Why is Estimating So Dang Hard

Posted At : June 19, 2017 5:47 PM | Posted By : Mark Kruger
Related Categories: Business Of Development

Let's go Live to Bob and the Muse

Consider this typical interaction with a prospective customer:

- **Muse**: Tell me what you need Bob.
- **Bob**: I have an application hosted on ColdFusion 9. I want to upgrade to the latest version and move it to the cloud.
- **Muse**: That's a good choice. The latest version of ColdFusion performs splendidly in the cloud and is much easier to manage. (yes the Muse uses words like "splendidly").
- **Bob**: Great! How much will it cost me.
- **Muse**: Well.... [lengthy pause]
- **Bob**: Did I lose you? Muse... Hello...
- **Muse**: Oh I'm here. I have some questions. What kind of application is it?
- **Bob**: It's a customer portal where folks log in and use our whizzbang service.
- **Muse**: Do you use any third party APIs?
- **Bob**: Pbbbt... of course not.
- **Muse**: Excellent - how about jar files. Do you remember using any jar files in your code to access Java stuff?
- **Bob**: Jar files? What in the ham sandwich is a jar file?
- **Muse**: It's a way you package Java classes. You might use it to do some complicated Java thing that ColdFusion can't handle on its own.
- **Bob**: Uh... [Lengthy Pause]
- **Muse**: Did I lose you? Bob... hello? Maybe we'll put a pin in that one.
- **Bob**: Sorry my eyes just glazed over there for a second. Anyway, how much did you say?
- **Muse**: Well I'm not sure I have enough information to uh...
- **Bob**: Oh! and we also need to upgrade our thingy that posts reviews to Amazon and Yelp. Something about bees... apiary or something.
- **Muse**: API Library maybe?
- **Bob**: That's it!
- **Muse**: [banging head on the desk] ok how about you walk me through the app.
- **Bob**: Sure ... but... how much will it cost?

Now it's not the Muse's fault that he's having a hard time communicating. After all, if it was easy I'd be flipping burgers and still driving my '72 delta 88 from College. And before you get all up in Bob's grill, it's not his fault either. We IT natives tend to use lingo as if everyone has been exposed to the concept of an "API" or "java integration" or a "mouse".

So I'm going to let you in on a little secret. The bald truth is that development companies *usually have no idea what something will cost* even after you describe it to them thoroughly. The most accurate estimate for enhancements will always come from the original developer, and even then, they are often guessing wildly.

Muse Wisdom: Don't trust quick estimates or high levels

of certainty!

Chances are if someone is throwing numbers around without asking a lot of questions and without being nervous about "guaranteeing" the cost, you are about to enter a relationship you will regret. Think Vegas chapel and too much bourbon and you'll get the idea. To help further the conversation about estimates I offer this 3-part blog series on "The hard task of estimating". My hope is that in the end, readers who are customers and readers who are developers will both come to understand a little more about this maddeningly ticklish task that we all must do.

Estimating is hard and estimates are often unreliable because of three main areas where issues arise:

- Applications
- Developers
- Stakeholders or Customers

In this post we are going to talk about your application.

"Oh what a cute little application... Is it friendly?"

Exploration

When you take your car to a mechanic he has access to thousands of pages of documentation. If he can diagnose what's wrong he can fix or replace it *and* he knows exactly what it will cost. Why? Because he (or someone in the shop) has done that

exact job before. When I say *exact* job, I mean that someone somewhere has literally removed and replaced the hanger for an exhaust system on a 1998 Ford contour, or replaced the wiring harness on a 2001 Volvo. There are no unknowns.

Now think about your application. The person you asking to work on may not yet even know what it does. He or she will need to investigate the actual useful functionality involved. Indeed, people tend to *use* an application differently as well. Your car isn't driven from the trunk by one person while another person likes to hang out on the grill like a hood ornament. Yet an application can suffer from exactly that sort of problem.

