

# SOFTWARE EXPERTS



How Outsource Software Developers Conduct Business

- an insiders view to the mind of the developer.



## ZEN & MYTHS

OF

# SOFTWARE OUTSOURCING

SOFTWARE EXPERTS  
18813 TRACER DRIVE  
LUTZ FL 33549 © 2006

[Http://www.swxperts.com](http://www.swxperts.com)

## Executive Summary

In software outsourcing not everything is as it seems...

Magicians are a wonder to watch; even when you know what they are doing. Perhaps the most impressive thing is how easy they make it look. The reason it looks easy is they have practiced; they made all the mistakes before they got it right. You just don't get to see the mistake, if you are lucky.

Magicians make magic appear so easy. Competent outsourcing partners make outsourcing seem easy. So easy, in fact, you may even want to give it a try. Just like trying to do magic, it doesn't take long to figure out it is easy only when you know what you are doing.

### In many ways outsourcing success is easy!

Though when experienced partners team up it's magic! But outsourcing can rapidly turn to a comedy of errors when inexperienced partners try it and rely on flawed assumptions. Herein this document are some of the myths of successful outsourcing. These myths, mistakes and the universe of outsourcing magic are explored in a world deeply rooted in hard fact reality.

Mostly the myths will seem like common sense, yet it is not the logic or belief that make them myths. Rather it is how people act and conduct business which makes them so. These common actions people take conflict with logic. The conflict between action and logic is what creates these myths.

Myth: All culture problems arise geopolitically or regionally.	1
Myth: I want a partner.	3
Myth: 'Yes' Means "YES!"	6
Myth: Partnership means Venture Capitalist.	7
Myth: Partnership may be magic and even seem like alchemy, but it is best when it includes a large dose of mind reading!	8
Myth: All proposals are created equal.	10
Myth: All coders are created equal.	12
Myth: Similar work for other clients will succeed at our project.	14
Myth: Our software engineers are standing by!	16
Myth: Coders are interchangeable parts.	18
Myth: The contract is absolute.	20
Myth: All software belongs to me. (He who pays - owns everything.)	22
Myth: You can control costs by tightly managing.	25
Conclusion	27

For expediency this report will skip the myth that outsourcing is new by commenting: It is as old as people. Farmers, farmed; Tailors sewed and both traded goods. Today as a society we outsource almost everything; our lunch to fast food, our taxes to an accountant, our health to a doctor and our car repair to a mechanic. Yet when we consider business outsourcing we choke up, "Oh what to do? What to do?"

## **Myth: All culture problems arise geopolitically or regionally.**

Picking an outsourcing partner need not be complicated. For success it is best to not to consider national culture but clarity in communication.

Most people are not really aware of their own personalities though they may be aware of their corporate culture.

Lets use listening as an example - personalities and cultures aside, the likelihood is someone who has poor hearing might be called a poor listener, but someone else with good hearing who understands but doesn't respond might be called a bad listener, and still someone who hears but would rather talk might also be referred to as a bad listener. Are you a good listener? Of course.

Though regional or national culture is important and discussed further on in this report it is NOT the most significant cultural issue. The myth of national culture is often articulated in things like "The QRS people are good programmers." Yet, reality is: the greatest impact on success or failure is the culture of language, the culture of personalities and corporate culture. Perhaps corporate culture being the most important. To make sense of the impact of corporate culture in picking an outsourcing partner it may help by first

comparing how you pick a personal service provider.

- ❖ Consider for a moment how you found your dentist or doctor. You probably asked family, friends or coworkers for a referral. And how did you find your auto mechanic? Did you also ask your friends; perhaps your neighbors? This is a simple method isn't it? It is easy but it doesn't guarantee that you will be happy with their choice. It only improves the probability for success.

Why not? Because there are personalities and styles involved in almost every business transaction, both for the buyer and vendor.

