How Spotify builds products

Henrik Kniberg

Version 1.1
2013-01-18

Product development isn't easy. In fact, most product development efforts fail, and the most common reason for failure is building the wrong product.

Spotify is a Swedish lean startup with an awesome track record of product delivery. Their products are loved by users and artists and spread virally – they have over 20 million active users, 5 million paying subscribers, and are growing fast. For example, it took roughly a year to go from zero to 1 million paying subscribers in the US, a foreign market with plenty of established players.

Spotify's vision is to bring you the right music for every moment. That is, unlimited access to all the world's music, and the ability to share it easily; and the more music that gets shared and played, the more money goes back to artists. Starting as a music player a few years ago, their products are now evolving into a ubiquitous platform for discovering new music and connecting artists with their fans directly.

The products are designed to be easy, personal, and fun. Even Metallica, long known as die-hard opponents to music streaming services, now say that Spotify is "by far the best streaming service" and are "stunned by the ease of it".

Here's the paradox though: Successful companies like Spotify only want to deliver products that people love. But they don't know if people love it until they've delivered it.

So how do they do it?

The purpose of this article is to provide a high level summary of Spotify's approach to product development.

Disclaimer: Like all models, this is a simplification of reality. They don't always follow this process to the letter, and there is a lot of local variation. But this article should give you the general idea.

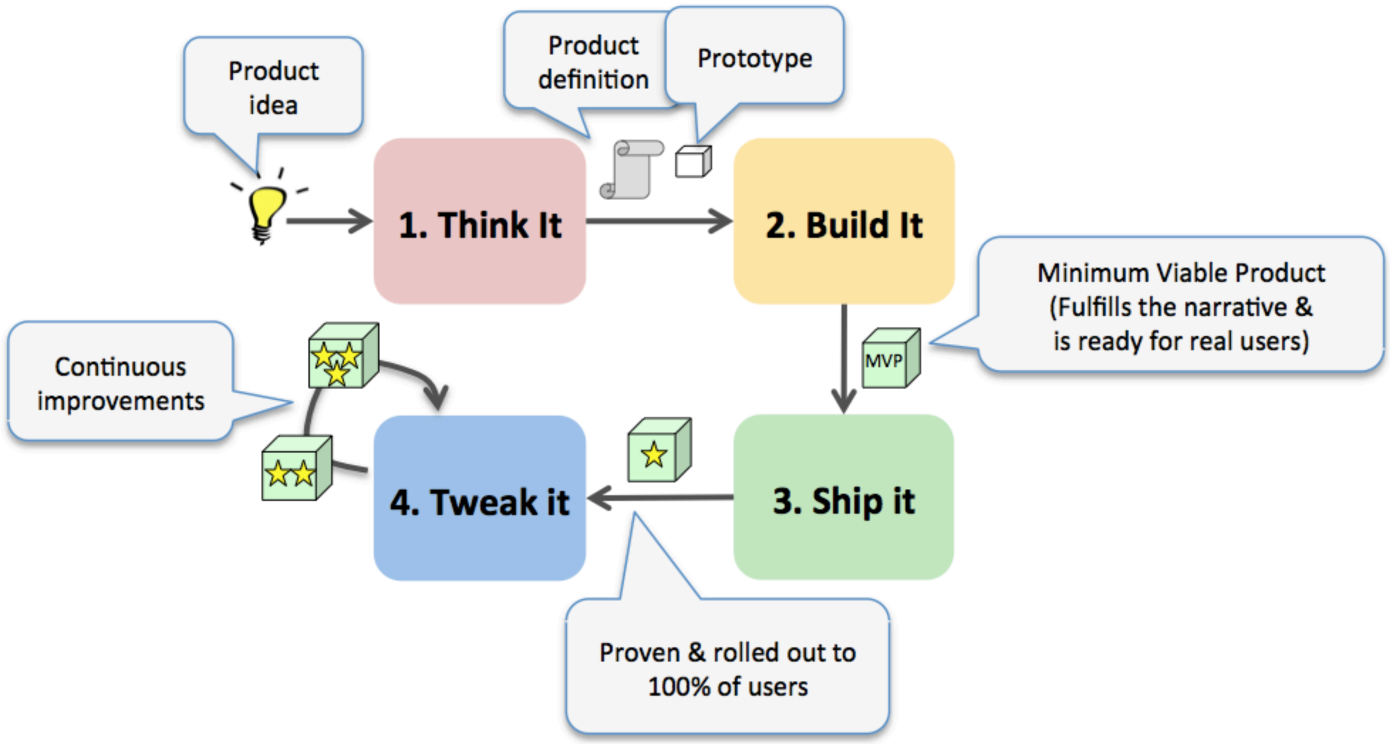
Acknowledgements: I didn't invent this model. The material in this article is based on discussions with Gustav Söderström, Oskar Stål, Olof Carlson and their internal docs and frameworks such as the "Think It, Build It, Ship It, Tweak It" framework. I also learned a lot talking to designers, developers, and agile coaches. Thanks everyone!

