

# Our Approach to Software Development

by Todd Wilson, President, ekiwi, LLC

## Overview

We've developed a lot of software over the years. I'm happy to say that, on the whole, we've had very satisfied clients, and have managed to stay profitable along the way. Having said that, we've also made a lot of mistakes, and learned a lot of lessons the hard way. The intent of this document is to explain the approach and philosophy that guides our work, which we've arrived at through both positive and negative experiences in software development.

If you're reading this document, you're probably either an existing or a potential client of ours. Our desire is to help your business to succeed, while helping to ensure our own success. As you read through this document, please keep this dichotomy in mind.

## Risk

In any software development project there is an element of risk. Based on our experience, the primary risk we deal with is simply that we won't create quite what you, as our client, had in mind. We do our very best up front to capture in a statement of work document your needs and requirements, but we're invariably bound to miss or misinterpret something. We've also found that clients sometimes have a hard time picturing and communicating what it is they'd like to have built.

Other less common risks include:

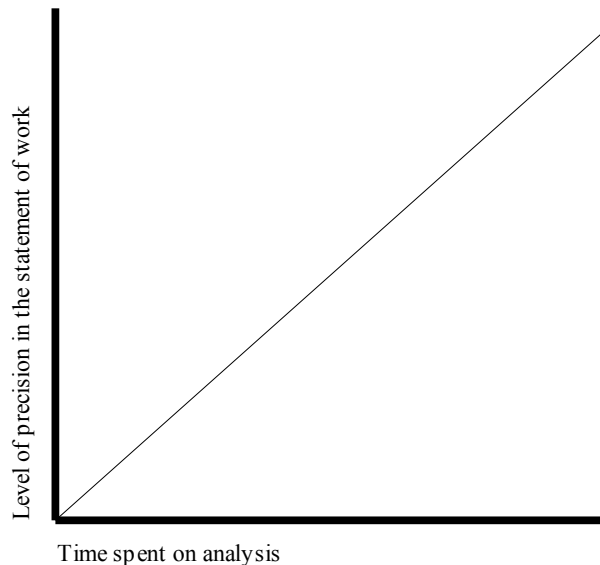
- Client needs change in the middle of a development cycle.
- Extracting information from a particular web site turns out to be more difficult than we anticipated. This can be because the web site:
  - Presents information in an inconsistent format. For example, they may vary the HTML tags they use to format a street address.
  - Blocks too many simultaneous requests. For example, the site may only allow up to 500 requests per hour from a given IP address.
- We get overloaded with work, and end up pushing deadlines out.
- We create software that contains bugs.

In the past we have shouldered virtually all of the risk in our projects. For example, if in our statement of work we didn't capture very well what the client had in mind, we've generally allowed for a significant number of tweaks and changes so as to deliver to the client what they truly wanted. For the most part, clients have been reasonable in this process and haven't demanded anything that was obviously outside of the scope of our statement of work.

In an ideal world, we would be able to capture perfectly in a statement of work what it is that you'd like to have us build. Unfortunately, this is essentially impossible, so we do our best to perform a reasonable amount of analysis of the project up front. Even at that, there will always be at least some

disparity between what you have in mind and what is contained in the statement of work.

Fortunately, there are ways to mitigate the risk of not delivering what it is you truly want. The best way we've found to do this is to invest more time up front in analysis. This generally involves many iterations of review with the client, and usually results in a longer and more detailed statement of work. The statement of work may even include such items as detailed screen mock-ups, entity relationship diagrams, and flow charts. The process can be visualized in this way:



In other words, the more time we spend on analysis, the more precise the statement of work is. Or, the longer we spend reviewing and revising the statement of work, the better we capture what it is that you truly have in mind.

This is a bit of a catch-22, however, as you may have already surmised. We could spend weeks detailing exactly what the client wants to have built. It could easily get to where we're spending an unreasonable amount of time trying to detail requirements and needs. Not only does this mean that we push the start date out for the project, but it also results in a higher cost for us. Time truly is money, and if we're investing a lot of time in analysis, it costs us more up front.

### **Sharing Risk**

It's only fair that the risk for a project be shared between both parties (i.e., you and us). We've had unfortunate situations in the past where, because we've shouldered virtually all of the risk, we ended up losing money on a project with the seemingly infinite number of tweaks and changes a client insisted upon. As such, we now put mechanisms in place so as to avoid these situations.

When we embark upon a project, it will be up to you how much analysis should be done up front. Given that the level of analysis will have a direct impact on the level of risk of the project, you should have a say in just how much risk we allow for. For example, it may be that you have a very tight budget for a project, and can't afford for it to go over a specific price point. Should you elect to have a significant amount of analysis done up front, you will need to pay for the time we invest in that analysis. In cases where budget and risk aren't big issues, we'll do a reasonable amount of analysis up front, for which we won't charge you. This obviously gets a bit tricky, since it's a gray area. At what

point does a “reasonable” amount of analysis become a “significant” amount of analysis? The number we use to guide us is 3%. That is, of the total time required to develop your project, we will not charge you for about 3% of the time spent in analysis. Even with that hard number it's still gray, so see the “We're Going to Be Nice to You” section for more on this topic.

In addition to sharing the cost of analysis in order to mitigate risk, we also attach a maximum number of hours to your statement of work. Once both parties agree that the statement of work is a good representation of what is to be created, we attach a hard estimate in hours to it. This estimate determines the cost. In addition, we attach a second number, which will indicate the “maximum number of hours” we will spend on your project. This number will be 15% higher than the estimated amount. For example, if we estimate your project to take 100 hours, we will spend up to 115 hours on it. After that point, you have the option of either deferring features to future development cycles, or paying us an hourly rate to continue with development. Our intent with the “maximum number of hours” figure is to acknowledge that it's only right for us to shoulder some of the risk. If we did a poor job of capturing what it is you had in mind, we should be liable to account for that by investing extra time into your project. On the flip-side, if you see all sorts of tweaks and changes you'd like to make to what we build, it's only right for you to pay for those if they're beyond a “reasonable” buffer of 15%.

