Agile product management framework

Creator: Manuel Mehring – Version 1.0 – Sept. 2021

Licensed under “Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) – see: https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

About this guide
This guide condenses all information available at www.AgileProductManagement.org into one single document to be used as reference.

For each step, there are links for further reading or links to the templates used, to make this guide easier to read. Thus the guide focuses on giving an understanding of the goals and interconnections between the steps in the framework – not detailing each step (otherwise this would be a book). Wherever there is more information available than necessary in this guide, links are provided for further reading.

About the framework
The “Agile Product Management Framework” aims to provide a structured approach to product management in agile contexts. Much has been said on how to structure and most effectively setup agile development teams, but little efforts have been made to structure the process before that, which tackles the vision building, the planning and the effective management of agile requirements within complex environments. This framework aims to bridge that gap and give product managers or people in charge of product development a structured and proven way of creating and structuring compelling products in agile environments. Without having to reinvent the wheel with every new product, product managers can focus on delivering the best value to their customers.

As mentioned above, this framework explicitly tackles the steps needed to be done before development starts, thus integrating well with other development frameworks, like Scrum, Kanban, SAFe, etc.

This framework has been created by combining many existing best practices and models around product management, business models, requirements-engineering and agile in general. If you are familiar with agile workings, some of the models presented may seem familiar. The goal of the framework is to bring all this together in a meaningful way that makes sense as one stringent process, which helps to reduce errors and enable people to use a set of proven practices to improve the quality of their products.

Core ideas
When developing complex agile products, this usually involves dealing with multiple agile (development) teams as well as various stakeholders. To deal with this complexity, and in order for
everyone to work on the same product effectively, the future product must be known and agreed between everyone involved. That’s why **alignment** is at the core of this framework. All the tools and steps have visual outcomes that can be easily understood by all people involved, helping to transport the product’s goals and visions easily and reach alignment between everyone involved.

Also it is highly recommended to place all these charts/visions/roadmaps in a place where everyone involved can easily access it, creating a shared vision and understanding. Product plans that are kept on a “management level” will lead to poor product decisions and low product quality. Thus **transparency** is key to reach alignment.

We believe that theory is interesting, but knowing how to put this in action is valuable as well – that’s why this framework provides a **ready-to-use-toolset** for each step.

On top of that, all the tools/templates provided work in a highly visual, non-technical way. This is especially important, as this allows us to quickly do changes with minimal effort. We can almost guarantee you, that there will be important changes that arise during the product’s lifetime – and the faster we can react to these changes, the more value we’ll be able to deliver for the same amount of money spent. This ability to quickly respond to changes is at the core of agile, which is why everything in this framework centers around what we call “**product agility**”.

**Roles**

This framework does not define any new roles, as all outlined steps lie in the accountability of the product manager/product owner – or whoever is responsible for defining the product scope, vision and direction. This does not mean, that one person has to deliver all of it alone – after all agile is about teamwork and collaboration - but the product manager/owner stays solely accountable for the creation and the quality of the steps outlined in the framework.
Overview:

The framework consists of 7 steps, which are outlined on more detail in this guide

1. **Vision** – Creating a product vision and defining the target audience, based on the company’s strategic goals
2. **User Journey** – Mapping how users will interact with the product – from the first touchpoint to the last and identifying opportunities and improvements that need to be developed.
3. **Prioritize** – prioritizing the envisioned opportunities and improvements
4. **Story Mapping** – breaking down the identified opportunities and improvements into actionable work items
5. **Estimate** – estimating the created work items and grouping them into meaningful releases
6. **Roadmap** – Creating an actionable plan(roadmap) that shows at what time what functionality of the product can be expected and allows for a high level management of dependencies.
7. **Deliver** – Delivering the product, based on the structure and plans made in the previous steps.
Product management is not only about building the product – it is also about creating products that support or even fulfill the enterprise strategy. Because ultimately, we should only be funding and building products that support our company’s strategy.