Is it important for a developer to know how you use your application and what it does exactly? Why yes... yes it is. Most applications contain hundreds - sometimes thousands - of features, events, fields, forms, and objects. Many of these are quite important. It's not hard for a developer to imagine that your customer onboarding process is a critical piece of code. But down in the weeds, where you have perhaps forgotten, there are things lurking that are not so black and white. Suppose a developer finds a tool for importing data from a spreadsheet. Does your customer use that? Do you? Is it a one-time utility created for some specific data and long forgotten? I have seen developers spend time spec'ing out requirements to migrate or retool a utility only to have the client scratch their head and say, "Hmmm... I'm not even sure why that's there." The longer your application has been in service and the more changes (no matter how minor you think they are) have been made the more likely this case will be.

Integration

Unlike a car, your system does not exist within its own shell. 95% of modern systems (that's my estimate so don't email me) have some level of integration with other systems. Of course, an even higher percentage connect to a database server, which is its own sort of system integration. Each integration must be examined. Here are some examples drawn from the over 120 codebases in our working set:

- *Mail server, mail service* - an application creates a distribution list on the fly against a third-party mail server using .NET webservice's WSDL file to compile a stub class.
- *Payment Gateway* - anyone selling things on line is sending the data somewhere to get those payments processed. Sometimes they use Java or .NET or (of late) a REST interface.
- *JS library* - jQuery and Angular are very popular right now. What version are you using? Do you know?
- *Google Maps, GIS, or other location services*
- *Financial Data* - several of our customers use engines from quote services to get quote and chart data
- *Webtracking or advertising* - does your site use a services like AdSense or Google Analytics?
- *Scientific or Financial calculations* - One of our sites uses a complex jar file that encapsulates dozens of financial studies for stock and option analytics. These and other scientific calculations are difficult and inefficient to do in a scripting language.
- *Non-standard data sources* - MySQL, MSSQL are still ubiquitous, but you may also have one off connections to non-standard sources like a flat file, a spreadsheet or even (ugh!) an Access database.
- *Non-relational Databases* - These stand-offish applications are increasingly

popular for caching, sessions, shopping carts etc. They include things like Mongo, NoSQL and CouchDB. Using them often includes jar files and java code.
- *External Resources via HTTP/S* - Does your site reach out on the web for an RSS file? an image? A Json packet of some sort? The security requirements of TLS are sometimes a dizzying array of cans and can'ts that trip up a migration or upgrade.

I could go on but you get the idea. Each of these (and the many other possibilities) come with pitfalls and possible obstacles. In order to estimate, your developer will need to have a grasp on what exactly is going on in your application.

Evolution

Finally, if your application is like thousands of others, it has evolved over time. When you created version 1 you probably sat back and waited for the money to roll in - only to find out that version 1 is little more than something new for people to gripe about. So, you got to work and added this feature and that feature while deprecating others and enhancing still others. By the time you got to version 3 your app was finally starting to groove the way people wanted. But you also found that no matter how great your application was, you still had to have it under development constantly. The result? Your (now quite mature) application is a collection of features, false starts, new attempts and conglomerations of ideas and concepts. Perhaps you never thought about the ramifications of that process.

For one thing no one, not even you, understands *all* the things it does or contains. For another, it is very likely interconnected in ways that make changing one area problematic for some other area you don't even know about. If you hear us talk about regression testing, this is the problem we are trying to solve.

Meanwhile, given the evolving nature of your application, your architecture - your code organization and database schema - may not be suitable for the size and scope of your application. We often work with clients who created a user-permission schema - a way for users to log in and do stuff - that is too simple. They started out with an admin user and a regular user, only to find they needed intermediate users or guest users or whatever. One common project we must do is to expand the user-permission schema to make it more flexible. This retrofit job may be easy, but more typically it involves *every area* of an application.

Conclusion

As you can see it can be daunting to simply wrap your head around an application you paid for and use every day. Now imagine being Bob and approaching the Muse to ask, "what is it going to cost?" The right answer... the correct answer... the one that I should give, but that will turn away sales... is "I DON'T KNOW". Of course I can't say that, so I say, "Well let's see, if the stars are aligned and I can hire Ben Nadel to do the work for $25.00 an hour it *could* take 50 hours. On the other hand, if we find A, B, C, and D and I'm forced to use my intern it could take 600 hours. Meanwhile, I have questions...

Of course, it's not just the nuances of your application that matter to the estimate. Next, we will tackle the pitfalls of how developers approach estimates and how their personalities influence the final result.