Consider the auto mechanic as an example: Some people want to chat with their car mechanic; yet some excellent auto mechanics simply don't like to talk. If you are a talker, you might think the mechanic was born without a personality. Some people are so afraid of an auto mechanic they will go out a buy a new car every

three or four years just to avoid having to deal with a repairperson. Salesperson -- yes, but a repair person -- no. It is obvious a fearful individual should not be partnered with a silent partner and it is also unlikely that the client will ever be satisfied with the relationship.

Perhaps the least considered element of software outsourcing is compatibility of corporate cultures. Sure we call it "culture" but it really is personalities or styles. Individuals and corporations both have definitive personalities. Some companies focus on the bottom line, others focus on the employees while others still focus on their clients. Some are service oriented others are... 'on hold forever' when you call. Compatible cultures refers not just to the quality of the product produced, or the level of expectation but the combination of product, communication (dialog) and the overarching relationship.

"What does this have to do with outsourcing?" You ask.

The answer is: "Everything." Culture is the woof and weave of the fabric of successful outsourcing. What makes for a successful partnership on a personal level and a corporate or business level is an understanding and acceptance of the other partner's way of doing things. If you take the fundamental premise that your partner is there to help you reach your objectives shouldn't you find someone who is of similar mind set with similar values? Surely you would agree. But how do you do that?

## Myth: I want a partner.

Most often when we consider language in outsourcing we often think of English Vs Indian, English Vs. Russian or English Vs. Xyz. Reality however is a different thing; communications, or lack thereof, actually it comes down to English Vs. English. Almost every time it renders to the question of what do words actually mean in English? Confusion often comes into outsourcing with terms like “partnership” or “easy project”

We do a good job, so with great frequency people and companies ask to have us do outsourcing as their partner. Sometimes it is VERY successful and others it begins with a red flag warning. For example when someone says “*We are looking for a partner who will share our success*” we immediately ask for their definition of partner.

Often ‘partnership’ means they want their partner to be a Venture Capital partner.

They want their partner to deliver at or below our development cost in exchange for future profits. Business opportunity overtures are usually accompanied with terms like “*easy project*,” “*clone*

*of ABC.COM... But with these changes*” or “*shared profit*.” Partnership is a wonderful catch phrase. And we, like all vendors, look for long term relationships which will extend beyond the initial offering. The problem arises when a partnership offering is one sided and not clearly defined.

What really does partnership mean to you? Does it mean your outsourcing vendor is a virtual CIO? Does it have a budget? Or is it simply a vehicle to deliver code on time? Is your development partner really a partner or just an employee? How much of a role do you expect them to play? This may seem a judgment about what kind of a relationship works but it is not. It is a statement; Outsourcing software development works best when the relationship is defined before it begins and the roles are not switched after coding begins. Problems arise when clients say one thing and mean another. It may make sense with an example.

Honesty in communications with your software development vendor will be rewarded. Often more than you expect. But always remember that teaming is not poker.

Poker is poker and teaming is teaming -- the strategies and rewards are very different..

In one instance a client asked us to be their partner, they had an idea for a retail software package which would 'revolutionize the web' but they had no experience in business or software development. They didn't have a business plan and had a limited budget. We entered into this by accepting to do only a project plan. This consisted simply to design the specifications from what we were told. We were given a user level functional description and a mock up of a couple of the reports to be generated. Our task was to scope the project. We were to define what resources would be required to produce the application, how long and what it would cost.

We completed a complete set of milestones, estimated delivery schedule and a full development plan. We came to logger heads after we delivered the specs. Once the review was complete the client became upset because we had failed to include a licensing module to lock or unlock the software. **Licensing and marketing** were not issues discussed or were they in any of the user level detail furnished. Failing not to provide even a limited business plan, licensing was never considered by our development team. Under direction of the client we had defined the scope of work to his specifications and limited the hours we were to spend on the project -- his form of cost control.

We separated because his expectation of partnership was for us to be his marketing experts, business advisor as well as his software development experts.

So what is the point? Ask yourself this before you begin: Is your software development partner really your business partner? Sometimes it is, most of the time it is not. Will you share your complete business plan? Will you share your financials?