Having this in mind, the first step is about aligning the product’s vision with our company strategy. This first stage is probably the most important one, as changes here are cheap and quick to be made. One additional post-it at this stage can mean months of additional work down the line. So don’t skip this stage too early. Here we get the most value of the work we put in.

Although the framework suggests a linear progression of steps, it is important that we regularly come back to this step and refine the vision as we learn and more information becomes available (See Vision Meeting for further info).

The main outcome of this first stage are two things:

1. Personas, describing our most important target audiences
2. The product-canvas

Describing our target audience with personas

When describing our product, we need to think about our target audience. Personas exemplify meaningful segments of our target audience which have different needs, thoughts, or channels, through which they interact with us. We use personas to make working with these target group segments easier, by imagining them as memorable persons to empathize with. The better everyone involved understands the target audience, the better their decisions will be – thus leading to a better product.

To create this alignment and to make our target audience tangible for everyone to work with, we use personas. Personas enable us agree on a common language while dealing with target audiences.
Take for example an airline product. A young family with kids has wildly different needs while travelling than a business traveler, while both use the same product (transportation).

To develop such a persona, we use the empathy map, which allows us to summarize a target group segment that we can empathize with, and thus make it easier for us to understand their needs. Below you can find an example of the empathy map template (With Courtesy of XPLANE\(^1\)).

![Empathy Map](https://xplane.com/worksheets/empathy-map-worksheet/)

The outcome of this exercise are tangible, expressive and catchy personas, like “Ellen the frequent flyer” or “John the family father” that everyone can effectively work with.

**Creating the product-canvas**

The product-canvas is the second outcome of this stage and is a main communicator on why we do things and how it’s supposed to work on a very high level. Creating this board bridges the gap between strategy and execution in our company, which is at the core of product management. The more people understand why and how we plan to do things, the better decisions can be made at the team- or individual level. A solid product-canvas thus enables decentralized decision making, which is crucial for agile teams to tackle their tasks the most effective way by taking everyday decisions, balancing technical solutions with our high level vision.

Below you can find a “filled” version of the product canvas that is filled with questions in each section to help you complete the canvas.

---

\(^1\) [https://xplane.com/worksheets/empathy-map-worksheet/](https://xplane.com/worksheets/empathy-map-worksheet/)
The board has three main sections:

1. The topmost section describes the strategic themes/initiatives of the company, that our product will support

2. The section directly below the strategy describes the product’s vision (left) and derived from that the product goals we want to achieve (right)

3. The bottom part lends itself to the business model canvas, helping us forge the business case for our product.

The most important thing about this board is that the information that is collected here, is not set in stone. It is crucial to regularly get back to this and alter the information according to new learnings or events that have happened. That process usually happens in the vision meeting, which usually takes place at least every quarter. (See meetings at the end of the guide for more information).

The product canvas is the most important and the most condensed view of our product. We use it to reach alignment and transparency, by showing and communicating it as often as we can. The most successful agile teams are those, where everyone knows the important cornerstones and vision of their product, so they can effectively self-manage.
Step 2: The user journey – or – How users will interact with our product

The user journey connects both elements from step 1, the target audience and the product, by describing **how** the target audience will interact with our product. The interaction takes place from the very first touchpoint with the product to the very last – a process we call “user journey”.

It is important that we model **all** steps, from the very first touchpoint, to the very last. If our product can be found online, the first touchpoint might be a search on Google while the very last touchpoint could be recommending the product after buying it. The more complete our journey is, the better we can understand the product we need to build. Always ask yourself if there is one more step prior to your first, or one after your last, that you might have missed. Because if your product is great, but no one can find it while shopping, it will not be successful.

The main goal in this step is understand how users interact with our product. Based on that knowledge, we then define and create a product. By doing so, we base all our product decisions and features upon the ideal customer’s experience, thus increasing our chances of success with our customers.

