to the third party platform cause significant changes (and delays) to the delivery of the increment?
- Is personnel availability required from the third party to integrate successfully?

Dependencies - Any dependencies that can influence the projects: