Hiten Shah

5 Habits

to Building Better Products Faster

Intro

5 Habits to Building Better Products Faster

Hiten Shah,
Co-Founder, Product Habits

Most people begin product development making the same fatal mistake: they try to build something that their customers want. They assume that there's one right answer to what their customers actually want.

Reality is much messier and will continually surprise you.

That's why it's so important to ground yourself with good **habits** before stringing together a single line of code. Your habits are the basic building blocks of everything you do and they seep into everything you create.

Stick with me, and I'll help you form the habits you need to create better products, faster. We'll dive into frameworks for how to think about products and build a stronger problem-solving process. For each product habit, we'll examine case studies of successful and failed products alike before getting to the nitty-gritty details of how to build the habit.

Ultimately, your process, your people, and the things you make all emerge from these building blocks. Putting together a rock solid foundation is what allows you not just to survive an unpredictable market, but thrive in the face of it by leading with product.

Chapter 1
Always Start with Customers

It's a new world of product development. Not that long ago, it took millions of dollars and a team of engineers to create a working prototype of a product. Today, a lone developer can hack together the same code on their own in half an hour. This creates a huge amount of freedom for product development, and a bigger window for what's possible—along with a completely new set of challenges.

No one spends three years designing one product anymore because the speed of implementation means that what you set out to build in the first year will be obsolete by the third. The challenge now is to be almost telepathic: know what your customers want, before you actually get anything into their hands. Cut overhead, ship fast, and iterate even faster.

The only way to do this is to *start* with the customer.

You Are Not Your Customer

Make starting with customers a **habit** that you exercise without thinking. It's the easiest "hack" you achieve as you develop your product.

As Lean Customer Development author Cindy Alvarez writes, "Customer development starts with a shift in mindset. Instead of assuming that your ideas and intuitions are correct and embarking on product development, you will be actively trying to poke holes in your ideas, to prove yourself wrong, and to invalidate your hypotheses. " Cindy brought this discipline to KISSmetrics where she helped us build out our analytics product before heading to Yammer and Microsoft.

If you can figure out what your customers need—*instead of what you think they need*—you can use that knowledge to create something they'll actually pay for while you're still building your product. That's customer development in a nutshell.

Most founders and product managers fall flat when they try to solve their own problems. They're constantly told to "build for themselves," and create products that alleviate pain points that they've experienced first-hand.

This is great for starting out, but it's not enough to reach a wider market or build a sustainable business. There's one simple reason why: **You are not your customer. There are way more of them than you.**

If you don't build the habit of starting with your customers and understanding what makes them tick, you're not going to know how to meet their basic needs, let alone improve your product down the line. You'll be unable to differentiate your product from the thousands already out there and will end up dead in the water.

It feels counterintuitive, but the early stages of product development aren't at all about your constraints or your resources. They're about focusing on what's important: the customer. The customer is the building block and everything you do should revolve around them.

Work Backwards Like Amazon

Amazon's approach to product development is called "working backwards," a process that each new product initiative at the company begins with. Ian McAllister, director of Airbnb and former GM of Amazon, describes this in detail:

> *For new initiatives a product manager typically starts by writing an internal press release announcing the finished product. The target audience for the press release is the new/updated product's customers, which can be retail customers or internal users of a tool or technology. Internal press releases are centered around the customer problem, how current solutions (internal or external) fail, and how the new product will blow away existing solutions.*

This is followed by an FAQ and an appendix. The total “planning” document created by working backwards is six pages long.

Working backwards is a framework for how to think about product without lengthy roadmaps that end up being scrapped. It’s a way to short-circuit the traditional product development track, and make sure that you build something that your customers will actually care about.

Amazon Web Services (AWS) is probably the best example of how far the “working backwards” methodology can take you. It’s a $9.6B run rate business, and helped launch the cloud. One-third of daily internet users visit websites built on top of Amazon’s cloud. AWS accounts for an incredible 36% of Amazon’s operating profits.

How Amazon Does It

Take a look at the 2006 press release for Amazon’s S3-Simple Storage Service, one of the first and best-known AWS products to be launched:

<< Back

Amazon Web Services Launches

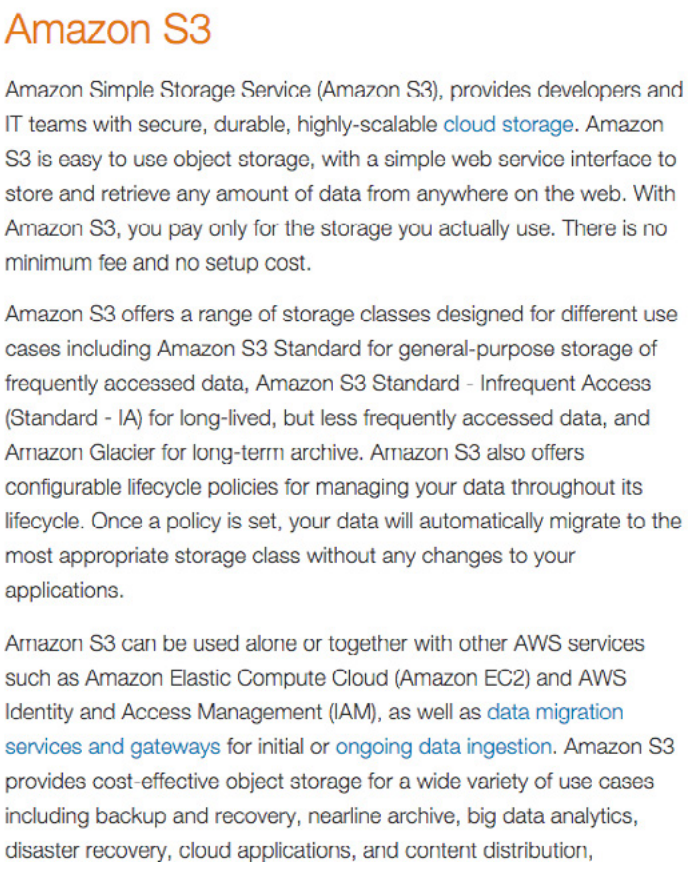
SEATTLE--(BUSINESS WIRE)--March 14, 2006-- S3 Provides Application Programming Interface for Highly Scalable, Reliable, Low-Latency Storage at Very Low Costs

Amazon Web Services today announced "Amazon S3(TM)," a simple storage service that offers software developers a highly scalable, reliable, and low-latency data storage infrastructure at very low costs. Amazon S3 is available today at http://aws.amazon.com/s3.

Amazon S3 is storage for the Internet. It's designed to make web-scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

It looks simple, but when he was putting together a pitch for Amazon Web Services, the current head of AWS Andy Jassy tore through 31 drafts of the initial press release before taking it to Jeff Bezos.

The landing page for Amazon S3 doesn't look so different today. What's really amazing is how little the press releases have changed over the past seven years, especially considering the speed and scale at which AWS took over the cloud. Just two small paragraphs have been added to the current product description:



Amazon S3

Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable cloud storage. Amazon S3 is easy to use object storage, with a simple web service interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. There is no minimum fee and no setup cost.

Amazon S3 offers a range of storage classes designed for different use cases including Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard - Infrequent Access (Standard - IA) for long-lived, but less frequently accessed data, and Amazon Glacier for long-term archive. Amazon S3 also offers configurable lifecycle policies for managing your data throughout its lifecycle. Once a policy is set, your data will automatically migrate to the most appropriate storage class without any changes to your applications.

Amazon S3 can be used alone or together with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2) and AWS Identity and Access Management (IAM), as well as data migration services and gateways for initial or ongoing data ingestion. Amazon S3 provides cost-effective object storage for a wide variety of use cases including backup and recovery, nearline archive, big data analytics, disaster recovery, cloud applications, and content distribution,

The massive scope of the original product vision—"a simple web service interface to store and retrieve any amount of data from anywhere on the web"—is unchanged. Amazon was working backwards from "a mental image of a college kid in his dorm room having the same access, the same scalability and same infrastructure costs as the largest businesses in the world." Today, Airbnb, Yelp, Slack, and Netflix are all hosted on Amazon's servers.

Starting with a press release or blog post changes the game. It allows you to launch your product in an intimate, customer-centric way. As you build your product out, you use it as a guiding light to ensure that you're on the right track.

In your "working backwards" document, describe the problem that you're trying to solve, and how your product does it in a unique way. If your product's features sound boring and dry to customers, don't build them. Rewrite the blog post until you have something you'd be happy publishing. Rewriting drafts is *much* less expensive than rewriting code.

How to Create Better Product Habits

It's not enough just to tell your product team to write a press release and expect them to automatically start thinking about the customer. The reason why the "working backwards" approach has been so successful for Amazon is that they've sewn customer-centricity into the fabric of the entire company.

Deep product habits have to be built up over time, through small triggers and reminders that motivate action.

Working backwards isn't a silver bullet that guarantees you success on the scale of AWS. It simply forces you, and everyone in the company, to build around the customer.

Here are a couple proven tactics for building better product habits that can help get your team on board:

- **Give the customer a chair in the meeting.** At Amazon, every meeting that Jeff Bezos attends has an empty chair representing the customer—the "most important person in the room." This is a strong reminder to everyone at the highest level of the food chain that the customer holds the throne.
- **Consider scrapping your product roadmap.** Product roadmaps never actually hold up past the first month or so. Often, product roadmaps represent commitments that shouldn't be kept. Instead of working forward on a product roadmap, work backwards from the customer.
- **Force customer-centricity into the code.** Back in 2002, Jeff Bezos issued a 6-point memo to engineers at Amazon. One of his key points was that "all service interfaces, without exception, need to be externalizable." This meant that everything engineers at Amazon built internally needed to be built as if the customer would use it. These constraints forced Amazon engineers and developers to think of the customer as they wrote code. Today this mentality is hardwired into the company.

Jobs-To-Be-Done at Intercom

People hire your product out to do a job. For a long time, a lot of these jobs were essentially taken care of by Microsoft Excel. Have a list of leads? Stick it in a spreadsheet. Need to visualize data? Stick it in a pivot table. People like to throw rocks at Excel, but at the end of the day it's a spreadsheet, and it does the job.

Most new solutions are hired to take care of old jobs—the solutions change but the jobs stay the same. Intercom's book on Product Management gives the perfect example of this:

> *People, particularly students and young people, wanted to pass notes and messages, without fear of other people seeing them...People still want this so today they use Snapchat.*

People used to burn sensitive letters. Now there's Snapchat. Understanding the job is the only way to define what your customers want from your product in their *own* context, instead of yours.

Des Traynor, Intercom's founder, explains:

> *If you're building a new product, it's because you believe you can create a better solution that people will want to use because it delivers a better outcome. A strong understanding of the outcome customers want, and how they currently get it, is essential for you to succeed in product development.*

Intercom is a company that makes jobs-to-be-done an extreme habit. They understand that product development isn't about how customers use your tool today. It's about the job your customers need handled. Whether your product becomes the new industry standard or gets chucked into the dustbin **doesn't matter**. The job still needs to be done.

When people start product pitches with phrases like "Wouldn't it be cool if..." and "What do you think about..." it's always speculative. The "jobs-to-be-done" framework is a way to constantly align product development back to first principles—the customers.

How intercom Uses "Jobs-To-Be-Done" for Marketing

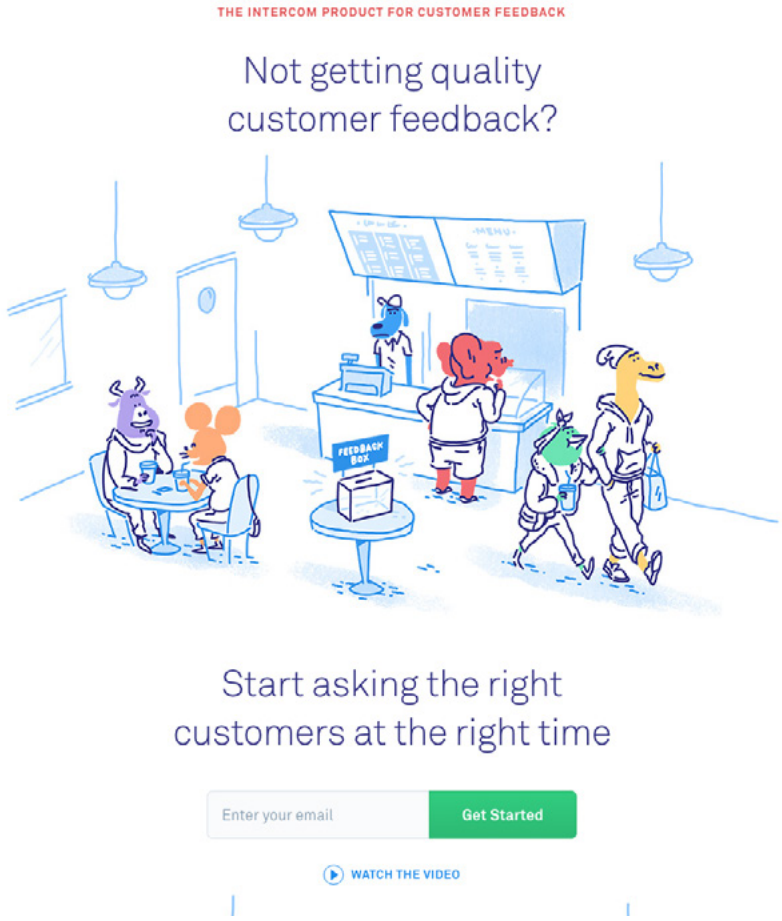
The basic formula for jobs-to-be-done is simple: **When _____, I want to _____, so I can _____.**

Intercom has written extensively about how it applies the jobs-to-be-done framework to product development. Looking at Intercom's marketing site, it's quickly apparent just how obsessively it practices this habit.

Intercom built a single workflow for managing customer communications, and each of the company's products solves a sub-job that's a part of the larger one.

On the homepage, you see marketing copy that's super broad: Intercom makes communicating with your customers "simple and personal for everyone." The landing page for each product—Acquire, Engage, Customer Feedback, Customer Support—literally paints the job-to-be-done, and reinforces it with clear marketing copy.

The Job: Customer Feedback



Jobs-to-be-done formula: When I talk to customers, **I want** to start conversations with the right customers at the right time, **so I can get** quality customer feedback.

Each of Intercom's products is framed in a similar way. The landing page starts with the problem, posed as a question. Each product presents a hyper-specific solution to that problem.

All this goes back to creating habits. The jobs are the basic building blocks that you create your product on top of. The speed, efficiency, and usability of your product will change over time—but how successful it is will always link back to the basic job that the customer's trying to accomplish.

The jobs-to-be-done framework forces you into the habit of always thinking about the customer and their problems, rather than just building for the sake of it. It makes the customer the compass that drives the direction of your product development.

How to Create Better Product Habits

Working backwards from Intercom's marketing site, it's easy to see how to apply the jobs-to-be-done framework after the fact. But to build strong product habits that can sustain a business, you have to start from scratch.

At the end of the day, getting your entire product team to focus obsessively on the job-to-be-done is simply a good habit. It allows you to build better products faster—ones that your customers will actually use.

Here's how to orient your team around jobs-to-be-done:

- **Build it in Excel.** Software engineers and developers often exist on a technical level that is far removed from the people they're building for. Get your software engineers to try and work out the job-to-be-done in Excel—the way that most of your customers are probably doing it—to get them closer to that pain.
- **Work the marketing circuit.** Intercom's genius is that it understood that the jobs-to-be-done framework can apply to everything. Getting your product people in the same headspace as your marketers forces them to think through *problems* and customers, instead of thinking about them abstractly.
- **"All Hands Support."** Try having everyone in the company, especially the engineers on your product team, handle customer support tickets at least once a month. It's one thing to look at reams of data and customer feedback in aggregate, and another entirely to work through specific support issues that customers face. This is something that Wistia, the video hosting and analytics company, does religiously. As Wistia CEO Chris Savage says, "employee motivation should be aligned with happy customers." This comes from getting your product people in front of your customers.