Summary

Our core philosophy is:

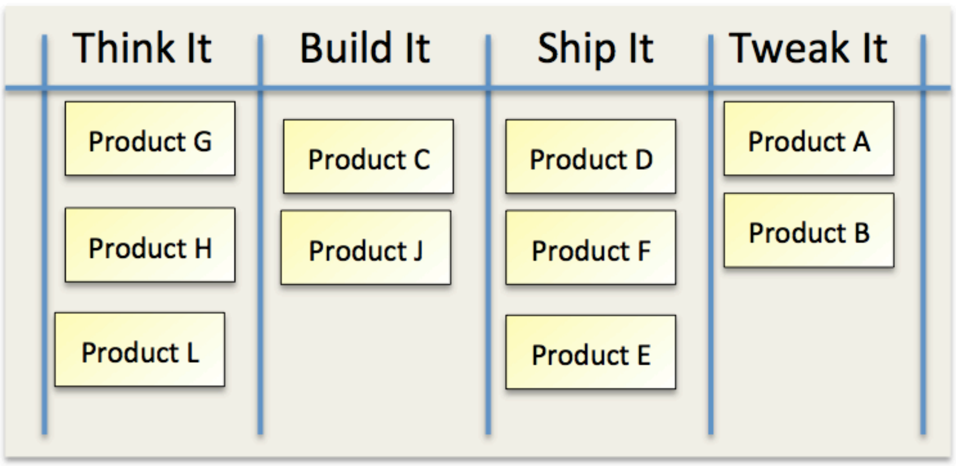
- We create innovative products while managing risk by prototyping early and cheaply.
- We don't launch on date, we launch on quality.
- We ensure that our products go from being great at launch to becoming amazing, by relentlessly tweaking after launch.

All major product initiatives go through four stages – "Think It", "Build It", "Ship It", and "Tweak It". Here's a picture illustrating the flow from idea to product, and what comes out of each stage along the way. You'll find a more detailed version of this picture at the end of the article.


Product idea
Product definition
Prototype
1. Think It
2. Build It
Minimum Viable Product (Fulfills the narrative & is ready for real users)
MVP
3. Ship it
Proven & rolled out to 100% of users
4. Tweak it
Continuous improvements

- **Think It** = figure out what type of product we are building and why.
- **Build It** = create a minimum viable product that is ready for real users.
- **Ship It** = gradually roll out to 100% of all users, while measuring and improving.
- **Tweak It** = Continuously improve the product. This is really an end state; the product stays in Tweak It until it is shut down or reimagined (= back to Think It).

Spotify has over 30 squads[1] and a number of different products, so to keep track of what's going on and visualize it to the rest of the company, we use a product status board that shows which products are in which stage. Roughly like this:

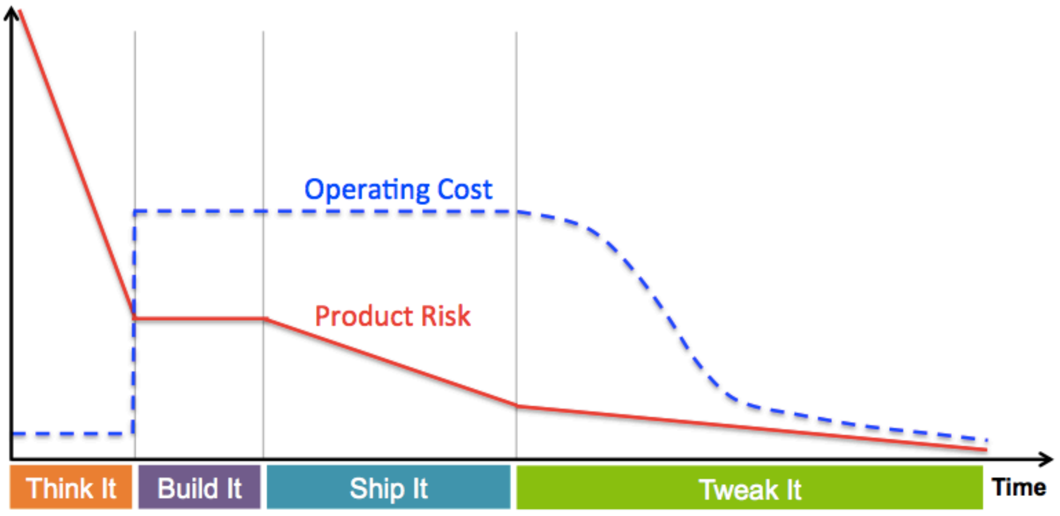


We are also experimenting with forecasting mechanisms, with squads responsible for providing a regularly updated date range (date X – date Y) showing when they think their product will reach the next stage.