Note that these early steps are highly conceptual and creative, meaning it is highly recommended to do this in a workshop setting with a diverse group to gather different views. Try to include actual customers for your future product into this. Other perspectives, like Marketing, Sales, Development or others all bring valuable insights to the table. If we make this a “Dev-Team-only” meeting we will miss a lot of opportunities.
The high level product flow

The high level flow shows the most important steps of using our product and connecting them in a way that the user interacts with it. Some (very basic) example steps for a new pizza delivery app might be:

1. Search app on appstore / Click on advertisement
2. Install & open app
3. Select pizza
4. Checkout
5. Rate experience in appstore
6. Recommend app to friends

If we have an existing product, we start with mapping out the status quo and from there develop the future state. If our product is a brand new one, we can directly map out how the product flow is supposed to be, viewed from the user’s perspective.

There are two important things to consider when doing this:

- **Level of Detail:** The main benefit of such a visual overview is that it gives us just that, an overview. If we divide this into too many steps, it will become messy and create more confusion than alignment. Of course level of detail depends on the product’s complexity, but the goal here is to get a complete overview of how users actually experience your product. So complete coverage of your product is much more important than detailing every step. As a general advice, just start mapping out the steps, and when you see it gets to complicated, combine steps to achieve a higher level of abstraction.

- **The user’s perspective:** It is imperative, that all this is captured from the user’s perspective. That means here it is not important that in step X all our data is stored in a fancy database, because our users most likely do not care (except if we are marketing a database product of course) – so in each step we describe what the user expects to do, not what technology we use to accomplish that.

User Journey Map

The user journey map is based on the previously completed high level product flow and focuses on two things:

- **How is the experience** of users along the high level flow with our product?
- **What could be done in each step to improve that?**

We can think of the user journey map as an extension of the high level product flow. The high level product flow visualizes what steps users experience when interacting with our product. The user
journey map goes more into detail on each step and visualizes how happy they are at each step and based on that lists potential improvements to the user’s experience at each step.

Here’s an example how a user journey map could look like, based on a previously completed high level product flow:

The user journey map has the following info:

- On the left hand side: The personas involved in this journey as well as a short summary, giving context, so it can also be understood by stakeholders who are less involved.
- In the middle: The emotion-curve, showing how satisfied/dissatisfied users are at each step
- And most importantly at the bottom: Opportunities and improvements (sometimes also called “features” or “epics”), encapsulating what to achieve to make our customers happier.

These captured high-level opportunities and improvements form the basis for all further requirements, as these will be prioritized, broken down into actionable work items and finally be planned in a roadmap.
Step 3: Prioritize

The goal of the prioritization step is to reach **early alignment** between stakeholders on what should be done first. Inexperienced agile teams often make the error of trying to reach alignment too late in the process. At this early stage, changes are still cheap and alignment makes sure everyone will row in the same direction. Besides aligning the prioritization, it is also important to align with the stakeholders how prioritization is done. Making the ruleset for prioritization **transparent** helps build a common understanding, thus streamlining the process and reducing the amount of conflict around prioritization during future development.

The inputs for this stage are the identified opportunities and improvements from the stage before. The output of this stage is a **prioritized** list of opportunities and improvements. To reach that conclusion, we need two things for each of those items: Their relative **effort** and their **value**.

Using effort and value, they are then graphically sorted. (See schematic below)
Items in the bottom right corner are in the sweet spot and should be the top priority items, as they can be done quickly, and yield the highest value.

On the other hand, items in the top left corner, which require high effort, but only yield minimal value should be avoided (or even skipped completely). If items are skipped, it’s also important to have that discussion and also reach alignment on these decisions.

Note that the placement of items in this diagram is relative to each other, which means it is not important to precisely estimate the effort of each item, but sufficient to make sure the relative position is correct. This makes the process much faster, and thus can be repeated at regular intervals, without requiring weeks of estimation work.

As a result of this step, all collected opportunities and improvements from the user journey map should be placed inside the diagram, relative to each other (see schematic below)
For your product, the result of this step is a prioritized list of things to be developed. As some opportunities/improvements may be skipped because of their low priority, this step strongly affects the scope of our product. As agreement on product scope is important, we recommend to do this alignment with product managers/product owners, Requirements Engineers, representatives from each team and key stakeholders. (For more detail see: Prioritization Meeting )
Step 4: Story Map

To actually develop the prioritized opportunities and improvements they have to be broken down into work items, which are small and precise enough to be implemented by development teams.

