Estimating Part 2: Quality, Quantity and Visual Blinders

Posted At : August 17, 2009 3:33 PM | Posted By : Mark Kruger
Related Categories: Business Of Development

In our first post on this topic, **Red Lobster Eyes With a Happy Meal Wallet**, we began the discussion of the difficulty involved in estimating projects for startup companies. We took the time to chat about the nature of the bright people with big ideas who choose to try something new on the Internet. Our main point was that Big Idea people often underestimate the time and money it takes to complete a project. Today we will talk about one of the principle pitfalls that arise in reaction to lack of resources.

Trading Quantity for Quality

That a start-up company is underfunded is not news to anyone. Having started a business myself I'm well aware that most of the time a start-up owner is doing what he is *able to do*, not *what he wants to do*. When a big idea person comes to me with a list of features I know I will be tasked with helping this customer see the possibilities while keeping his feet on the ground. My first goal is to create a list of the core features that are essential to the site's success. I suggest to the customer that he (or she) put his money into making these core features rock solid, thoroughly QA'ed, scalable and extendible.

Sometimes this results in a great application with foundational code that becomes the bedrock of all future development on the product (let's hope anyway). Often however, the customer focuses intently on trying to squeeze more *features* out of the budget. In other words, cutting quality in favor of quantity. In fact this is a somewhat natural outgrowth of my next point. Most big idea people really see the totality of their project or product as a set of features - not as a "system" with lots of interlocking parts. The customer's true understanding of the project centers around the "screens" that you build for him or her. This is pretty obvious when you think about it. Consider for a moment that the customer will probably *never see* most of the code you write for the project. Instead, he or she will interact with the system in the way you designed, using the screens or applications or widgets that are built for customer interaction.

The "Visual" Blinders

Because of this screen-centric view the customer may have extreme difficulty understanding how development time breaks down. This is important to you as well as the customer. When you present them with an invoice for $4,000 or $5,000 for a single feature they will have some questions about what they are getting for that amount of money.

Let's take a simple messaging system as an example. A short description of our system is as follows:

> Allow an admin user to create a message that is sent via email and/or SMS to all users, a single user, active users, inactive users or users in a particular group. The message should also be displayed on the home page when the user logs in.

Given that explanation, here is what the *customer* sees.

- A form to send out messages.
- A formatted email template.

- A new area of the member home page with a message in it.

The thing to note here is that each of these is a *visual* item. The customer does not see a diagram of web servers, data, SMTP, SMS etc. They see what they will be *using*. They have a sort of tunnel vision that boils everything down to what they see on the page.

We as *developers* might see the following tasks or hurdles:

- The need to alter the signup and profile pages to collect the users Cell number (for SMS).
- The need to implement an SMS gateway.
- The need to log/audit process in case it is necessary to resend due to a technical difficulty.
- The need to estimate the total number of potential messages for a global send. If the customer has 25 thousand users for example, we probably don't want to send all 25k into the spool at once (especially on a shared server). So we might need "plan B". This will apply if the customer even *expects* to have that many in the near future as well. Obviously we don't want to create something that works now but fails in six months.
- In the "send messages" form we will need some queries for things like status, group and individual users.
- We might possibly need search interface for selecting users. This is especially true if the number of users ranges into the thousands. We can't have them all in a single drop down right? We will need a way to tease out individual users to meet our requirements.
- We will need to display the message on the home page, but the messages that display there will depend on the user login. We will need a query that filters the messages so that they apply to the users status and group as well as individual and global.
- We will need to check on things like SPF and domain keys to make sure these messages can get through appropriately.

The thing to note here is that only a few of the things that concern the developer are *visual*. A developer sees systems, workflows and technical hurdles while the customer sees only the visual aspects of this task.

Now a messaging system like the one described is a common feature. Sure there are few unknowns, but it is the sort of thing we ColdFusion developers do over and over again for various suites of applications. So I have a good idea of exactly what such a feature might cost. The problem is that even though I might be very sure of my numbers, when I tell the customer it will take 30 hours (just as an example - there are no real requirements here so please no comments!) he might go away scratching his head. "30 hours? For a simple form and an email?"