Why 4 stages?

The biggest risk is building the wrong product - a product that doesn't delight our users, or doesn't improve success metrics such as user acquisition, user retention, etc. We call this "product risk".

The 4-stage model helps us drive down risk and get products out the door quickly. The graph below shows how product risk is reduced at each stage, and how cost-intensive each stage is.



As you can see, the Think It stage drives down risk at a low cost. You can also see why we want to shorten the Build It stage as much as possible (high operating cost and little risk reduction). The gradually reduced operating cost in Tweak It reflects that, over time, the product doesn't need to be updated as much and squads can start moving on to other things.

The duration of each stage varies a lot, the ratios above are just an example. The total time varies too; some products get to production within a few months, others take a half year or more. Within each stage, though, releases (even if only internal) are done on a fairly continuous basis.

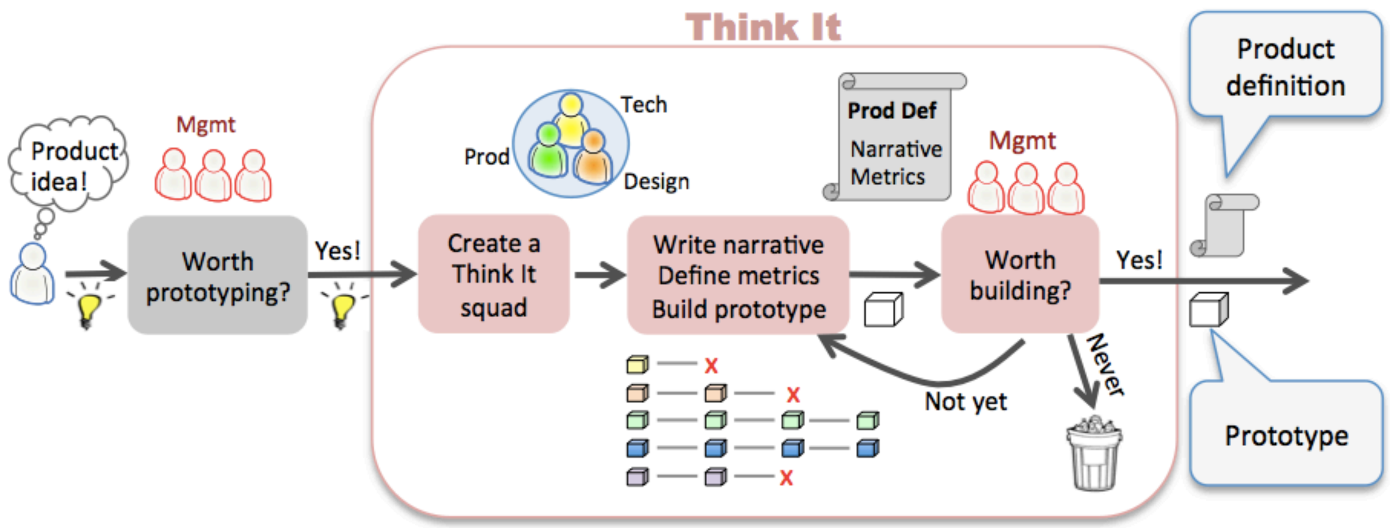
[1] A "squad" is a small, cross-functional, self-organizing development team. For more info, see "Scaling Agile @ Spotify with Tribes, Squads, Chapters, and Guilds".

So, let's take a closer look at each stage.

Think It

Product ideas are born all the time, and can come from anyone in the company. Most ideas are improvements to existing products ("tweaks") and the squads will simply implement and release on their own.

The "Think It" stage is for when someone comes up with a whole new product idea, or wants to reimagine an existing product.



If management agrees that the idea is worth exploring, a small cross-functional "Think It" squad is formed. This typically consists of a developer, a designer and a product owner. Their job is to write a product definition, and build a compelling prototype.

The product definition is a short document that answers questions such as:

- Why should we build this? Who will benefit from this and how?
- What are the key metrics that we expect this product to improve? This could be measured in terms of how much music is streamed, how many downloads, how many logins, etc.
- What are the hypothesis? How will we know if this product is successful?
- Is this a "step change" (that is, a product that we expect will give at least a 2x improvement on the chosen metric)? If we expect only minor improvement on the metrics, there better be some other strong reason for building it, for example a strategic reason.

The product definition is not a requirements document or a project plan. It does not contain lists of features, budgets, resources plans, and such. It is more like a data-driven purpose statement.

