Peter K. Shultz

Peter is currently a product manager at Google.

Being a PM
22 Mar 2019

I started my job at Microsoft five months ago today. I had the opportunity to write a small post about what I do for Wes Weimer, my former software engineering professor. I thought I'd share some of it here.

Hi everyone! I'm a program manager (PM)1 at Microsoft, working on high-performance computing. Professor Weimer asked me to write about my job to give you some industry perspective on what you're learning right now.

Large software companies have an interesting problem: they have a lot of work to do on each of their products. Whether that's implementing a new feature, improving performance, or working with another team to create a product integration, there's often so much work going on that it's hard to prioritize what to do next. That's where PMs come in!

For every 5 to 15 software engineers writing code to build or improve a product, there's one program manager figuring out what those software engineers should be working on next. With that comes a whole other bunch of responsibilities: specifying the behavior of new features, working with partner teams who have a stake in decisions you make; representing your product in meetings with finance, marketing, and documentation folks. The list goes on and on.

Everbody has a different definition of what PMs do. Many people say a PM's job is to be "the voice of the customer", but that makes software engineers sound ignorant of the customer, which they're not. Software engineers and PMs both have the needs of the customer in mind–it's just that PMs have a better understanding of what customers need most because they're busy prioritizing everything that needs to get done to improve the product.

At least at Microsoft, PMs don't actually tell software engineers what to do. Rather, program managers persuade software engineering managers2 that certain things need to get done. If our arguments are persuasive, software engineering managers will allocate their software engineers to work on what we think matters most. Software engineering managers often call this "funding" a certain piece of work (i.e. "we'll fund that feature you seem to care about so much").

From the perspective of a PM, slide 6 of the "Requirements and Specifications" lecture really drives home an important point: if a mistake happens and it's not discovered until later, it

becomes very expensive to fix. So expensive, in fact, that you won't be able to deliver on all the other features you promised (and who likes breaking a promise?). As a result, priorities need to be reset, and somebody–you, a friend, that team you were working with–is going to be unhappy.

That's why PMs are often assigned to write specs–a document that describes [in painstaking detail] the requirements for how a new feature or integration is going to work (see slide 33 of the "Elicitation, Validation and Risk" lecture). There's no doubt that software engineers could write specs if they wanted to–it isn't too hard. But why burden them with more work when they could be coding?3

Feel free to ask any questions in the follow-up section. I'll answer as best as I can.

1 Depending on where you work, PM can mean program manager, product manager, or project manager. Each of those job titles means something different depending on who you ask. ↩

2 If you become a software engineer, your boss is a software engineering manager. At Microsoft, software engineering managers assign programming tasks ("work items") to the engineers that report to them (their "direct reports" or "directs"). They're often responsible for leading design and [re-]architecture efforts as well. ↩

3 Probably the easiest way to describe what a PM does is "anything that would get in the way of a software engineer from checking in code". ↩