Why the disconnect? Largely it is because of perspective. A customer with no IT background simply does not have the points of reference to digest the whole enchilada. Additionally, Big Idea people often come from sales and marketing - folks who live for visual media. They might be the only people left in the world who still like Flash splash screens and banner ads. They are simply not wired to see the system as more than the items they can touch and handle. Moreover, it is difficult for them to grasp that working out the system and workflow and data interactions is *more* time consuming than creating the forms and templates. In fact, such customers may spend a

dozen iterations tweaking the labels on the form, or asking for modifications in font sizes or styles. The same customer might never ask about speed, fault tolerance, scalability, extendibility etc.

How to Educate the Customer

Given the nature of how customers often see an application, it is important that we as developers work hard to expose the customer to the whole picture. This does not mean that the customer must be educated as to the nuances of DB indexing or why ORM is the way to go, or why jQuery is better choice than YUI. It also does not include high-handed, condescending technical jargon that is designed to awe and impress. It means that the customer needs to be exposed to some of the inner workings of developers. He or she doesn't need to fully grasp everything that is going on. But it is reassuring to be exposed to some of the complexity of the work.

Its Nice to be Included

When I was a younger man I enjoyed working on cars. I purchased a number of cars before I went to college. I would buy a fixer upper, make repairs and body work, get a cheap Earl Sheib paint job, and then resell it for (hopefully) a profit. These days when I open the hood of a car and poke around, but I don't really see a lot of things I recognize from working on my 1972 Oldsmobile 98 like the one pictured her - with its Rocket 350 V8 engine. Add some fading, a few dents, and a blue driver side door to this pic and you have the car I drove through all my years of college. Ahh... happy memories. I'm also old enough to not want to bother poking around under the hood of my car any more. So I tend to let the guys at the shop do all the work these days.

Still, when I go to the garage I do enjoy a full explanation. Maybe I just enjoy hobnobbing with folks who are not afraid of grease under their finger nails. One purpose of such chats is to *reassure myself* that the technician is competent as well as to *understand as much as possible* the work to be done. If a technician provides a thorough explanation I am more sanguine about the repairs, ***even if I don't fully understand it***. On the other hand if he is condescending or impatient with his explanation, my confidence tends to diminish (indeed I might even become a problem customer).

The same phenomenon is at work in IT projects as well - perhaps even more so. In fact, I believe that one of the things that is poorly perceived across the board in IT projects is the importance of communication to stakeholders. I really think that developers and project managers put insufficient value on the task of making sure the customer *feels good* about the project. If you are one of the folks who just said to yourself "What?? What does it matter how he 'feels' about the project??" ... listen up.

I'm talking to you.

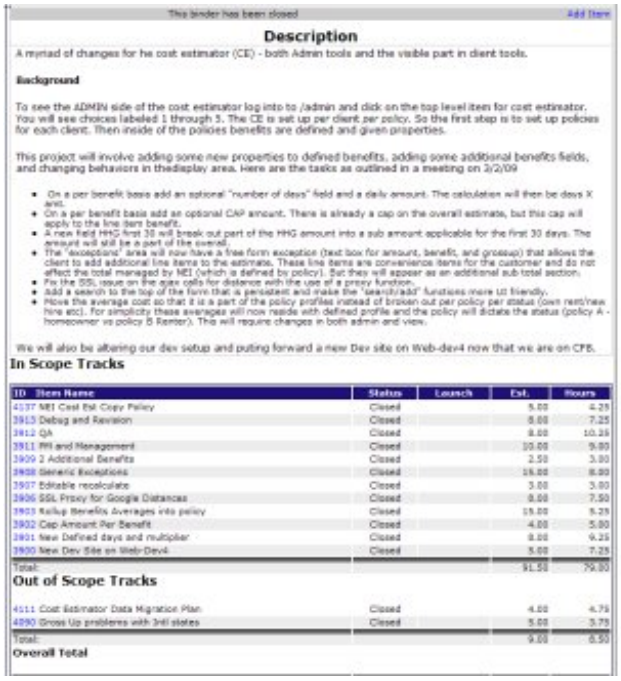
You need the support of the stakeholders to succeed. Many projects fail because they lose steam largely due to *discouragement* on the part of customers. Project stakeholders need reassurance and they need confidence in you and your team. Finding ways to build confidence can mean the success or failure of the project. It is far from a mere "politicking" task. It is an essential part of communication with your customer. Some customers might pop the hood enough to say "Yep, that's quite an engine" and others might enquire as to the purpose of "that little thingy there" - but all of them need attentive, patient communication so that they are fully aware of the tasks at hand. This communication should exist whether the customer is Laissez-faire or not.