To achieve that, we create what is called a user story map. Based on the prioritized list of opportunities/improvements from the previous step, each of them is now broken down into actionable work items. (See schematic below)
At the top (shown in blue) we have the user’s steps when interacting with our product (taken from high level product flow) and on the right (turquoise & red), we have the opportunities/improvements A, B, X and Y, that we identified as our top priority items earlier.

Now ordered by priority, we can start breaking down each of these opportunities/improvements into actionable user stories, that make that opportunity/improvement come to life.

**Example:**

Let’s assume “Opportunity A” of our pizza delivery app would be “Order Pizza” we now start breaking down what development efforts our teams would have to do to achieve that. Collaborating in cross-functional teams, they come up with the following user stories:

- **User story:** As a user, I would like to select my pizza directly from the menu, so I don’t have to remember their names or numbers when ordering
- **User story:** As a user, I would like to be able to review my selected food in a cart-like-view to prevent giving out wrong orders
- **Spike (shown in pink):** At this early stage, we don’t accept digital payment methods – but as easy payment is identified as high priority, we need to research what payment providers we could use in the future and how much effort that might be.
- **Nonfunctional requirement (shown in green):** The servers have to be able to take very high peaks of users ordering simultaneously, as the football season is coming up, and at each game, orders usually rise up to 10x as high as normal.
As shown in the example, breaking down opportunities/improvements can produce various types of requirements. These might be user stories, but could also be nonfunctional requirements\(^2\) or spikes\(^3\), where more research or experimentation is required.

This step is highly collaborative, so we strongly recommend doing this in a physical space, or with adequate online collaboration tools, like Mural, Miro or similar tools.

Within each opportunity/improvement, the vertical position of the requirements represents priority.

During requirements elicitation, visually mark dependencies between stories with arrows. This is especially important as dependency management is only possible, if they are transparent and known to everyone.

It’s important to understand the story mapping process not as something we do once, but rather something that we regularly repeat to select the most valuable things to do next based on the currently available information. (See also Story mapping & estimation)

**Step 5: Estimation & release grouping**

When doing agile estimates, there are many techniques out there. There are even movements that advocate having no estimates at all. We recommend to start with the practices laid out in this chapter to have a solid foundation and evolve estimation practices from there.

---

\(^2\) **Nonfunctional requirements (NFRs)** are requirements, that describe general requirements, like performance or stability. These cannot be directly developed, but rather create the boundary for other requirements. For example, if we have specific performance targets, all development work has to make sure, that the targeted performance is possible. Depending on our agile process, these NFRs usually are placed as some sort of quality or test criteria to make sure all other work complies with these (i.e. called “definition of done” within Scrum).

\(^3\) **Spikes** are work items, which aim to reduce uncertainty around a specific area. For example, a new technology has to be evaluated first, to assess if it should be used within the project. This assessment work is called a spike. A spike can result in adding more work, as more is learned during evaluation or experimentation.
Estimation is a learning process. Over time, we will naturally alter the estimation procedures based on our learnings and individual organizational needs.

Note that in “Step 3: Prioritization”, a relative estimation already has been done. At this stage, estimation gets more accurate, as all necessary steps of work have been collected and a more thorough understanding has been gained during user story mapping.

