

4. Managing Clients

You can't do freelance software development without clients. Well, you can, but you may find it tough to bring in any money. In order to be a successful freelance developer, you need to develop skills in finding clients, keeping them after you find them, and sometimes even getting rid of them. In this Section, I'll share with you what I know about the care and feeding of clients.

Marketing Yourself

As an employee, you may never have thought about marketing. As a freelancer, it's essential. If your potential clients can't find you, then there's no chance that they'll hire you. Over the years I've tried a variety of things. Here are some of the marketing ideas that have *not* worked for me:

- Google AdWords and similar programs
- Craigslist advertising
- Local business groups (Chamber of Commerce, Kiwanis, etc.)

Now, it's possible that some of these will work for you, and I don't pretend to be a marketing genius. But reflecting back over my years in this business, the vast majority of work has come in through a single source: word of mouth. And that word of mouth doesn't happen by magic: it happens because I am fairly relentless about letting people know that I write code and that they can hire me.

These days, being relentless means being online. Among other things, I maintain:

- An active Twitter account (<http://twitter.com/MikeGI>)
- A LinkedIn profile (<http://www.linkedin.com/in/larkware>)
- A Working With Rails profile (<http://www.workingwithrails.com/person/7738-mike-gunderloy>)
- A frequently-updated blog (<http://afreshcup.com>)
- A corporate brochureware site (<http://larkware.com>)
- An active Lighthouse user account (<https://rails.lighthouseapp.com/users/7211>)

- An active record of contributing to Rails itself (<http://contributors.rubyonrails.org/contributors/mike-gunderloy/commits>)

You get the idea. My goal is to make sure that plenty of people think of me when they think of Rails consulting - and that they know where to find me.

There's another piece to this picture, of course. The best word of mouth comes from satisfied customers - and, conversely, the worst word of mouth comes from unhappy customers. If you want your name to be passed along, you need to make sure you keep your customers satisfied.

Marketing Rails

I also find that I have to do a certain amount of marketing Rails itself. This need is decreasing as the framework matures and gets more well-known, but you'll still find people out there who get wary as soon as you bring Rails into the conversation. There are a few points that usually go over with customers who are unsure about hiring a Rails developer.

First, you can make a great case against Microsoft and other heavyweight commercial alternatives on the basis of price. If you want to build a dynamic web site with Microsoft technologies, you have to be prepared to buy licenses of Windows Server and SQL Server - and that is not inexpensive. With a typical Rails app, your total license cost is likely to be zero. In addition, Rails hosting on a VPS tends to be much cheaper than Windows hosting for an equivalent site - though you may not want to bang on that too hard, as there are some very lowball bottom feeder Windows hosts out there.

Second, if they're worried about Rails being immature or unscalable (the two most common complaints, I've found, from those who are only vaguely aware of Rails), you can point to success stories. If Twitter, Scribd, Hulu, Slideshare, whitepages.com, yellowpages.com, and more can bet their business on Rails (and succeed) it's very likely that your prospective customer can too. The lists of top Rails applications at rails100 (<http://rails100.pbworks.com/>) can be useful in this regard.

Lastly, you can move the discussion from Rails to yourself. This works best when you have a personal track record to point to - but when you do, it's reasonable to tell customers something like, "Rails isn't suited for every site, but I think it will work fine for yours. The only things I'm not immediately certain about are X and Y. Let's do a quick technology spike to remove that uncertainty, and then make a final decision as to how to move forward. I've built my career on keeping customers happy and I'll make sure you're happy too. And if Rails isn't right for you, I'll be happy to give you a referral to another developer."

Estimating Basics

One of the key steps in the dance between finding a client and signing them to a contract is coming up with an estimate for how long the work will take (and therefore how much it will cost, unless you ignore my advice and enter into fixed-bid contracts).

Estimating is a tricky skill and one that you don't often need in a full-time job. The good news is that it's a skill that can be learned, like any other. Here are some tips for providing better estimates.

Upgrade The Envelope

If you're figuring out your estimates on the proverbial back of the envelope, or cocktail napkins, or some similar device - stop. There's a reason "back of the envelope" is a synonym for "rough and inaccurate." You need to keep careful track of how you came up with the number that you send to the client. At the very least this means putting it into a document or spreadsheet somewhere. If you're fond of agile project management, there's a lot to be said for using your project management tool to come up with your initial estimate, by breaking down the job into small pieces and estimating each piece. I use Pivotal Tracker for this, which has the advantage of naturally rolling into project management when I get the job.