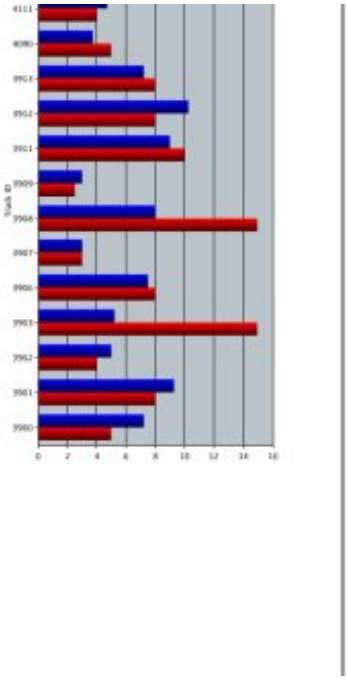
Our Approach

When estimating it is important to *take the time to describe* your tasks and requirements in a way the customer will understand - or at least sense. Descriptive requirements are important. Such requirements are not *just* written to satisfy the development team. They are also written (you might want to write this down) ***to provide substance to the tasks involved so that the customer can be reassured and informed as to the complexity of a project or feature*** . In other words, good requirements help *justify* the hours and cost. They are a customer relationship and sales tool as well as a development tool.

Every company or developer is different but here is what we do at CF Webtools. We have created a collaborative project system that we use to track all hours and tasks associated with a particular project. As developers have questions or finish items on the list they use this system to add notes to individual tasks. These notes are sent to the project team *and* to the customer in individual or digest form. Hours are tracked along with the tasks so the customer knows how much time is being used and can compare it with the estimate. The idea is not only to keep the customer involved in the process (which is an important aspect of creating acceptable deliverables), but also to demonstrate the pace and complexity of the work. It helps educate the customer about the scope of the work beyond just the visual aspects.

Sample Project Binder Screen





Of course some customers use the system effectively and appreciate the level of detail, and others rarely log in. But even those who do not log in see the notification emails and often respond via email (which is shuttled to the assigned team) with questions and comments of their own. In this way the customers see what is going on. It's not perfect and we are enhancing, revising and rethinking all the time - but it is working well for most of our customers.

Developers Need Knowledge Too

Finally, I don't want to leave the topic of communication without taking the time to address the knowledge gap that runs in the other direction - between the customer and developer. Many developers do not take adequate time to understand the customer's business. When a new customer comes to you your first task is to *understand the business* they are in. It is not to figure out how many bleeding edge technologies you can pitch them because you like to shop in the toy store. Your new manufacturer probably doesn't need a SIM game, or a 3D flex visualization of ordering trends, or a social networking site for assembly line workers. As we use to say in Sunday School, "Stop, Look and Listen". Take an interest in the business. Be curious. Step out of *your* box in the same way you want him to step out of his. In the past year I have learned new things about:

- Large scale farming
- Option trade signals
- Retail standards for markup
- FDA rules for Pet medications
- How charter jet flights are booked
- How travel agents work
- Marketing PGA tournaments
- The upscale jewelry market
- Vacation house rentals
- High school athletic programming

...as well as dozens of other items both trivial and substantial. I learned all these things through conversations with customers about projects proposed or underway. Of

course not everyone is wired this way. Part of what motivates me the acquisition of knowledge and (more importantly) wisdom. But it doesn't take an (ahem) genius to understand that the more you know about a customer's business the more likely it is that you will meet his expectations.

Conclusions

I hope this post has brought out a few basic truths. Quality can be a more cost effective way to spend project money than quantity. Customers need your communication efforts even with very technical details. And you as a developer also have a box that deserves to be stepped out of on occasion. In our final post in this series we will discuss how to account for the "hidden costs" associated with estimating.