9. Requirement Engineering

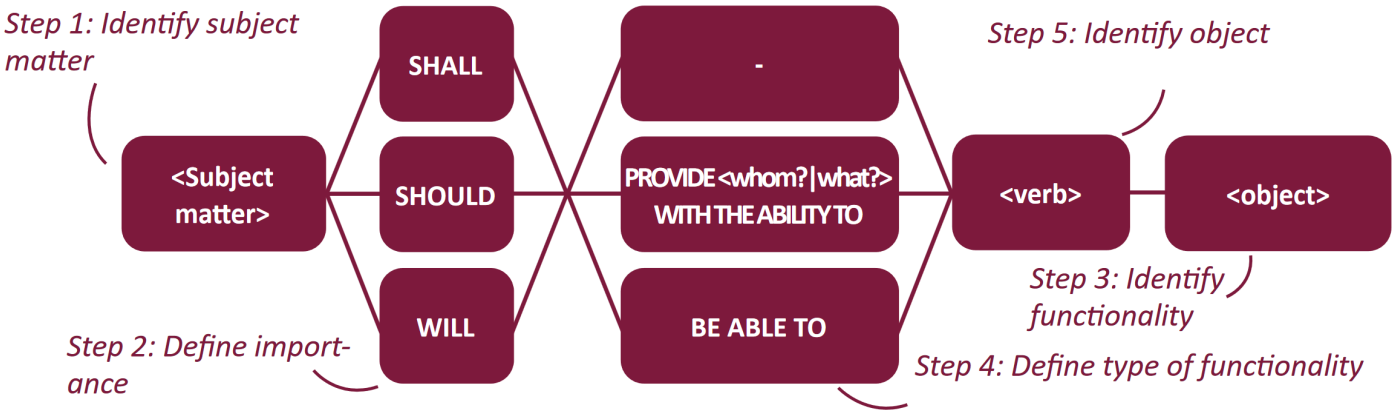
9.1 Requirements

As you have learned or learn in software engineering there are functional and non-functional requirements. We mainly focus on the functional requirements. For a real product requirements of all stakeholders are relevant, e.g. time constraints etc.

Requirement engineering is a course by itself. However, try to write down and prioritize all requirements learned during the user research.

9.2 Template for functional requirement

It is difficult to write clear and unambiguous requirements, the following template helps.



The Sophist: Requirement template

Or with an additional condition

Example

"If an error message has been generated, the system shall provide the administrator the ability to print the error message to the network printer."[1]

