agileader

TOP PRIORITIZATION METHODS

THAT WILL ELEVATE YOUR PRODUCT TO ANOTHER LEVEL agileader

Throughout my professional career I have seen multiple products. Most of them had one thing in common. Their Product Owners used the simplest possible prioritization technique - Stacked Ranking. They have just ordered product backlog items relatively one to another and hoped that their 'experts opinion' would be good enough.

In many cases, this approach was one of key reasons why those products struggled to succeed.

You don't have to experience such a failure by yourself!

Try one of following prioritization techniques in order to ensure your product maximal value in shortest time and its great success.

Kano Model

I think that this is the best technique for product road map and backlog prioritization. Period.

As this method is customer oriented, it provides really accurate insight into your market. Its core element is quite simple. This method explores functional and dysfunctional aspects of customers attitude toward your features and how they are going to address customer needs. You have to ask your customers two questions with potential answers ranging from 1 to 5.

Question 1 - Functional

How would you feel if **you have** this new feature?

- 1 - I like it that way
- 2 - It must be that way
- 3 - I am neutral
- 4 - I can live with it that way
- 5 - I dislike it that way

Question 2 - Dysfunctional

How would you feel if you **don't have** this new feature?

- 1 - I like it that way
- 2 - It must be that way
- 3 - I am neutral
- 4 - I can live with it that way
- 5 - I dislike it that way

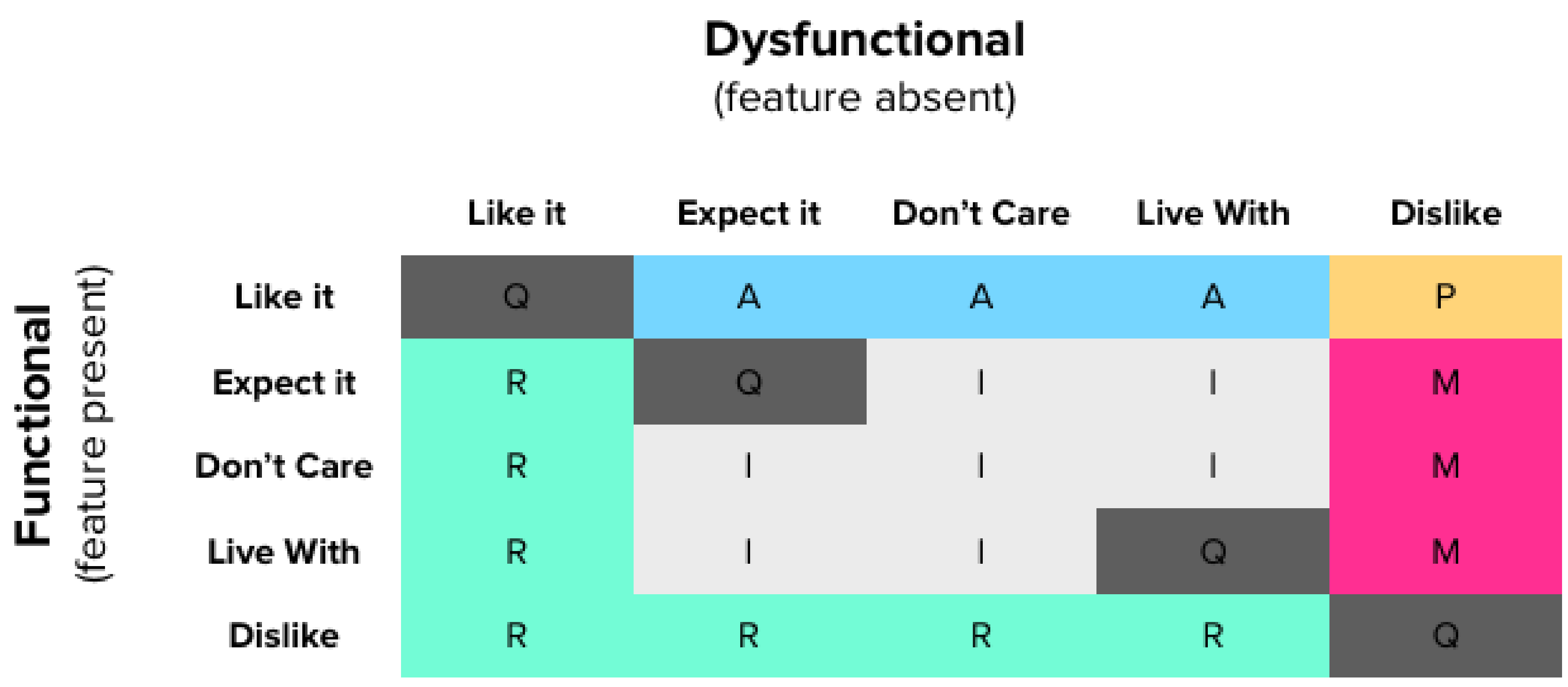
By looking at these two aspects, you can easily detect features that are within following five* main groups. To describe them I will be using an example in which we are proposing to our potential customers to buy a new car.

