15: What's the Big Idea?

Explicitly define the one-point vision for your app

What does your app stand for? What's it really all about? Before you start designing or coding anything you need to know the purpose of your product — the vision. Think big. Why does it exist? What makes it different than other similar products?

This vision will guide your decisions and keep you on a consistent path. Whenever there's a sticking point, ask, "Are we staying true to the vision?"

Your vision should be brief too. A sentence should be enough to get the idea across. Here's the vision for each of our products:

- **Basecamp**: Project management is communication
- **Backpack**: Bring life's loose ends together
- **Campfire**: Group chat over IM sucks
- **Ta-da List**: Competing with a post-it note
- **Writeboard**: Word is overkill

With Basecamp, for example, the vision was "Project management is communication." We felt strongly that effective communication on a project leads to collective ownership, involvement, investment, and momentum. It gets everyone on the same page working toward a common goal. We knew if Basecamp could accomplish this, everything else would fall in line.

This vision led us to keep Basecamp as open and transparent as possible. Instead of limiting communication to within a firm, we gave clients access too. We thought less about permissions and more about encouraging all participants to take part. The vision is why we skipped charts, graphs, tables, reports, stats, and spreadsheets and instead focused on communication priorities like messages, comments, to-do lists, and sharing files. Make the big decision about your vision upfront and all your future little decisions become much easier.

Whiteboard philosophy

> *Andy Hunt and I once wrote a debit card transaction switch. A major requirement was that the user of a debit card shouldn't have the same transaction applied to their account twice. In other words, no matter what sort of failure mode might happen, the error should be on the side of not processing a transaction rather than processing a duplicate transaction.*
>
> *So, we wrote it on our shared whiteboard in big letters: Err in favor of users.*

> *It joined about half-a-dozen other maxims. Jointly, these guided all those tricky decisions you make while building something complex. Together, these laws gave our application strong internal coherence and great external consistency.*

—Dave Thomas, The Pragmatic Programmers

Make Mantra

> *Organizations need guideposts. They need an outline; employees need to know each day when they wake up why they're going to work. This outline should be short and sweet, and all encompassing: Why do you exist? What motivates you? I call this a mantra — a three or four-word description of why you exist.*

—Guy Kawasaki, author (from Make Mantra)

16: Ignore Details Early On

Work from large to small

We're crazy about details.

- The space between objects
- The perfect type leading
- The perfect color
- The perfect words
- Four lines of code instead of seven
- 90% vs 89%
- 760px vs 750px
- $39/month vs. $49/month

Success and satisfaction are in the details.

However, success isn't the only thing you'll find in the details. You'll also find stagnation, disagreement, meetings, and delays. These things can kill morale and lower your chances of success.

How often have you found yourself stuck on a single design or code element for a whole day? How often have you realized that the progress you made today wasn't real progress? This happens when you focus on details too early in the process. There's plenty of time to be a perfectionist. Just do it later.

Don't worry about the size of your headline font in week one. You don't need to nail that perfect shade of green in week two. You don't need to move that "submit" button three pixels to the right in week three. Just get the stuff on the page for now. Then use it. Make sure it works. Later on you can adjust and perfect it.

Details reveal themselves as you use what you're building. You'll see what needs more attention. You'll feel what's missing. You'll know which potholes to pave over because you'll keep hitting them. That's when you need to pay attention, not sooner.

The Devil's in the Details

> *I really got over the "get into details right away" attitude after I took some drawing classes...If you begin to draw the details right away you can be sure that the drawing is going to suck. In fact, you are completely missing the point.*

> *You should begin by getting your proportions right for the whole scene. Then you sketch the largest objects in your scene, up to the smallest one. The sketch must be very loose up to this point. Then you can proceed with shading which consists of bringing volume to life. You begin with only three tones (light, medium, dark). This gives you a tonal sketch. Then for each portion of your drawing you reevaluate three tonal shades and apply them. Do it until the volumes are there (requires multiple iteration)...*
>
> *Work from large to small. Always.*

—Patrick Lafleur, Creation Objet Inc. (from Signal vs. Noise)

17: It's a Problem When It's a Problem

Don’t waste time on problems you don’t have yet

Do you really need to worry about scaling to 100,000 users today if it will take you two years to get there?

Do you really have to hire eight programmers if you only need three today?

Do you really need 12 top-of-the-line servers now if you can run on two for a year?