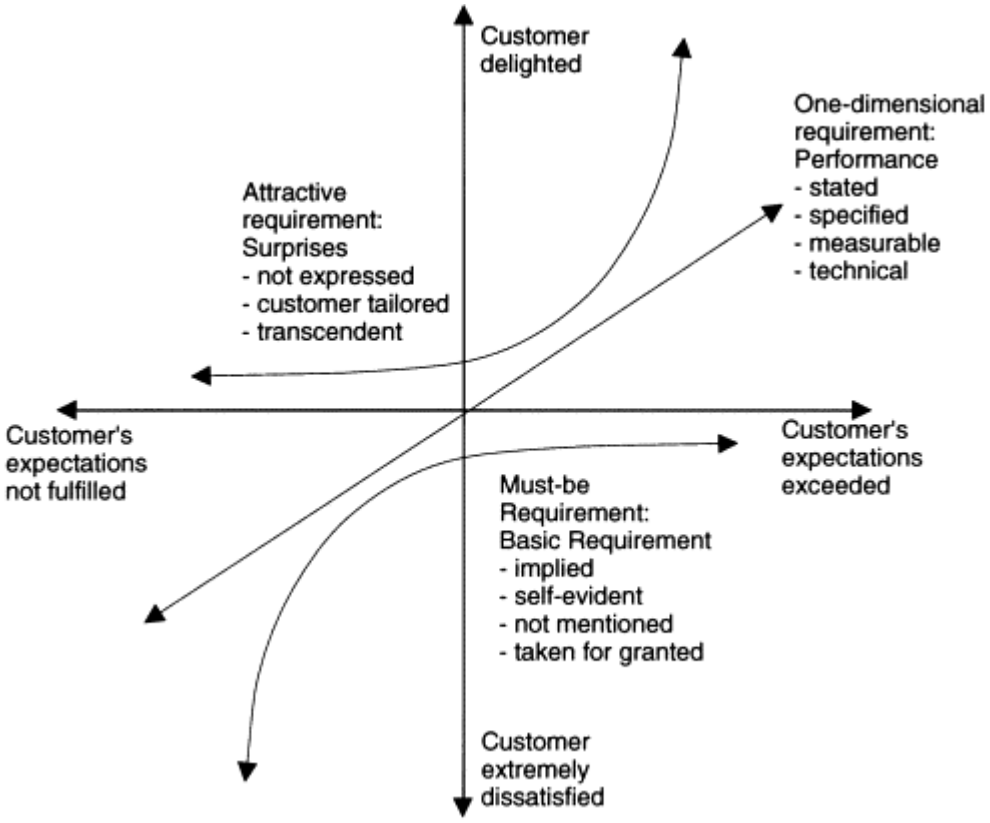
Here is a check list for clarity

- Are the requirements clear and unambiguous? (Are all aspects of the requirement understandable and not subject to misinterpretation? Is the requirement free from indefinite pronouns (this, these) and ambiguous terms (e.g., “as appropriate,” “etc.,” “and/or,” “but not limited to”)?)
- Are the requirements concise and simple?
- Do the requirements express only one thought per requirement statement, a stand-alone statement as opposed to multiple requirements in a single statement, or a paragraph that contains both requirements and rationale?
- Does the requirement statement have one subject and one predicate?

This list is very similar to ISO/IEC/IEEE 29148

9.3 Kano Model

The Kano model, created by Noriaki Kano in the 1980s, is a framework for product development and customer satisfaction.

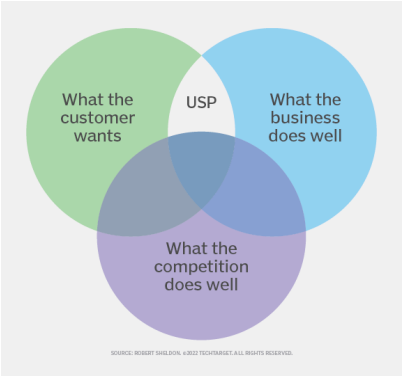


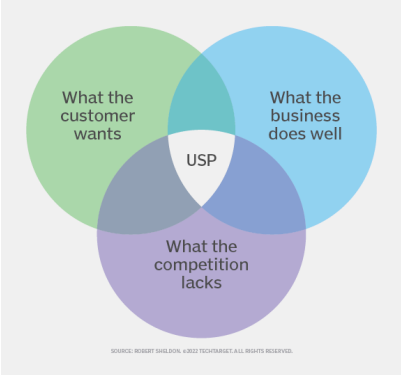
Kano Modell [^2]

Attractive requirements or delighters "are the product criteria which have the greatest influence on how satisfied a customer will be with a given product. Attractive requirements are neither explicitly expressed nor expected by the customer. Fulfilling these requirements leads to more than proportional satisfaction. If they are not met, however, there is no feeling of dissatisfaction." [2]

A "delighter" refers to a feature that creates customer excitement or *delight* when included in a product. These features go beyond basic expectations and can significantly enhance customer satisfaction. Delighters are considered unique innovations or surprises that generate an outsized positive response from users. A delighter sets your product or service above your competition. It might be your unique selling point (USP). However, not every USP is a delighter nor vice versa. While delighters enhance customer satisfaction by surprising and delighting users, USPs aim to differentiate a product in the market and attract new customers by offering something unique that competitors do not provide.[3] [4]

"A unique selling point can be thought of as *what you have that competitors don't.*"[5]





From Personas to MVP

9.4 MVP

Cite

A Minimum Viable Product is "*a version of a new product, which allows a team to collect the maximum amount of validated learning about customers with the least effort*".[6]

It is basically the first version of a product, which shows the main idea, contains a minimum set of features. Do not focus on tedious work, but ensure at least one delighter and your unique selling point is addressed properly.

The following approach might help to define, which features should be part of the MVP and which might be added in later versions.