5* groups of features

- **Must-Be**
  - These features are expected by customers and if you don't provide them, customers might be dissatisfied.
  - Example: You can open doors to your car.
- **Performance**
  - The better such features perform, the happier your customers are.
  - Example: Fuel efficiency.
- **Attractive**
  - Positive addition to your product that customers didn't expect.
  - Example: Extra set of tires as a gift when they buy the car.
- **Indifferent**:
  - Customers don't care about these particular features regardless if your product has it or not.
  - Example: Ultra fancy screen saver for car's display.
- **Reverse**
  - Your market reacts negatively to your new feature and you should better amend it to do exactly an opposite. Alternatively, do not implement this feature until you have a chance to redesigning it.
  - Example: Taking away driver's seat to make the car lighter.
- **(*)Questionable**
  - These answers indicate that the person, that you have spoken to, does not fully understand either questions themselves or benefits that the feature provides. You should rephrase your questions and ask again. Until that, you should ignore such answers when prioritizing features.

Here is how answers for main two questions are related to those five* groups:

Kano model - results distribution [1]



Dysfunctional (feature absent)

Functional (feature present)	Like it	Expect it	Don't Care	Live With	Dislike
Like it	Q	A	A	A	P
Expect it	R	Q	I	I	M
Don't Care	R	I	I	I	M
Live With	R	I	I	Q	M
Dislike	R	R	R	R	Q

M - Must-Be; P - Performance; A - Attractive; I - Indifferent; R - Reverse; Q - Questionable

Priorities

Order of those main five groups is important. You should always start a development from features that have the highest score in ‘Must-Be’ group. When all ‘Must-Be’ features are done, you can go to ‘Performance’ group and, at the end, to ‘Attractive’ one. You shouldn’t bother yourself to work on ‘Indifferent’ features unless they are necessary for other, not-customer-related reasons (compliance, security etc). Redesign 'Reverse' ones before the implementation.

To conclude, when you will be preparing your questions for selected features, you should design them through the prism of benefits that those features can provide to users. Avoid just listing dry facts about features' implementation. It is easier for your customers to judge their feeling about your product's functionalities if they see how those feature impact them. Additionally, you should also tailor answers for your needs so that they work well in your context.

For more details, I recommend you to visit a comprehensive guide on Kano Model [2]:

LINK

MoSCoW

This lightweight technique can be used as starting point for many Product Managers. Take your stakeholders or/and customers and ask them to divide your product's planned features into four main categories:

- **Must have**
  - You will have to implement all of these.
- **Should have**
  - These features are still important and should be implemented but they are not urgent so they can be done slightly later.
- **Could have**
  - These are 'nice to have' if time permits.
- **Won't have**
  - You shouldn't even bother doing these features.

When you have all results aggregated, you need to conclude them. If you are using this method as part of onsite workshop, try to find the consensus with your stakeholders before the session is over so that everyone agrees to place each of features into single group. If you had used this method offline, you can just average the value out.

As you can see, this technique is pretty simple, but comes with a cost. You cannot differentiate feature priority within each of groups. You will have to find this out by talking to your stakeholders to better understand rationale behind their decisions.

If you want to read more about this method, feel free to check posts like this one [3]:

LINK

Story Mapping

Jeff Patton described this technique in his book: "User Story Mapping: Discover the Whole Story, Build the Right Product" [4]. From my perspective, this is great method to learn about the concept of MVP - Minimal Viable Product [5].

Story mapping in general can be considered as lengthy approach for prioritization but it produces plenty of output. You will get a good break down of user stories prioritized into increments / releases that will last for long. You can use it, for example, to kick off new initiatives within your product. Also, consider hosting this activity on site with physical board and sticky notes. However, as the plan generated using this method spans across multiple iterations, you can find it outdated quite quickly if you work in highly dynamic environment.

Your goal is to take one dimensional product backlog and present it using two axes.

Horizontal axis

On horizontal line, you need to plot detailed sequence of actions that user needs to perform in the case that you want to prioritize. You should put those actions just below the axis on the board, from left to right, in chronological order. Try to find related actions and group them together - to represent that, place cards above the line and we will call them Activities.

Vertical axis

Vertically, you are going to visualize the break down of each of written down actions. Take each of them and split it into tasks and user stories. Place them, in columns, below their 'parent' action. Then, it's time for the prioritization. You need to order cards so that the higher the item is on the board, the higher priority it has. To do that, you can use other techniques like MoSCoW.

