# Win everyone over with your roadmap

Cheat sheet: top 10 tips for getting buy-in from your technical team

BY MARTIN SUNTINGER



The presentation of a roadmap can be nail biting for both developers and product managers; one party has worked hard to come up with a vision while the other party waits to see the unknown that is going to affect their work.

I felt this tension when I worked as a developer and I often found myself unsatisfied with the roadmaps product management put together. I didn't fully buy into the decisions, and I'd often walk out of a planning meeting with the sentiment of "Well, this doesn't make sense to me, but if they think so...", or even worse, a "We'll have to figure out things ourselves and make it look like what we do fits this roadmap.

You might argue the problem was likely me suffering from NIH (Not Invented Here syndrome) and you might be right. As a developer, I had a very strong opinion on what was the right thing to do. But now as a product manager, I see what caused this disconnect, and what general learnings product mangers can draw from this disconnect to hit a homerun with your next roadmap presentation. After all, if the technical team agrees with and understands the big picture, day-to-day design and implementation decisions will be made with the right context, and you'll get the product you envisioned.

## 1. Choose substance over buzzwords

Whilst buzzwords like "big data analytics", "machine learning," or "an Internet of Things initiative (IoT)" might resonate with business stakeholders as high-level anchor points, they aren't helpful and actionable items for technical teams. Engineering needs to know exactly what it is they're building, how it relates to current products, how it should be delivered, and who will use it in the end, and for what purpose.

Setting high-level themes is great for structuring the roadmap and context, but make sure you don't stop there and have good answers for what is behind each high-level item. For example, if you've called a theme "intelligent services," make sure to break this down into key initiatives and epics that are needed in order to deliver this theme.

## 2. Set the right context

Technical teams need context for why you are building something on a roadmap. No technical team is going to say, "Just tell me what you want and I'll build it." On the contrary, engineers need to see evidence for why you see demand for a feature. Let data speak, but also tell a powerful story from the perspective of your users. Use personas, and talk about some alternatives that you've excluded, and why. To help the entire team understand the roadmap the "why" matters as much as the "what."

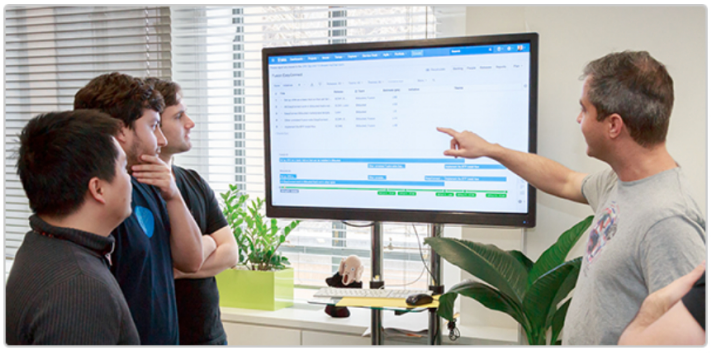
## 3. Consider commitments carefully

If a feature doesn't seem well thought through or realistic, yet it is still on the roadmap, this should scream red flag. You don't want the technical team getting the impression that they have to build stuff just because you promised it to someone. This could be a commitment to a customer or because because "management wants it." So be wise about making commitments. Even if you have a forcing function behind yourself that requires a particular change, make sure you understand and pass on the rationale to the team, and stand behind it yourself.

## 4. Make realistic plans

A vision is good, but it's even better if everyone feels confident that it can be achieved. The plan doesn't need to be precise, but if the first thing your development manager sees when looking at a roadmap is a huge bottleneck – for example, if the roadmap looks design heavy and frontend centered, but the team only has one designer and has struggled with frontend topics in the last few months – then you'll have he or she struggling with the roadmap throughout the rest of your presentation.

Make sure you do a reality check upfront with the team and play with what-if scenarios. You need to have answers, a clear plan of action, and concise consideration about the "how it can be done" before asking for everyone's commitment.



## 5. Think big, start small

You need to be aware of where a product and team skills are today versus where you want them to be. It's great to advance into new fields, which might require new skills in the team or moving away from existing technology, but don't just write down dreams of where you'd like to be in a year. Instead, think about how to get there realistically. Acquiring talent takes time, adopting new technologies takes time, and abandoning existing products requires clear commitments and a transition plan.

## 6. Build a business case

Business case? Yes. Technical teams care about businesses cases. Maybe not to the same extent as senior management, but an entire development team cares about why something is relevant to the business, if it produces real results, and how this will be measured. Tap into the business-street-smarts of your technical team. Everyone has a vested interest in the business succeeding as a whole, and it can be great source of feedback or additional ideas.

Also, full clarity on the business impact and seeing it happen can be a great motivator; driving results is satisfying beyond just having built and shipped a feature.

## 7. Balance mundane with motivating

Engineers want to build unique, innovative products that they can take pride in. If it's just the same old story others have told before, it can be demotivating. Make sure you do research to confirm that your story is as innovative as you think. Aside from demotivated developers, the business impact of the lack of innovation is even worse.

With this said, even roadmaps will always be a balancing act between exciting new features, and technically not too interesting must-haves that just have to be done. Try to make sure that even the mundane is motivating on your roadmap.

## 8. Think beyond MVP and v1

Creating a minimum viable product, and then a version 1 is one thing, but there's also everything that happens post-launch: operations, maintenance, feature requests from users, continuous improvements, and new versions of other products and/or platforms that are integrated. A quick think on what the challenges and obstacles might be after a launch will bring these to light without much effort, and engineers will be thankful that you didn't ignore these realities. Rough estimates suggest that the initial effort of building a new feature is often only a third to one half of the total effort spent on it over its entire lifetime. In

other words: the aftermath is more costly then the initial build, and some "quick small things" become very costly in the long run.

## 9. Roll with the punches

Estimates are a good thing. They give you guidance, and are created to the best of a product manager's knowledge at each given point in time. However, many assumptions made for estimates turn out very wrong once you go into implementation or refine a design. Make sure you prepare and present the roadmap so that it's flexible to changes.

## 10. Be open and honest

A roadmap is there to provide guidance. It's not a detailed plan for execution and everyone on a software team knows that. There's no need to sell it as something different than it is. So if you don't have all the details on a topic yet, be open about it. Share what you have, what the intention is, what the open questions are and highest risks that still need to be addressed. Point out areas that require experimentation and more research to nail down the "what." Just remember to account for this experimentation time in planning.

## The bottom line?

Your team deserves a roadmap that clearly paints the big picture, but doesn't neglect realities. Your team also deserves a roadmap that is motivating, exciting, and has enough details so the entire software team knows what to do in the next 1-2 sprints with a feeling of confidence that they'll achieve great results with material impact for the business.



MARTIN SUNTINGER

Martin currently leads Atlassian's ecosystem team. Having successfully founded a startup in the Atlassian Marketplace before, he's incredibly passionate about making sure our customers get access to a great choice of high quality apps in our Marketplace, as well as provide a great experience for all developers building on our platform. Outside the office, you'll likely meet him mountain biking, kayaking, or chasing down the best coffee in town.

TUTORIAL

ARTICLE

## Learn scrum with Jira Software

A step-by-step guide on how to drive a scrum project, prioritize and organize your backlog into sprints, run the scrum ceremonies and more, all in Jira.

Try this tutorial →

## Creating a lean, mean product requirements machine

Learn how to create a lean, agile product requirements document by following these principles with this agile product requirements document template.

Read this article →

### Agile Topics

Agile project management

Software development

Product management

Sign up for more agile articles and tutorials.