21: Half, Not Half-Assed

Build half a product, not a half-ass product

Beware of the “everything but the kitchen sink” approach to web app development. Throw in every decent idea that comes along and you’ll just wind up with a half-assed version of your product. What you really want to do is build half a product that kicks ass.

Stick to what’s truly essential. Good ideas can be tabled. **Take whatever you think your product should be and cut it in half.** Pare features down until you’re left with only the most essential ones. Then do it again.

With Basecamp, we started with just the messages section. We knew that was the heart of the app so we ignored milestones, to-do lists, and other items for the time being. That let us base future decisions on real world usage instead of hunches.

Start off with a lean, smart app and let it gain traction. Then you can start to add to the solid foundation you’ve built.

22: It Just Doesn't Matter

Essentials only

Our favorite answer to the “why didn’t you do this or why didn’t you do that?” question is always: “Because it just doesn’t matter.” That statement embodies what makes a product great. Figuring out what matters and leaving out the rest.

When we launched Campfire we heard some of these questions from people checking out the product for the first time:

“Why time stamps only every 5 minutes? Why not time stamp every chat line?” Answer: It just doesn’t matter. How often do you need to track a conversation by the second or even the minute? Certainly not 95% of the time. 5 minute stamps are sufficient because anything more specific just doesn’t matter.

“Why don’t you allow bold or italic or colored formatting in the chats?” Answer: It just doesn’t matter. If you need to emphasize something use the trusty caps lock key or toss a few *’s around the word or phrase. Those solutions don’t require additional software, tech support, processing power, or have a learning curve. Besides, heavy formatting in a simple text-based chat just doesn’t matter.

“Why don’t you show the total number of people in the room at a given time?” Answer: It just doesn’t matter. Everyone’s name is listed so you know who’s there, but what difference does it make if there’s 12 or 16 people? If it doesn’t change your behavior, then it just doesn’t matter.

Would these things be nice to have? Sure. But are they essential? Do they really matter? Nope. And that’s why we left them out. The best designers and the best programmers aren’t the ones with the best skills, or the nimblest fingers, or the ones who can rock and roll with Photoshop or their environment of choice, they are the ones that can determine what just doesn’t matter. That’s where the real gains are made.

Most of the time you spend is wasted on things that just don’t matter. If you can cut out the work and thinking that just don’t matter, you’ll achieve productivity you’ve never imagined.

23: Start With No

Make features work hard to be implemented

The secret to building half a product instead of a half-ass product is saying no.

Each time you say yes to a feature, you're adopting a child. You have to take your baby through a whole chain of events (e.g. design, implementation, testing, etc.). And once that feature's out there, you're stuck with it. Just try to take a released feature away from customers and see how pissed off they get.

Don't be a yes-man

Make each feature work hard to be implemented. Make each feature prove itself and show that it's a survivor. It's like "Fight Club." You should only consider features if they're willing to stand on the porch for three days waiting to be let in.

That's why you start with no. Every new feature request that comes to us — or from us — meets a no. We listen but don't act. The initial response is "not now." If a request for a feature keeps coming back, that's when we know it's time to take a deeper look. Then, and only then, do we start considering the feature for real.

And what do you say to people who complain when you won't adopt their feature idea? Remind them why they like the app in the first place. "You like it because we say no. You like it because it doesn't do 100 other things. You like it because it doesn't try to please everyone all the time."

"We Don't Want a Thousand Features"

> *Steve Jobs gave a small private presentation about the iTunes Music Store to some independent record label people. My favorite line of the day was when people kept raising their hand saying, "Does it do [x]?", "Do you plan to add [y]?". Finally Jobs said, "Wait wait — put your hands down. Listen: I know you have a thousand ideas for all the cool features iTunes could have. So do we. But we don't want a thousand features. That would be ugly. Innovation is not about saying yes to everything. It's about saying NO to all but the most crucial features."*

—-Derek Sivers, president and programmer, CD Baby and HostBaby (from Say NO by default)

24: Hidden Costs

Expose the price of new features

Even if a feature makes it past the “no” stage, you still need to expose its hidden costs.

For example, be on the lookout for feature loops (i.e. features that lead to more features). We’ve had requests to add a meetings tab to Basecamp. Seems simple enough until you examine it closely. Think of all the different items a meetings tab might require: location, time, room, people, email invites, calendar integration, support documentation, etc. That’s not to mention that we’d have to change promotional screenshots, tour pages, faq/help pages, the terms of service, and more. Before you know it, a simple idea can snowball into a major headache.