Iteration Scope

At the end, you should review how to divide your map per iterations. Think through the prism of the whole user journey = all activities. Keeping that in mind, select which tasks and user stories should be part of first iteration / first release. Draw a line to separate them. This is your first MVP that you will deliver! Take the next set of tasks and create scope for next iteration. Continue this process until you have no more tasks to choose from. As a result, you have perfectly planned out scope for your product.

To learn details of this method, you can read blog posts like the one bellow [6]:

LINK

Weighted Shortest Job First

This method (a. k. a. WSJF) is one of key artifacts of SAFe methodology. It helps to ensure that Product & Solution Managers focus on the economic aspect of their product when they prioritize features for the development. Weighted Shortest Job First relies on two main factors:

- **Cost of delay**
  - Amount that you lose for every moment a feature is not available to be used.
- **Job duration**
  - Any metric that shows how long would it take to develop a feature.

Cost of delay

To calculate 'Cost of delay' for each of features, you need calculate three parameters:

- **User business value**
  - How significant relative value does this feature provide to customers?
- **Time criticality**
  - Would the value decrease if we wait?
- **Risk reduction /opportunity enablement**
  - Would we lose less if we have this feature? Can we get extra benefit from this feature? Does this feature enable us other opportunities?

During the feature prioritization, you will be taking features and, for every of factors mentioned above, you will assign them a value from predefined scale. The SAFe recommends using quasi-Fibonacci scale: 1, 2, 3, 5, 8, 13 ,20.

Start with 'User business value' factor and select a feature that should have the lowest score in this category. Give it a '1'. Then, score relatively remaining features assigning them appropriately larger values. When you finish with the 'User business value' factor, do exactly the same for remaining two parameters. To calculate the 'Cost of delay' of a given feature, use the following equation:

Cost of Delay = User business value + Time critical + Risk reduction

Job duration

Unless you have any historical data, I recommend to use the same approach as with 'Cost of delay' components to score the 'Job duration' of your features. Assign relatively values from 1 to 20 for each of prioritized items. However, when you spot a feature which would require a 13 or 20, you can consider splitting it into smaller features. Then, you will have to review the 'Cost of delay' for those sliced features.

Rule of a thumb - you will get much better results when all your features have relatively low size.

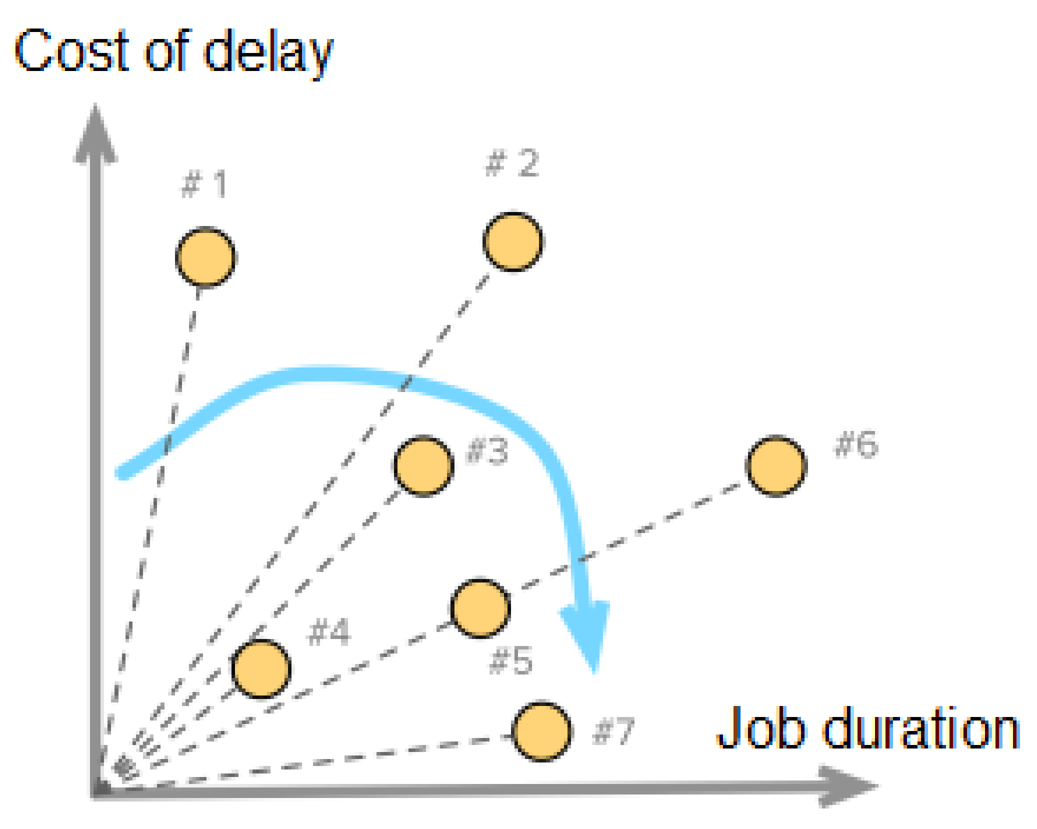
Weighted Shortest Job First

When all factors above are known, the WSJF of a feature equals:

WSJF = Cost of delay / Job duration

Now you can order your features basing on the final score - start with features that have the highest overall WSJF.