The main reason not to use the cocktail napkin is that it doesn't allow for the back-and-forth discussion that goes into a good estimate. You'll almost certainly want to ask questions after completing your first round of estimating, and revise the estimate based on customer responses; it's a rare job where things are specified well enough to come up with a price in the initial email. If you take a documented approach to your estimates they'll be much easier to adjust for such changing circumstances.

2+2 = ?

If you said "4" then you're thinking of some domain other than time estimating.

Think about a single task you estimated at taking two hours. Could you get lucky and finish it in 1? Sure. Could you get so lucky that you finish it in .5 hours? Nope. Could you have blown the estimate entirely and find it taking 3, 8, 15 hours? Sure.

Accuracy in estimates is asymmetric. If you have two 2-hour tasks, you might get really lucky and finish them both in 2 hours total. But no matter how hard you try, you can't take less than no time at all to do the work. And you might get wildly unlucky and discover those two tasks put together take you 20 or 30 hours. You need to allow for this asymmetry when you're looking at a bunch of little task estimates and adding them up to get a job estimate.

There's another reason estimating by summing task numbers is dangerous: like most developers, you probably suffer from a combination of optimism and hubris that makes it hard for you to allow for bugs, mistakes, and just plain disasters. Sure, if everything goes well your individual little estimates can be added together to get a job estimate - but where's your margin for error?

Put these factors together and I like to use a simple rule of thumb for reasonably-sized projects: estimate the tasks, add the total time together, and then multiply it by 2.5 to get an estimate for the job.

Inch-Pebbles Are Your Friend

Avoid the temptation to produce an estimate in terms of huge milestones: 2 weeks for UI design, 1 week for database design, and so on. Estimating accurately in huge chunks is next to impossible. Instead, break those milestones down into inch-pebbles: 2 hours for styling the entry form nicely, 1 hour for a well-tested Customer model. Any task that looks like more than 6 hours of work should be decomposed into smaller pieces before you estimate it.

Use a Scale

Get in the habit of estimating tasks according to a fixed scale. For very early estimates, when I'm just trying to get an idea of overall budget, I tend to use 1/2 day, 1 day, 2 days, 3 days, 5 days. When you're able to refine to smaller tasks, 1/2 hour, 1-2 hours, 3-4 hours, 5-8 hours is a good set. It's easier to look at a set of tasks and sort them into buckets than it is to consider each separate task as a unique and beautiful snowflake.

Don't Speak from Hunger

It can be very tempting to come up with lowball estimates to get the job, especially if you know what the customer's budget is. Unless this job is the only thing between you and starvation, stick to your guns and provide a fair estimate. If this is a \$50,000 job, don't bid it at \$17,400 just because you know that the prospective client wants to spend no more than \$17,500. If a more realistic project comes along, you'll be kicking yourself for underbidding this one and then getting stuck with it.

Allow Yourself Some Wiggle Room

So you've worked through the estimating exercise and come up with a number of \$42,350 as your best guess. Don't send that number over to the client. You know as well as I do that the chance of you hitting the estimate on the nose is next to zero. You need to allow for two things here. First, the estimate itself isn't nearly as precise as a single number would indicate, so you need to send it over as a range rather than as a number. Second, you need to allow some billable overhead time for things like meetings and specifications (unless you included all of these things in your inch pebbles that you estimated).

If \$42,350 was my best guess, I'd probably quote \$45,000 to \$50,000 as my best guess. If I beat that, the client will be happy, especially if I never tell them what my original best estimate was.

TIP: The more uncertain the job, the wider your range should be. If the spec is very fuzzy, a two-to-one difference between high and low estimates is not unreasonable.

Allow the Client Some Wiggle Room Too

If you fear the number is high enough to make the client gasp and run away, it's a good thing to give them some choices as well. You can take two approaches to this. First, give them an idea of tradeoffs they can make:

"It'll take \$45,000 to \$50,000 to build everything you want, but if we postpone the PDF output requirements to the next release we should be able to get that down to more like \$30,000 or \$35,000."

Second, you can steer them into an agile approach:

"You've got about \$45,000 to \$50,000 worth of work here. Why don't we plan on doing about \$5,000 worth of work per week, and we'll teach you how to use our project management tool to prioritize the work. We'll make sure that the application stays in working order, so at any time you can call a halt and just ship the features we've implemented to that point. This gives us the flexibility to do the highest-priority work and lets you make the tradeoffs if the budget gets tight."

