Vendor Lock and the Devil

Posted At : February 7, 2006 11:15 AM | Posted By : Mark Kruger
Related Categories: Project Management

Reoccurring Revenue is the Holy Grail of development. Every development shop wants to find those customers who produce work for them (and hence revenue) time and again. These so called "maintenance" customers or "ongoing development" customers go by another less flattering name - the "cash cow". You can look a bit further into the future if you have customers who will regularly provide you with projects or maintenance level work. Developing a pool of these customers is an excellent way to grow your business. In fact, some applications that you build for customers are so complex they engender a form of vendor lock.

Listen Here

When a product is "live" and mission critical the company or developer that built the application is usually well aware that *they* are the ones who are best suited to support the application. This is especially true when the application was built by a single developer - which happens *surprisingly often*. When something critical needs to be done, the original developer will know where to go looking. Someone else will spend twice as long just "learning" where to make changes. This might seem like a good situation for the developer - they are needed and critical for the success of the application. Unfortunately it can often be a trap.

Being irreplaceable doesn't make you a better developer. In fact, it stifles growth and creativity. In the inner voice of every person beats the heart of a petty extortionist. Unless we are willing to seek the "better angels of our nature" (in the words of Lincoln), we all fall into the trap of gathering to ourselves such power as we are able to grasp. When it becomes evident to you as a developer that you have a customer over a barrel, you will inevitably be tempted to weild that power in ways that do not *benefit your customer*. When they call, you may be in less of a hurry to return the call. If a deadline is missed it won't bother you as much. The relationship between you and your customer will begin to deteriorate. But because you know they need you, you will find it harder to care. This is a sorry state of affairs both personally and professionally.

The Devil You Know

There's an old saying that the devil you know is better than the devil you don't. The idea is that sometimes it's easier - or at least safer - to stick with a bad situation than to change and make it worse. Your customer may find himself in this exact situation. he or she has invested a great deal of time and money in a product, and a side-effect of doing so is that he is "joined at the hip" with the existing developer or development company. Moving away from the current provider means expense, inconveniences and (not to be discounted) a new level of acrimony with the existing vendor. Some customers fear the vendor will do something vindictive like destroying code or data. Like a spouse trying to find a divorce lawyer without arousing their spouse (arousing their suspicion I should have said), the customer may put out feelers for a new development team.

Still, because moving to a *new* vendor is such a daunting task, they often stick with the old vendor and try to "work around" the relationship. When the relationship goes bad the old vendor often starts treating the customer poorly - over charging, being less responsive, and generally adding to the level of rancor. The customer too may delay payment, make unreasonable demands or constantly stir up conflict. It really is a lot like a marriage (ha). We've been involved in 3 situations like this and we were uncomfortably "in the middle". As a part of our company vision we clearly state that "We (CF Webtools) do *not* want to be the devil".

3 Tips on Not Being the Devil

So here are some tips on *not* being the devil. It seems like it would be obvious, but folks often forget themselves and yield to the extortionist within.

1. Don't Yield to the "Power" Trap

The goal of the extortionist is to extract money. He or she will see each situation in terms of a *win* or a *loss*. More "wins" for his team (the developers) and more losses for the other team (the customers) mean more *leverage* - which is another word for power. So each time the customer needs the developer, he has to come hat in hand to the developer-extortionist to ask for help. Pretty soon, a new language evolves. The customer is asking for "favors" and the developer is taking about "squeezing him" into the schedule. The customer is "pleading" with the developer. The developer, having effectively leveraged his power to a new level, feels *less* inclined to meet the needs of the customer quickly. Pretty soon the developer is a king of his own little cyber kingdom and the customer has to come as a supplicant and kiss his cyber-ring (or maybe fiber ring) to get anything done. The problem with this situation is that it spirals downward to an eventual crash. Sooner or later the "pain of change" will become less than the "pain of remaining the same" and the customer will bolt.

Instead of gathering power I recommend that you *love your customer*. Now before you go sending flowers let me explain. You *love* your customer in a business sense when you put the needs of the customer ahead of your desire to make more money. You love your customer when you analyze each situation from the standpoint of the customer needs. So the first question is "what is best for my customer". I'm not suggesting you take a vow of poverty - and this is not really a post about money. I believe that you should be paid well for what you do. I'm only saying that thinking about your clients using the language of "control" and "power" and "money" will not benefit you or the client in the long run. On the other hand, considering the clients needs as a primary motivating factor in your business will allow you *both* to succeed.

2. Write For Posterity

As a developer always keep in mind that some *other* developer will someday have to manage your code. If you are a slap-it-up-and-get-it-done type developer with no sense of structure and no compelling need to document some developer somewhere sometime is going to curse you ("may fleas infest your armpits and your breath smell like a camel!!"). Instead, recognize that you, as a developer, belong to a *community of developers.* We are all working together in the same occupation and there are connections between all of us. I'll make a deal with you. If you write with me in mind, I'll do the same for you.

3. Learn to Communicate

It's an accepted axiom among CIO's and HR managers that *relational skills* and *development skills* are dialectically opposed. If someone is a good developer (says the

HR mavin) they are unable to speak a coherent sentence and they most definitely will be unable to smile - except at special occasions like meeting William Shatner at a Star Trek convention. Obviously, knowing Java or C# or Coldfusion or nonsense like that precludes them from being able to string together any English, and unless they are allowed to speak entirely in acronyms there is no point in letting them in front of people at all.

Now I'm not buying this, but there *is* a kernel of truth here. Very often the conflicts in a developer/client relationship are the result of lack of communication. Your relational skills are important. In fact, they are *more important* than your technical skills because they mean something outside of work. Your technical skills will not help resolve disputes between your children. They won't help you tell your wife you love her (although you could program a nice flash valentine's day card...). They won't help you absorb requirements from your client stakeholders or be able to adequately explain your new fancy pants program. It's amazing how "life skills" make you a better developer. It's often not the kind of *developer* you are but the kind of *person* you are that makes the difference. Work on your *people skills*. Mediocre developers with good people skills get more business, charge a higher rate and (in general) are *happier* than gurus who have isolated themselves by a lack of relationship skills.

So... here's to happier and healthier developers, and happier and healthier relationships with our clients.