Alex Roe's Product Management Principles

Last updated: May 2020

Background

These are the principles / lessons / mental models I have taken away after three years as a PM at Google (Search, Photos) and two years as the first PM at an enterprise startup (Cresta). This is largely inspired by Ray Dalio's book and I plan to keep this a living doc and continue to update as I learn more. If you have any questions or have found this useful - I'd love to hear! Please email me at alex.roe48@gmail.com.

1. General

1.1. Be nice to people; assume everyone is trying their best given the information they have
1.2. A PM does not need to have all of the answers. This is a classic new PM logical fallacy.
1.3. Appreciate your team, after all, they're the ones producing the output
1.4. It's never as good or as bad as it seems
1.5. Know thyself: Seek out feedback, know your strengths and weaknesses, what excites you, what bores you
    1.5.1. *A simple and effective way to do this is to send an email to folks you closely work with and ask (1) what should I keep doing (2) what should I stop doing (3) what should I start doing*

2. Making decisions

2.1. Seek to get to the best answer regardless of who it comes from (ux, eng, uer, prod, etc.)
    2.1.1. **Note* not everyone communicates the same way or is comfortable speaking up in a group; go out of your way to make sure your team is included when decisions are made and that their voices are heard*
2.2. Always refer back to the user need and/or goal you're trying to solve
2.3. Get the data (UER, experiment) when in doubt; this speaks louder than words
2.4. However don't be paralyzed by acting without data if it is difficult or time consuming to get; "let's wait til we get the data" is often a scapegoat and leads to punting the decision
2.5. Figure out if the decision is a type 1 vs type 2 - critical for velocity
2.6. When a decision is made, document what it is and send to stakeholders; this prevents later finger-pointing and confusion
2.7. If you cannot resolve a decision amongst parties, escalate to the decision maker asap with clear reasoning from both sides; don't make it personal - refer to 2.1
2.8. Familiarize yourself with cognitive biases and start developing a filter for detecting when these are in play

3. Prioritization

3.1. Have a clear definition (metric) of what success is for your product / project; evaluate all new projects against this metric; focus on the ones that move the needle the most
3.2. 80/20 rule - find ways to do 20% of the work and get 80% of the benefit; too easy to get caught up trying to do 80% for 20%
3.3. Any feature to be prioritized must have at a minimum (1) an estimate of user/business impact and (2) an estimate of engineering cost. A simple S-XL grade for these two criteria is sufficient and makes deciding whether to prioritize something or not much easier.

3.3.1. *Note: you should be in the habit of reflecting after doing this to make sure your estimate for (1) matches reality. Your engineering stakeholders should be responsible for (2) and likewise should make sure matches reality over time*

4. Presentations

4.1. Slides are a tool, not your presentation; don't have essays; be a minimalist w/ content; images + speaking over them > words
4.2. Tell them what you're going to tell them, tell them, tell them what you told them
4.3. Be a storyteller - villain, victim, hero framework
4.4. Tailor your presentation to your audience; make 'em laugh
4.5. Demos/prototypes speak louder than slides

5. Delegation

5.1. You will not scale as a PM unless you delegate and trust your team
5.2. Define clear requirements and expectations (document these), then get out of their way and let people do their job
5.3. When giving feedback, refer back to the requirements and expectations you defined
5.4. Don't be critical of how someone does the task if it meets the requirements but might be different than how you would've done it; this leads to resentment. It is fine to give feedback and make suggestions but should frame it as "consider…" vs directive
5.5. Give feedback along the way vs at formal performance review points

6. Productivity

6.1. OKRs are an effective tool for planning what you want to get done at a high level for a long period of time; when new tasks come in or you think about your week, refer to these to make sure what you are working on aligns with your goals
6.2. Don't let others define what you do, you control what you do. You let others control what you do by being reactive to email and not asking "how soon do you need this by?" and "if I do this I drop X, is that ok?" if reactive asks come in
6.3. Email is async, minimize checking this; aim for only 3x / day; don't set a precedent of having to reply immediately - especially outside of work hours; for even more productivity, don't start your day by checking email, do an hour or two of work before opening up email.
6.4. Inbox zero. Inbox zero. Inbox zero.
6.5. Use your calendar to block off time to get work done
6.6. Maximize time spent on important, not urgent tasks. Minimize time spent on urgent, important. Delegate urgent non-important. Spend no time on non-urgent, non-important.