Value vs Cost [7]



To learn more about the SAFe or this technique, you can visit following website [8]:

LINK

Speedboat

This technique is more qualitative than the Kano model. At the beginning, you need to imagine your product as a "fast boat travelling thought the sea of the business value". The faster it goes, the more benefit it provides to users. The goal of the speedboat method is to ask your stakeholders and customers what are the biggest anchors, that they see, that prevent them from having optimal value?

The easiest way to conduct a workshop, using this approach, is to sketch a boat with some anchors. Then, ask your participants to write down problems that are slowing the boat the most on sticky note cards. Then, place them on anchors. You should accept if more than one card have the same problem written on in. On the card itself, stakeholders should also add their prediction how fast can the boat travel if the problem is resolved. When all participants have finished, your consider as your top priority items either:

- The mostly repeated problem.
- Cards with the highest sum of predicted velocity improvement.

Both of those options reflect issues that, overall, were the most painful ones for your customers. You should start building your road map from these problems.

For more information, about this visualization exercise, you can click here [9]:

LINK

Buy A Feature

A technique great for exercise when you can gather your stakeholders in a single location. In general, I consider this method more as an internal activity, but it is fully acceptable to use it with customers or users. The plan is to organize a market place of features. Here is how you can do that:

- Write down or print out cards with your features. You need to have tangible elements that participants will interact with.
- Each of features, that you want to prioritize, gets a price tag. This can represent how complex a feature is to develop or any other metric, as long as it is consistent for all features in the game.
- Every participant of workshops gets their play currency that they will be spending. However, make sure that this amount is less that the total cost of all features. So that people will have to carefully choose which cards they are interested in.
- After that, everyone uses money they have to "buy" features they want to. People can either write down feature numbers they are acquiring or get a copy of the feature printout if provided. Most importantly, there is no restriction that two people cannot buy the same feature. Hence, let's consider that you have infinite quantity of each feature.
- When all participants decide that they are not going to buy anything more, the workshop is over and you can check your results. Please keep in mind that it is acceptable to have unused money.
- To sum up, as your top priority items you should consider features that have been bought the most times.

Variations of the trading mechanism

There are two variants of the main part of the workshop:

- **Individual**
    - Everyone have their own shopping cart to take care of. Therefore, participants are not allowed to share their money.
- **Cooperative**
    - People can combine their cash to buy features together. For example, if one person has spare money, she can share it with a player that is short on budget.
    - Moreover, in this version, you can set some feature prices to exceed amount given per each person. That forces people to work together as nobody can buy the biggest feature by herself.

To conclude, this method can be conducted online - there are tools for that. Although, to achieve full potential of this technique, you should consider collocation so that people will be able to physically interact with cards and cooperate to buy optimal set of features. Also, this will provide you the chance to speak with your stakeholders and ask them why are they interested in buying certain features?

If you want to read more about this technique, you can find some details here [10]:

LINK

Conclusions

Each of techniques has pros and cons and can be applied in different situation. For customer facing products, you always have to focus on your users. I recommend using, for such products, more customer-oriented techniques like 'Kano model' or 'MoSCoW'. On the other hand, 'WSJF' and 'Story mapping' serve well in internal projects - even big ones. Finally, I really like to consider 'Speedboat' or 'Buy a feature' as great exercises if I want to do unusual workshops and activate my stakeholders.

Hyperlinks

[1] Evalulation Table
http://foldingburritos.com/wp-content/uploads/2015/06/EvalTable-Mod.png

[2] The Complete Guide to the Kano Model
https://foldingburritos.com/kano-model/

[3] MoSCoW Prioritization
https://www.productplan.com/glossary/moscow-prioritization/

[4] User Story Mapping: Discover the Whole Story, Build the Right Product 1st Edition
https://www.amazon.com/User-Story-Mapping-Discover-Product/dp/1491904909

[5] Minimum Viable Product (MVP)
https://www.productplan.com/glossary/minimum-viable-product/

[6] A Guide to User Story Mapping: Templates and Examples (How to Map User Stories)
https://plan.io/blog/user-story-mapping/

[7] Value vs Cost Prioritization
https://foldingburritos.com/product-prioritization-techniques/value-vs-cost/

[8] Weighted Shortest Job First
https://www.scaledagileframework.com/wsjf/

[9] Speedboat Technique
https://info.obsglobal.com/blog/2014/01/speedboat-technique

[10] How to play the 'Buy a feature' design game
http://www.uxforthemasses.com/buy-the-feature/