Just Wing It

People often spend too much time up front trying to solve problems they don’t even have yet. Don’t. Heck, we launched Basecamp without the ability to bill customers! Since the product billed in monthly cycles, we knew we had a 30-day gap to figure it out. We used that time to solve more urgent problems and then, after launch, we tackled billing. It worked out fine (and it forced us into a simple solution without unnecessary bells and whistles).

Don’t sweat stuff until you actually must. Don’t overbuild. Increase hardware and system software as necessary. If you’re slow for a week or two it’s not the end of the world. Just be honest: explain to your customers you’re experiencing some growing pains. They may not be thrilled but they’ll appreciate the candor.

Bottom Line: Make decisions just in time, when you have access to the real information you need. In the meanwhile, you’ll be able to lavish attention on the things that require immediate care.

18: Hire the Right Customers

Find the core market for your application and focus solely on them

The customer is not always right. The truth is you have to sort out who's right and who's wrong for your app. The good news is that the internet makes finding the right people easier than ever.

If you try to please everyone, you won't please anyone

When we built Basecamp we focused our marketing on design firms. By narrowing our market this way, we made it more likely to attract passionate customers who, in turn, would evan gelize the product. Know who your app is really intended for and focus on pleasing them.

The Best Call We Ever Made

> *The decision to aim Campaign Monitor strictly at the web design market was the best call we ever made. It allowed us to easily identify which features would be genuinely useful and, more importantly, which features to leave out. Not only have we attracted more customers by targeting a smaller group of people, these customers all have similar needs which makes our job much easier. There are loads of features in Campaign Monitor that would be useless to anyone but a web designer.*
>
> *Focusing on a core market also makes it much easier to spread the word about your software. Now that we have a tightly defined audience, we can advertise where they frequent online, publish articles they might find interesting, and generally build a community around our product.*

—David Greiner, founder, Campaign Monitor

19: Scale Later

You don’t have a scaling problem yet

“Will my app scale when millions of people start using it?”

Ya know what? Wait until that actually happens. If you’ve got a huge number of people overloading your system then huzzah! That’s one swell problem to have. The truth is the overwhelming majority of web apps are never going to reach that stage. And even if you do start to get overloaded it’s usually not an allor-nothing issue. You’ll have time to adjust and respond to the problem. Plus, you’ll have more real-world data and benchmarks after you launch which you can use to figure out the areas that need to be addressed.

For example, we ran Basecamp on a single server for the first year. Because we went with such a simple setup, it only took a week to implement. We didn’t start with a cluster of 15 boxes or spend months worrying about scaling.

Did we experience any problems? A few. But we also realized that most of the problems we feared, like a brief slowdown, really weren’t that big of a deal to customers. As long as you keep people in the loop, and are honest about the situation, they’ll understand. In retrospect, we’re quite glad we didn’t delay launch for months in order to create “the perfect setup.”

In the beginning, make building a solid core product your priority instead of obsessing over scalability and server farms. **Create a great app and then worry about what to do once it’s wildly successful.** Otherwise you may waste energy, time, and money fixating on something that never even happens.

Believe it or not, the bigger problem isn’t scaling, it’s getting to the point where you have to scale. Without the first problem you won’t have the second.

You have to revisit anyway

> *The fact is that everyone has scalability issues, no one can deal with their service going from zero to a few million users without revisiting almost every aspect of their design and architecture.*

—Dare Obasanjo, Microsoft (from Scaling Up and Startups)

20: Make Opinionated Software

Your app should take sides

Some people argue software should be agnostic. They say it's arrogant for developers to limit features or ignore feature requests. They say software should always be as flexible as possible.

We think that's bullshit. The best software has a vision. The best software takes sides. When someone uses software, they're not just looking for features, they're looking for an approach. They're looking for a vision. Decide what your vision is and run with it.

And remember, if they don't like your vision there are plenty of other visions out there for people. Don't go chasing people you'll never make happy.

A great example is the original wiki design. Ward Cunningham and friends deliberately stripped the wiki of many features that were considered integral to document collaboration in the past. Instead of attributing each change of the document to a certain person, they removed much of the visual representation of ownership. They made the content ego-less and time-less. They decided it wasn't important who wrote the content or when it was written. And that has made all the difference. This decision fostered a shared sense of community and was a key ingredient in the success of Wikipedia.

Our apps have followed a similar path. They don't try to be all things to all people. They have an attitude. They seek out customers who are actually partners. They speak to people who share our vision. You're either on the bus or off the bus.