For the estimation process, we recommend using these good practices:

- **Estimation takes time.** There are faster, but more inaccurate techniques, like T-shirt sizing (Estimating all items into “buckets” of T-Shirt-Sizes, like S, M, L, XL) or more accurate but also more time consuming techniques, like planning poker⁴. **If we have to estimate large amounts, start with faster but more inaccurate techniques. If we want to plan more accurately, i.e. for the next 2 months, we should use more precise techniques.**
- **Estimations are always made with the information currently at hand.** As information changes and new things are learned, estimations that were made in the past may not be accurate anymore. That means, **estimations decay over time.** This **requires us to estimate on a regular basis.** Doing estimates as a one-off session at the beginning of a project is highly risky and will lead to many problems down the road.
- **Estimations always include uncertainty.** The bigger the estimation is, the higher the uncertainty. That’s why it has proven effective to use scales that are not linear when estimating (e.g. planning poker uses scales like 1, 2, 3, 5, 8, 13, 20 and so on, which reflect the increasing level of uncertainty).
- **Numeric vs. non-numeric estimations: Both types work.** But having some sort of numeric estimations allows us to do neat things like adding up the estimates of multiple work items that form a release. This usually makes working within larger organizations much easier. That’s why **we recommend using numeric estimation methods (like planning poker) or transferring the results of non-numeric estimations back to numeric ones:** E.g. by assigning a number to each T-Shirt-Size – i.e. $S = 1, M=5, L = 13, XL = 40$).
- **Agree to estimation references to reach alignment with estimations across teams:** It has been a proven practice to set a reference-work-item for each estimation size. So after some time, our teams should have a reference story for each estimation size. I.e. if planning poker is used, there should be a reference story for all common sizes, like 1,2,3,5,8,13. For each new work item, teams then compare the new item to the references to come up with an estimate. Having estimation references reduces estimation variance, which in turn makes estimations more reliable. With multiple teams, it is recommended to align these references between teams.

That said, we recommend doing the estimation process in the following order:

1. Once all currently known requirements are collected by breaking down the opportunities/improvements, **start estimating all of them with a fast, but inaccurate method to give rough estimates.** (Like T-Shirt-Sizing)
2. **Reorder items and releases as necessary:** Once estimations are done, usually the priority changes, as some items are so much effort, that it does not make sense to develop them, whereas others are so easy to implement, that they are prioritized higher.

---

⁴ Planning Poker: Usually done within Scrum teams. For each item, everyone picks a hidden numeric card and then all cards are revealed, compared and discussed in depth to come up with an agreed upon estimation. For more info see: [https://www.mountaingoatsoftware.com/agile/planning-poker](https://www.mountaingoatsoftware.com/agile/planning-poker)
3. If things are too large, **break them down into smaller items**. (Also known as user story splitting⁵)

4. For the most important items, **use a more precise estimation technique** to achieve a more precise basis for short-term planning. (Like planning poker)

5. **Repeat this continuously** so we always have a precise estimate on the work for the next 3 months, a rougher estimate on the work forth the next 3-6 months and a rough estimate for everything beyond 6 months.

When all requirement elicitation and estimation is done, start grouping them into releases that will enable meaningful functionality to end users. Note, that this has to be aligned with the development methodology and release management plans for the product.⁶

Smaller, more frequent releases are preferred, as learning and continuous improvement with real feedback can happen more often, which increases product quality.

It’s important to view this from a customer point of view. It’s of no use if the database is created, but the user’s data is not yet stored in it. So we always have to make sure to slice releases in a way, that there is meaningful functionality visible for end users with each release. Also take into account all outlined dependencies and restructure the requirements or the releases accordingly.

The following image illustrates the grouping into releases. Usually releases group one or multiple opportunities/improvements, but sometimes it’s not possible and they have to be split across multiple releases. In the example below, “Improvement Y” can only be partly completed in release 2 and thus was split. Note that the splitting takes into account all dependencies, so all interdependent parts are in one release, enabling full functionality to be delivered to users.)

For new products: the Minimum Viable Product (MVP)
As we can see, in the schematic above, our Release 2 is marked as “MVP”. The Minimum viable product (or MVP in short) defines the earliest stage at which we can release our product to our customers to validate our product goals. (First principle of the agile manifesto: “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”⁷) It is highly important, that this MVP is released as early as possible, as the learnings gathered from real user feedback usually have a high impact on product development. Releasing an MVP early massively

---

⁵ See [https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories/](https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories/) for more info on user story splitting

⁶ Note that this grouping happens from a user’s point of view, to outline what group of work items would form a useful improvement for the customer. Also note, that releasing is not the same as deployment. You can deploy the current version of a product to your servers, without actually releasing it to customers.