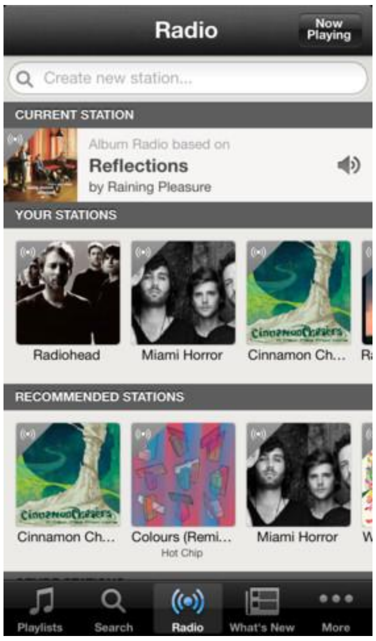
The most important part of the product definition is the *narrative*. What story are we going to tell the world? What will the press release look like?

For example, one recent product is Spotify's "Discover" tab. Here's an excerpt from a 2 minute video presenting this narrative:

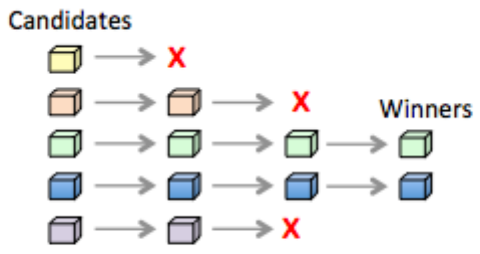
> **Introducing a better way to discover music.**
> Look! Your favourite artist just shared a song with you. We're bringing artists and fans closer than ever before. Like an artist? Just follow them, and share your discoveries with friends.

Another example is Mobile Free Radio, with the narrative "Radio you can save". In this case we used Google Adwords to try several different narratives live and learn which was the most compelling.

The key thing here is: the narrative is written *before* the product is built! That way we make sure that the product is compelling before we even build it.



In addition, the Think It squad builds lots of different prototypes to experiment with the look & feel of this product – both "lo-fi" paper prototypes and "hi-fi" runnable prototypes (but with fake data sources and such). Internal focus groups are used to help figure out which prototypes best convey the narrative, until we've narrowed down to just a few winning candidates.



This is an iterative process with no deadline. The product is simply not worth building until we can show a compelling narrative and a runnable prototype that fulfills it, and we can't decide upfront how long that will take.

As illustrated in the risk vs cost curve above, the Think It stage lets us drive down product risk in a very cost-effective way – we are just prototyping and experimenting. This gives us a cheap and safe way to fail, so we can keep trying until we find out what the right product is to build.

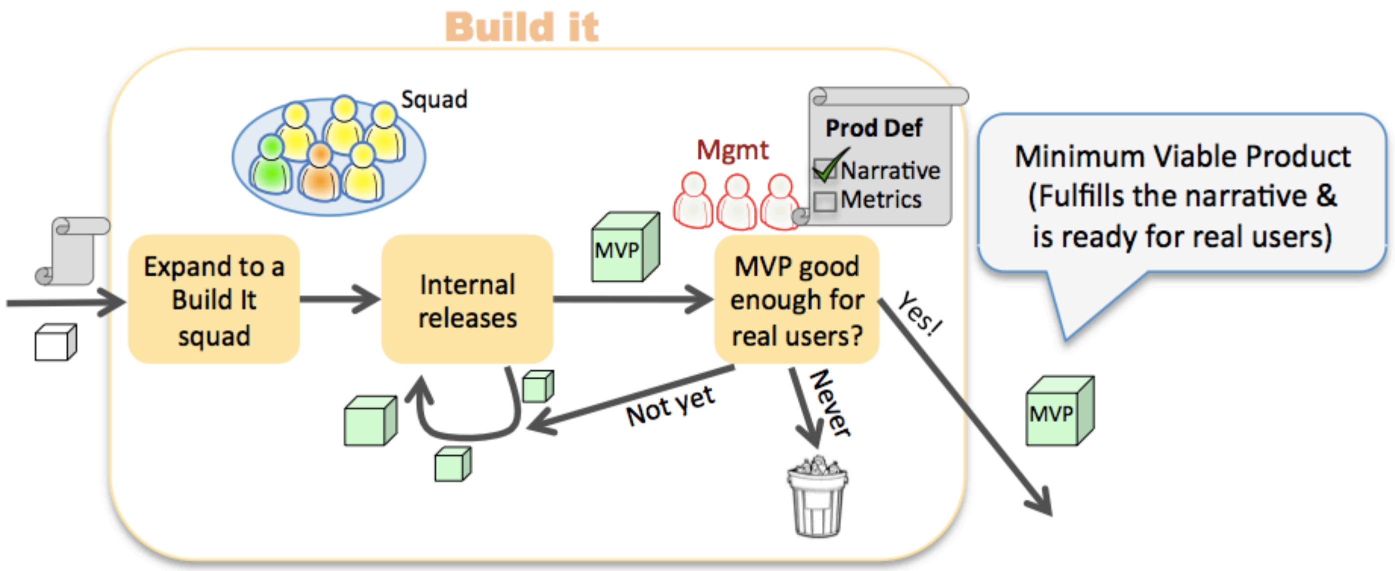
Definition of Done: The Think It stage ends when management and the squad jointly believe that this product is worth building (or that the product will never be worth building and should be discarded).