Feature scoring rubric

-1 Hurts persona
0 Neither helps nor hurts persona
+1 Helps persona
+2 Critical for persona

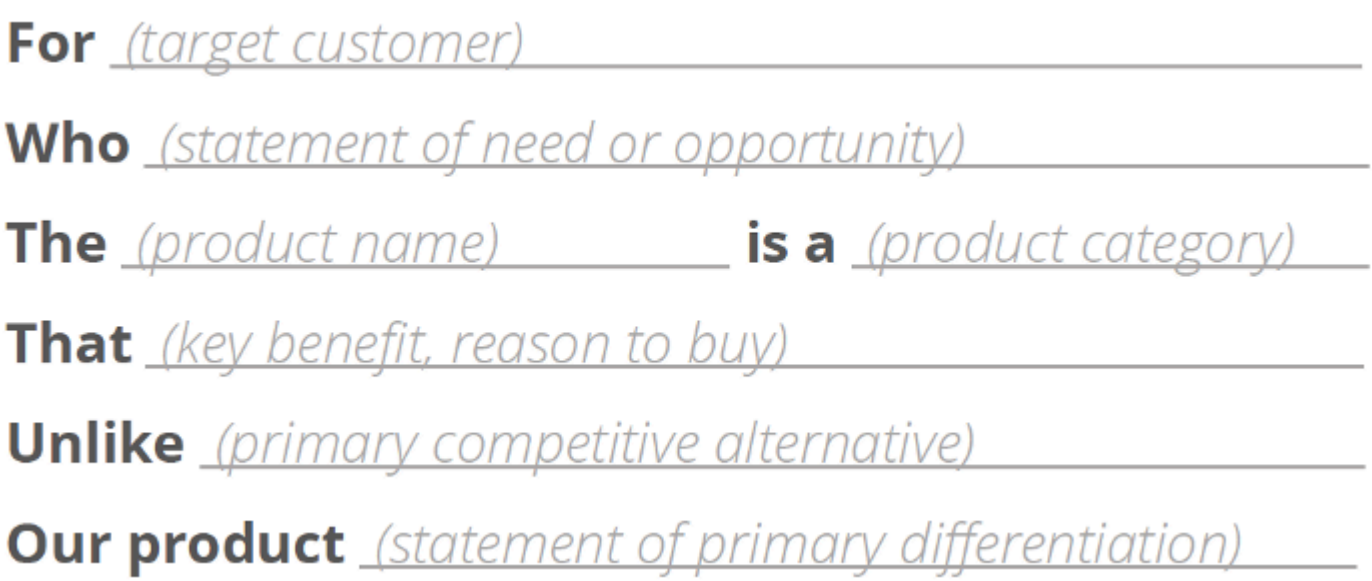
	Jessie 50%	Amit 30%	Jennifer 20%	
Feature / User story 1	2	0	-1	80
Feature / User story 2	0	1	0	30
Feature / User story 3	1	2	2	150
Feature / User story 4	0	-1	1	-10
Feature / User story 5	2	0	2	90
Feature / User story 6	-1	1	1	0
Feature / User story 7	1	2	1	130

From Personas to MVP

A more detailed discussion may be found in the article: Feature Driven Agile Product Innovation Management Framework.[7]

9.4.1 Vision

[8]"A good vision is inspiring for people. It's a description of a dream, of a state to become. It's a dot on the horizon that you're working towards. It's also stretching for people. On the other side, a vision statement should also be short and clear for people. By short and clear we don't mean short term. It should be a dream for the long term, but people also need to be able to understand the vision, so it should offer clarity. A nice way to start with your vision statement is by using the vision statement template (see below). Beware: it's just an example to get you started, you don't have to force your vision into this template!



9.4.2 User Story

After you decided, which user stories or features should be developed in the near future. It is time to write them down in a clear and readable format. The typical template for a user story is

```
    As < role > I want to < function > so that < benefit >
```

And an example

Example

"As a library user, I want to be able to search the inventory for books by a specific author. The optional third part (the intention) was omitted here because it was almost identical to the core of the requirement.