Focus on the Things That Don't Change

The building block of Working Backwards, jobs-to-be-done, and customer development is your customers. If you make them the center and beginning of every product initiative, no matter how big or small, you hardwire them into a central part of your company culture.

Your approach to your customers is a reflection of your company's mission. It's not just about product-market fit, but positioning, marketing and sales. A customer-centric ethos will keep your entire company focused on building the best, most useful product it can.

Chapter 2
Continuously Improve

In November 2005, my co-founder Neil Patel and I launched a podcast advertising network called Fruitcast. We didn't know a thing about podcasting, but we did think it made sense that there should be a marketplace like AdWords for the web for companies to get their ads into podcasts.

We outsourced development to a dev shop and had it built in around two months for $45,000. When Michael Arrington wrote about it for Techcrunch, he pointed out one fatal flaw in our product:

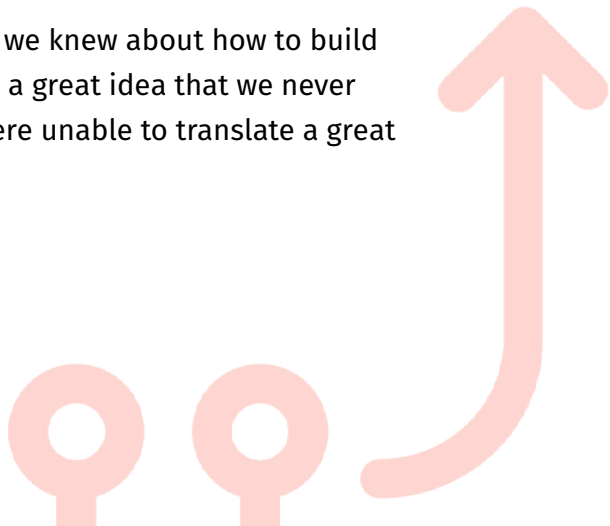
> *It looks like they are paying out on a per-download basis, and the site suggests that podcasters can expect to receive $0.25 per listener. That's a hefty $250 CPM to the advertiser... which is way above what radio stations charge for ads.*

Wow, we had no clue. We didn't know anything about radio stations, or podcasts, or advertising. We were shooting blind.

Around a year after launch, we shut down Fruitcast for good. One of the developers who worked on the project wrote a line in an email that said, "I was sincerely hoping we'd find some magical combination of events to jump start."

There were a lot of reasons why Fruitcast failed, but that was the big one. A lot of companies never find product-market fit, fizzle out, and die—and that's fine.

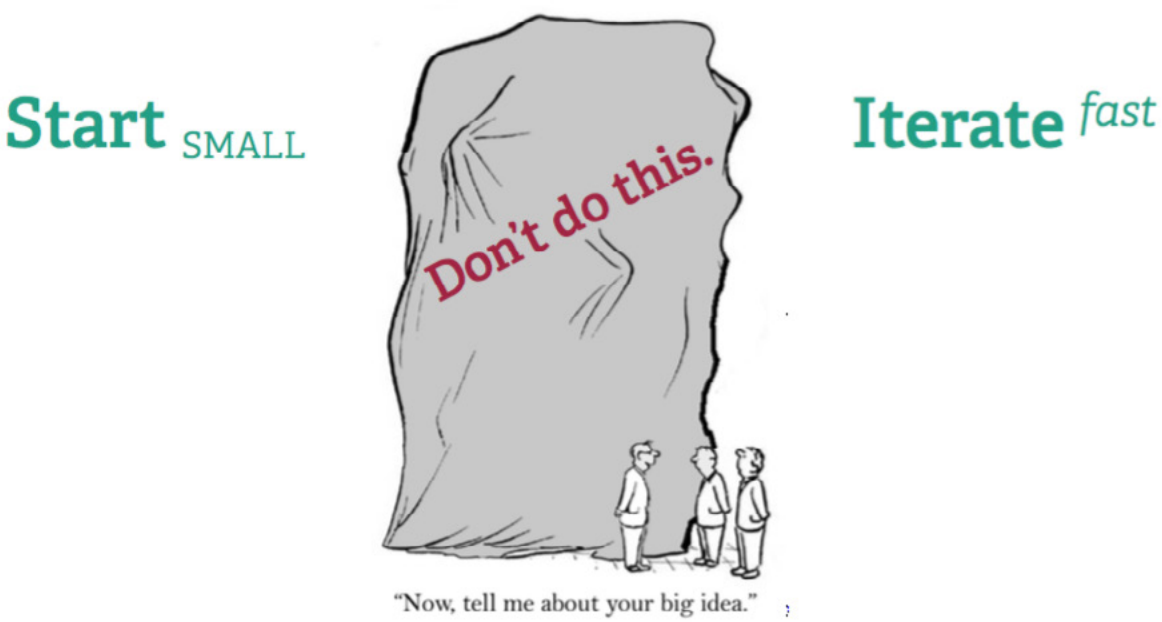
Where we really kicked ourselves was that—against everything we knew about how to build product—we fell into that trap of being so assured that we had a great idea that we never went out to learn everything we needed to. Without this, we were unable to translate a great idea into a product with product-market fit.



That's why building a habit of continuous learning and improvement is one of the most important things your product organization can do—because it counterbalances your ego and fear of being wrong and looking stupid.

Don't ship sh!t. Ship products that people want. Here's how.

Minimum Viable Everything



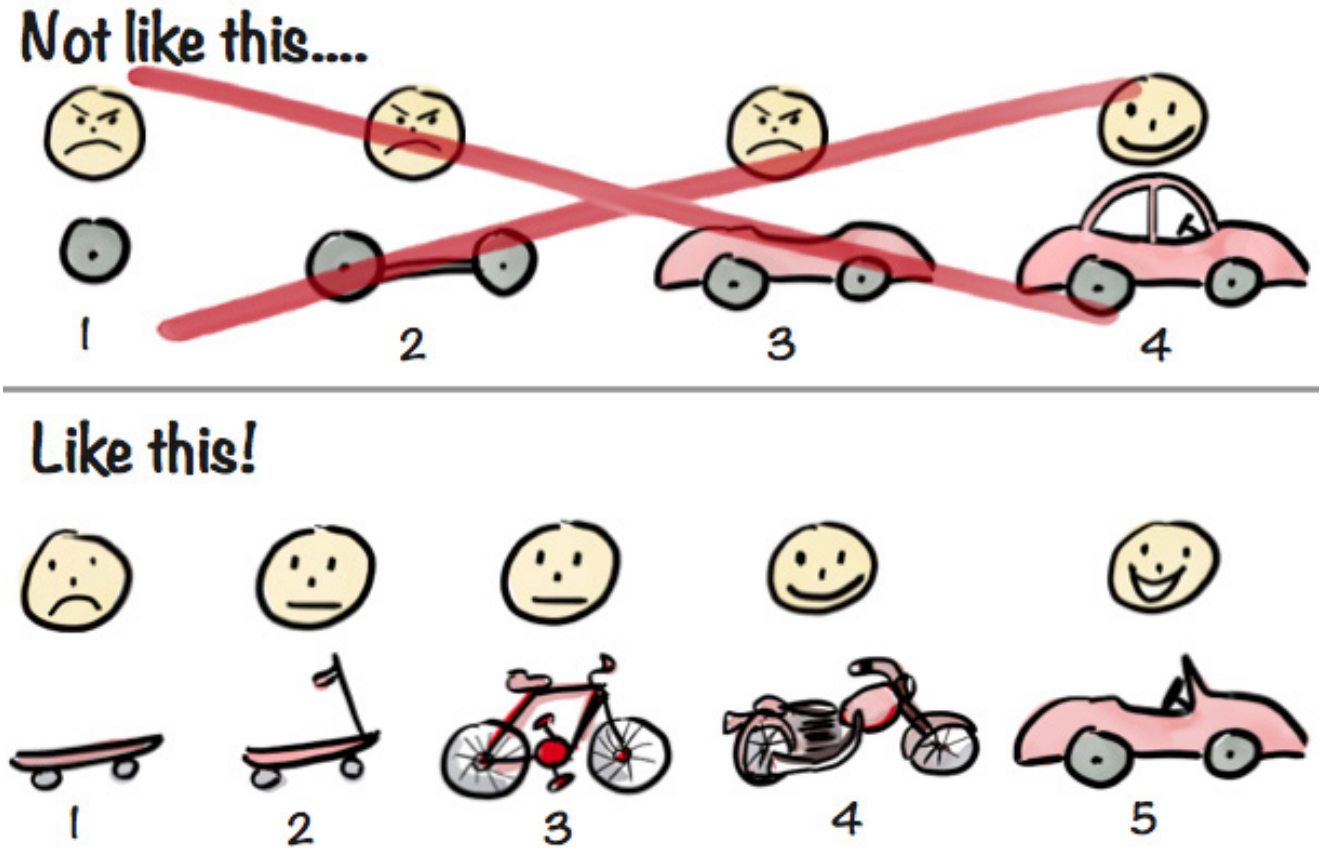
Minimum Viable Product has become the default product development methodology. The basic idea behind MVP is that, rather than clinging on to your big idea, you build something small and iterate fast.

The mistake that most people make is that they confuse MVP for a go-to-market strategy. It's not—MVP is about building product in a structured way that forces you to learn.

- **Minimum:** Make your learning cycles as short as possible. "If you are not embarrassed by the first version of your product, you've launched too late." —Reid Hoffman, co-founder of LinkedIn.
- **Viable:** Test a simple, crystal-clear hypothesis. "Complex ideas are almost always a sign of muddled thinking or a made up problem." —Sam Altman, President of Y Combinator.

The MVP methodology allows you to break product development into a series of steps so you can keep iterating—and improving. But what makes it work is a focus on *learning* rather than *building*. Nail the first part down, and the second follows naturally.

A Skateboard for Henry Ford



(image source: Henrik Kniberg)

Henry Ford famously said, "If I asked what my customers want, they would have said faster horses." He gave them cars instead. Today, you should probably give them a skateboard.

Spotify's analogy for MVP is building a skateboard when a customer asks for a car. Instead of setting out to build a car by making a wheel, start with a skateboard.

- Nobody can do anything with a wheel, or two wheels—it's not **viable**. You can't learn anything about whether they want a car or how they'd use a car by watching them play with two wheels.
- Building a car is equal in time to building 1,000 skateboards—it's not **minimal**. Trying to jump from nothing to a car first falls into the fallacy of the "big idea." And after all that time and money spent on a car, you have no idea if it's what your customer actually wants.

As you build and iterate on MVP, you're learning whether your product has a market. What Marc Andreessen of Andreessen Horowitz said 7 years ago applies equally today: "The #1 company-killer is lack of market." The skateboard stage tests the **riskiest hypothesis**: that customers want to get from point A to point B.

The skateboard will still take the customer from point A to point B, which means you can learn whether this is actually something they care about. It's the most minimum stage of your product, but also the most important because it's the entire premise of the undertaking.

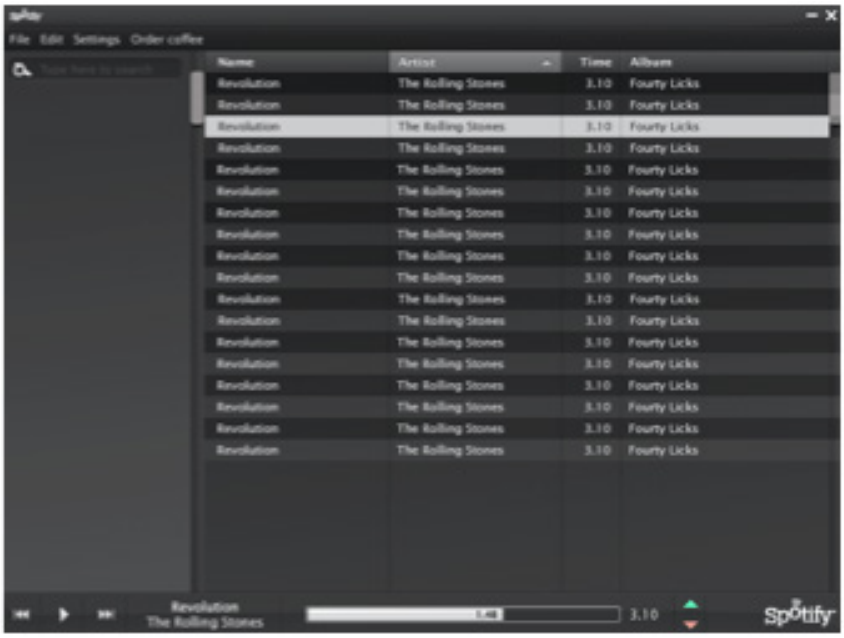
As you build from the skateboard to the car, you're learning at every iteration and getting closer and closer to product-market fit. Your goal is to build less so you can get your product in your customer's hands *sooner*, and learn *faster*.

MVP in Action

When Spotify started building its product in 2006, streaming technology was extremely nascent. Products like Real Player were incredibly slow, buggy, and often just didn't work. With its MVP, Spotify set out to obsessively make streaming music the same as if it were playing from your hard drive.

There was one key hypothesis that Spotify set out to test: **People don't mind streaming music if it works as well as owning it.**

The minimum viable product looked like this:



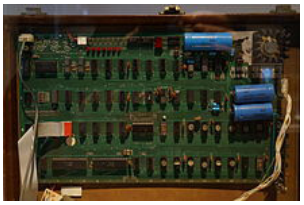
(image source: Henrik Kniberg)

There isn't much to it on the surface. It basically didn't have any features, and was only able to play a couple of pre-determined songs that were "hard-coded." What the folks at Spotify were doing with their MVP was to see if their basic hypothesis held up.

They quickly learned that they needed to focus on latency, which was something that most of Spotify's competitors at the time overlooked and didn't put obsessive focus into. As Daniel Ek, co-founder and CEO of Spotify, put it:

> *"We spent an insane amount of time focusing on latency, when no one cared, because we were hell bent on making it feel like you had all the world's music on your hard drive. Obsessing over small details can sometimes make all the difference. That's what I believe is the biggest misunderstanding about the minimum viable product concept. That is the V in the MVP."*

Here are a few other examples of successful MVPs:



- **Apple:** The first Apple computer (Apple I) was basically just a motherboard. It didn't include a keyboard or a case, and had to be assembled manually. The early mode of MVP came from Wozniak's hobbyist mentality of playing with technology and tweaking it with friends. If you look at the sleek unibody design of a Macbook Pro Retina today, you can trace it back to a series of continuous improvements upon this over the past 30 years.

- **Dropbox**: When Dropbox was developing its early product, no one knew that syncing files to the cloud was something they needed in their lives. Instead of adopting the old "build it and they will come" mentality, Dropbox founder Drew Houston created an MVP in the form of a 4-minute demo video of how Dropbox *would* work. Dropbox's waiting list shot up from 5,000 to 75,000 people overnight, which gave him a massive community of people to learn from.

- **LinkedIn:** LinkedIn's MVP included user profiles, the ability to invite people to connect, search for users, and send email-like requests along with a page summary of network stats. No muss, no fuss.

- **Groupon:** Groupon started off as a Wordpress blog that posted daily discounts, gift certificates and vouchers for the Chicago area. According to co-founder Andrew Mason, the interest in the MVP "was enough to prove the concept and show that it was something that people really liked."

How to Build the Habit

Crucial to actually developing your MVP is building the habit of MVP as a mindset. You can't just pay lip-service to the principle—you need to actually buckle down and focus on learning.

By building the habit of MVP, you give yourself the objective distance you need to rigorously test the viability of your ideas.