⁷ [https://agilemanifesto.org/principles.html](https://agilemanifesto.org/principles.html)
reduces our risk to lose a lot of money by developing something the customers don’t need/want. So when slicing up the first releases, the product manager/owner marks the point where the product incorporates valuable features that can be released as MVP. The MVP might only be released to a very small user group or in a small geographic region, but it should be as early as possible to learn from actual customer-market feedback.

6. Roadmap

The roadmap is a key element to communicate our product’s future and to reach alignment, inwards (teams) as well as outwards (stakeholder). It is the major baseline that everyone else will reference. Thus, the roadmap should always be up to date, transparent and available to all people involved. The roadmap is regularly updated within the roadmapping meeting. (See Roadmapping meeting)

As the last step before delivery, the roadmapping step is a sum of all previous steps in the framework. It shows why we are doing it (strategy & vision) as well as a plan on how to achieve it.

To create a meaningful roadmap, on top of all the information from the previous steps, three additional bits of information are needed:

- Team capacity
- External dependencies to be managed
- Important dates/milestones

**Team capacity** means how much “workforce” is available over time. In an ideal world, this is a flat line, with a constant capacity – but on many occasions it isn’t, which is why it has to be part of the plan. It is represented by a line, showing the available capacity over time. (Shown in blue in the following schematic)

**External dependencies** can have a big impact on delivery dates. Which is why they have to present on the roadmap and made transparent, so they can be managed. For example: A specific opportunity might need legal approval before starting development, or there is a technical dependency on
another product within the company. Managing these has to be the primary responsibility of the product manager/product owner.

**Important dates/milestones** can be dates for releases (like the MVP), specific trade fairs or any other important date, where a specific state of the product is expected to be available.

The following schematic shows how all this information comes together in one visual picture.

Note that at the top, the condensed strategy and vision elements from Step 1: Laying the foundation: Personas, strategy and vision are present. These act as anchors, ensuring, that while diving deep into roadmapping details, the overarching vision and goals are fulfilled.

Serving our two core ideas of **transparency and alignment**, the roadmap should be made transparent and shown as often as possible to stakeholders and teams. This enables self-managing agile teams, being able to take effective decisions on a day to day basis, while staying true to the overarching plan/strategy.

**Blocks of work**
The y-axis in the schematic above has to represent a meaningful metric of our agile delivery method. Depending on the agile delivery method used, there is at least one major metric, measuring (and predicting) how much the team(s) can deliver. In Scrum, this would be velocity (based on story points), in Kanban, this would be cycle time (based on number of work-items).

**Example:**
Let’s say we are using scrum, and story points. Let’s assume that, in our previous step, the first opportunity (“Opportunity A”) was estimated to be 24 story points in size. We decide that 8 is a meaningful scale on our Y-axis, thus “opportunity A” would be represented by 3 blocks (each worth 8 story points).

This is just a visual representation of the estimated work. These are not user stories on a wall or descriptive items. Just blocks, visualizing the amount of work. The advantage of this is, that it can be
easily understood and even more important, easily changed. Staying true to our **core idea of product agility**, we have to embrace change, and thus require a planning method, that can quickly adapt to changes. As new things are learned during delivery, estimations and requirements always change. The block method allows us to visually grasp the amount of work and allows us to easily rearrange items during daily delivery routine and quickly grasp where changes might be needed.

**Agile metrics and agile leadership**
Agile planning relies on accurate agile metrics. That means, it is vital that the agile metrics collected are reliable. If these are 300% off from reality, our plan will also be 300% off from reality.

It is the responsibility of the product owner/product manager to lead their teams to supply these metrics with a reliable accuracy - Of course this can happen with the help of agile coaches/Scrum Masters, but setting the requirement and leading the teams to quality metrics is part of agile leadership that product owners/managers have to ensure.

7. Deliver

The last step is the delivery itself, crafting the product based on all the previously completed steps.