What voice does the partner have in the relationship? What voice do you want them to have? Do you perceive your partner as a bunch of coders who have no personality? Do you expect them to come to work every day and do their assigned tasks and leave when the clock strikes five? Are they an extension of your in house staff? Do you expect them to work overtime? Will you pay them to work overtime? If they speak up and say something like:

*"1) We think that error codes should be declared as an enumeration not as int. It is more consistent, especially since the CWdCryptography class already uses enumerations for encoding method.*

2) Some of error code names interfere with code names defined in windows headers (e.g. `ERROR_FILE_NOT_FOUND`). We would offer adding an error code prefix, say `WD_` (So `ERROR_FILE_NOT_FOUND` would become `WD_ERROR_FILE_NOT_FOUND`).

Will you embrace them as offering critical design guidance or will you be concerned they are trying to raise the project cost by project creep?

Actually, what most people mean when they say 'partner' is they want a reliable vendor.



## Myth: 'Yes' means: 'Yes!'

Regional customs and culture do have an effect on the success of an outsourcing project. Again, most often regional issues have to do with understanding the imprecise nature of English as much as they are regional. But regional issues should not be discounted. Even within the US there are regional variations. Beware however, in certain countries the word "Yes." Doesn't necessarily mean "Yes, I agree" or "Yes we will deliver on time." It means "Yes, I hear you, I understand but not necessarily agree." Or even worse for the neophyte it can also mean "Yes, we will try, but we know we can't do it, but we don't want to make you angry."

The issue here is not language per se. Say Indian Vs. English, rather it is just another permutation of the meaning of words in the English language. Our personalities and corporate cultures determine the meaning of English words to us and to your vendor.

Clarity, or rather lack of it, in usage of the English language is the biggest cause of misunderstanding, though there are clearly cultural variations which can exacerbate problems. The point is, you can reduce misunderstandings and associated problems addressing regional interpretations of language. If you want to have a good experience in offshore outsourcing ask what the meaning of the words are.

Be advised to check culture and usage; ask your partner what things mean. Remember it is your money; It is your project, therefore it is your responsibility to communicate clearly

Language remains open to interpretation even within our borders. A recent President of the USA was quoted as saying: *"It depends on what the meaning of the word 'is' is. If the--if he--if 'is' means is and never has been, that is not--that is one thing. If it means there is none, that was a completely true statement...."*

Does this statement make sense to you? It did to him.

## Myth: Partnership means Venture Capitalist.

Software developers, like ourselves, are bombarded daily with great ideas and even more **not** so great ideas. People ask vendors to be partners to develop clones of successful sites. They come to us when they have no, or little money, no business plan, no understanding of the project requirements, no understanding of what it will take to bring their project to market, no software specifications, no revenue projections and no contract. Many prospects will explain there is an opportunity to make millions and millions of dollars. Inevitably these prospects will promise to reward us with obscene riches if only we will do all the work for free, or close to it, and provide business guidance, and marketing guidance. They want us to do this gratis so long as it doesn't conflict with their view of themselves as the "Gods of new ideas." No feedback or input is allowed except for "It's the greatest idea in the heavens."

The dichotomies of these thoughts conflict when on the one hand they want an experienced and talented partner, but on the other want us to be ignorant enough not to ask for standard business back up documentation. **Software developers are not Venture Capitalists.** If software developers were venture capitalists they would demand a business plan, resumes of key people, projections, legal documents and the like before entering into any meaningful discussion

Software developers are just that -- software developers, and though the quality of work product may vary from one to another the role remains predominantly to produce functioning code.

Partnership isn't of itself a taboo word. In fact it is a beautiful thing when it works.

One prospect came to us with truly a killer app! We read his overview and all were awed; it was fantastic. He offered a partnership because he had no money. Unfortunately (or not) we made the decision to pass, not because the idea was defective, but because the plan was. He needed money to advertise, and with no money for development and no business plan we felt it would fail.

