Bryan Reinero: What is product management?

Harsh Sinha: Product management is a multifaceted discipline. It consists of the tasks needed to take a fledgling idea to strategy, to execution, and ending with bringing the product live to the customer. And then, you learn from the customer and iterate on the idea to improve the product. The ownership of that entire process belongs to an individual called the product manager [PM].

What are a PM’s responsibilities?

I break down the responsibilities of a PM into three main themes. The first one: inspiring—having an idea. A company has a larger vision. From that, the PM has to come up with ideas, and then work with the team and the customers to understand what to build that would add more value and align the product towards the goal set by the company.

In this phase you are answering questions like, “What is the problem you are solving?” and “Why does the customer care about this problem?” It’s a lot of work to understand the market, customer needs, how your current product is placed in relation to competitors’ products, and the value proposition that you’re providing to the customer.

Along with that, you have to define success and failure criteria for every product that you launch. If you are going to run an experiment to test whether a new feature is going to improve the customer experience, before writing your first line of code you have to define success and failure criteria.

The second theme is prioritization. PMs get feedback from many places: the customer; analytics; what the market’s doing; what the competitors are doing. They also get feedback from the engineering team about what is and is not possible to build.

After taking in this data and feedback, [the PM’s job is to] prioritize the next top things to build. That doesn’t mean that a PM goes into a conference room [alone] and puts [his or her] head down into an Excel “priorities” spreadsheet. It’s more about having the hard conversations with the team about scoping, sizing, and what the market wants right now versus what the technology can do. All of that has to be taken into account to come out with a prioritized list.
The third theme is execution. Execution is a critical part of product management. That starts with defining the product specifications based on the first two steps that I just described.

The next part of execution is not only defining the happy flow but also defining the unhappy flow—the edge cases when the product would not work at all, or would not work as expected.

The third part is making sure things are delivered. If there are blockers impacting the team, then executing means driving conversations cross-team and across the company to remove those blockers, so that the team can move forward.

And finally, in the build-measure-learn process, ensuring that analytics are set up correctly and interpreted correctly (to determine whether the experiment is a success or a failure), and then feeding that back into the iteration cycle so that the product continues to iterate and improve.

Why should this job be done by an individual PM? Would it be easier just to have customers talk to the engineering team directly and tell engineering what they want?

At TransferWise we tend to do that. We believe in having as few people between the customer and the engineers as possible, so we have set up cases where engineers take customer calls [while] working with our CS [customer support] agents. Engineers answer emails and chats directly. The idea is to not to have the engineers become CS agents, but to give some of the customer’s pain to the engineers. That way, when the engineers are building the product, they understand the why.

That said, there are only so many hours in a day. Engineers do have to write code. From an efficiency perspective, it is good to have one person (or a group of people) who can talk to customers or partners (if you’re in a B2B business). Those people can distill that down to a few larger themes that the team can execute.

Also, there is a skill-set question here. For engineers it is not always natural to get out in front of customers, to speak their language, and to understand the specifics of the requirements, because engineers tend to be more involved in the technical details. A PM can also act as a good translator to convert those customer voices into implementation specifics.

Is there a format a PM should provide that articulates the customer needs?

It varies from organization to organization how structured this is. I’ve worked in different-size organizations and teams. In some cases, PMs write down specifications in Word documents, and those go through multiple peer reviews before arriving in the engineering team’s in-boxes. That is the traditional approach, and I’m sure there are organizations running that way even now.

When I was at eBay in the early days, there used to be a template called a PRD: a product requirements document. That was used by every PM. It was a full-on 40-page document with questions about every proposed feature. It contained screenshots and wireframes. The engineering team would interpret the PRD into an ERD (engineering requirements document), which was even longer and contained technical-implementation details.

What I just spoke about is the waterfall life cycle, which used to take months to ship a product. Nowadays, the way product specifications work is at a much higher level.

I can give you an example of how we do this at TransferWise. We run smaller product teams. We usually have one PM with anywhere from three to six engineers on a product team. These product teams are full-stack, including customer support, operations representatives if we have a business partner, or a banking
person given that we’re in financials. The team plans together, prioritizes together, and understands why they are building what they’re building.

The product specification usually starts off in a ticketing system like Jira. We start with an epic for the primary value proposition. From there, the epic is broken down into specific tasks which are iterated on by the product team.

Having smaller teams and usually one- or two-location teams that work well across [video chat or instant messaging] also helps because you don’t have to write as many documents. But really there’s an iterative process, so the feedback loop is going faster.

You said that the financial team might be part of the product team. Does that mean that the product team is a cross-functional team? Is that difficult to achieve and still maintain a small team?