Watch Your Track Record

Keep track somewhere of the estimates you produce on jobs that you actually land - and how much it actually takes to do them. If you find that you're constantly estimating high, you can shave the initial numbers that you give to clients to make up for your natural pessimism. On the other hand, if you're consistently underestimating, then it's time to up the correction factor you apply before you send the estimate over to the client.

Keeping Clients Informed and Happy

I'm grouping these two things into a single section because in my experience an informed client is not always a happy one - but an uninformed client is always an unhappy one. Even though you might be happier talking to your computer than talking to people, you can't neglect this side of your freelancing business if you expect to succeed.

There are a variety of ways to keep your clients in the loop about what's going on with their project, and different clients will require different amounts of attention. Some will be happy with the occasional email and access to the project's issue-tracker. Others will insist on formal weekly reviews. Still others will call you on the telephone on a daily basis, despite your best efforts to break them of this habit.

Unless it gets to be a serious drag on your productivity, I recommend dealing with clients on their own preferred terms. But in most cases, here's what I try to steer towards when it's up to me to manage things:

- If the client is reasonably tech-savvy (ie, they've heard of an RSS feed and know what to do with it) I prefer to use tools that will generate RSS feeds of their activity. There are plenty of these out there: GitHub, Lighthouse, Redmine, Pivotal Tracker,

and No Kahuna among others. This has the obvious benefit of extending the Don't Repeat Yourself principle to clients. Just update the systems you're using anyhow, and let the details flow through.

- My second choice is to schedule written progress reports. Weekly is right for most medium-sized projects. These don't have to be extensive: what we accomplished last week, what we're planning for next week, and what the biggest threats are. Using three bullet points for each of those three sections makes for a nice, digestible high-level view of the project.
- If the client wants to have actual phone meetings, that's my last choice. Sometimes you can't escape, in which case I recommend scheduling for Friday afternoons or Monday mornings, when you wouldn't be getting much real work done in any case.

There's more to keeping clients happy than keeping them informed, of course. The key thing to keep in mind is that successful freelance work is a combination of their money and your advice. You can offer your best guidance and advice, but if the client really wants something, you have to choose between satisfying their wishes or walking out on the job. You can read more about the second choice in the next section.

Firing Clients

Yes, I said "firing clients." Getting rid of them. Telling them to vamoose. Letting them know that their services are no longer required. You get the idea.

Painful though the idea may be when you're just starting out and trying to get your first client, there comes a time in every freelancer's life when you have to recognize that things are just not working out. Think of it this way: some clients brighten your day every time they call, by being enthusiastic about the work you're doing, reasonable about deadlines, and paying your bills on time. Then there are the others. As your business grows and changes, you should maximize the first kind - which can mean getting rid of the second kind.

Why Fire a Client?

There are a variety of reasons why you might feel it's time to get rid of an old client. Here's a (certainly not exhaustive) list:

- They're verbally or physically abusive (yes, this happens)
- They refuse to pay their bills, or pay alarmingly late
- They insist you continue to do work at an old billing rate, even though your current rate is much higher
- They can't supply specs or design elements on a timely basis

- They expect kickbacks or free work in order to keep their business
- They act like they're doing you a huge favor to hire you
- They imply (or state) that they don't trust your professional skills or ethics

I'm sure you can extend the list if you've been in business for any length of time at all.

How to Fire a Client

Just because you've decided to fire a client is no reason to be a jerk about it. Instead, you should try your hardest to part ways in a polite and professional manner. You can't prevent them from badmouthing you if that's the sort of person they are, but you can keep your own karma in order. Here are some things to keep in mind when you need to sever a client relationship:

- Do it in writing, not over the phone.
- Don't leave room for ambiguity.
- Don't be argued into changing your mind. Would you want to work for someone who wanted to get rid of you?
- Complete all work in progress, or leave it in a state that's easy for someone else to pick up.
- Deliver your final bill with the termination letter.
- If you know someone who you think would be a better fit for this client, send over a referral.
- Be honest. If you're getting rid of them for persistent nonpayment, tell them. It won't help you to get a reputation for dishonesty.

Even with the best of intentions, firing a client is a painful process, and I wouldn't advise making a habit of it. But if you must do so, and you've searched your conscience to know it's the right thing, then don't let any misplaced loyalty stop you.