- **"No tasks longer than one week."** At AngelList, they have a rule that "you have to ship something into live production every week—worst case, two weeks." This time constraint means that you have to design your projects to be viable in a minimal time frame.
- **Sell before you build.** Social media marketing startup Buffer moved from idea to paying customers in 7 weeks. Its MVP was a simple landing page that described a basic scheduling tool for Twitter—only when you clicked on "Plans and Pricing," it took you to a newsletter sign-up. Buffer hadn't actually built its product yet, but the interest they received was a clear market that they had an idea worth selling.
- **Make Product accountable to your customers.** Some do this with a forced cadence of product email updates. Others do it with all-hands support. Both make Product conscious that they have to deliver something that improves customers' lives, quickly, not just ship changes in the abstract.

Write or Die

In a rush to move quickly, startups often get sloppy with one of the most important habits for continuous learning and improvement: writing stuff down.

If you don't write stuff down, it's impossible for your whole organization to continuously learn and improve.

Every time you do something that another team member has to do later, you should take an extra 5-10 minutes to write it down or even sketch it out. If you make documentation digital and shareable for everyone on the team, everything will improve more quickly from building product to onboarding new team members.

Ben Horowitz of Andreessen Horowitz says:

Why writing stuff down matters:

- As a company, it's how you build your knowledge base, and create a living repository of everything you've learned developing product. That's the foundation you build on for learning with each iteration and product experiment.
- It forces your processes, people, and quality of thought to be better. As Intel CEO Andy Grove once said, documentation is often "more a medium of self-discipline than a way to communicate information."

> *Good product managers gather information from engineering informally and verbally, but good product managers give direction in writing to engineering....Written communication to engineering is superior because it is more consistent across an entire product team, it is more lasting, it raises accountability.*

If team members want to know why a certain process is in place, they can look at all the documentation behind it and understand the decision behind it. The only way to rapidly improve as a small product team is to have everyone involved in improvement, and the only way to achieve this is by putting pen to paper.

The KISSmetrics Golden Motion PROCESS

Back in 2012, with 25 different team members spread across 4 different time zones, everything at KISSmetrics felt like it was moving glacially. Meetings were bungled and inefficient, and emails were a nightmare. It was like the left hand didn't know what the right hand was doing.

This all changed dramatically when we did something super simple—we wrote down our product development process.

Our CTO John Butler put together what he called “Golden Motion Process,” which made explicit the implicit product development process that we aspired to. The Golden Motion was a two-week period of time where as a company we focused on building a product feature or set of features to improve a single metric.

Our lead designer simplified the basic steps into a graphic:

This gave everyone on the team an easily digestible diagram to understand the process and refer back to.

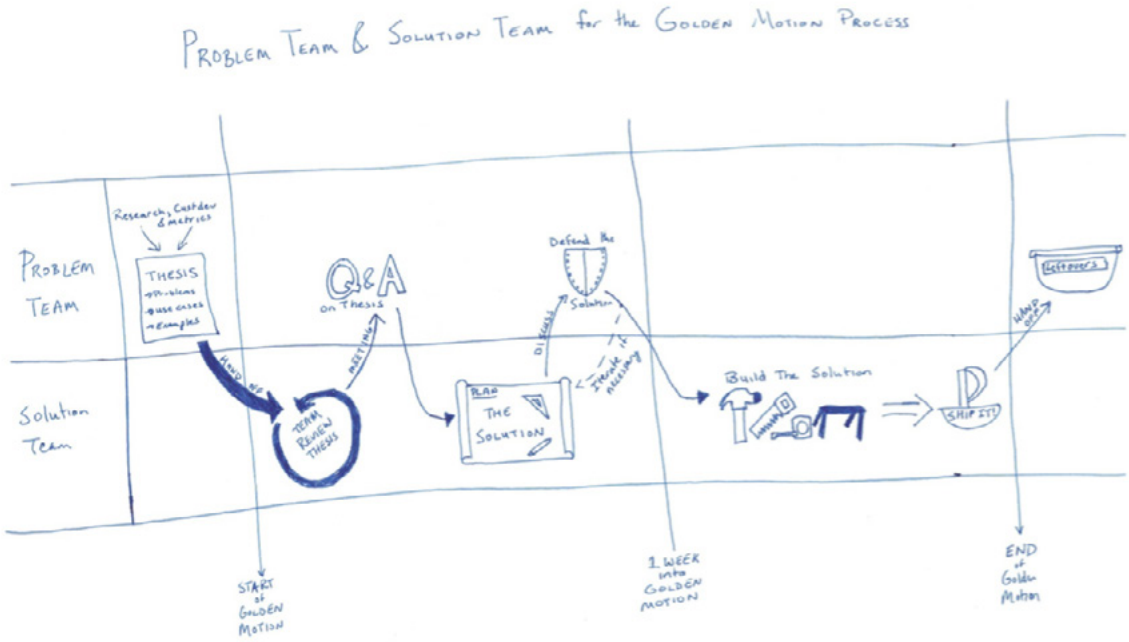
To kick it off, and demonstrate our commitment to it, we flew team members in from around the world to a weekend summit in San Francisco. From the beginning, the focus was 100% on the idea that the Golden Motion was a “work in progress”—something that we could add to and iterate upon over time, just as we were improving our product.

Here’s the beginning of the document that was passed around the team:

> *First off, I want to point out that the GM process is a work in progress—we learn something new from each GM and we iterate on the process every time we start a new GM. This GM is no different. We’ll be testing some new ideas and getting feedback from you (and the other team) as to how the process is working. (In fact, we just made a few tweaks this afternoon as you may have noticed from the email that Steve sent out earlier today.)*

That’s when the remarkable happened. Before we wrote it down, we could only vaguely complain about the problems with our process and see its negative effects on the team. After we wrote it down, a switch flipped—we could actually work on, tweak and improve our process.

The next iteration of the process started out with a sketch that looked like this from one of our designers:



What started as someone's idea sketched out on the back of the napkin developed into a formal process that improved how the whole team operated. Stuck inside one person's head, it would have been useless.

The emphasis on the Golden Motion Process was constantly about shipping faster—but with an intent on learning, rather than building. By the end of the Golden Motion, we had blown our 25% OKR out of the water.

At a very basic level, the principle behind documentation is simple. It's literally about getting everyone on the same page. Everyone, from leadership down to individual team members, is essential to how product is developed. If a designer creates 5 different versions of a design, it's worth taking the extra 5-10 minutes to clarify to the next person why each version was made.

This iteration introduced the problem team and the solution team:

- **The Problem Team** determined the direction of the product and figured out what we'd build next. It focused on conducting continuous customer development to dive into the root problems and opportunities around our product. It also was responsible for digging into the analytics around our product and A/B tests.
- **The Solution Team** was responsible for solving the problems identified by the Problem team. It was responsible for improving existing features and creating new ones.

Building better products faster comes down to one guiding light: Do less and make it count for more.

How to Build the Habit

Writing it down is about taking what's in someone's head and putting it on paper. It's important to make sure that the answers exist—and they exist because someone can pick them up later.

Creating good product habits, like thorough documentation, can seem menial when you're just two people in a room. But as you grow and scale, spending an extra 5-10 minutes to clearly communicate what's in your head is the only way you can scale. Documentation is a **reusable asset,** and one that accrues in value and in quantity over time.

In order to be good at building products, you have to be able to break apart the problems behind them and approach these problems from a multitude of different angles. What enables continuous improvement is the ability to build off of the universal first step, and the second, and the third, synchronously as a team. Documentation provides the steps, and habit makes it come alive.

Here are a few of methods to get documentation right:

- **Get outside the product team.** To make sure your product team is on the same page, Ben Horowitz advises getting "someone outside the product team to ask 5 different people in engineering, QA, and doc what their product is supposed to do and why and get the same answer."
- **Make someone else do it.** The stress test for whether or not a piece of documentation is good enough to pass the bar is to have someone else read what you've written and try to apply it. If someone else can't repeat the process that you've written down, you've written it down wrong.
- **If it's not on paper, it doesn't exist.** Distributed knowledge on product processes throws a wrench in the gears—they may as well not exist. In order to reinforce the importance of documentation for your team, treat them as if they don't exist. If you only give credit for something that's written down, chances are high that your team will write it down.

Fail Fast, Be Foolish, LEARN

Fruitcast was neither my first failure nor my largest. A couple years later, my co-founder and I sank a million dollars into a website-hosting marketplace, which we never bothered to release. But, following these were the successes of Crazy Egg and KISSmetrics.

As a founder or a product leader, you will fail a lot. The only way to cope with it is to make those failures mean something, and the only way to achieve that is by continuously improving. If you build a system around product development that is truly focused on learning, then controlled failures all become assets that you can turn into a competitive advantage. This doesn't guarantee that your product will see traction, and take off—but the alternative is to rely on *dumb* luck.

Chapter 3
Think Deeply

For Newton, deep thinking didn't start with abstraction. It started with an apple falling on his head. In their book, 5 Elements of Effective Thinking, Edward Burger and Michael Starbird write:

> *Newton described the universe—the behavior of the sun, the planets, and distant stars—using the same laws that describe everyday occurrences like apples falling from trees. The simple and familiar hold the secrets of the complex and unknown. The depth with which you master the basics influences how well you understand everything after that.*

This is a simple, fundamental truth to building product. Start by thinking deeply about the basics, instead of everything that you're hoping to achieve. The surprises come as you build. Thinking deeply allowed Newton to understand physics. He may have had grand dreams and ambitions, but he started from thinking deeply about what he knew, and the motion of an apple dropping.

Good products rarely begin with technical know-how—they arise from deeply considered solutions to problems.

Reduce the Number of Steps

Thinking deeply isn't about increasing complexity. It's about breaking problems down to their most basic form.

Evan Williams, one of Twitter's co-founders, gives his secret to building a billion-dollar internet company: "Take a human desire, preferably one that has been around for a really long time. Identify that desire and use modern technology to take out steps."

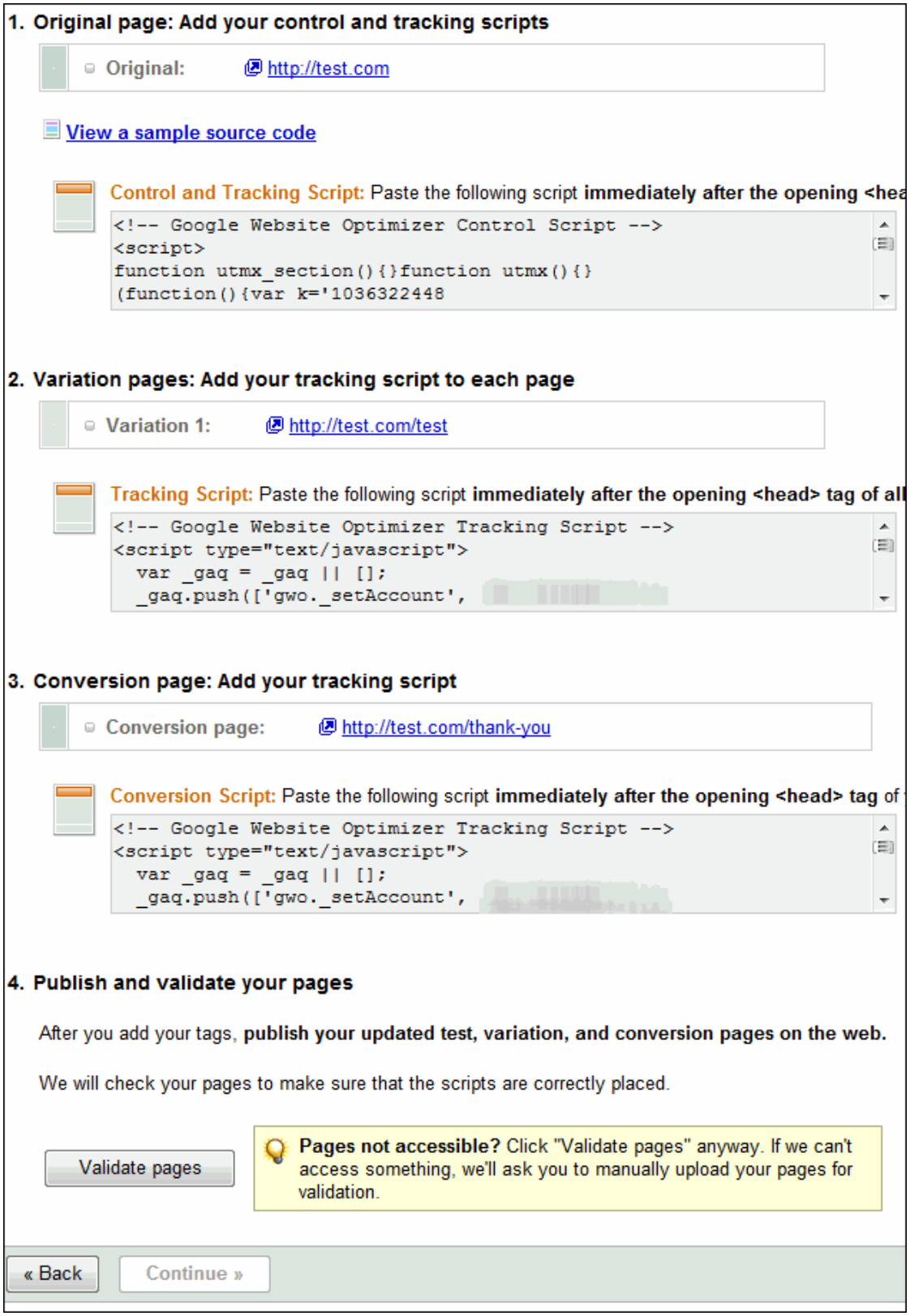
The desires that Williams describes aren't about summoning a car or a side of fries with a tap on your phone. They're more basic and immutable: getting from point A to point B, or just eating food. We want what we've always wanted. What technology allows us to do is get it faster, and without having to think about it.

The best ideas come from deeply thinking about how to make things easier for other people, and help them accomplish their goals with the minimum amount of effort. You might get there with tech, but that's also the part you should think about last.

The Origin of A/B Testing

One of the best examples of deep thinking comes from the evolution of the A/B testing space. Back in the early 2000s, it was pretty difficult to gain a quantitative understanding of how your marketing efforts were working out.

In 2006, the only real A/B testing tool available was Google Website Optimizer. It looked like this:


1. Original page: Add your control and tracking scripts
Original: http://test.com
View a sample source code
Control and Tracking Script: Paste the following script immediately after the opening <hea
<!-- Google Website Optimizer Control Script -->
<script>
function utmx_section(){}function utmx(){}
(function(){var k='1036322448
2. Variation pages: Add your tracking script to each page
Variation 1: http://test.com/test
Tracking Script: Paste the following script immediately after the opening <head> tag of all
<!-- Google Website Optimizer Tracking Script -->
<script type="text/javascript">
var _gaq = _gaq || [];
_gaq.push(['gwo._setAccount',
3. Conversion page: Add your tracking script
Conversion page: http://test.com/thank-you
Conversion Script: Paste the following script immediately after the opening <head> tag of
<!-- Google Website Optimizer Tracking Script -->
<script type="text/javascript">
var _gaq = _gaq || [];
_gaq.push(['gwo._setAccount',
4. Publish and validate your pages
After you add your tags, publish your updated test, variation, and conversion pages on the web.
We will check your pages to make sure that the scripts are correctly placed.
Validate pages
Pages not accessible? Click "Validate pages" anyway. If we can't access something, we'll ask you to manually upload your pages for validation.
« Back
Continue »

The process was painful. You had to take your page, think about all the different sections you wanted to A/B test, and put in script tags. For each test you wanted to run, you had to create a new page.

