1: What is Getting Real

Want to build a successful web app? Then it's time to Get Real. Getting Real is a smaller, faster, better way to build software.

- Getting Real is about skipping all the stuff that represents real (charts, graphs, boxes, arrows, schematics, wireframes, etc.) and actually building the real thing.
- Getting real is less. Less mass, less software, less features, less paperwork, less of everything that's not essential (and most of what you think is essential actually isn't).
- Getting Real is staying small and being agile.
- Getting Real starts with the interface, the real screens that people are going to use. It begins with what the customer actually experiences and builds backwards from there. This lets you get the interface right before you get the software wrong.
- Getting Real is about iterations and lowering the cost of change. Getting Real is all about launching, tweaking, and constantly improving which makes it a perfect approach for web-based software.
- Getting Real delivers just what customers need and eliminates anything they don't.

The benefits of Getting Real

Getting Real delivers better results because it forces you to deal with the actual problems you're trying to solve instead of your ideas about those problems. It forces you to deal with reality.

Getting Real foregoes functional specs and other transitory documentation in favor of building real screens. A functional spec is make-believe, an illusion of agreement, while an actual web page is reality. That's what your customers are going to see and use. That's what matters. Getting Real gets you there faster. And that means you're making software decisions based on the real thing instead of abstract notions.

Finally, Getting Real is an approach ideally suited to web-based software. The old school model of shipping software in a box and then waiting a year or two to deliver an update is fading away. Unlike installed software, web apps can constantly evolve on a day-to-day basis. Getting Real leverages this advantage for all its worth.

How To Write Vigorous Software

> *Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts. This requires not that the writer make all sentences short or avoid all detail and treat subjects only in outline, but that every word tell.*

—From "The Elements of Style" by William Strunk Jr.

No more bloat

The old way: a lengthy, bureaucratic, we're-doing-this-to-coverour-asses process. The typical result: bloated, forgettable software dripping with mediocrity. Blech.

Getting Real gets rid of...

- Timelines that take months or even years
- Pie-in-the-sky functional specs
- Scalability debates
- Interminable staff meetings
- The "need" to hire dozens of employees
- Meaningless version numbers
- Pristine roadmaps that predict the perfect future
- Endless preference options
- Outsourced support
- Unrealistic user testing
- Useless paperwork
- Top-down hierarchy

You don't need tons of money or a huge team or a lengthy development cycle to build great software. Those things are the ingredients for slow, murky, changeless applications. Getting real takes the opposite approach.

In this book we'll show you...

- The importance of having a philosophy
- Why staying small is a good thing
- How to build less
- How to get from idea to reality quickly

- How to staff your team
- Why you should design from the inside out
- Why writing is so crucial
- Why you should underdo your competition
- How to promote your app and spread the word
- Secrets to successful support
- Tips on keeping momentum going after launch
- ...and lots more

The focus is on big-picture ideas. We won't bog you down with detailed code snippets or css tricks. We'll stick to the major ideas and philosophies that drive the Getting Real process.

Is this book for you?

You're an entrepreneur, designer, programmer, or marketer working on a big idea.

You realize the old rules don't apply anymore. Distribute your software on cd-roms every year? How 2002. Version numbers? Out the window. You need to build, launch, and tweak. Then rinse and repeat.

Or maybe you're not yet on board with agile development and business structures, but you're eager to learn more.

If this sounds like you, then this book is for you.

Note: While this book's emphasis is on building a web app, a lot of these ideas are applicable to non-software activities too. The suggestions about small teams, rapid prototyping, expecting iterations, and many others presented here can serve as a guide whether you're starting a business, writing a book, designing a web site, recording an album, or doing a variety of other endeavors. Once you start Getting Real in one area of your life, you'll see how these concepts can apply to a wide range of activities.

2: About Basecamp

What we do

Basecamp is a small team that creates simple, focused software. Our products help you collaborate and get organized. Millions of people and small businesses use our web-apps to get things done. Jeremy Wagstaff, of the Wall Street Journal, wrote, “[Basecamp] products are beautifully simple, elegant and intuitive tools that make an Outlook screen look like the soft ware equivalent of a torture chamber.” Our apps never put you on the rack.

Our modus operandi

We believe software is too complex. Too many features, too many buttons, too much to learn. Our products do less than the competition — intentionally. We build products that work smarter, feel better, allow you to do things your way, and are easier to use.

Our products

Basecamp turns project management on its head. Instead of Gantt charts, fancy graphs, and stats-heavy spreadsheets, Basecamp offers message boards, to-do lists, simple scheduling, collaborative writing, and file sharing. So far, hundreds of thousands agree it’s a better way. Farhad Manjoo of Salon.com said “Basecamp represents the future of software on the Web.”