This is a subjective decision, with no hard data to support it. The hard data comes in the Ship It stage, so we want to get there as quickly as possible.

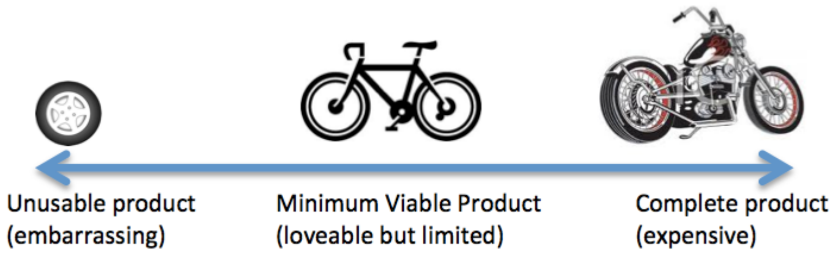
Build it

The Think It squad is now expanded to form a more permanent squad (sometimes multiple squads), with all the skills needed to build, test, and ship the real product. This squad will own the product over the long term, not just during Build It.

The goal of the Build It stage is to build an MVP (Minimum Viable Product) that is good enough to be released to external users, and good enough to prove something about the product. The MVP is built iteratively using Agile software development methods such as Scrum, Kanban, and eXtreme Programming.


Build it
Squad
Mgmt
Prod Def
✓ Narrative
☐ Metrics
Minimum Viable Product (Fulfills the narrative & is ready for real users)
Expand to a Build It squad
Internal releases
MVP
MVP good enough for real users?
Yes!
Not yet
Never
MVP

There is a balance to be found here, illustrated on the useless-to-perfect scale below:


Unusable product (embarrassing)
Minimum Viable Product (loveable but limited)
Complete product (expensive)

On the one hand we don’t want to build a complete product before shipping it, because that would delay our learning. We can’t be sure that we are on the right track until we’ve delivered real software to real users, so we want to get there as quickly as possible. On the other hand, we don’t want to release a useless or embarrassing product. Even if we say it is a beta or alpha, people expect great software from Spotify and judge us by what we release.

So the squad needs to figure out the *smallest possible thing* they can build to fulfill the basic narrative and delight the users. We need it to be narrative-complete, not feature-complete. Perhaps a better term is Minimum Loveable Product. A bicycle is a loveable and useful product for somebody with no better means of transport, but is still very far from the motorcycle that it will evolve into. We do need to fulfill the basic narrative though, or our measurements will be misleading: “Hey, we released a wheel, and nobody used it, so the product is a failure and we shouldn’t build the rest of the bike!”