Here are the steps Google outlines for setting up an experiment:

Name experiment and identify pages:

- Step #1 Name your experiment. For example: Sign-up Form AB Test
- Step #2 Set your original test page URL: http://www.mysite.com/sign-up.html
- Step #3 Set your first variation test page URL: http://www.mysite.com/sign-up-b.html
- Step #4 Set your (optional) second variation test page URL: http://www.mysite.com/sign-up-c.html
- Step #5 Set your conversion page URL: http://www.mysite.com/thank-you.html

Install and validate JavaScript tags

Original test page:

- Step #6 Add the Control Script at the top of the page.
- Step #7 Add the Tracking Script at the bottom of the page

First Variation test page:

- Step #8 Add the Tracking Script at the bottom of the page

Second variation test page:

- Step #9 Add the Tracking Script at the bottom of the page

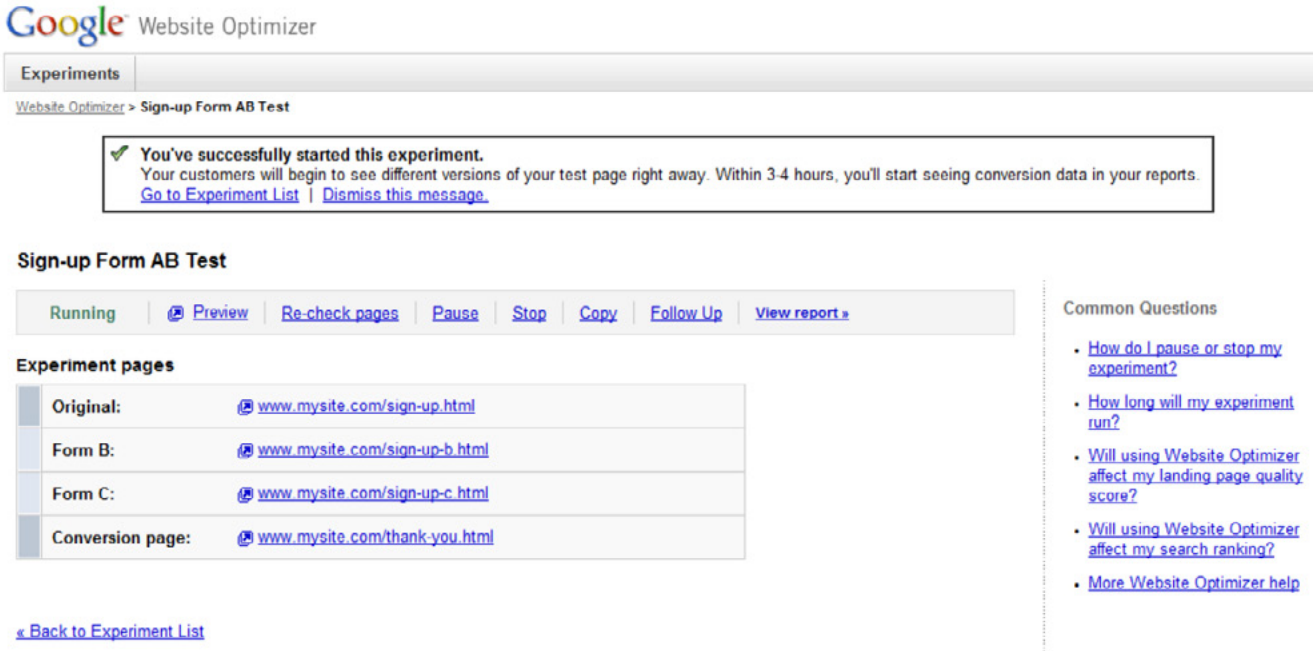
Conversion page:

- Step #10 Add the Conversion Script

A/B experiment set-up: Preview and start experiment

- Step #11 Review the data, test the pages using the 'preview' function, and then launch your test

It was an 11-step process for a static web page. If you wanted to add different variations, or were using a website with custom CSS, you often had to spend even more time poring through the code of your site. At each step of the way, because you were altering the code of every page on your website, there was a chance that you'd introduce an error and have to start over.



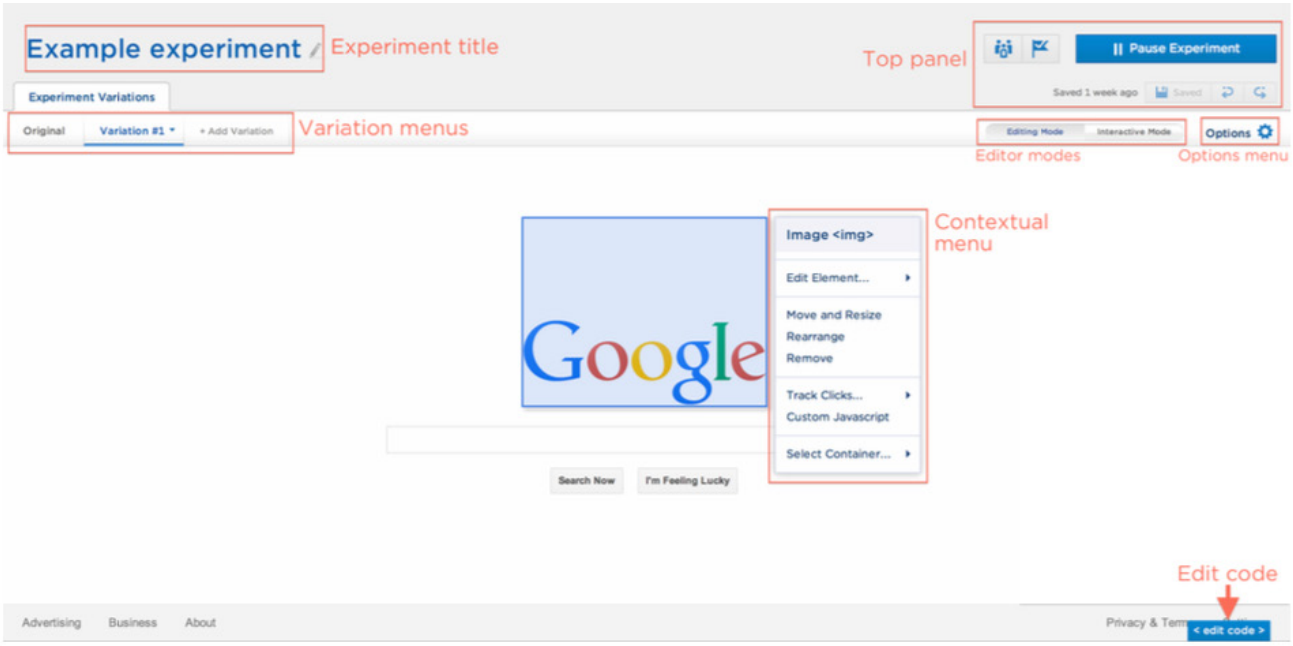
Sean Ellis, who was VP of Marketing at Logmein at the time, wrote:

> *One way I have worked around my engineering deficiencies has been to hire the skills onto the marketing team. For example, in my last long-term VP Marketing role I hired a front-end designer/engineer to design and code landing pages and a dedicated DBA to build reports and run ad hoc queries.*

Most marketers, especially at startups, simply didn't have the resources to accomplish this. Those like Sean Ellis, who were able to tap into engineering talent, could run more tests, learn faster, and grow.

How Optimizely Cornered 77% of the A/B Testing Market

In 2010, former Google product managers Dan Siroker and Pete Koonen launched Optimizely, which was their deeply thought-out solution to this problem. Optimizely couldn't launch better tests than Google, but it did provide a graphic interface to run and manage the tests.

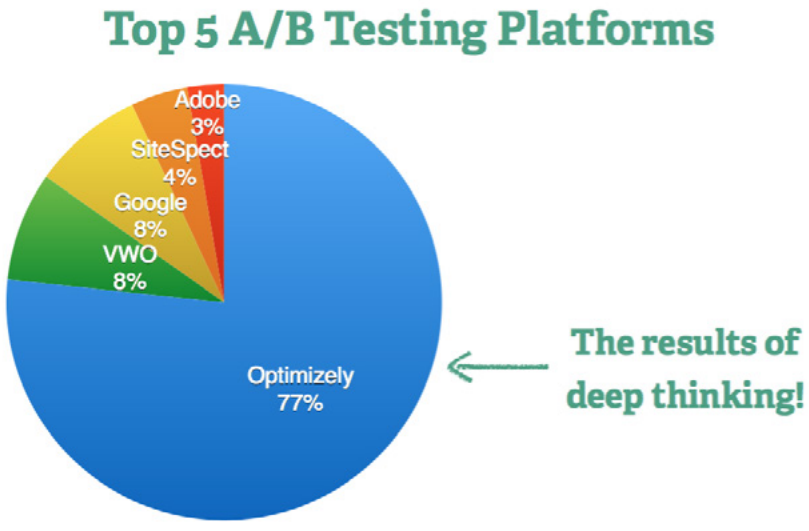


All marketers had to do was install a single line of javascript on their site. They could then click on anything they wanted to change on a WYSIWYG editor instead of staring blankly at lines of code.

The number of steps it took to A/B test—from turning to an engineer, pointing out the elements of your web page that you wanted to test, and having them insert script tags for each—was drastically cut down.

This basically cut down Google Website Optimizer's 11-step process into four short steps.

1. Insert Optimizely javascript snippet in the head tag of your site.
2. Open up the visual editor, and click on each section of the site you want to test.
3. For each variation, click "add variation."
4. Click start experiment to preview and launch test.



Optimizely thought deeply about the problem, and solved some of the hard technical problems behind building a WYSIWYG.

Marketers were suddenly able to run many types of experiments on their own, and all the A/B testing tools you see today solve the problem in the way that Optimizely did first. Six years later, Optimizely still has the lion's share of the market.

How to Build the Habit

When we hear about a problem, our instinct is to jump in with a bunch of different solutions. Far more often than not, these are scattershot fixes that don't get at the root causes of the problem. They do more harm than help.

Building the habit of thinking deeply starts with baby steps.

Thinking deeply about reducing steps isn't something that comes naturally to us—it takes practice, and time.

- **Ask the opposite.** When building product, the internal biases and assumptions you hold often obscure the real problem. When you come up with a hypothesis about a product—or really anything—ask yourself, "What if the opposite were true?" Force yourself to keep to this line of reasoning for an hour, and challenge each assumption you make.
- **Storyboard the steps.** Create a chart of the problem your problem's trying to solve, and write out each step—no matter how large or small—it currently takes for them to solve it. Look for high-friction areas where you can make things easier and more efficient.
- **Create a concept car.** Optimizely takes the idea of the "concept car" from auto production to describe prototypes of a futuristic looking thing, only with software. Instead of thinking about your limitations, think deeply about what the future could look like. This is another way of working backwards. It allows you to build from the end-goal out.

Take Time to Do Thorough Research Upfront

A lot of product managers limit research to a specific stage of building product—during customer development, or while they're piecing together an MVP. But those who build the most successful products don't just spend time thoroughly researching the problem before they write a single line of code—they continue to do so at every stage of the product lifecycle.

Don't just do research when you're building your product, or doing some customer development. Limiting research to one specific phase of product development will make it obsolete.

Putting in thought upfront means that you can accomplish more with less. You spend 10x the time thinking, and 1x the time doing.

The Apple iPod

Apple is a perfect example of a company that thinks deeply and obsessively about real-world problems. There's no dedicated R&D department—research is expected from each department, from design, to product to marketing. There isn't a separation between research and execution.

The iPod is one of the best examples of how deep research can create a wildly successful product. Portable MP3 players had been around for years, but Steve Jobs and the Apple team thought they were garbage.

What made the iPod so successful is the level of research and prototyping before launch. The iPod started with a pretty basic problem:

- MP3 players on the market didn't work well. Those with flash memory could only hold a CD worth of songs, while those with a hard drive were too big and too difficult to use. For each, it took too long to transfer songs.
- With the rise of Napster and MP3 downloads, there was a clear shift in the distribution of music, which none of the portable MP3 players available were meeting. iTunes was already being developed to include a CD burning feature.

Tony Fadell, who led Apple's iPod team, spent six weeks in stealth mode looking at the competition. The conclusion that he came to was that the iPod had to be something that could hold a lot of music, and a long battery life. He brought three prototypes of the iPod to Steve Jobs before product development began.

The iPod included these elements:

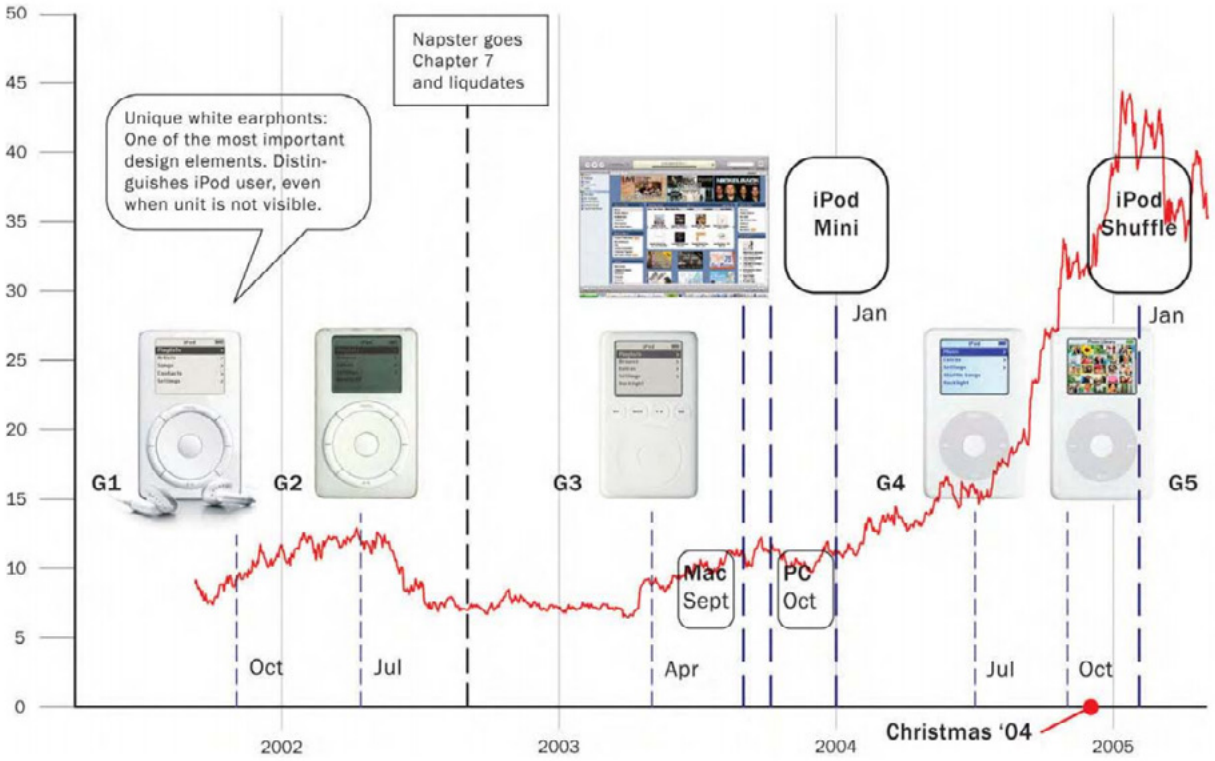
- a 5GB Toshiba hard drive that was just 1.8 inches wide
- a Firewire connector for faster data transfer
- a rotating wheel that would allow people to scroll through thousands of songs

None of these were groundbreaking innovations—they weren't even all things that Apple created internally. What made the iPod work was that each of these small elements involved deep thinking about how the iPod could be the best product on the market. As Johnny Ives said of the iPod, "Every single component, every process, has been considered and measured to make sure that it's truly useful and that it actually enhances the user's experience."

What the iPod did was look into a future where you could have thousands of songs in your pocket. Competing products forced you to click a button to scroll through each song, while the iPod's rotating wheel allowed people to easily browse through a massive library of music. It united what was capable with technology, with what customers actually wanted to do.

Each successive "iPod killer" launched by the competition, from Dell's DJ to the Creative's Zen, outpaced the iPod in terms of technical specifications. But none of them was executed with the same thought as Apple. What made the iPod were the details.

Distribution As Product



(source: Sketching User Experiences)

It wasn't just deeply thought-out design that made the iPod a hit, either—the bigger distribution strategy is what drove it home. Companies like Gillette sell razors at a loss and make money back on the blades. The iPod flipped this traditional profit model upside-down. Apple didn't make money by selling songs, but by selling the hardware.