For every new feature you need to...

- 1. Say no.
- 2. Force the feature to prove its value.
- 3. If “no” again, end here. If “yes,” continue...
- 4. Sketch the screen(s)/ui.
- 5. Design the screen(s)/ui.
- 6. Code it.
- 7-15. Test, tweak, test, tweak, test, tweak, test, tweak...
- 16. Check to see if help text needs to be modified.
- 17. Update the product tour (if necessary).
- 18. Update the marketing copy (if necessary).
- 19. Update the terms of service (if necessary).
- 20. Check to see if any promises were broken.
- 21. Check to see if pricing structure is affected.
- 22. Launch.
- 23. Hold breath.

25: Can You Handle It?

Build something you can manage

If you launch an affiliate program do you have the systems in place to handle the accounting and payouts? Maybe you should just let people earn credit against their membership fees instead of writing, signing, and mailing a check each month.

Can you afford to give away 1 gb of space for free just because Google does? Maybe you should start small at 100 mb, or only provide space on paying accounts.

Bottom line: Build products and offer services you can manage. It's easy to make promises. It's much harder to keep them. Make sure whatever it is that you're doing is something you can actually sustain — organizationally, strategically, and financially.

26: Human Solutions

Build software for general concepts and encourage people to create their own solutions

Don't force conventions on people. Instead make your software general so everyone can find their own solution. Give people just enough to solve their own problems their own way. And then get out of the way.

When we built Ta-da List we intentionally omitted a lot of stuff. There's no way to assign a to-do to someone, there's no way to mark a date due, there's no way to categorize items, etc.

We kept the tool clean and uncluttered by letting people get creative. People figured out how to solve issues on their own. If they wanted to add a date to a to-do item they could just add (due: April 7, 2006) to the front of the item. If they wanted to add a category, they could just add [Books] to the front of the item. Ideal? No. Infinitely flexible? Yes.

If we tried to build software to specifically handle these scenarios, we'd be making it less useful for all the cases when those concerns don't apply.

Do the best job you can with the root of the problem then step aside. People will find their own solutions and conventions within your general framework.

27: Forget Feature Requests

Let your customers remind you what's important

Customers want everything under the sun. They'll avalanche you with feature requests. Just check out our product forums; The feature request category always trumps the others by a wide margin.

We'll hear about "this little extra feature" or "this can't be hard" or "wouldn't it be easy to add this" or "it should take just a few seconds to put it in" or "if you added this I'd pay twice as much" and so on.

Of course we don't fault people for making requests. We encourage it and we want to hear what they have to say. Most everything we add to our products starts out as a customer request. But, as we mentioned before, your first response should be a no. So what do you do with all these requests that pour in? Where do you store them? **How do you manage them? You don't. Just read them and then throw them away.**

Yup, read them, throw them away, and forget them. It sounds blasphemous but the ones that are important will keep bubbling up anyway. Those are the only ones you need to remember. Those are the truly essential ones. Don't worry about tracking and saving each request that comes in. Let your customers be your memory. If it's really worth remembering, they'll remind you until you can't forget.

How did we come to this conclusion? When we first launched Basecamp we tracked every major feature request on a Basecamp to-do list. When a request was repeated by someone else we'd update the list with an extra hash mark (II or III or IIII, etc). We figured that one day we'd review this list and start working from the most requested features on down.

But the truth is we never looked at it again. We already knew what needed to be done next because our customers constantly reminded us by making the same requests over and over again. There was no need for a list or lots of analysis because it was all happening in real time. You can't forget what's important when you are reminded of it every day.

And one more thing: Just because x number of people request something, doesn't mean you have to include it. Sometimes it's better to just say no and maintain your vision for the product.

28: Hold the Mayo

Ask people what they *don't* want

Most software surveys and research questions are centered around what people want in a product. "What feature do you think is missing?" "If you could add just one thing, what would it be?" "What would make this product more useful for you?"

What about the other side of the coin? Why not ask people what they don't want? "If you could remove one feature, what would it be?" "What don't you use?" "What gets in your way the most?"

More isn't the answer. Sometimes the biggest favor you can do for customers is to leave something out.

Innovation Comes From Saying No

> *[Innovation] comes from saying no to 1,000 things to make sure we don't get on the wrong track or try to do too much. We're always thinking about new markets we could enter, but it's only by saying no that you can concentrate on the things that are really important.*

—Steve Jobs, CEO, Apple (from The Seed of Apple's Innovation)