- A search for a specific author's name (last name and/or first name) will display a list of available books by that name.
- If no results are found for an author's name or term, the user will be shown an appropriate message.
- A maximum of 20 books will be displayed per screen page.
- If more than 20 titles are found, the user can navigate between pages and narrow down the results.
- The search will not take longer than five seconds."[@rupp]

To make the user story more personal and let the team emphasize with the persona and her needs it is recommended to use the name of your persona instead of the role only, hence use the following template

User Story Template

< name of the persona > wants to < function > so that < benefit >

Each user story should follow the **INVEST principles**

- **Independent**: Backlog items should be independent of each other. This means that a backlog item can be considered on its own and implemented without the need for further backlog items.
- Negotiable: Backlog items are negotiable. They do not represent a contract that has to be implemented in exactly the same way, but leave the developers enough freedom to work out the details in a sprint together with the stakeholders.
- Valuable: Backlog items provide a value. Usually a value for the customer, or for the future user. In this context, it is irrelevant how large a backlog item is. Even the smallest backlog items must deliver a value. If this is not the case, then you should seriously question their existence and the need to implement it.
- **Estimable**: Backlog items must be estimated (see chapter 6.3). This estimation helps, for example, to sort the backlog item into the product backlog, or to be able to make a statement about how large the backlog item is. In order for the backlog item to be estimated, it needs to be understood by everyone involved. Otherwise, estimation will not be possible.
- **Small**: Backlog items should be small. The smaller they are, the more specific they usually become. But this is not the only decisive factor; the sprint length also plays a role. After all, the backlog item should be able to be fully implemented in one sprint. Accordingly, we have to choose the size so that the item can fit into the sprint but also leave enough buffers so that the smallest surprise does not immediately jeopardize the implementation in the sprint. But be careful. Do not tend to cut the backlog items too small. This will only increase the administrative workload without adding any value.
- **Testable**: A backlog item should be testable. This means that it is sufficiently well understood as to be able to describe how it can be verified whether the backlog item has been implemented according to the wishes. The use of acceptance criteria for the respective backlog items is suitable for this purpose."[9]

The following pragmatic rules will help you to write good user stories

- Is everybody able to implement the story without further discussion and questions?
  - Beginners miss clarity in their stories
- Is it possible to implement the story in 4h?
  - Beginners write too big stories
- Are acceptance criteria clearly defined, i.e. you could give the implemented issue and the criteria to another person and this person could clearly say, if the criteria are fulfilled
  - Beginners formulate the issue itself and acceptance criteria too vague
- Is it possible that different team members work on different stories?
  - Beginners are often not possible to write independent stories, they tend to interweave stories which requires a sequential work and thus slows down the development process

9.5 Issue

User stories are one type of issues. Use gitlab issue templates for the following types of issues