7. Communication

7.1. Never ping someone just "hey". Ping someone the entire question that they can respond to when they can; if they don't respond, send an email
7.2. Keep your emails and docs short; no one has time to read them; if needed give a tl;dr and link out to a doc instead of writing an essay in email
7.3. Bold action items in emails to make them explicitly clear
7.4. Over-communication is rarely a bad thing - especially with your managers / stakeholders; avoid the "hey what's up with X project" emails
7.5. It's fine to admit you don't know something; admit so vs trying to sound smart
7.6. Overpromise and try your effing hardest to deliver (h/t Jeff Grimes)

7.6.1. *Common saying is underpromise and overdeliver, but great products get built aiming for something difficult (make sure it's possible tho)*
7.6.2. *Contrarian point: if you don't have credibility within the org yet, it's wise to underpromise and overdeliver and then leverage that organizational capital into bigger projects down the road (h/t Michael Hopkins)*

7.7. if you can't do something immediately, let the stakeholder know and give an ETA up front

8. Meetings

8.1. Be allergic to meetings; treat meetings as if you are paying for everyone's time. Is it still worth it? Can it be handled over chat/email?
8.2. If you have to schedule a meeting, find a time where people already have blocks of meetings and cluster there as to not fragment their time too much (especially with engineers!)
8.3. Status meetings are time sucks; get team to update status of tasks ahead of time and use meeting for resolving open questions / discussion
8.4. Have a clear agenda and desired outcome defined ahead of time; cancel meeting if there is no set agenda / outcome needed; meetings for the sake of meetings suck
8.5. Keep meeting notes and send them out to everyone afterwards
8.6. Keep meeting moving, be quick to take things offline or follow-up separately
8.7. Be present - laptops and phones down

9. Enterprise / B2B product management

9.1. There are three different classes stakeholders that have different needs: the person/group writing the check; the manager/group who will help implement/operationalize the software; and finally the end user of the product.
9.2. Sales is as/more important than the product. Get to a min-viable-sales product that you can close a deal then grow that product to a set of customers; leverage that set as distribution for additional product offerings + feature prioritization; there is danger in overfitting to early customers.
9.3. The buyer of your product is often different from the end-user of the product. This means that although you could have an incredible end-user experience, unless the product meets the needs of the buyer, then it doesn't matter, so you need to deeply understand what your buyer wants.
9.4. What do buyers of enterprise software want? A promotion! In my experience at Cresta our buyers are using Cresta for a specific need: increase sales revenue, and ultimately want a good ROI on their purchase; I *think* this can be generalized to most enterprise product buyers (e.g. if you spend $1 on technology, you should be expecting to get $5 back measured in some way)
9.5. To the point above, you should have a really good way to demonstrate clear ROI for your product

10. Working with customers

10.1. Visit them in person. Talk to them, understand who they are as people, develop relationships with them. Cultivate a win-win mentality. Spend time with them outside of the office.
10.2. Be front-line support. Make sure your customers have a direct line to you and you are responsive. Stuff will go wrong, it's how you respond that matters.
10.3. Every day observe your customers. If you're not with them, observe using a tool like Fullstory. Replicate their setup at your office if not the same (e.g. they use windows but you're on a mac). So crucial to never lose sight of how your users are using the product *especially* if you're not the target user yourself
10.4. Let your customers write your roadmap. Refer to 9.1 above, think about the roadmap from the perspective of all three of them and solicit their feedback and listen, listen, listen. After listening

(seriously, listen), compile into prioritized list; go back to them with prioritized list and get their agreement. That's how you get buy-in.

11. Startup product management (especially when you're the first PM)

11.1. No ego. Engineers are used to making product decisions themselves before you were there. CEO was used to driving product direction before you were there. Don't rush to "be a PM" and make product decisions. Listen, understand, and make sure you understand deeply the customer, tech stack and the business. Take on grunt work, fix some bugs, ask lots of questions.

11.2. At times you'll have to do everything. Write code, build dashboards, make mocks, visit customers, help with marketing, etc.. Don't forget to block out time for product thinking every week.

11.3. You'll have fewer resources, less time, and more things to do than at a larger company. Prioritization becomes even more important. Even though everything feels critical and you want to move fast because "startup", take a breath. Say no more often, think about what's important in the grand scheme of things, make sure you maintain some work life/balance (at least one day / week with no work) to prevent burnout.

11.4. Block out days to help with focus. At one point I was responsible for product work, marketing work, customer success work and a plethora of other random things. I struggled with prioritization and getting stuff done until I broke my week into chunks of time: 2 blocks (morning, afternoon) each day; then ahead of time, I defined what those blocks were for e.g. meetings, marketing work, product work, etc. Then, whenever I had a task, I would categorize it into what kind of work it was and batch it until I had a block for that type of work. This helped keep me sane and helped me get stuff done

11.5. Coach others to be product thinkers: the whole company benefits from not just "product managers" being able to be product thinkers; help your teammates out, overcommunicate context for decisions being made, lead others to water and decisions instead of making them yourself especially when they are type 2 type decisions that are correctable. This way of operating gives the company and yourself massive leverage.

11.5.1. Caveat: as PM your job is to ensure that the product is going towards the correct direction; if you give up all decision making and don't keep a pulse, then the product risks losing cohesion and devolving to show your organizational seams. This is a tricky balance to get right.