I wish he had done his homework because we would have enjoyed participating. But still we are not Venture Capitalists! And an NDA prevents us from discussing it.

## **Myth: Partnership may be magic and even seem like alchemy, but it is best when it includes a large dose of mind reading!**

Prospective clients actually say things like *"I want my site to go **BAM!**"* Or maybe they are more subtle -- *"I can't explain what I want but -- I will know when I see it!"* *"We want a clone of ABCD.com, except..."*

Herein are the pieces of myth (excuses):

- ❖ You don't know how to articulate so you use vague terms.
- ❖ You don't want to provide project detail because you are lazy. (Oh, sorry, the Politically correct term: you are too busy)
- ❖ The partner is smarter than you and can figure it out what you want.

We are not in high school; We are however, in business. We earn our living writing quality code not mind reading. But we know the excuses and the language of failure lie resident and disguised in generalities and incomplete thoughts -- what happens is this: If, and this is a very big "IF", your partner has the knowledge, business acumen and skill to ask, they will ask you the questions you should have asked yourself in order to complete the detail.

This is where the myth conflicts with reality: **when you don't provide the detail you will pay for both the development of the questionnaire and the development of the software.** (THE COST GOES UP WHEN YOU DON'T DO YOUR HOMEWORK.)

Should you make the mistake of relying on the vendor-partner to know what questions to ask, be prepared for surprises. See when it is done and you've signed off on the specs all changes cost extra. If they forgot to ask about an item or you forgot to tell about a feature or report or requirement, you will get to pay for it as an overage. In spite of this -- you may succeed though you should expect to be on the phone or on instant messaging forever as you hash out the most simple questions. Question after question will cause delays. YES! Expect delays. More on this later.

The reality or consequences of mind reading may end when there are so many questions the client gets frustrated and blames the vendor. Why? Because the vendor continuously

asked questions like: "Do you need a control panel? What is the security structure? How often do you purge accounts? When do you want to run Database synchronizing?" Additionally it places the burden of responsibility on the vendor, as was explained in the lock - unlock example previously.

It is, after all, your money. You can elect to have someone read your mind for \$10 to \$80 per hour.

There are those who would say "If you don't know where you are going you can't get there." The unfortunate reality is if you don't know where you are going you will get there, only you won't know you arrived and it may not be where you want to be.

Isn't this a lot like mind reading?

Bad partnering more often results when the definition stage is incomplete or skipped. Without a clearly defined goal or specifications it is closer to mind reading than it is code writing.

Mind reading usually concludes with the client and vendor entering the Twilight Zone of outsourcing. It is therefore best to avoid mind reading

whenever possible.

## Myth: All proposals are created equal.

When prospects arrive with vague specifications for their project (*I want a clone of...*) The vendor has two choices in responding.

- First they can bid an inclusive package price.<sup>1</sup>
- Second, they can bid an à la carte price.

In the first case the proposal is based upon the business strategy it is best to bid accurately from a full set of specs, failing that it is appropriate to bid worst case. Here the assumption is the client wants everything. This gives some wiggle room to deliver in the event there are surprises or undisclosed requirements. Often when a client has to budget, it is best to have an accurate assessment of cost up front. More importantly, bidding it to a fair expectation of anticipated requirements, cost and delivery provides both the vendor and client an accurate expectation. The vendor is able to guarantee the delivery schedule while allowing for some minor variations without upsetting good relations with the client.

The second case scenario employs a different strategy. Vendors will bid low intentionally with the knowledge that the price will rise as the client adds all the requisite features. Cost overruns and delays are expected with this strategy as the vendor and client hash out the detail. The post contract overruns is expected by the vendor even if not known by the client. The detail will be worked out over the most obvious things: *"Oh you want management reports? You didn't say you wanted reports, they weren't in the specs that will be extra. Etc. Etc."*

This strategy is no inherently evil or trickery, in fact we see this strategy for pricing around us every day. It is not unique to outsourcing. For example you can go to a restaurant and be offered this choice: Do you want a full course meal price or à la

We are offered the choice of à la carte and inclusive pricing every day.