In a 2006 interview with Businessweek, Clayton Christensen predicted the downfall of the iPod: "But once the technology matures and becomes good enough, industry standards emerge... At that point, the competitive advantage of the early leader dissipates, and the ability to make money migrates to whoever controls the performance-defining subsystem."

Christensen was almost right. What ended up killing the iPod wasn't a cheaper knockoff. It was Apple's iPhone. Where the iPod began with the idea that you could carry thousands of songs in your pocket, the iPhone crystalized the idea that you only want one thing in your pocket.

Cannibalizing your own product is a side effect of thinking deeply. This is something that companies like Apple, Google, and Facebook have always known. When you get complacent, someone else comes along and builds the next thing better.

(source: Ben Barry)

Instead of trying to protect your own turf, focus on building products that think deeply around the problem and the market, rather than just how well they sell.

How to Build the Habit

You don't need billions of dollars to research well. You need to build a habit of actually *doing* research at every point in product development, from building a new feature, to after you launch your product into the market.

Your product doesn't exist in a vacuum. Research is how you connect what you're building with the rest of the world.

- **Eat the dog food.** When Facebook launched on Android, the experience was awful. Most developers preferred to use iOS, and the resulting Android experience was so bad that chief product officer Chris Cox forced parts of his team to switch to Android phones. Get your product team to live through the problem firsthand.
- **Start from an empty state.** Part of the reason why the Apple team chose the name "iPod" was because it didn't describe what the iPod actually did. The iPod itself was an empty state that wasn't limited to one function, but could constantly evolve over time.
- **Help the competition.** In 2014, Elon Musk open-sourced all of Tesla's patents. He wrote on the company blog: "We believe that Tesla, other companies making electric cars, and the world would all benefit from a common, rapidly-evolving technology platform." Look at what the competition is doing constantly, and learn from them.

How to Compete

The market is a graveyard of failed products. Yesterday's wonders quickly turn dull. Most product people's answer to this is to build more, and at speed—and this is how they often fail.

The law of product development is that products grow obsolete quickly. If you stay complacent, someone else will come along and solve the problem better than you. The upshot of this is that there is always opportunity, even in a crowded space, to solve the problem better.

Thinking deeply doesn't mean moving slowly, it means moving deliberately. It means having a clear direction for your product, and knowing how to implement it. In today's crowded space, those who are able to build a habit of thinking deeply have a huge advantage.

Chapter 4
Use Data

In 2005, Steve Jobs asked Intel CEO Paul Otellini to build a new processor for the first iPhone. Otellini said no, and Intel was caught pants down as smartphones ate the world.

Otellini owned up to his mistake after the fact:

> *"I couldn't see it. It wasn't one of these things you can make up on volume. And in hindsight, the forecasted cost was wrong and the volume was 100x what anyone thought.... The lesson I took away from that was, while we like to speak with data around here, so many times in my career I've ended up making decisions with my gut, and I should have followed my gut. My gut told me to say yes."*

The problem was that Intel used data to back up what it *already* knew, and not what it could find out. The companies behind the best products of today understand that in order to survive, you have to reinvent yourself constantly. This means leveraging experience and collected data to drive a vision of the unknown.

Otellini's decision was completely logical and also completely wrong. Moving to a cheaper ARM chipset ran counter to 50 years of how Intel has pulled in profit—each year manufacturing a high volume of premium microprocessors, and selling them at wide margins.

Year after year, Intel has thrown billions looking at the same data for "forecasted" volume and costs. Instead, they should have been looking at how to use data to reinvent themselves.

Product development today is filled with a demand for "more data." But what's far more important than what you have at your fingertips are all the things that you can't know. Without context and a larger vision for the future, data means nothing. It fills out spreadsheets on a piece of paper and a couple of pretty graphs. You might as well build product by tracking the way sticks fall onto the ground.

This chapter of the ebook is not about how you accumulate data. It's about how you use data.

Allow Data to Guide Product Vision

When you're building a company or creating a new product from scratch, the data available to you will often tell you that what you're doing is completely illogical. That's because creating something is an act of instinct on a very primal level.

Later on in the product lifecycle, you can use more data to leverage tactical, short-term optimizations—increasing the number of sign-ups, or driving up your active user-base. As you're starting out, though, you have to carve out your long-term product vision. Use the limited data you have as a starting point to launch yourself into the things you don't know.

> *"Being data-informed means that you acknowledge the fact that you only have a small subset of the information that you need to build a successful product.""*
>
> *—Andrew Chen*

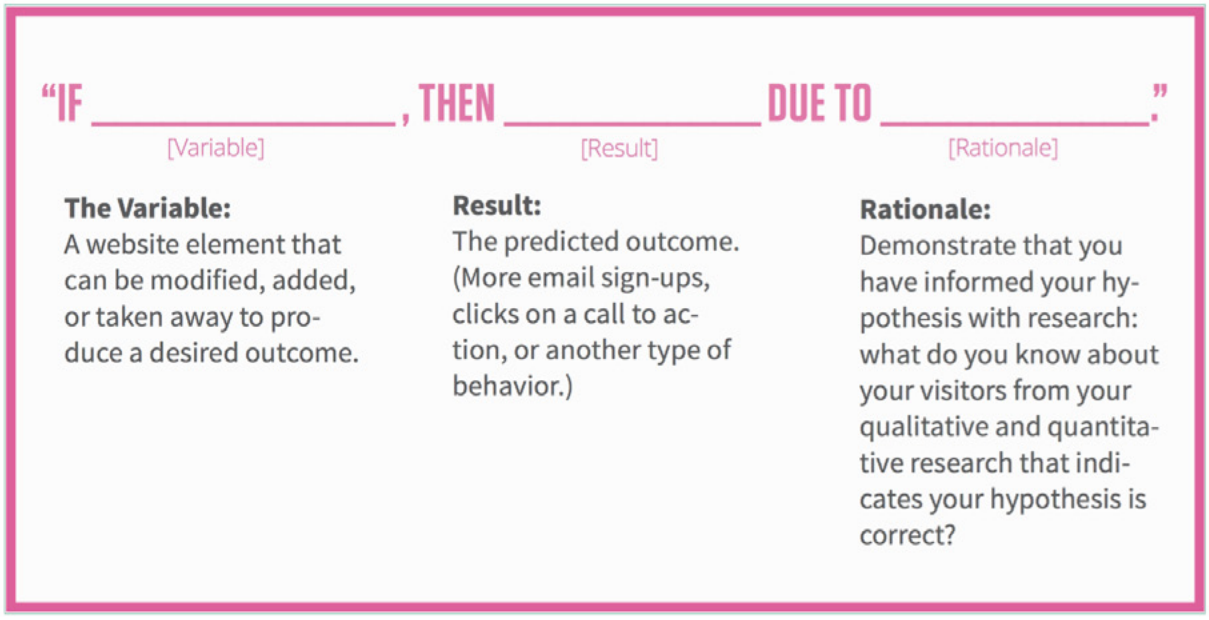
Being data-driven means that you're trying to drive up data points (DAUs, MAUs, etc.) simply for the sake of it. Being data-informed means you're trying to build your product vision, working backwards from the problem or the job-to-be-done.

Companies that have been around the block—like Intel—tend to be data-driven. They know what works, and use data to optimize for the local maximum. Growth is linear, and maxes out at 10%. Being data-informed allows you to set your sights higher. Being data-informed and willing to take calculated bets is how you set your sights on the global maximum, and take over an entire industry.

Data doesn't tell you what decisions to make. Data informs *your* decision making. You'll never have the complete picture when you're building product and that's a good thing—this is where the **opportunity** is. If all the data is out there for you to find, it means that someone else probably beat you to the punch.

How We Built KISSInsights with a Data-Informed Hypothesis

Letting data inform your product starts with an educated guess, or hypothesis, keeping in mind that simpler hypotheses often make for better products. This is how KISSinsights (now Qualaroo) began.



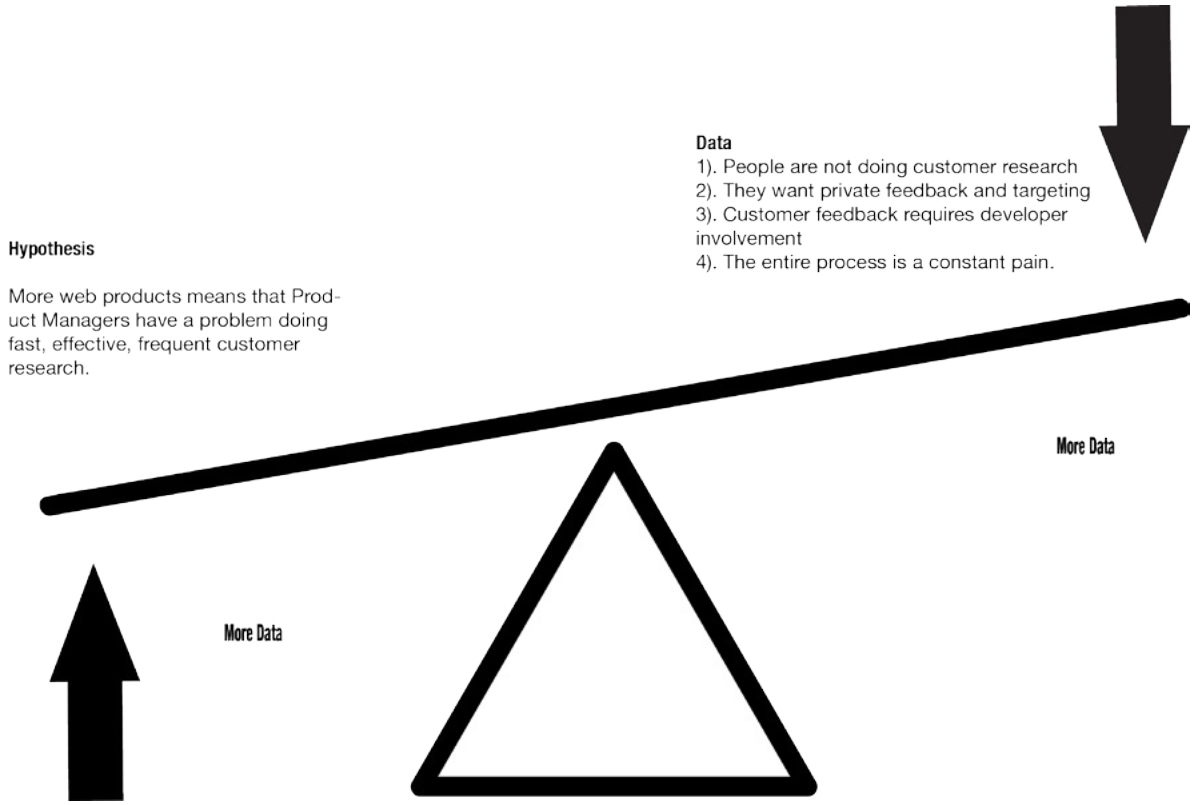
(source: Optimizely)

When we were conducting customer development interviews for KISSmetrics, our analytics product, we heard product people and marketers saying one thing over and over: “It’s too hard to figure out what people are thinking when they visit my website.”

This problem had nothing to do with KISSmetrics and the questions we were asking, but it popped up frequently enough to be interesting. Cindy Alvarez, our team lead for research, says, “one person might be a nutcase. Ten people are not all nutcases.”

Our hypothesis for KISSinsights was informed by this qualitative data. Product managers have a problem doing fast, effective, and frequent customer research. There were two parts to this that we had to figure out:

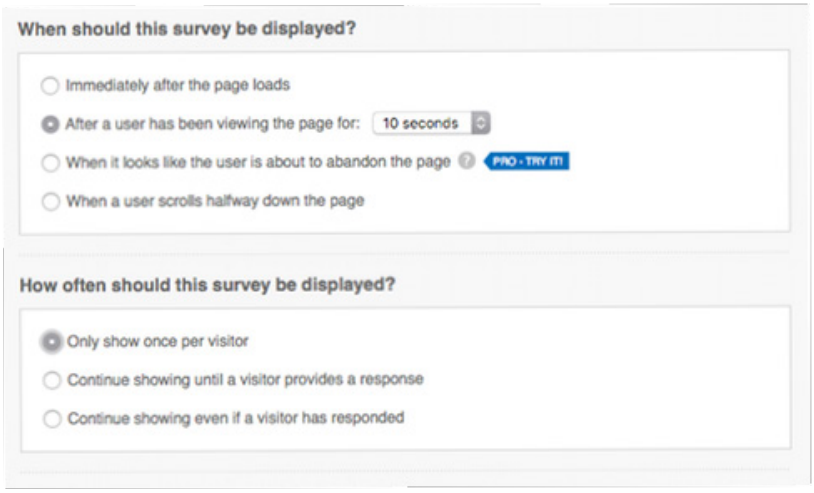
1. Is this a recognizable, interesting pain?
2. Where exactly is the pain?

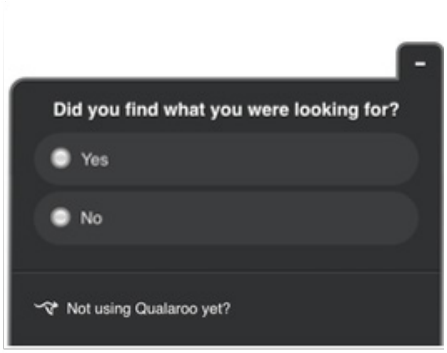


The survey tools available to product people at the time forced users into a separate page, and interrupted whatever they were doing on your site. This meant that they rarely filled out surveys, and it was incredibly painful to get user feedback.

We had our hypothesis for the product. If you can send a user a highly specific question that's relevant to what they're doing on your website, then they'll actually answer it because that it poses minimal interruption to the user workflow.

The resulting product sent a pop-up survey to users who had been on a specific webpage of a site for a specific period of time, and took 5 minutes tops to set up. Instead of the 1% to 2% response rate typical to existing survey tools, KISSinsights netted customers anywhere from a 10% to 40% response rate.





20 customer development interviews provided us with enough qualitative data to build and launch an MVP of KISSinsights. We didn't have enough data to know it would be successful, only enough to inform a hunch—but it was all we needed. Acting on limited data trumps getting more data.

Within a year, KISSinsights was acquired by Sean Ellis. Today, it's called Qualaroo.

How to Build the Habit

In order to build better products quickly, look to how you can use data more effectively. This doesn't necessarily mean having *more* data—it means deploying resources in the most efficient and effective way.

Ultimately, relying on your gut isn't separate from data at all. Building product based on instinct goes back and taps into the experience that you've built up, both individually and as an organization.

- **Summarize findings into a single sentence.** This forces you to look away from the numbers, and into what they actually mean for your product. At Yammer, the user research team gives updates during monthly meetings. They're pushed to summarize information in a short sentence—"Here's what we learned from you and here's what we've done as a result."
- **Experiment constantly and open up your inputs.** Don't rely on data from a single source to determine the success or failure of your target goal. By using data from a variety of sources, from A/B testing to usability testing to customer development, you can stress-test your hypothesis against your own biases.
- **Use data across teams.** In order to build a data-informed culture, you need to invest time in educating your team. At Sailthru, chief data scientist Jeremy Stanley spent 30% of his time teaching a class for engineers on statistical learning.

ONE Metric Per Product Team

When you're working on product as part of a small, closely-knit team, there are always a million things you could be doing better. But your advantage as a startup is the ability to move quickly to validate your hypotheses with results. One of the most powerful ways to accomplish this is simply by limiting the amount of data your team looks at.

> *"All product initiatives need a single metric and a target goal."*
>
> *— Steve Cox, former VP of Product KISSmetrics*

The idea here breaks down to something pretty basic: As a team, your success is rooted in everyone knowing why they're working on what they're working on, and what they're trying to accomplish.

Any time you add a feature or try to improve your product, you need to be able to know what success would look like.