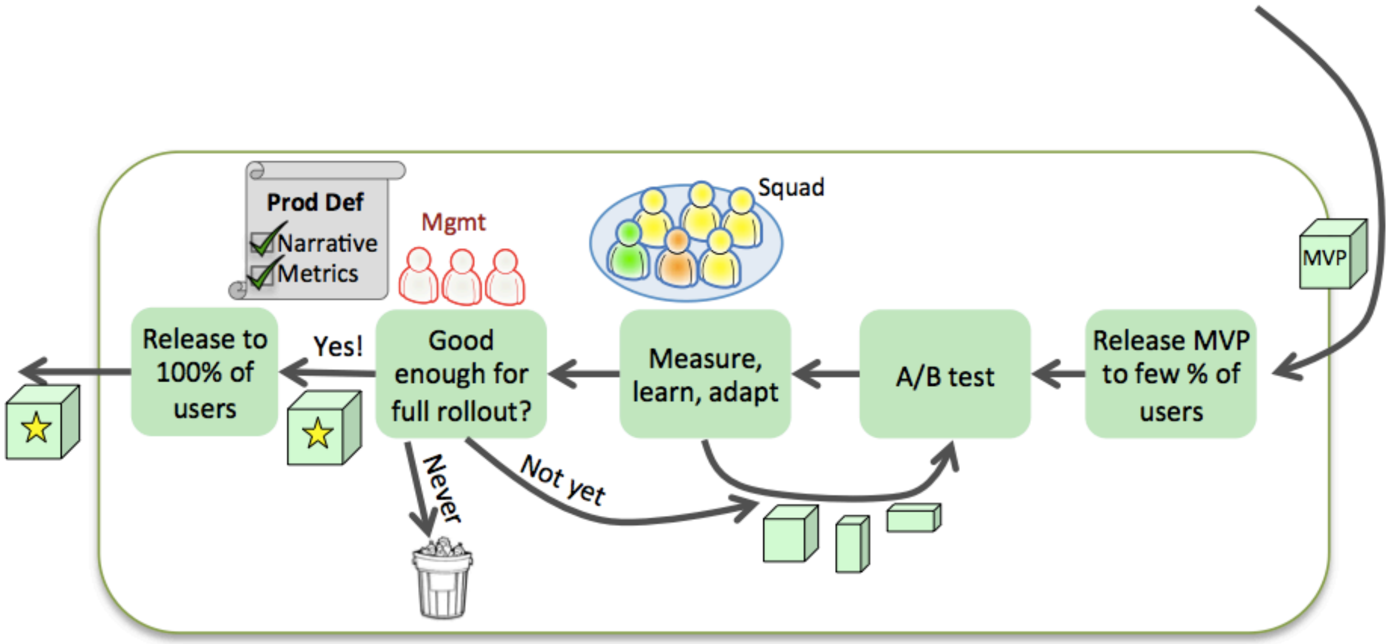
The key difference between Think It and Build It is that, in Think It, we take all shortcuts we can and don’t worry about technical quality. In Build It, we write production-level code and build quality in.

Definition of Done: The Build It stage ends when management and the squad jointly believe that this product fulfills the basic narrative and is good enough to start releasing to real users.

We are ready for the Moment Of Truth!

Ship It

The purpose of the Ship It stage is to gradually roll out the product to 100% of the users, while measuring and ensuring that the product fulfills its promise out in the wild.



The squad starts by releasing to a small percentage of all users (typically 1-5%), in order to collect data. How do those users act, compared to the other 95-99%?

Remember, in the Think It stage we defined a few hypothesis for the product. Now we can finally test if these hypothesis hold true, and iteratively improve the product as necessary. We'll rarely get it right on the first try, and part of the strength in this model is that we don't have to.

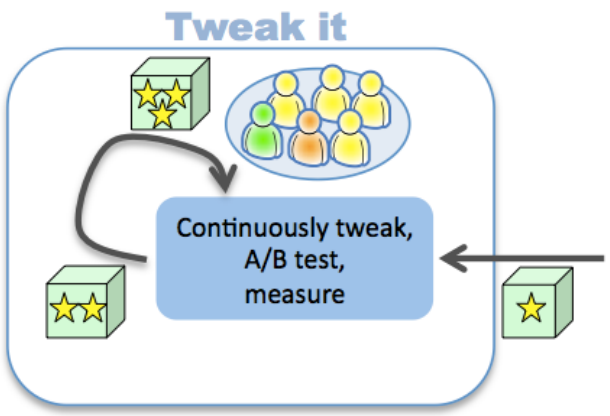
When management and the squad agree that the product is having the intended impact on the small user group, we gradually roll it out to more users, while still measuring and improving. This gives us time to deal with operational aspects such as hardware capacity, monitoring, deployment scripts, scalability, etc.

Definition of Done: The Ship It stage ends when the product is available to all users.

Note that the product is still not "feature complete", finishing Ship It just means that the product (MVP + necessary improvements) has been 100% rolled out. There is no such thing as "feature complete", since the product continuously evolves even after Ship It.

Tweak It

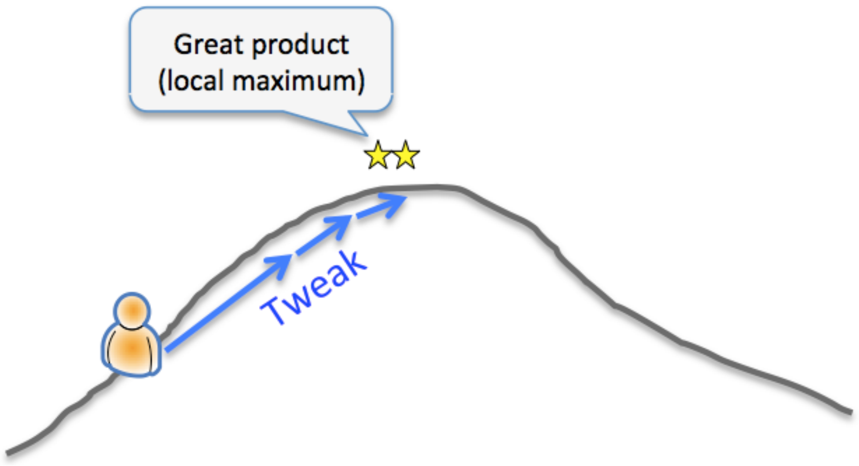
This is the most important stage, since this is where all products end up (unless they get trashed along the way), and the place where products spend most of their time.