Do you want a "Happy Meal" or A burger, fries and a soft drink?

Do you want a fly, drive, hotel package?

In software development usually the pricing packages are not as clear, but it is up to you to decide if you want a "Happy Meal" or not. Just be aware that the "Happy Meal" it is often less expensive than a lesser meal purchased à la carte.

---

<sup>1</sup> For reference, unless instructed otherwise, we at Software Experts always have our vendors bid for the worst case scenario. (We will bid to the expected actual cost and deliver to that expectation. Ala carte bids must be specified.)

carte? Complete versus à la carte is similarly deployed in software pricing. The unfortunately many first time clients do not realize they are being offered this pricing structure. That itself is the myth that an à la carte proposal is the same as a inclusive proposal.

All proposals are **NOT** the same. There is a definitive and measurable cost to success. The cost to fail is often higher especially after you factor in emotional capital, delays, frustration and the total dollars spent. The point is you usually will pay for success either way whether you get it or not.

The rule of thumb, is the lowest price vendor is likely using an à la carte structure. What often happens is when the client discovers this they want to change vendors. The problem is not as obvious as just switching say office supply vendors. When this change happens the client is usually invested heavily into an à la carte application development. The impact often is the program was developed à la carte. In other words it was developed as pieces, first the foundation and each feature added as a patch so the application becomes a patchwork instead of an integrated program. When the realization comes they will not get what they bargained for the emotional cost can be devastating. Sometimes they have to start all over from scratch other times the project cost is 50% or more higher than initially expected.

Though every client wants to get the very best deal they can they forget it is in their best interest to have the vendor be profitable. The vendor needs to be in business a) to complete the project and b) later to do upgrades and additional work.

*In proposal review it is the responsibility of the client to know which pricing schema the vendor is using and whether that schema is the best for them.*

## Myth: All coders are created equal

Many prospects wrestle with this myth in odd ways. It appears to be rooted in the false logic and fundamental misbelief which often manifests in one of two thought processes:

1. "All programs are equal. So being equal, it would make sense without a justification I understand, to go with the low bidder"
2. "All programs are **not** equal. So, it would make sense that the highest cost would equate to better quality, therefore I should go with the high bidder"

Neither of the above is an accurate representation, but still, sometimes we all get lucky. Knowing that there are differences, most people don't understand the return on investment (ROI) of software. The value in software and its ROI are related to the purpose and function of the software.

Consider that by definition, half the people in the world are below average. And consider that half the programmers in the world are also below average. Some should not be allowed near a computer. Fortunately half the programmers are above average. Your challenge is finding the ones above average.

The myth is really all about understanding value. Software is a tool, and all tools have an underlying principal of value. To make sense of the value of tools lets look at car tires.

When you need new tires you go to the shop and pick out a set. You have choices. You can buy cheap tires with a 30,000 mile rating for say \$50 each or you can buy 60,000 mile tires for \$75 each. The decision which to buy is based on the vehicle purpose and usage. If you drive 100,000 per year and the car is only 9 months old 60,000 mile tires make sense. If the car is 6 years old and you are selling it in 3 months and only drive it 8,000 miles per year then the 30,000 mile tires make sense.

Cost per tire	Usage 30,000	Usage 60,000	Usage 120,000
50	<b>50</b>	100	200
75	75	<b>75</b>	<b>150</b>

In a good, better, best scenario, if you use a tool it is usually the better or best quality produces the greater value. If it is not mission critical than RIO is less important.

Here is the rub, if the vehicle is a truck and you use it for earning a living then to go 60,000 miles the cheaper ones cost more than the more expensive tires. Software development works the same way. If it cost less to build, but takes longer to run, the cost associated with operations will manifest in higher operational labor cost and associated taxes.

When I was in High School, a neighbor offered to provide me any dentistry I needed for free. The problem was he was a plumber. I am sure he would have used a ¼ inch drill to do a root canal. Now he would likely say I deserved it, but I never took him up on his offer.

