Finding the Communication Sweet Spot

Posted At : June 19, 2005 4:17 PM | Posted By : Mark Kruger
Related Categories: Project Management

The hardest task in completing a project as a contractor or outsource company is figuring out what *really meets the needs* of your client. As a programmer this is partly because you cannot fully understand the *industry* of your customer, and partly because you don't understand the *culture* of the company. Other things can complicate the issue including lack of communication skills on your part, or the part of the company. Add to that the fact that the process of discovery is often cut short by the urgency of the project and you have a recipe for a failed project. To be effective you must *identify the true stakeholders* in the company, *learn about the company and its business* and finally *use your wits to navigate* the business culture of your client.

h4>Identifying the Stakeholders

You might be tempted to think of the person directly in charge of your project as the *only* stakeholder. In fact, they may see it that way too. There's no doubt that the IT manager in charge of your project is a stakeholder, but if your communication and discovery go only through him or her you are *not likely* to build an effective application. Sure, decision making must funnel through the manager, and he or she needs to be involved in each step of the process. Unless the manager is the target *user* however, he may not be your best source of planning information. The folks who are tasked with actually using your application will often have the best and most comprehensive input.

Now this is a slippery slope and you have to be "people savvy" to manage it. Often the person who is your chief contact has a narrow view of the company and sees his own job outside the context of the users. Some IT managers think their job is about computers, networks and software - and not about helping people leverage technology to grow and manage a company (if you are an IT manager please re-read that last sentence :). My advice is, if at all possible, meet with people who's task it will be to *use* what you are creating. Make the IT manager a collaborator in this process. Take time to get to know his users and how they do their job. Do your initial requirements, build some prototypes and have a "stakeholders" meeting. Describe your ideas to the folks who are supposed to use the application. Ask pointed questions designed to elicit input from them.

This is where you ***don't pitch***. Generally speaking, pitching is a waste of effort when it comes to IT projects. Stop worrying about all the things you *can* do for them. This is where you listen. Ask questions about people's jobs and how they work. Take a genuine interest in your clients business. Be a joyful learner!! Anytime you learn something new you have just benefited yourself both personally and professionally. The longer you work with a company the more you will understand their lingo and vocabulary - and your questions will get better. Don't be afraid to look dumb. They didn't hire you to know everything. Stop and ask them about an acronym they use, or a particular task you they do that you don't understand. You are not a manufacturer or retailer or contractor (or whatever). People love to talk about what they do for a living - listen carefully and use that knowledge to help them!

Try to figure out how things actually get done in a company. There is the way things are supposed to work, and then there's the way things actually work. Building an

application based on the theoretical way things are supposed to work will probably not result in a killer ap (though it may work as designed). Instead, absorb some of the business culture. Here's 3 structures to look for in a company (there are no doubt many more).

Brain Trust Centralization

Knowledge is power. I remember working for a company that had a legacy CRM system running on Cobol. A single individual who had been with the company for 25 years knew everything there was to know about that system. It was so old that it was difficult to acquire the knowledge necessary to modify the application. The result? That one guy could dictate how many internal processes in the company were accomplished. They just couldn't afford to lose him.

That's an extreme example, but I bet you have seen people for whom the path to security and advancement is to gather as much proprietary knowledge as possible and then share selectively. This is sometimes referred to as a *knowledge transfer* problem. If this is what you run into you may have a very difficult time accomplishing your objectives - especially if your application is designed to affect the tasks or operations of the brain trust. He or she will see it as a loss of personal power or influence. Your best bet is to collaborate *with* him or her, but sometimes the brain trust isn't the best collaborator because it goes against their modus operendi.

Passive Collaborators (Bandwagon hoppers)

In this situation a group of people has been burned by situations where they have taken initiative and been shot down for one reason or another. Or perhaps they are a concentration of passive agressives. Instead of being interested and pro-active they are watchful and super sensitive. They wait till someone takes initiative and then become followers and collaborators - carefully keeping the responsibility focused on the initiative taker. If you don't get this, consider this scenario:

Wife: want to go out to eat?
Husb: Sure, where do you want to go?
Wife: Anywhere is fine...
Husb: How about chinese?
Wife: That would be ok... but we had chinese on Tuesday...
husb: Ok... how about italian
wife: I like italian... but that restaraunt is so far....
husb: Mexican...
wife: Mexican sounds good...

The thing to note here (other than the fact that this is a wise husband (ha)) is that wife *does have* an opinion about where she wants to go to eat, but she is unwilling to face being shot down. Instead, she waits for the initiative that matches her desire and joins it. That's the way it works in a lot of projects as well. Folks will give lip service to how great the new application is, but wait for modifications or criticisms that they support to surface and then hop on the bandwagon. How to recognize it? Simple. When you initially talk to folks they have few ideas for you, but when you bring the prototype they are quick to point out the items that don't appeal to them. In short, they know what they *don't* like.

To remedy this situation you simply must be adept at "drawing out" what people really think. Find different ways to ask the same questions. Draw scenarios for them and ask

how they work through this or that problem. People will often give you a better response when asking about a specific scenario than when talking about hypothetical situations.

Glue Pot

Ever see that episode of Gilligan's Island where he forgets and leaves the lid on the glue pot - then it blows up and they all get covered with gluey feathers? Well some offices are like the glue pot. In fact, like it or not you are often called in to consult *because of* the fact that the pot is about to blow. There may be some intractable problem that they hope to solve with automation or an application - but underneath is a bubbling mess of sticky glue where management is at odds with workers, or departments are at odds with each other.

In this situation listen carefully. Try to discover the issue and don't take sides. Try to find a neutral solution that helps both sides feel like they are making headway. Technology is a wonderful thing. It can't put people's feelings back together, but it can do things like make collaboration or notification more possible or equal.

Circle the Wagons

This is often a situation where you can help a great deal, although it's not a good business structure. In this scenario, management is passive until there is a crisis, then everyone pulls together to solve a pressing need or problem. In situations like this the IT manager is often not in his position because he is an effective manager or a good analyst, but because he's a fireman. He knows how to "get things done in a pinch". One might wonder why the company suffers so many pinches. He probably has a talent for "work-around" solutions. You can help this person and this company in the short run by building applications that serve the immediate need at hand. In the long run, however, this situation will eventually lead to disaster.

The Real Point

I guess the real point here is that you have to see your task as analyzing the relationships that exist between people, as well as the ones between objects in your code. If you do that you will find that stakeholders respond to you, and transitions will be smoother.