As stated in the very beginning, this framework focusses on product management, and is delivery-framework-agnostic. It will work well with all common (agile) frameworks that you would want to use.

There are only two things to keep in mind, which form the vital connection between this framework and your delivery framework of choice:

1. **Metrics**: As stated above, agile planning depends 100% on accurate agile metrics, captured and calculated in the delivery phase. It is vital that these are delivered with high quality standards.
2. **Discover and Learn**: Delivering agile products always implies reacting to changes. That means, all feedback and learning gathered during delivery has to be regularly reflected in the
product management process. That’s why all steps do not happen only once, but are regular events to adapt to the latest learnings. (See Meeting structure)

Meeting structure

To make sure the most important changes are made regularly and with the right set of participants, there are several meetings. All these meetings refer to the steps in the framework outlined above. Of course the initial creation of the artifacts, like vision or user journey takes more time and has to take place in a workshop-like format. After the initial version is available, the following meeting structure should be followed to ensure all is regularly updated as new information is available.

<table>
<thead>
<tr>
<th>Meeting</th>
<th>Interval</th>
<th>Input</th>
<th>Output</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision meeting</td>
<td>At least every 3 months</td>
<td># Strategic company goals # Current product canvas # Current roadmap</td>
<td># Updated product canvas # Required changes in subsequent steps</td>
<td>PM/PO, RE, Teams representative, key stakeholders</td>
</tr>
<tr>
<td>Prioritization meeting</td>
<td>At least every 3 months (depends on delivery rhythm and frequency of high level changes) Should be closely after vision meeting</td>
<td>Identified opportunities / improvements</td>
<td>Prioritized and agreed upon list of opportunities / improvements</td>
<td>PM/PO, RE, Teams representative, key stakeholders</td>
</tr>
<tr>
<td>Roadmapping meeting</td>
<td>At least once per month. Should be aligned to delivery rhythm. Should be closely after prioritization meeting.</td>
<td># Estimated opport./improvem. # Key agile metrics # Capacity for 6 months # Product Vision # Key milestones # Ext. Dependencies</td>
<td>Updated roadmap for at least 6 months</td>
<td>PM/PO, RE, Teams representative</td>
</tr>
<tr>
<td>Product- Retrospective</td>
<td>At least every month</td>
<td>-</td>
<td>Actionable changes and improvements to the process and the product itself</td>
<td>PM/PO, RE, Teams representative</td>
</tr>
<tr>
<td>Story mapping &amp; estimation</td>
<td>No fixed schedule. Part of daily routine.</td>
<td>-</td>
<td>User Story map and estimated requirements</td>
<td>PM/PO, RE, Teams representative</td>
</tr>
</tbody>
</table>

Vision Meeting

The vision meeting takes place at least every 3 months. During this meeting, the product canvas is updated with the most current information and necessary actions are agreed upon.

The participants should include the product manager and/or product owners, the requirements engineers, and a representative from each development team as well as key stakeholders.
The meeting has two goals:

- If major cornerstones of the product-canvas have to be changed (i.e. a competitor has made a major move that requires a shift in product strategy) the product’s vision/goals have to be adjusted.
- In the second part of the meeting the team checks if the current state of the product and the planned roadmap reflects the product’s vision and goals. If goals and/or vision are not met, or are endangered, necessary steps are discussed and agreed upon.

The product vision is the highest abstraction level of the product that all following steps are based upon. Thus the vision meeting should always be the starting point in the meeting cycle.

**Prioritization Meeting**

The prioritization meeting should happen at regular intervals of at least every 3 months. The participants should include the product manager and/or product owners, the requirements engineers, and a representative from each development team as well as key stakeholders.

Usually this is aligned with the delivery rhythm (often also called “cadence”) of the teams. I.e. if the agile teams use sprints of 4 weeks the frequency of the meeting should not be every 3 weeks, but every 4, 8 or 12 weeks, to run in the same rhythm as the delivery teams. Also the experienced change frequency should be considered – if major changes happen quite often – which often happens at the beginning of a product’s lifecycle - a shorter interval is recommended.