## **Bugs**

We're not perfect. This might come as a surprise to you, but code we write occasionally has bugs. It's only right that we fix bugs for you as part of the cost of development. We do our best to account for time required to do bug fixes in the initial estimate. In case we're off, though, we can always overflow into the 15% buffer. This helps to mitigate the risk for both of us. If we deliver software that requires more bug fixing than we initially anticipated, you have a way to address this by spilling into the buffer.

So, what's a bug? This is a much trickier question than it might appear, at first glance. Our experience has shown that very often a client will cite as a bug an issue that we're certain should be classified as a change request. The nice thing about the 15% buffer is that it doesn't matter. You can classify an issue as a bug or a change request. Either way, we spend up to 15% beyond the initial estimate.

Right about now you might be thinking, “But what if they send me software that's riddled with bugs?” That's certainly a possibility, but our experience has shown that it's very unlikely. We've been doing this for many years, and the number of functional bugs we discover in our software is typically relatively small. At this point it becomes a matter of trust. You simply need to trust that we're going to deliver you quality software. By the same token, we're acknowledging that problems can occur by giving you a buffer through which we can account for them.

## **Change Requests**

In software development, it's hard for any of us to have a clear picture of what's wanted. This will likely include you, as the client. Please know that it's all right for you to change your mind while we're developing something for you. You even have the right to request significant changes in the middle of a development cycle. There are certain ramifications of doing this, though, of which you should be aware.

When a change request is made during a development cycle we will first do an impact analysis. If the change is small, we might simply amend the statement of work and incorporate the change. If the requested change is significant, it may mean that we halt development, declare the current statement of

work void, and draft a new one that reflects the direction you'd like to take. In cases where we change the direction dramatically, though, there will likely be costs associated with restructuring what we've already done to allow for the changes.

Think of it like building a house. We draft an initial set of blueprints, and start building. While we're in the process of hammering away you might walk by and comment, "You know, that light fixture should really be moved down about six inches." In such a case it's probably not a big deal, so we would just take care of it, after making an amendment to the statement of work to account for the cost. Instead, if you decide one day that you'd like to move an entire wall seven feet back, the ramifications are going to be larger. This could affect the overall structural integrity of the house, so it may mean re-drafting our blueprints so that we can account for such a major alteration. It's likely doable, but it would mean creating an entirely new contract, and the costs associated with the change would be incorporated into the new document.

### **Project Management and Quality Assurance**

We do project management and quality assurance work on any project we undertake. We hope you see this as a good thing. We also hope that you understand that these activities cost us time and money. As such, you'll see two sections on your statement of work for these two items. In cases of very small projects these sections may not hold any time. In larger projects, though, we'll take a percentage of the overall development time to determine the number of hours. This will generally be about 5% for project management and around 10% for quality assurance. If you'd like, you're free to request that we reduce the amount of time we spend on testing. If we do that, though, then it increases the likelihood that you'll burn through your 15% buffer as we fix bugs that we would have caught, had we been allowed to complete our testing.

Regarding quality assurance, we do our best to ensure that the product functions as it should. Ultimately, however, you determine when it's done. This means that you should expect to perform some of your own quality assurance on our work.

### **Time-lines**

As part of the statement of work, we also include an estimated deadline whereon we expect to deliver a working version of your product. At this point we will have completed all of our code and quality assurance work. After that day, you will be given a window of time in which to perform your own testing. It's not uncommon for you to find bugs during that time, and it's important that we fix them quickly. In nearly all cases we'll have bugs fixed in less than 24 hours. We provide a fixed amount of time for you to do your testing, but please know that we're going to be flexible should you find an inordinate number of bugs. Having said that, it's not always easy to distinguish a functional bug from a desired change, which may or may not be found in the statement of work. This is another gray area, but we'll always do our best to be flexible. Take a look through the "We're Going to Be Nice to You" section if you're worried about this.

### **Payment**

Another hard lesson that we've learned from past experience is that we need to require payment at certain key milestones. We're happy to work with budgets and such, but at times it can get rough for us if we're too generous about deferring payment or spreading it out over time. If we seem like we're a bit pushy in this area, I'll apologize up front, but please understand that it's a critical aspect of our business

that we need to pay attention to.

### **We're Going to Be Nice to You**

Our clients are the lifeblood of our business. If you're unhappy with us or our work, you'll likely take your business elsewhere (and you rightfully should). We do our very best to exceed your expectations. Oftentimes this will mean granting some leeway in areas such as the maximum number of hours, time-lines, or payment. It may mean that we spend a bit of extra time on the up-front analysis, without charging you for it. Our intent isn't to nickel-and-dime you, or squeeze every penny out of you that we can. We want you to like us. We're going to be nice to you.

At the same time, we need to do what we can to ensure that our business prospers. That may mean that we need to be a bit hard-nosed about things on occasion. When we do this, please know that it's only so that we can ensure that we're going to be around for a while. That way we can continue to serve you.

On the flip-side, we hope that you'll be nice to us. If software bugs creep in, we hope you'll be patient while we work to make things better. If we get really busy, and need to push a deadline out a day or two, we hope that you'll afford us a bit of leeway as we will do with you. In the end we'll all be most successful if we treat each other fairly and kindly.