Let's say you're building in-app notifications for your B2B SaaS product. Maybe you want to cut churn. More users or less churn would make your boss happy, but if you can't justify the feature for any better reason, then the motivation and focus of your team are completely out of sync. Success in this context means trying to reduce churn to make your boss happy.

A better way to approach this is to say, we want our 90-day retention rate to go from 28% to 40%. We think by building a notifications feature, we'll be able to hit this target, because we noticed that users who turn their email notifications on during Week 1 are 40% more likely to hit the 90-day mark.

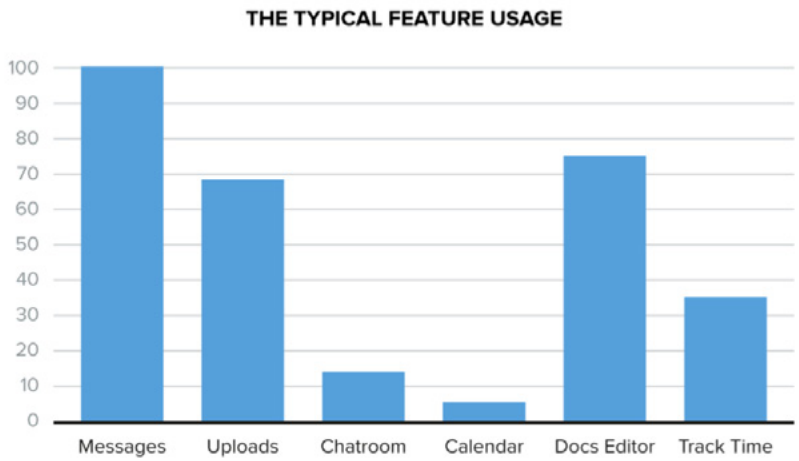
Uniting everyone on product behind a single metric is critical for creating a stickier product. If you've thought deeply about the problem, and justified your hypothesis from the jobs-to-be-done perspective, you can align your team towards working at a singular goal, with a clear reason why.

This idea applies to everything, from optimization projects like increasing landing page sign-ups, to building an entirely new functionality for your product.

Data doesn't make decisions for you. Data informs *your* decision making.

What to Look for

One of the best ways to use data to find your North Star metric is to look at the usage data you already have on hand. When you've already got usage, analyze it.



(source: Intercom on Product Management)

Map out product usage by feature to see what levers you can find to improve. Your core metric will be different depending on where you are in the product lifecycle:

- **Beginner:** In the early stages of your product, focusing on upping your total active user base is perfectly acceptable.
- **Intermediate:** As you hone your approach to data, look for more specific levers of growth. For an analytics product, this would be something like getting users to create 3 reports.
- **Advanced:** When you get more at home with your analytics data, you can start using a secondary metric to supplement your core metric. As you have multiple product teams, you can have each of them devoted to upping other specific core metrics.

Here are some examples of single metrics with target goals:

- **Twitter:** Twitter very early on knew that its ability to generate revenue was tied to the amount of views that Twitter feeds received. They used number of feed views to help predict potential revenue.
- **Remind:** Remind, a communication tool for teachers, focused on getting new users to refer other teachers to their product at the optimal moment. Internal data showed that momentum in growth was tied to when the first teacher at a school used Remind, so the product team focused its efforts on getting these users to refer.
- **Zynga:** Zynga, an online game platform, looked specifically at Day 1 retention. If a user came back the day after they had downloaded a game, they were more likely to stick around longer. They were also more likely to spend money in the game.
- **Dropbox:** For Dropbox, the key action for engagement was getting users to place at least one file in a Dropbox folder.
- **Slack:** Slack focused on getting companies to hit a milestone of 2,000 messages sent, which corresponded to a 93% retention rate. Their reasoning was simply that that's how long it takes for an organization to really use the product and realize its value.

As Slack CEO Stewart Butterfield says, for each metric used to build product "you have to figure out what conversion means in your case. What does retention mean? What does activation mean? For every business, it's going to be slightly different because of the nature of the product and the kinds of people who use it."

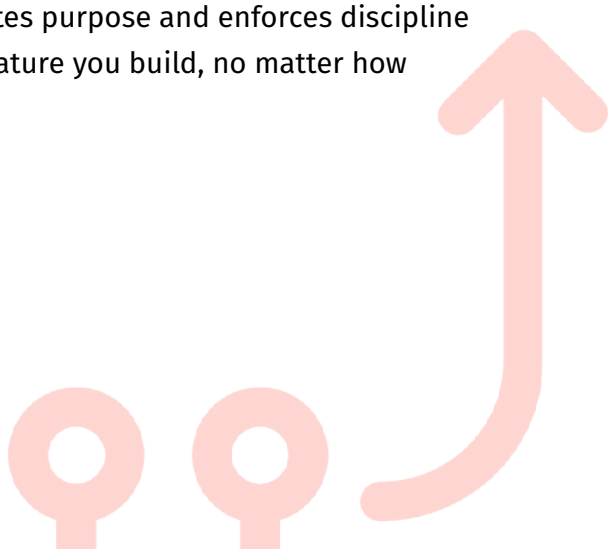
Different products will have different core metrics. Data only becomes useful for product development when you can figure out how it fits into the bigger picture.

- **Set an expiration date.** At Y Combinator, startups are expected to hit 8% growth a week on their core metric. Creating a timeline for product metrics is the only way you can build up an organizational rhythm that uses data to drive results.
- **Tape your core metric to the heads of your entire team.** As Aditya Vempatay of analytics platform Amplitude says, "It's impossible to come up with individual tactical goals that contribute to the overarching core metric if data is siloed and available only to the growth team or the data scientists." Your overarching core metric should be in a dashboard where everyone can see it, from Product to Sales.
- **Drown out the noise.** Looking at Facebook's metrics and how it hit viral growth 10 years ago is fine, but doesn't relate to what you're building today. In order to actually build a product people care about, you need to collect your own data within the context of your product.

How to Build the Habit

But in order to build this into a habit, you have to start by making sure everyone's on the same page.

Ultimately, uniting your team around a single metric creates purpose and enforces discipline for every feature you build, no matter how small.



Moving Beyond Blind Data

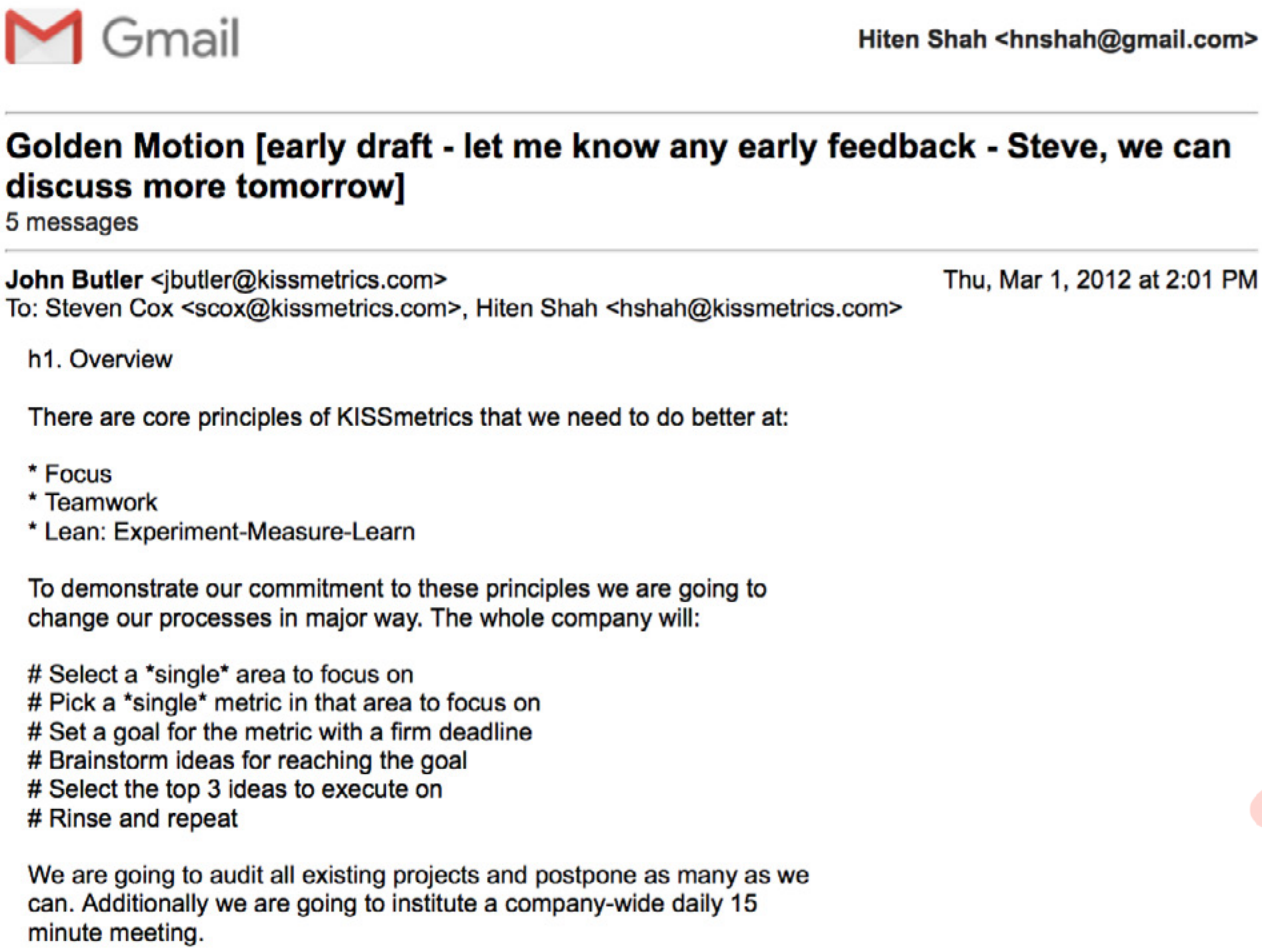
In product development, data is often used as a crutch to play things safe. When you want to justify the decisions you've made on product, paying lip-service to using data provides you with the easy answers.

Good product managers do the exact opposite. They use data to constantly challenge their assumptions, to validate hypotheses, and to solve urgent problems. At the end of the day, using data to build product is incredibly simple. It shows you how well you're doing, and where the opportunity is to do even better.

Chapter 5
Focus

"A company can only do one thing at one time" —Arjun Sethi, Founder & Troublemaker

In 2012, KISSmetrics was stumbling. We were having a hard time getting things done, wasting time and burning through cash. There were a lot of reasons for this—our team was remotely distributed, bad documentation, any number of things—but the root cause was a lack of focus.



Gmail

Hiten Shah <hnshah@gmail.com>

Golden Motion [early draft - let me know any early feedback - Steve, we can discuss more tomorrow]
5 messages

John Butler <jbutler@kissmetrics.com> Thu, Mar 1, 2012 at 2:01 PM
To: Steven Cox <scox@kissmetrics.com>, Hiten Shah <hshah@kissmetrics.com>

h1. Overview

There are core principles of KISSmetrics that we need to do better at:

* Focus
* Teamwork
* Lean: Experiment-Measure-Learn

To demonstrate our commitment to these principles we are going to change our processes in major way. The whole company will:

Select a *single* area to focus on
Pick a *single* metric in that area to focus on
Set a goal for the metric with a firm deadline
Brainstorm ideas for reaching the goal
Select the top 3 ideas to execute on
Rinse and repeat

We are going to audit all existing projects and postpone as many as we can. Additionally we are going to institute a company-wide daily 15 minute meeting.

Whether you're self-funded or working with venture dollars, you have a limited amount of money and time to build traction for your product. The only advantage that a startup has over larger companies is the ability to move quickly, unhampered by the governing processes that rule larger organizations.

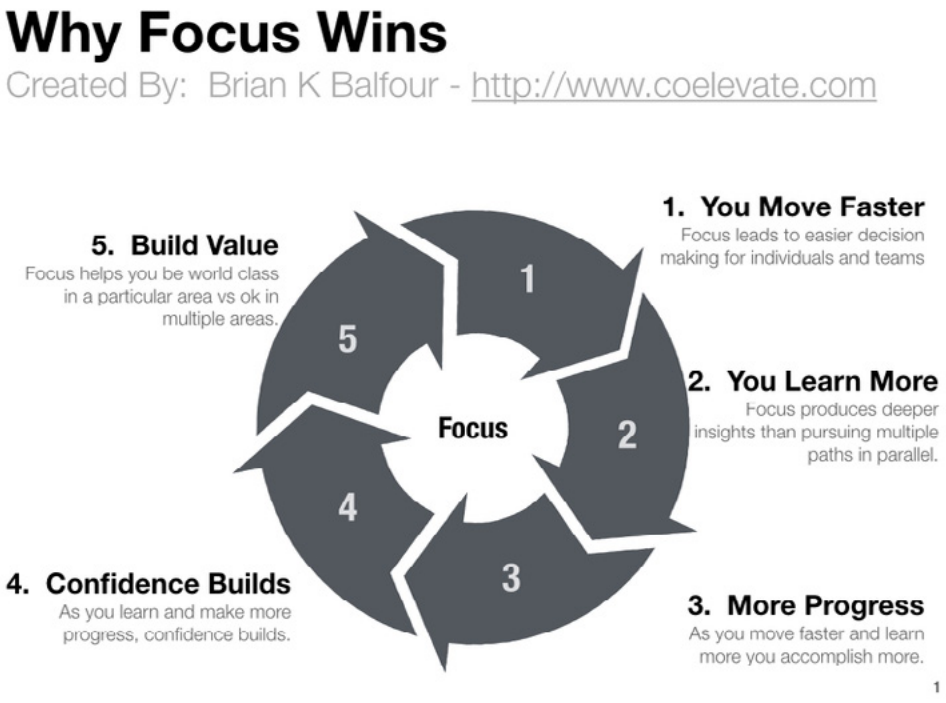
When you're developing your product, there will always be a million different things to build and fix. You have bug reports and feature requests pouring in from your customers, while the stakeholders of your company might be asking for something in a completely different direction.

What ultimately got Kissmetrics back on track was figuring out how to focus on building the right things at the right time.

Focus on One Thing at a Time

In order to successfully build a product, you need to constantly move the needle. Large companies build SWAT teams that are made for this purpose. As a startup, you are your own SWAT team. Your competitive advantage comes from your ability to attack one problem at a time.

A focus in developing products doesn't mean exchanging long-term product vision for short-term thinking. It means setting meaningful goals that allow you to track your success, adjust, and iterate.



(source: Coelevate)

Startups operate under conditions of extreme uncertainty, and it's tempting to try a lot of different things rather than put all your eggs in one basket. Trying too many things at the same time, however, means that you'll do none of them well.

By focusing on one goal and one metric at a time, you can constantly adjust to circumstances and change course, while learning quickly enough to grow.