It varies. I’ll give you an example of customer support. At TransferWise, we have a product which allows you to move money from the US to the UK. There is a core team within customer support for that route. We would not have the entire team in the stand-up and the planning sessions, but we do have one representative who understands the scope and the challenges of that team and gives input.

Why is the customer so important?

In this product-driven world, we are here to solve customer problems. If we just are building stuff without getting input from the customer, then it’s probably a hobby project. But given that we are here to solve customer problems, having the customer’s voice input and feedback throughout the process to understand what’s working [and] what’s not working are super-important, because otherwise teams can drift very quickly based on their own assumptions.

How do cognitive biases play a role in this?

Quite often we see anecdotal evidence taken as real evidence. People think that their problem is the biggest problem to solve. If you are moving money between the US and UK, and you see a specific behavior that is true for you, then you think that the entire world is like you, as opposed to having quantitative feedback from a larger group of customers transferring between the US and UK. That’s why it’s important to make sure you don’t get the unconscious proximity bias or the context of just working solving your own problems.

Besides use cases, what else would I find in a PRD?

You want flow showing what the problem is you’re solving and how the customers would solve that problem with the product. That would include use cases, edge cases, and success or failure criteria.

Going deeper into how you would even set up analytics: What are the things to track to make sure that eventually as the product goes live, you know what the parameters were and what the markers were or what the attributes were that you were testing for? How is that fed into the reporting engine?

Can you give me some examples of useful analytics?

Let’s say you started a new product. You’re trying to understand how many people are seeing the product landing page, for example. How many people are exploring the product? And from there, look at non-logged-in or session-based visitors.

You build a conversion funnel. If your product requires somebody to take an action, there is a process of converting that person from a visitor, to a signed-up and registered user, to [a person] actually doing that first transaction.

I’ll give you an example from TransferWise again. For us, a user is a user once they’ve completed their first transfer. That means money has moved on the platform. If you’ve just signed up, created an account, given us your email account, and set up your account, but you haven’t created and funded your first transfer, we don’t count you as a user until some money is sent and arrives at the other side.

Throughout this process we look at a conversion funnel. Somebody lands on our website landing page as a visitor. Then, what did they do on the page? Did they scroll? Did they interact with elements of a page that explains what we do? How much time did they spend on the page?

From that we can see things like, did they fall off before they completed the first transfer, and if so, where? After they clicked the sign-up page? Did they fill out the user name and email address, and then set up their account? Did they begin creating their first transfer? Did they run into difficulties due to errors in filling out any of the fields? Did they spend a lot of time on the home page but then dropped off there?

These are some of the metrics that you would calculate to understand how the product is doing. You go one level deeper into these metrics to see whether they’re dropping off at the registration page.
Eventually, the success or failure criteria would be things like, how many visitors who visited the site converted (meaning, exited the funnel by creating a new transfer)? Then we measure metrics like conversion rate. Is it 10 percent? Is it 20 percent? Is it 35 percent?” From there, you understand what you can do to improve to increase that conversion rate.

When you’re developing a new product, how do you get that feedback when you don’t yet have customers? How do you understand who your customer might be?

You start from the basics. What is the problem you are trying to solve? And why would a customer care that you are solving this problem? Who are the other people that are solving this problem? Are you solving problem no. 40 on their list, as one of my professors from Berkeley, Steve Plank, would say, or are you solving problems no. 1, 2, and 3? This is important because a lot of times we tend to think if we have a specific problem, then that is the biggest problem to solve.

When you’re doing new-product development, you will spend a lot of time researching whether it is a big enough problem that the market needs it to be solved. Look at how customers are solving this problem right now. What is the offering that exists in the market? And then, you ask, “Could I build a product which is an order of magnitude better than what exists in the market?” If it’s only 20 percent better, if it’s 10 percent better, the probability of a customer switching or changing their entire workflow is very, very low. It can be an uphill battle to acquire users.

Once you have an understanding of what the market is, that’s when you start defining the success and failure criteria. You start thinking about what the flow would be. You start doing bits of user testing. This is a more qualitative look where you develop wireframes and mock-ups. You get out there and talk to potential customers.

In that process you identify who your potential customers would be. If it’s a consumer-facing product, sit down and explain the product to the customer. See if they’re getting it. Do they understand the value proposition? Is this something that they would use? Would they pay for it? Those are the things that you can learn quickly in the early phases without writing any code.

From there, you get more involved. You start doing some basic prototypes. Build a small product and drive some traffic to it to see how people use it. That’s the iteration of a new-product process. 😊