Not everyone who is willing to drill my teeth should and not everyone willing to write your code should. And a low bid should be viewed with the same skepticism of a plumber practicing dentistry.

You pay for quality whether you get it or not.

What are some of the hidden costs? Well, if you have a website delivery system which is slow, potential clients may leave if the process is cumbersome. You may need to add staff to do all the work because they are waiting for the program to respond. It may crash or not be portable to newer versions of the OS or need re compilation when the capacity is exceeded. There are many considerations which make software ROI good or bad.

Here is a simplified example of software ROI:

Proposed development days	Proposed cost	Actual time to delivery	Post delivery Bug reports & delays	Post delivery cost	Actual cost with problem resolution & delay factored
50 days	50	110	8	8	98
75 days	<b>75</b>	75	2	0	<b>75</b>

This begs the question: *“If all coders are not the same and I am not able to know enough about software to value code, how do I make a decision?”* Isn't this the same question we asked in an earlier section? *“How did you choose your dentist?”* Though you may see it as asking your friends, the reality is you asked your dentist's client(s).



## Myth: Similar work for other clients will guarantee our project.

Why is this a myth? Because looking at other sites or sample projects will not tell you the whole story. This is not to suggest you avoid looking at vendors other work as a benchmark, but you need to talk to the vendor's clients. Try to ferret out if the clients have the same set of values as you, do they use English the way you do? Ask them what worked best, what they could have changed. Ask about the relationship, because your project is different then theirs, but how they interact in the (here is that word again) partnership which is what you need to understand. Check references for the technology and expertise but always speak with people on the relationship.

Consider that it is easy for a firm to show you work which was produced by an ex-employee. Looking at only samples will not tell you if they are currently qualified, though the project you want is 80% identical of what the firm did in the past.

Frequently prospects call wanting us to 'take over' an existing project or to complete a project currently under contract

with another developer. Sometimes we can and sometimes we have to tell them it needs a complete rewrite it from scratch. Clearly it is an expensive decision when they must change vendors.

The client sometimes is one who may have asked us to bid earlier and left us because of price. Their selection process was based upon review of similar work by other vendors but there is often a gap in pricing. The comparison was 1 - price and 2 - experience on similar work. Yet they failed. Why?

They failed because they didn't consider the relationship, the people involved. How can they avoid that? Consider paying the premium for the delivery and operational guarantee.<sup>2</sup> If in the example above the 75 day vendor guaranteed his with a stop loss, not to exceed the preset price limit or better yet if they guaranteed it to be up and running in 75 days or they would take a late delivery penalty the ROI will be as predictable as your ability to value the software.

Here is a useful rule of thumb to understand how to find the appropriate partner for you: The best of a profession will congregate together naturally. The best programmers will congregate

---

<sup>2</sup> Software Experts' vendors offers on time, on point and on budget guarantees on many of our projects.

together because of the challenges. The best in any field enjoy the challenges of leading their industry. This will manifest itself in a few ways. A firm with a high turnover is a clue to a management problem. Ask about turnover. Over 10% turnover is a warning sign. It means that in 2 years you could see anywhere above 20% of the original development team gone. Anything over 10% turnover is a red flag.<sup>3</sup>

Your partner needs certifications such as Brainbench, Microsoft and Sun. These will tell you something about the quality of their code, though you should check to be sure they actually have certifications. You really want to know is what percentage of the team is certified.

**But be advised**, many inept programmers have been certified, and many skilled ones never bothered to get tested. Certification is not a requirement but 'icing on the cake'. To be sure there is good cake below the icing, ask for resumes, look at sample code, talk to their clients, etc. This will give you an understanding of their ability, but once you have identified qualified candidates it is the way they do business and communicate which will make or break a project.