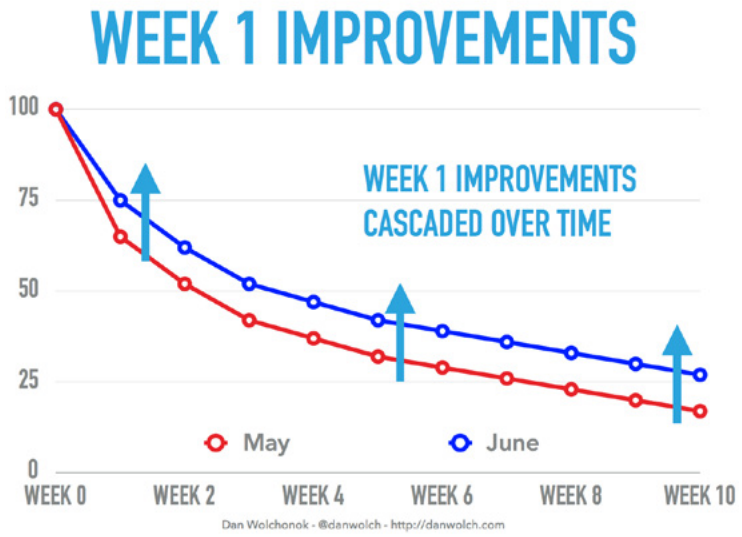
Former HubSpot VP of Growth, Brian Balfour, gives a 4-step process for building focus:

1. **Identify one long-term meaningful goal:** This might mean boosting the single metric we discussed in the last chapter. The alternative to focusing on one goal that matters is to spread yourself thin on short-term optimizations in order to hedge your bets.
2. **Distill the most important thing to make progress toward that goal.** Say that your goal is to increase inside sales revenue by 40%. Using data has shown you that users integrating other services increases free trial conversions by 3x. You could then focus on getting these trial users on a call with a sales rep.
3. **Create a timeline for making progress long enough to gather data.** All product initiatives need to have a clear goal, a measuring stick for what success looks like, and enough time to measure. For small teams, this should range from 30-60 days. Sticking to a timeline ensures that you don't sink too many resources into a product goal that you can't achieve. It allows you to move on and focus on the next thing.
4. **Editing your longer-term goal according to data.** The fourth step is why it's so important to set a single measurable goal in the first place. It's what allows you to figure out what you're doing right and wrong, and improve. Making mistakes is forgivable and inevitable in product, but failing to learn from them is wasteful.

Focus in Action: HubSpot's SideKick

HubSpot's email product, Sidekick, was facing a massive churn problem. Retention was growing worse by the day. Most users would take a single action in-app before leaving for good. The users who did stick around dropped off after only a couple weeks in.

Your customer's first impression of your product lasts over the course of the entire customer journey. As Appcues founder, Jackson Noel writes, "Just a small improvement right from the start has drastic effects further down the line. By increasing retention by a couple of percentage points, you can double your revenue only a few weeks later—and the easiest way to get there is through better user onboarding."

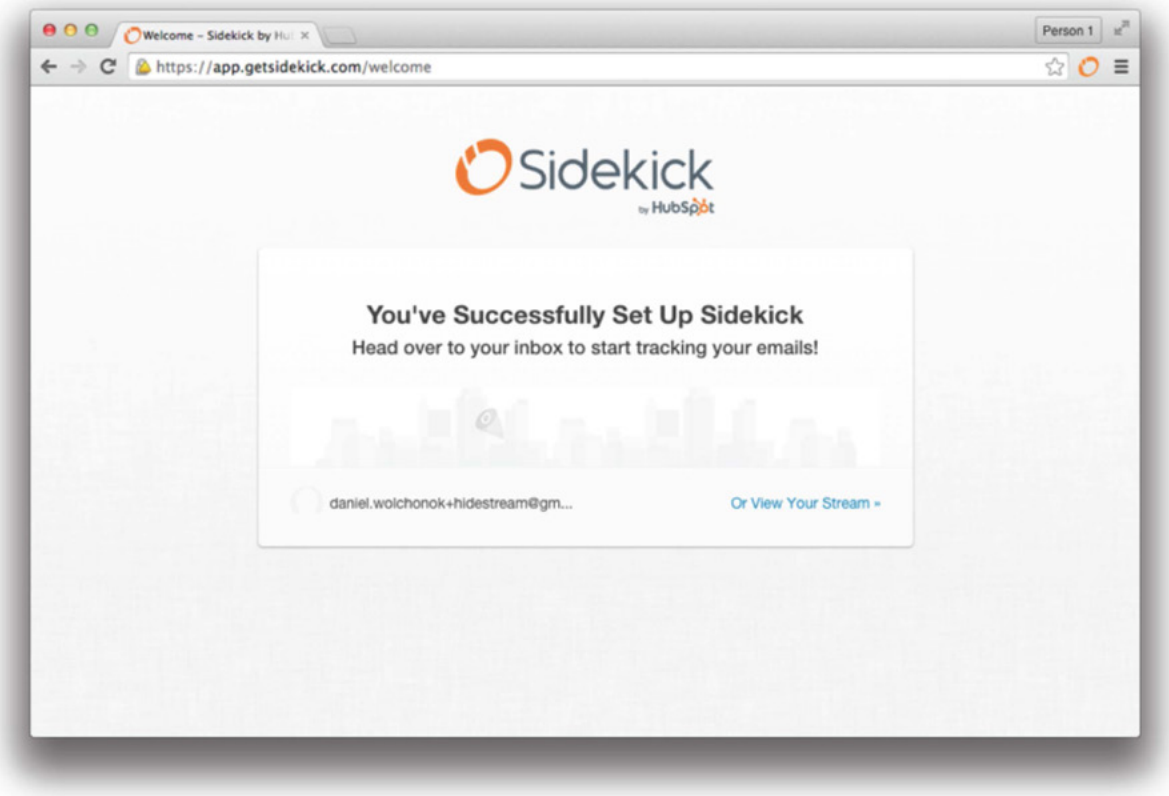


Dan Wolchonok, a product manager on HubSpot's growth team, found that a 15% increase in Week 1 retention had a cascading effect across the board. It meant a 15% lift to the rest of the retention curve. If they could improve onboarding and get Week 1 users to stick around, they could take the first step in addressing Sidekick's churn problem.

The key here is that HubSpot focused on one thing. Rather than trying to fix retention as a whole, they broke the problem down and started with the single thing they could do with maximum impact—improving Week 1 onboarding.

They tried:
- **adding an explanation to the empty login screen: failed**
- **inserting sample data into the onboarding screen: failed**
- **onboarding via video: failed**

It took 11 different hypotheses and failed experiments before they found a winning formula. But by maintaining focus, HubSpot was able to slowly work its way to a solution. Instead of trying to teach new users about how to use Sidekick, they focused instead on getting them to send more emails:



Dan Wolchonok - @danwolch - http://danwolch.com

Sending emails is the core value of Sidekick's product, which offers open-tracking and email templates. By driving users toward the stickiest features of the product during onboarding, they built up behaviors and patterns that led to long-term retention. Only then did the team look to boost retention through smaller optimizations, like reengaging with inactive users.

This is why focus wins.

How to Build the Habit

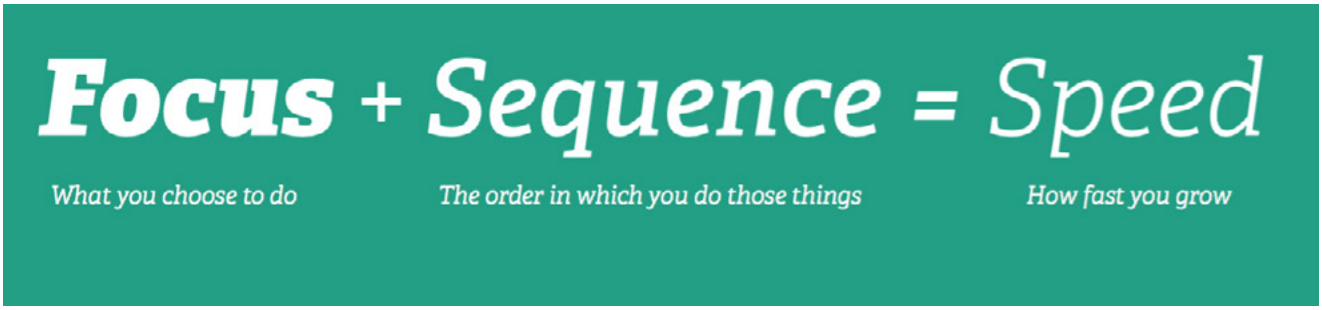
Having focus means that you don't optimize for the quick wins and immediate return on investment. More often than not, it means finding the right things to focus on.

- **Press-gang a DRI (Directly Responsible Individual).** When you're working with a cross-functional team of engineers and designers, lack of focus often develops from a lack of clear accountability. For your one target goal, hold one person on the product team directly responsible. As Gloria Lin of Flipboard says, "you don't have fifteen people worrying about the same things."
- **Look inside the crystal ball.** VP of Product at Facebook, Julie Zhou, recommends asking yourself, "If I could know anything in the world about how people are using my product, what would I want to know in order to tell me whether or not I was successful?" Work backwards from the customer to find something you can use as an approximate measure of success.
- **Minimize interruptions.** Make sure you and your team are ruthlessly protective of your own time. Everything, from Slack notifications to being tapped on the shoulder, can strip away focus. Heroku found that its top-performing engineers reported being interrupted 38% of the time compared to 76% of the time for bottom performers.

Focus + Sequence = Speed

As a startup, speed is your advantage. But if you don't ruthlessly prioritize, you'll end up spinning your wheels and burning through your cash.

In over 12 years of building products, I've learned one simple formula for moving rapidly:



The things that you decide to build and the order you build them in determines how fast you move on the opportunity. The combination of focus and sequence is how you focus on building products without losing sight of the bigger picture. If you don't focus, you don't build something good. If you don't obsessively think about sequence, you risk building the wrong thing at the wrong time.

Things change, quickly. The assumptions you make when you decide to build in-app messaging into your product might be nuked three months later when the competition gives that feature away for free. Every feature, bug fix, and improvement to product is a drain on money and time.

If you can focus on building product and sequence according to your larger product vision, you're able to move more quickly, try more things, and minimize the risk of getting derailed from your larger goals.

Freebie: You can take your pick from Basecamp, Trello, or any number of product management tools—my favorite option is still this Google Sheet. The most important thing to do with this kind of tool is to make one person (typically the product manager) the only person with edit access.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Project Name**	**Status**	**Owner**	**Q1'15**												
2	Jan 2015				Feb 2015				Mar 2015					Apr 2015		
3	1/5/2016	1/12/2016	1/19/2016	1/26/2016	2/2/2016	2/9/2016	2/16/2016	2/23/2016	3/2/2016	3/9/2016	3/16/2016	3/23/2016	3/30/2016	4/6/2016	4/13/2016	4/20/2016
4	Website															
5	Google auth	Started	Team A	x	x	x	x									
6	Design splash screen	Complete	Person B	x	x											
7	Invites															
8	Optimize invite messages	Complete	Team A	x												
9	Product															
10	Code up weekly digest	Blocker	Team B		x	x	x									
11	Brand & Styleguide															
12	Logo, typography, color guides, naming conventions	Started	Person A	x	x											
13	UI	Will Not Deliver	Person A			x	x	x								
14	Tone and voice	Planned	Person A					x	x							

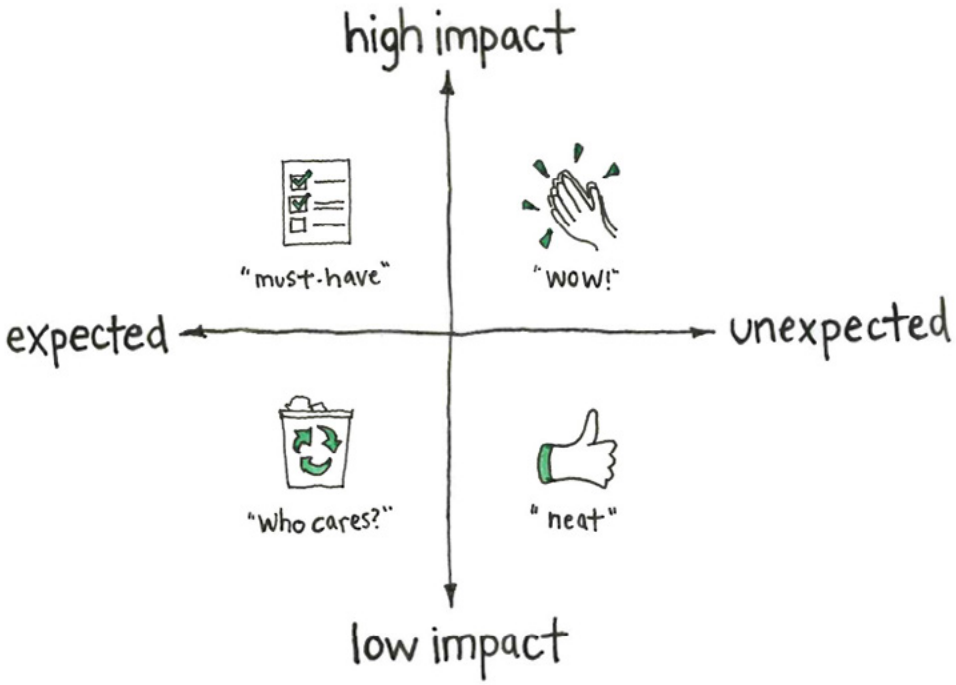
If the CEO wants to get Person A to shift from creating a logo to UI, she has to consult with the product manager first. It's a way to establish a system of checks and balances to maintain focus and proper sequence.

What to Focus on, WHEN

With your product, you always want to exceed customer expectations in a way that will significantly improve their lives—that's what draws them in and keeps them coming back. But it's just as important to make sure that the basic features are working and in place.

There are a lot of ways you can prioritize your feature roadmap. What's crucial is to have a system in place.

Tomer London, co-founder of Gusto, recommends measuring impact and the customer experience (Y-axis) against customer expectations (X-axis):



(source: Gusto)

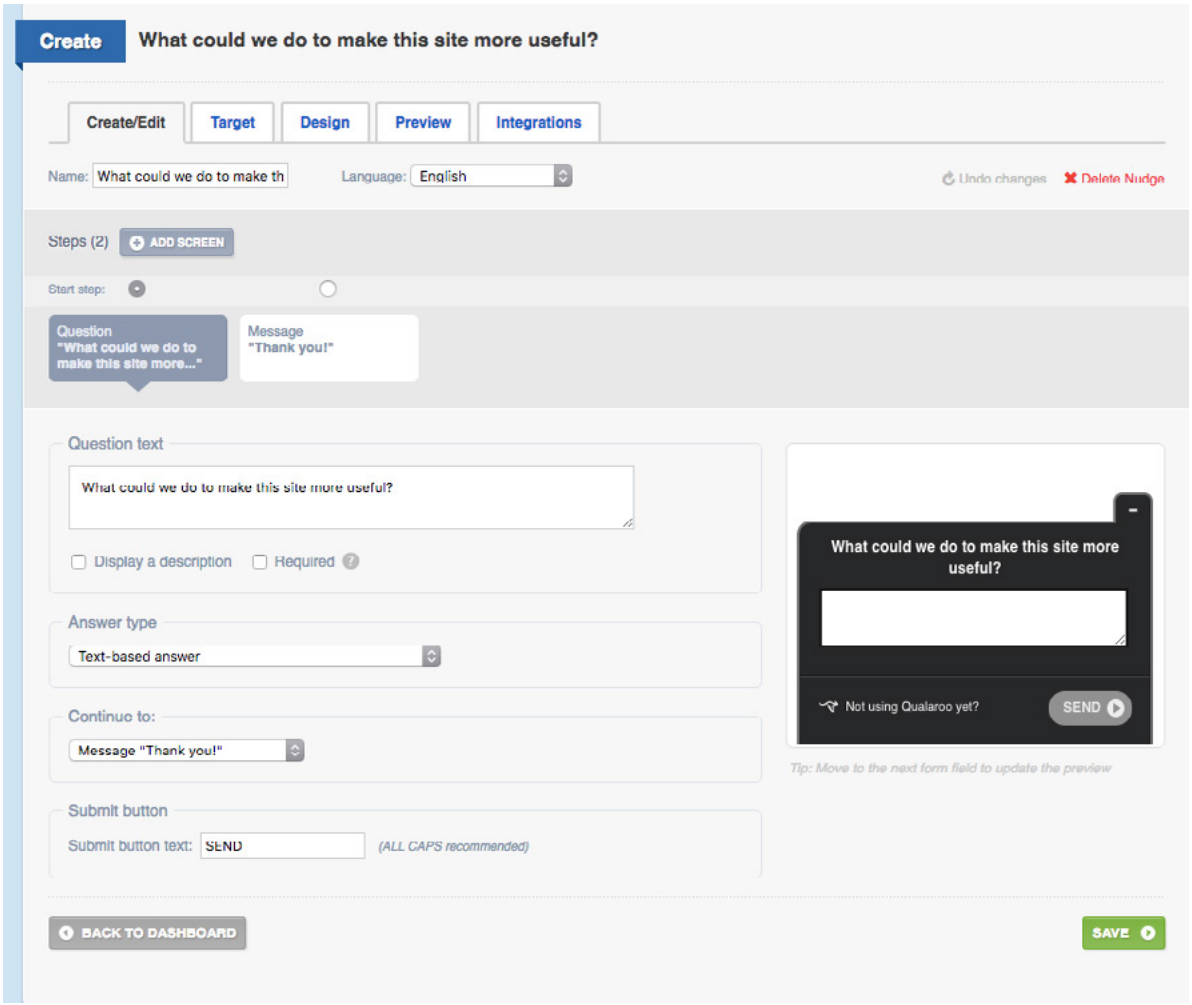
The way that you weight product initiatives against these axes will depend on your company and your product. But a good general rule of thumb is that a small improvement to a high-impact feature is far more important than a large improvement to a low-impact feature.