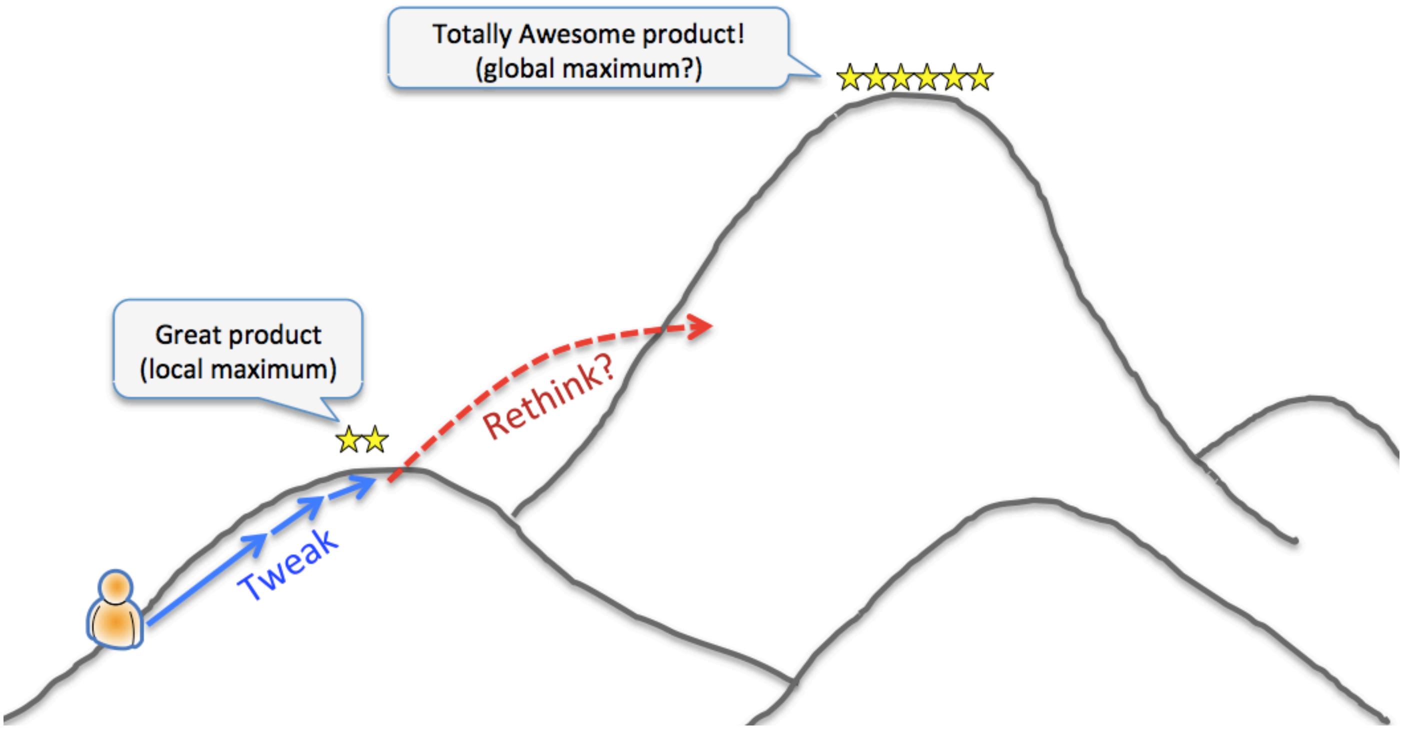
The product is now in production and available to all users. Although it is has proven itself to a certain extent in the Ship It stage, there is always plenty of room for improvement. The squad continues to experiment and A/B test and improve the product, while following up on the metrics. This can include significant new features as well as minor tweaks.

Some time in the future, however, the squad may reach a point of diminishing returns for the product. The product is great, the most important improvements have been made, and the cost/benefit ratio of new feature development is less attractive. Looking at the metrics, new features and improvements don't seem to be moving the needle a lot.

This means the product is approaching a "local maximum".