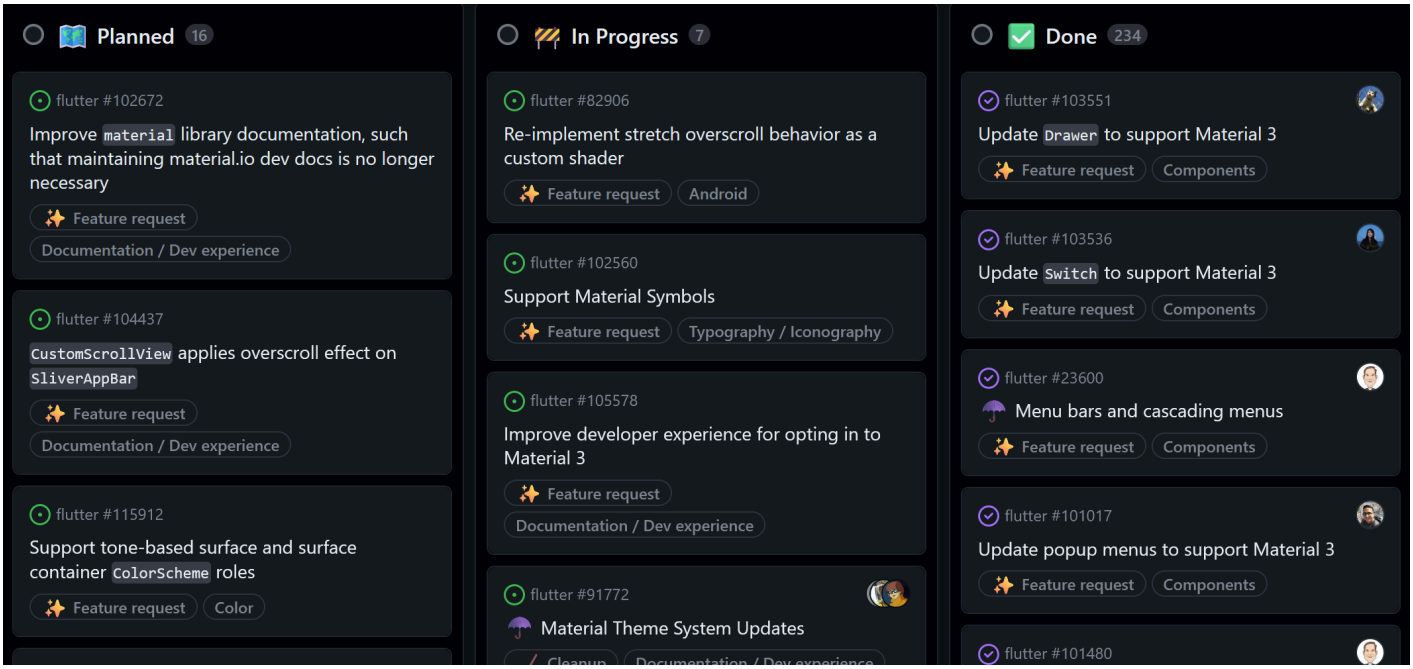
- bug
- user story/feature
- technical evaluation, proof of concept, or spike
- task -- if the story is too big, break it down into tasks -- if it is really not possible to write a smaller user story
- refactor

Use the following properties for issues to be organized better

- labels
- weight
- health status
- time tracking
- milestone

9.6 Story Board

Organize your issues in a story board and plan the next sprint in detail and the sprint goals up to the next release.



Flutter issue board

---

1. Chris Rupp. Requirements templates — the blueprint of your requirement. 2014. URL: https://www.sophist.de/fileadmin/user_upload/Bilder_zu_Seiten/Publikationen/RE6/Webinhalte_Buchteil_3/Requirements_Templates_-_The_Blue_Print_of_your_Requirements_Rupp.pdf (visited on 15.02.2024). ↩
2. Kurt Matzler and Hans H. Hinterhuber. How to make product development projects more successful by integrating kano's model of customer satisfaction into quality function deployment. *Technovation*, 18(1):25–38, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0166497297000722, doi:https://doi.org/10.1016/S0166-4972(97)00072-2. ↩
3. Karen Goldstein. Kano analysis: the kano model explained. 2023. URL: https://www.qualtrics.com/experience-management/research/kano-analysis/ (visited on 12.03.2024). ↩
4. Kano model. 2021. URL: https://www.productplan.com/glossary/kano-model/ (visited on 12.03.2024). ↩
5. Robert Sheldon. What is a unique selling point (usp)? 2022. URL: https://www.techtarget.com/whatis/definition/unique-selling-point-USP (visited on 12.03.2024). ↩
6. Eric Ries. *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Crown Business, New York, first edition edition, 2014. ISBN 9780307887894. ↩
7. Suchitra Damodharan, Vishal Muralidharan, and Vivek Muralidharan. Feature driven agile product innovation management framework. In *2020 IEEE Technology & Engineering Management Conference (TEMSCON)*, volume, 1–5. 2020. doi:10.1109/TEMSCON47658.2020.9140124. ↩
8. Scrum.org. 10 tips for product owners on the product vision. 2017. URL: https://www.scrum.org/resources/blog/10-tips-product-owners-product-vision (visited on 15.02.2024). ↩
9. The SOPHISTs. Agile requirements engineering. 2022. URL: https://www.sophist.de/fileadmin/user_upload/Bilder_zu_Seiten/Publikationen/Wissen_for_free/Agility_Brochure_Int/Brochure_Agility_Interactive.pdf (visited on 15.02.2024). ↩