The inputs are the identified opportunities/improvements which will be prioritized during the meeting. As estimation and prioritization at this stage is relative, adding one new item, may impact the whole prioritization queue, so having that prioritization discussion at regular intervals is crucial. We recommend to make a meaningful preselection of the items brought to this meeting, as there are many people involved and time is valuable.

Making this meeting fast and efficient is key to get the commitment to have that meaningful discussion at regular intervals – thus an experienced agile moderator is highly recommended.

**Roadmapping meeting**

The roadmaping meeting should be held at least once a month, as the resulting roadmap is the major baseline that always has to be up to date. Similar to the prioritization meeting, the roadmaping meeting should be aligned with the delivery rhythm of the delivery teams. If you are using Scrum for example, the roadmapping meeting should always take place shortly before the sprint planning, so all product-information is up to date for the development teams to base their plans upon.

As the roadmapping meeting is more of an internal meeting, the participants should the product manager and/or product owners, the requirements engineers, and a representative from each development team.

The needed inputs for the roadmapping meeting are

- Key agile metrics that is used in the roadmap (i.e. velocity of teams – see also Blocks of work)
- Up to date prioritization of opportunities/improvements
- Development-Team(s) capacity for the next 6 months
- The updated product vision canvas
- Key milestones (and their most current dates)
- External dependencies that need to be managed
It is the responsibility of the product manager/product owners to make sure that the required inputs are available before the meeting starts.

As the roadmapping meeting depends on the up to date prioritization of opportunities/improvements, it is strongly recommended to do a roadmapping meeting shortly after the prioritization meeting has taken place.

The output of the roadmapping meeting is an up to date roadmap for at least the next 6 months. Usually the roadmap spans even 12 months. Remember, that a longer outlook in the roadmap does not mean, that it all has to have the same level of detail. The further the point in time is from now, the rougher the outline of the product can be.

### Product Retrospective Meeting

The goal of the product-retrospective is to improve the product management process (not the product itself). Product manager, product owner(s), requirements engineers and team representatives meet at least once a month and conduct a retrospective meeting.

No matter what format the retrospective takes, it always follows these three key steps:

1. Gathering of input
2. Prioritization (usually “dot voting”)  
3. Discussing the highest priority items and deriving action items

Over time, numerous retrospective formats have emerged. For starters, we recommend something like a starfish retrospective. Over time, product teams can (and should) evolve their retrospective formats (See “Retromat” for various examples and inspirations)

It is highly recommended to have this meeting facilitated by an experienced agile coach/scrum master.

### Story mapping & estimation meeting

For story mapping and estimation, there is no fixed schedule as this usually happens on a “do when needed basis”. Several development frameworks also have events set for this, like the “refinement meeting” in Scrum.

It usually makes sense to plan a user story mapping workshop at least after the prioritization meeting, as the newly prioritized opportunities and improvements have to be broken down further. But story mapping & estimation is not limited to a specific meeting or event, and can be done when needed during daily routine.

Keep in mind that we don’t have to break down all available opportunities and improvements down to the same level of detail. Precise requirements analysis and estimation takes time and in the need of deterministic project plans, teams can easily get lost in estimating and planning. Managing and estimating requirements the agile way means that the near future should be precisely planned and estimated, but the far future can be less clear as it most likely will change. Spending too much time in detailing and estimating something that may or may not be developed in a year is often a waste of time, when after some time the requirements are skipped due to changes in product development, or market shifts.

So keeping the balance between precision in planning for the near term and less precise but more rough and visionary planning for the mid/long term is key here.
Conclusion

After going through all stages of the framework, from creating the product canvas, user journey creation, prioritization, story mapping, estimation and finally roadmapping you now should have a good understanding of how to effectively structure an agile product management process. Following this framework allows you to make your product management process truly agile, streamlining product management and agile development teams to make working together a breeze.

With the ready-to-use tools and templates you can start right away, striving to reach the ultimate goal to achieve true product agility, keeping up with the ever changing market developments and outpacing your competition.