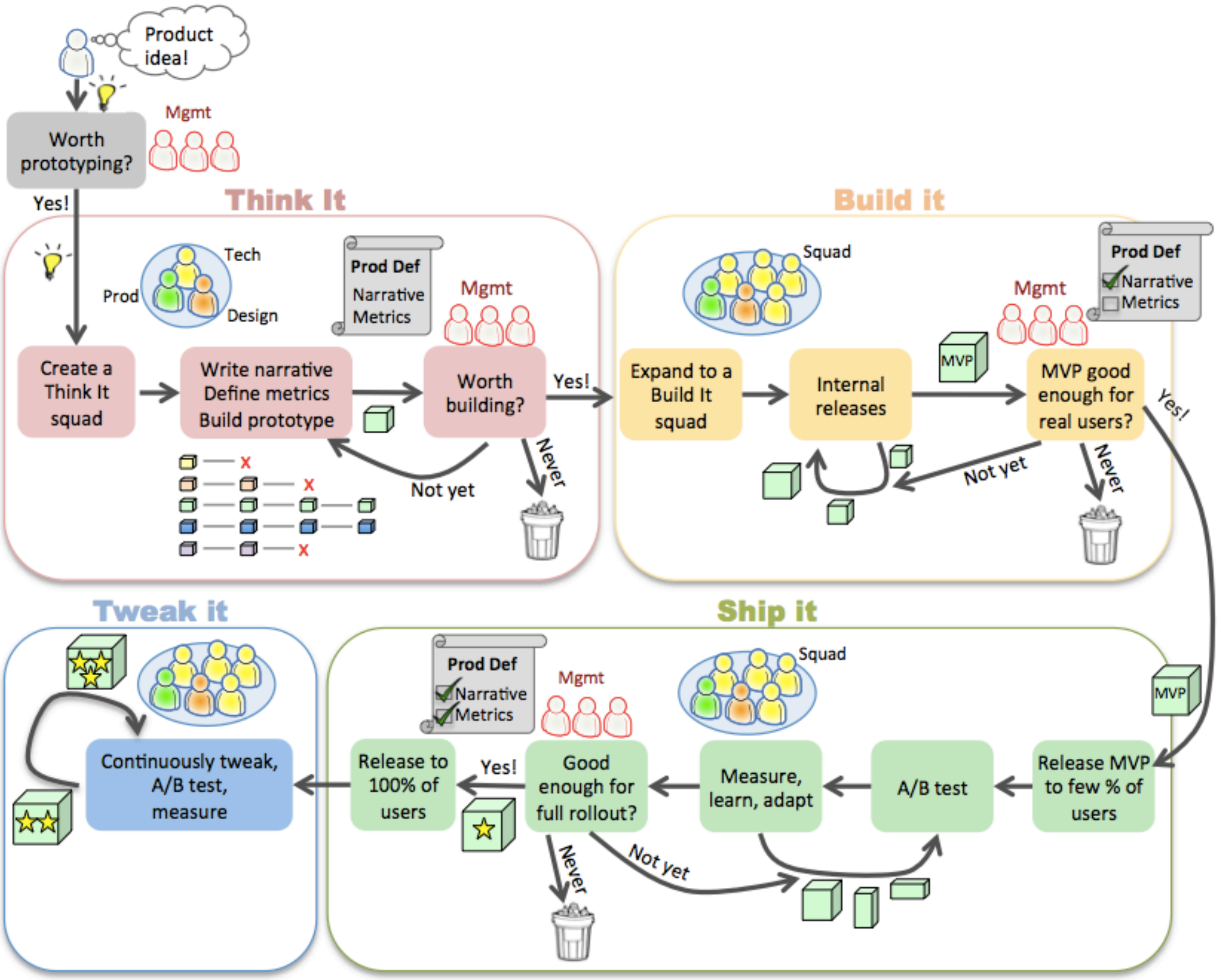


At this point the squad and management will discuss: are we happy being at the top of this hill, or is there a higher peak to be found? In the former case, the squad may gradually move on to other products. In the latter case, the squad may go back to “Think It” to reimagine this product and make a leap for the global maximum (or at least a higher peak…).


Totally Awesome product! (global maximum?)
Great product (local maximum)
Tweak
Rethink?

One example of this is the spotify.com website. The website was tweaked for 4 years before we decided to re-think it in the summer of 2012. The website now conveys the Spotify vision in a completely different and dramatically more effective way.

Graphical Overview



Final words

Hope you enjoyed this article!

If some parts of this model made you think “duh, I already knew that, we’ve been doing that for decades”, you’re probably right. This model isn’t about New And Amazing, it is just about Stuff That Works - new or old doesn’t really matter. I find this combination of practices to be very inspiring and powerful, and I hope you find something that could be useful in your context.

If you have any feedback or questions, feel free to get in touch, or write comments on the blog. We probably won’t have time to answer detailed questions, but might add an FAQ to this article based on the most common questions.

/Henrik Kniberg
henrik.kniberg@spotify.com
http://www.crisp.se/henrik.kniberg