It goes back to focus. Focus on the one thing that you can do that will have the highest impact.

1. Focus on the high-impact unexpected features to "Wow!" customers

If you're in the early stages of product development, you need to focus on building "Wow!" features that will have the highest impact, and keep your customers coming back for more. Wow! features are what differentiate your product from all the other ones out there.

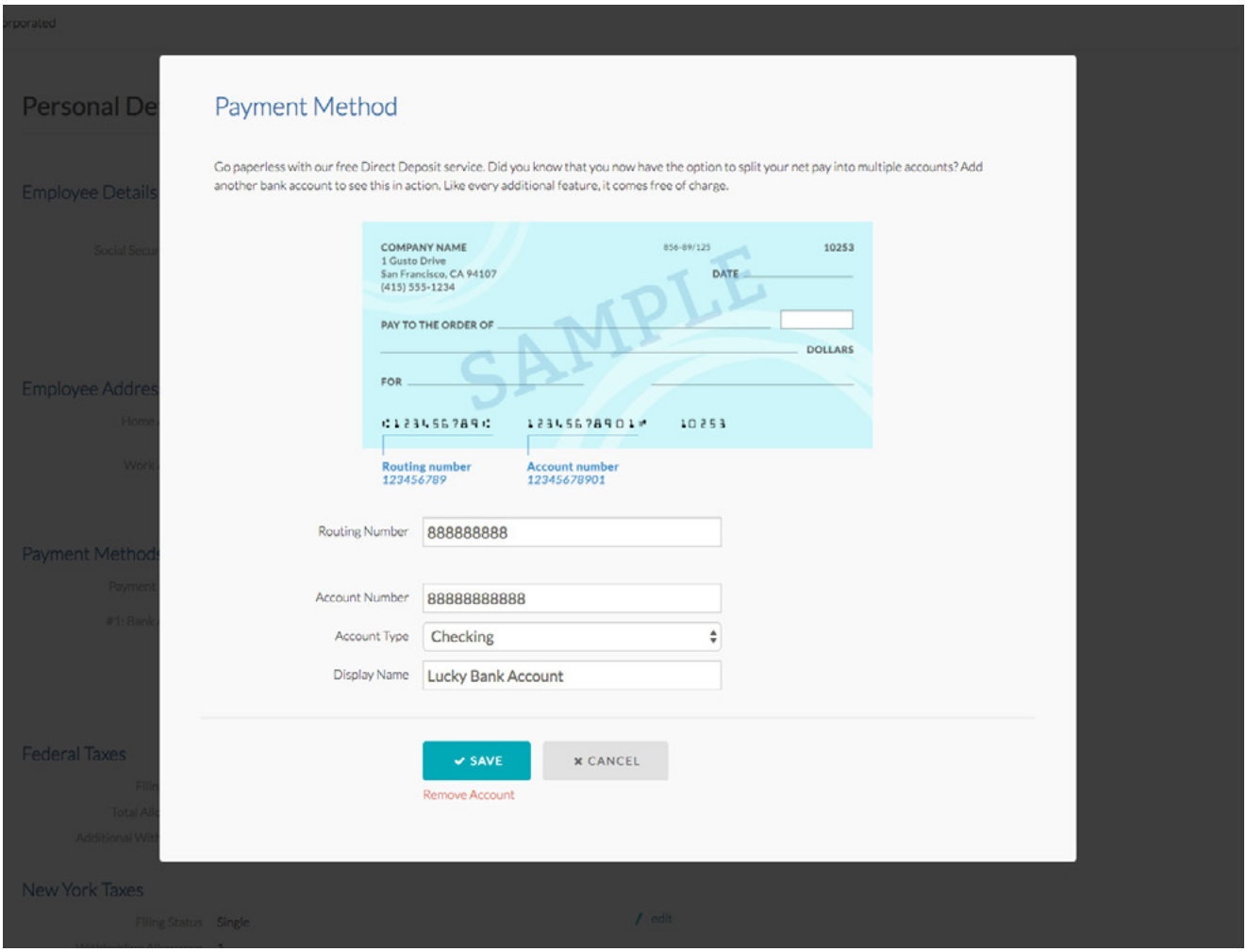
For KISSinsights (now Qualaroo), this was giving product managers and marketers the ability to send timed on-site surveys for feedback, at a time when everyone needed more customer feedback to learn and grow. Other products forced users to go to a separate survey landing page, which meant that surveys were interruptive to the experience and were rarely filled out.



2. Focus on “must-have” expected features next

Your “must-have” feature set is often what cements product-market fit after customers have been wowed. These are the features that customers simply expect to have and that most of the competition offers. For cloud-based storage, like Dropbox or Google Drive, the ability to work offline is a good example of a must-have feature.

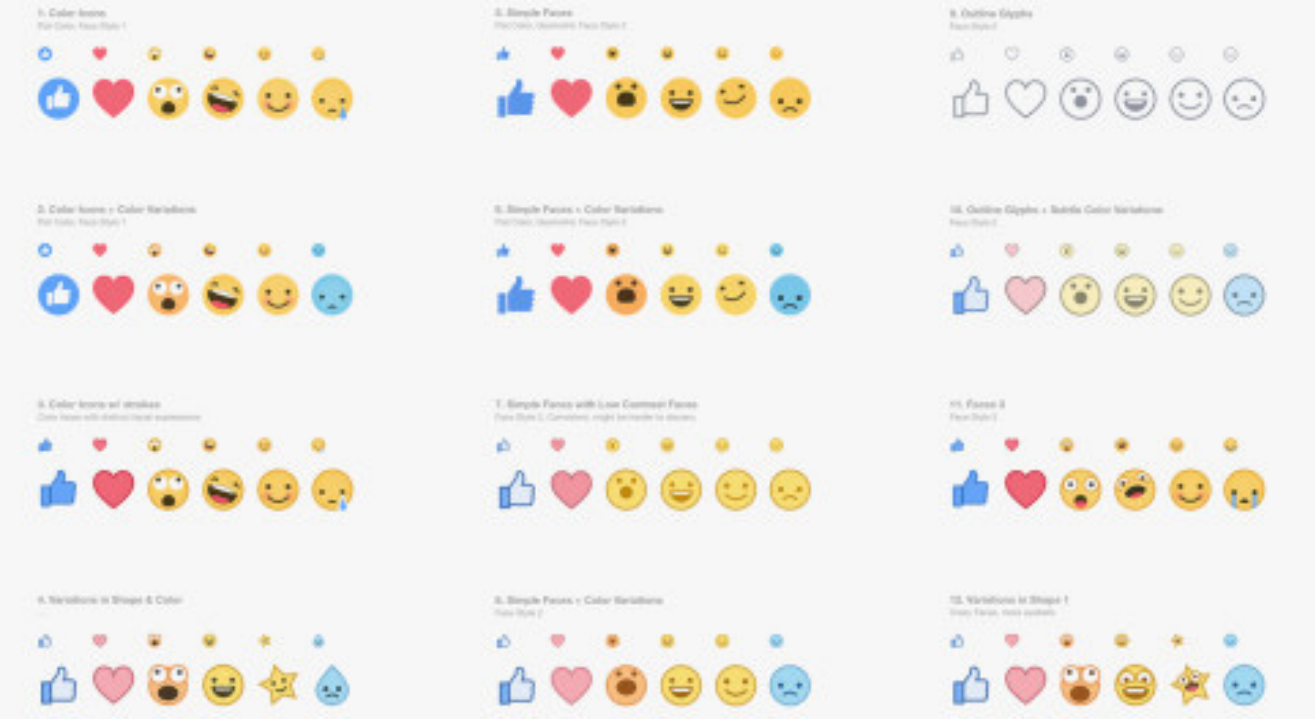
Tomar London gives the ability to direct deposit payroll into an employee bank account as one of Gusto’s “must-have” features. It’s something that customers consider absolutely necessary for any payroll system, and assume that relevant companies will have it.



3. Move on to “neat” unexpected low-impact features after product-market fit

Neat features are nice-to-have. They’re not incredibly urgent, but, like everything else on your product roadmap, have to be considered carefully.

A good example of a neat feature is Facebook’s reactions. The ability to “like” things was a must-have. It allowed Facebook to target users with content it knew users would like, based on what they had liked in the past. Facebook users have asked Facebook for a “dislike” option since the beginning of the company and the ability to include a more complex range of emotions.



(source: Wired)

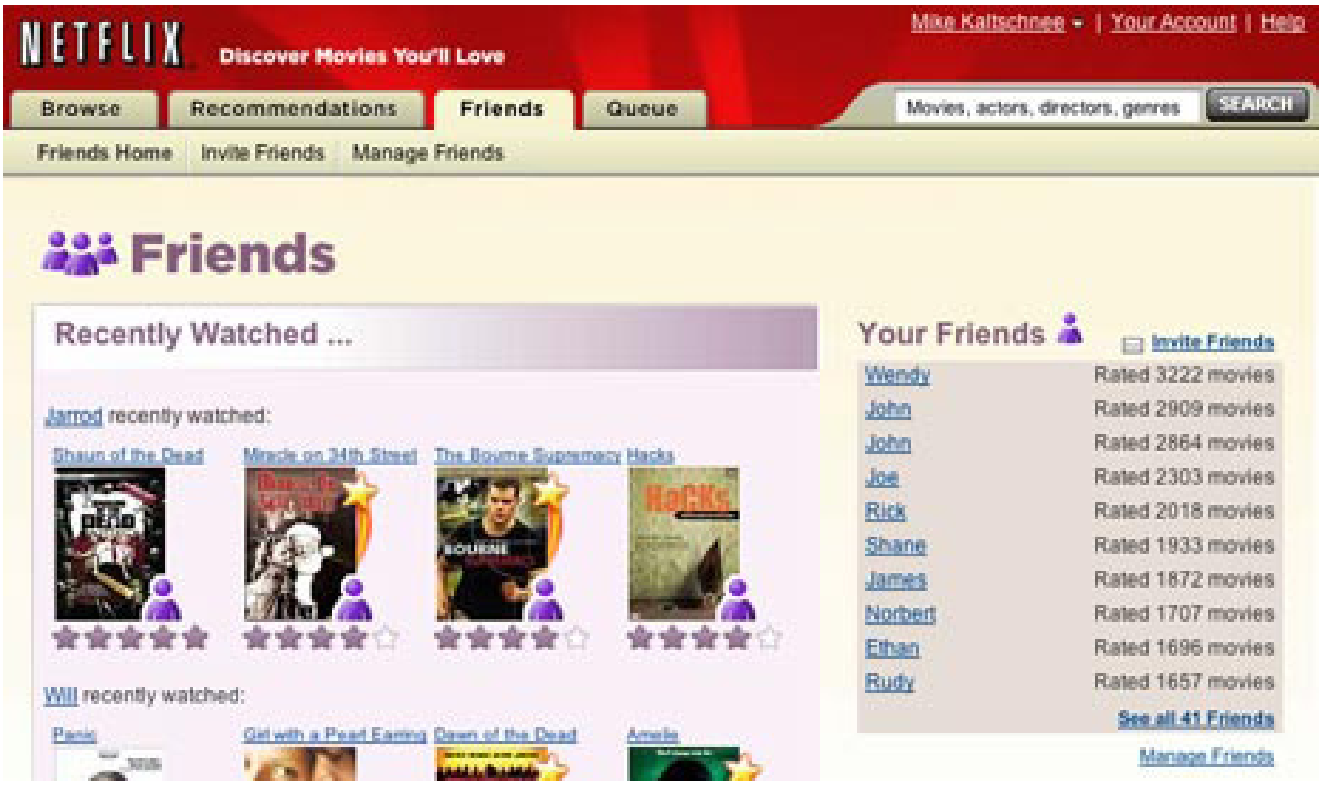
Reactions introduced a new complexity into the mix. The product team at Facebook combed through dozens of reactions. They pulled data from Facebook feeds to see which emojis were most frequently used. They stress-tested each set of reactions through multiple rounds of user-testing before settling on one for launch.

The lesson here is that there are not small changes to product. If you build a neat feature, make sure to do it right.

4. Focus on removing "who cares" low-impact expected features

Having a single product that does too many things can often be distracting, and what's worse is that it puts you at risk against someone else coming along and doing that thing better. Scrapping features can actually be one of the hardest things to do when building product. The technical complexity can be daunting, and there's always some (if tiny) subset of users who still depend on it.

In 2010, Netflix axed its Friends feature, and phased it out in a succession of web redesigns. The Friends feature was used by less than 2% of subscribers. It was distracting to the product team, and they figured that no one would miss it.



(source: Hacking Netflix)

Unfortunately, it turns out that these 2% subsets of your users also tend to be the most vocal—a lesson that Netflix quickly learned as its blog was ripped into by angry customers.

You'll always have to cut certain features from your product and it will always piss some of your customers off. Just don't try to sweep them under the rug.

How to Build the Habit

Product managers like to get things done. It's what makes them good at their jobs. That's why it's so incredibly hard to focus on building the right things, in sequence. In the moment, taking a step back feels like slowing down.

It seems counterintuitive, but building the habit of thinking about focus on sequence is necessary to move quickly.

- **Every day, ask yourself one question:** "Am I working on the right thing, right now?" If you're stumped for an answer you need to reassess everything you do.
- **Build deep relationships among members of your product teams.** As Brian Balfour points out, closely-knit teams that are willing to speak frankly and quickly about problems get them out of the way and are able to tackle them faster.
- **Use your budget to prioritize the features that you build.** As Basecamp co-founder David Hansson says, "One version of the feature might take 2 weeks, another might take 6 months. It's all in where you draw the line, how comprehensive you want to be, and what you're going to do about all those inevitable edge cases."

Move Fast, Focus

In 2004, just as Apple's iPod was taking off and Google's stock increased 4x in price, Steve Ballmer posted a memo that outlined Microsoft's focus areas:

> *"Non-PC Consumer Electronics: The opportunity is virtually unlimited to integrate the richness and intelligence of the PC world with everyday devices such as mobile phones, handheld devices, home entertainment and TV. At the center of our efforts are products such as Pocket PC and Smartphone, Portable Media Center, MSTV, MSN TV, Windows Automotive, the Windows Media Center Extender, and other electronic devices built on Windows CE and Windows XP Embedded."*

(Source: Ben Thompson)

When companies grow big, they inevitably lose focus. The more products you have, the harder it is to know what to focus on and when. Unlike Microsoft, however, as a startup you don't have nearly infinite resources to throw at problems. Your single advantage is the ability to press on one thing at a time.

How Can I Help You?

Before my co-founder and I built our SaaS product Crazy Egg, we had built over 12 different SaaS products between 2004 and 2006. Crazy Egg was the first product we built that actually stuck, and we made a lot of mistakes before getting it right.

Building better products is something that comes with time and it's still something I'm trying to improve at. I'd love to learn more about how you develop products, and offer any advice and help that I can. Fill out this Typeform survey to give more context around your specific challenges and I'll respond!