HEY is a ground up reinvention of email. HEY’s fresh approach transforms email into something you want to use instead of something you’re forced to deal with.

Ruby on Rails, for developers, is a full-stack, open-source web framework in Ruby for writing real-world applications quickly and easily. Rails takes care of the busy work so you can focus on your idea. Nathan Torkington of the O’Reilly publish ing empire said “Ruby on Rails is astounding. Using it is like watching a kung-fu movie, where a dozen bad-ass frameworks prepare to beat up the little newcomer only to be handed their asses in a variety of imaginative ways.” Gotta love that quote.

You can find out more about our products and our company on our web site at basecamp.com.

3: Caveats, disclaimers, and other preemptive strikes

Just to get it out of the way, here are our responses to some complaints we hear every now and again:

“These techniques won’t work for me.”

Getting real is a system that’s worked terrifically for us. That said, the ideas in this book won’t apply to every project under the sun. If you are building a weapons system, a nuclear control plant, a banking system for millions of customers, or some other life/finance-critical system, you’re going to balk at some of our laissez-faire attitude. Go ahead and take additional precautions.

And it doesn’t have to be an all or nothing proposition. Even if you can’t embrace Getting Real fully, there are bound to be at least a few ideas in here you can sneak past the powers that be.

“You didn’t invent that idea.”

We’re not claiming to have invented these techniques. Many of these concepts have been around in one form or another for a long time. Don’t get huffy if you read some of our advice and it reminds you of something you read about already on so and so’s weblog or in some book published 20 years ago. It’s definitely possible. These techniques are not at all exclusive to Basecamp. We’re just telling you how we work and what’s been successful for us.

“You take too much of a black and white view.”

If our tone seems too know-it-allish, bear with us. We think it’s better to present ideas in bold strokes than to be wishy-washy about it. If that comes off as cocky or arrogant, so be it. We’d rather be provocative than water everything down with “it depends…” Of course there will be times when these rules need to be stretched or broken. And some of these tactics may not apply to your situation. Use your judgement and imagination.

“This won’t work inside my company.”

Think you’re too big to Get Real? Even Microsoft is Getting Real (and we doubt you’re bigger than them).

Even if your company typically runs on long-term schedules with big teams, there are still ways to get real.The first step is to break up into smaller units. When there's too many people involved, nothing gets done. The leaner you are, the faster — and better — things get done.

Granted, it may take some salesmanship. Pitch your company on the Getting Real process. Show them this book. Show them the real results you can achieve in less time and with a smaller team.

Explain that Getting Real is a low-risk, low-investment way to test new concepts. See if you can split off from the mothership on a smaller project as a proof of concept. Demonstrate results.

Or, if you really want to be ballsy, go stealth. Fly under the radar and demonstrate real results. That's the approach the Start.com team has used while Getting Real at Microsoft. "I've watched the Start.com team work. They don't ask permission," says Robert Scoble, Technical Evangelist at Microsoft. "They have a boss that provides air cover. And they bite off a little bit at a time and do that and respond to feedback."

Shipping Microsoft's Start.com

> *In big companies, processes and meetings are the norm. Many months are spent on planning features and arguing details with the goal of everyone reaching an agreement on what is the "right" thing for the customer.*
>
> *That may be the right approach for shrink-wrapped software, but with the web we have an incredible advantage. Just ship it! Let the user tell you if it's the right thing and if it's not, hey you can fix it and ship it to the web the same day if you want! There is no word stronger than the customer's — resist the urge to engage in long-winded meetings and arguments. Just ship it and prove a point.*
>
> *Much easier said than done — this implies:*
>
> *Months of planning are not necessary.*
> *Months of writing specs are not necessary — specs should have the foundations nailed and details figured out and refined during the development phase. Don't try to close all open issues and nail every single detail before development starts.*
>
> *Ship less features, but quality features.*
> *You don't need a big bang approach with a whole new release and bunch of features. Give the users byte-size pieces that they can digest.*
>
> *If there are minor bugs, ship it as soon you have the core scenarios nailed and ship the bug fixes to web gradually after that. The faster you get the user feedback the better. Ideas can sound great on paper but in practice turn out to be suboptimal. The sooner you find out about fundamental issues that are wrong with an idea, the better.*

> *Once you iterate quickly and react on customer feedback, you will establish a customer connection. Remember the goal is to win the customer by building what they want.*

—Sanaz Ahari, Program Manager of Start.com, Microsoft