How To Use the MoSCoW Prioritization Method?

WHAT IS THE MOSCOW METHOD?

The MoSCoW method is one of the most popular prioritization techniques to establish what is most important to clients and stakeholders. By using this method, stakeholders can better understand the importance of different features in a release. It is extremely quick and simple to apply as a prioritization solution, classifying features in four different priority buckets: Must Have, Should Have, Could Have, and Won't have.

Let's break them down:

Mo	**Must have**	These are the essential features that need to be included into the product. Failing to include one would result in a failed release.
S	**Should have**	They are important requirements but not essential. They are initiatives that are of great importance and add significant value, but are not crucial.
Co	**Could have**	These are nice-to-have initiatives, as they don't quite affect the core function, and would have a very small impact if left out.
W	**Won't have**	These are definitely not a priority for the foreseen timeframe, and therefore will not be included in this specific release. In other words, they are out of scope.

WHEN AND WHY SHOULD I USE IT?

The MoSCoW model sets your initiatives by order of priority, and can therefore be applied to any phase of the product life cycle. However, it is most applicable to product launches, market launches, particularly early stage products and MVPs.

This is a good method to get the whole organization involved in the prioritization process, which in turn creates a broader set of perspectives by getting different departments involved.

Effort is another reason why you'd want to apply this method. Using it will enable your team to quantify the amount of effort allocated to each feature or initiative, resulting in the right combination of features per release.

HOW DOES THE MOSCOW METHOD WORK?

In order to run the MoSCoW method smoothly, your product team and stakeholders need to decide on the objectives and factors that will be decisive to the criteria. This will be immediately followed by reaching a consensus on what initiatives or features you'd like to select.

Setting these ground rules is of extreme importance — in particular how to settle disagreements — as this can become serious bottlenecks down the line.

Lastly, define how much effort should be split between the Must-Haves, Should-Haves, and Could-Haves. This typically varies by team and project, but a rule of thumb suggests that you should dedicate about 20% of your total effort to Could-Haves.

Now your team is ready to sit down and discuss your initiatives. Let's look into them:

MUST-HAVE INITIATIVES

The category name doesn't come as a surprise as they will be the lifeblood of your product or release. These are non-negotiable features. Without them, your release could be a guaranteed failure. You should reach an agreement on how much time and effort you spend on your Must-Haves —you should focus on them, but shouldn't allocate more than 60% of overall effort.

! Ask your team

- Will this project work without this feature?
- What happens if we release without it?
- What's the simplest way to accomplish this?

SHOULD HAVE-HAVE INITIATIVES

Being just under the Must-Haves in importance, they are still highly important to the product but not crucial. The product will still manage to function without them. On the other hand, you wouldn't want to leave them out as they generate a significant amount of value.

To put it into perspective, you should include them in the release, but you could schedule them for a future release without having a negative effect on the current one.

COULD-HAVE INITIATIVES

These are nice-to-have initiatives, meaning that they are not necessary. They add to the product and generate value to the user, but they are not exactly a core component or function of your product. There would be no repercussions if you were to leave them out.

WON'T-HAVE INITIATIVES

These initiatives are still important to take into account. You should always identify them as it'll help the team decide what will not be included in the scope, thereby allowing them to prioritize other initiatives. They prevent you from wasting resources that your team needs for this release.

Lastly, the subgrouping of these is also beneficial. Perhaps there are Won't-Have initiatives that will not be included within this scope, but could be included in the future, and others that simply won't be included at all.

Why we love it

- Gets business-side stakeholders involved in the feature prioritization process.
- Powerful and simple way to prioritize with time boxes.
- Highly based on the expert opinion of the team, both technical and business side.

A few downsides

- Often prone to bias from managers worried that their initiatives will fall into “should” or “could”, which is exacerbated by bad KPIs.
- Deciding where your initiatives will fall often becomes a never ending discussion when team members have different levels of familiarity with the product.