<sup>3</sup> Our vendors' turnover is under 5%

## Myth: Our software engineers are standing by!

Imagine people starting a project like building the software application of their dreams. It's just like building a house, and for whatever reason -- some one gets ill, funding or shifting priorities stalls the project. Clients will return believing that even after a month of non communications the development team will be instantly plugged in and work resumed. Software engineers are not stored in vending machines.

Besides sitting on the bench effecting turnover, check the ability of the outsourcing vendor to schedule work and meet that schedule. Instant restarts are unhealthy for all the vendors clients. Why? A development partner who robs Peter's team to work on Paul's will rob Paul's for Jim's. Conversely a firm which says it will finish one project before it takes another will usually show in some delays in the proposal and drafting stages. However, **this delay will not show up in the delivery schedule** after the contract is signed. Proposal delay may be a sign of

poor management or it may be a sign of good management. In either case quick response/slow response are both a sign of how firms manage their people, projects and whether they have a bench of people ready to work or whether they are properly deployed. Your task is to find out which it is.

It's like this: Coding shops, particularly top quality development shops like Enterra, have a standing book of business. When a project stalls, the team

will be temporarily reassigned during short delays to work on other projects and return. When the delays are excessive, say a week or more, the teams are reassigned to new projects and are not available until their current assignment is complete.

This is an important distinction between quality shops and below average shops. See, if your development team is standing by waiting, they don't have much work. This means you may not get the post delivery support because the programmers or the company will not be there when you need them.

If you want to find a great restaurant in a new town look for one which has a full parking lot and people gathering at the door. Probably is the food and value keep the restaurant very busy. They have a 'waiting list'

It is similar in software outsourcing. Good value means their clients return for more work.

When people say "I can get it done in XYZ for ½ your bid." I know it's a bait and switch and they will pay the higher price anyway in delays and poor code. It's a good sign to go on the software outsourcing waiting list. A good firm will have a short one.

We always tell our clients what impact to expect when there are delays from the client side.

Look for a 'client centric' company who will take their promise to deliver quite seriously. Robbing Peter's staff to work on Paul's project doesn't do either Peter or Paul any good.



## Myth: Coders are interchangeable parts.

Occasionally clients act like their coders are machines forgetting that though in another country their coders might be well educated and highly civilized. Everyone wants the very best. All will express this during the selection stages. But after selection some clients will demand instant response from their vendor. Perhaps it is because they don't have a personal relationship with their vendor. It is easy to forget they are people and to see them as "Coder Man" by day ..."Super Coder Man" by night. But all joking aside, a good vendor relationship will be built on strong communications channels and personal interaction.

In computer programming like any other business there is a high quality of business and personal ethic which highly educated people adhere to.<sup>4</sup> Some companies personnel will see their role as a factory task i.e. 9 AM to 5 PM, while others nurture a culture of creativity and personal pride by their staff and will stay late or arrive early to meet their commitment.

Notwithstanding any companies commitment, even the best are not available 24/7/365.

To appreciate this myth in the client vendor relationship is to understand how you treat your local staff. Isn't your 'partner' just an extension of your staff? Your partner is comprised of people! Some are married and all have personal lives, just like you. Though most coders are very logical people some may not be the most diplomatic, but it is true the higher the education a person has, the more civilized. You should treat them with the same respect you would if they were in the next cube.

Coders are fully aware there are rush requirements from time to time, and moments of client panic. Many will take personal pride and accommodate these 'emergencies.' However it is essential in order to elicit their cooperation, you treat them as humans and express that message in your communications with them. For example we had one client who sent a cashier's check to his program manager (PM) because the PM had come in over a couple weekends to help get the client's application up and running. As this was a revenue generating application the client showed appreciation appropriately. Coders will bend the rules

---

<sup>4</sup> Software Experts only works and represents those who have shown a broad understanding of business ethics.

when they are treated with respect and will be inflexible when treated like an interchangeable part. Who can blame them?

If you treat your coders special you will likely get something special.



## Myth: The contract is absolute.

Most first time entrants outsourcing can be expected to make a number of assumptions regarding contracts. The first is that an abusive Non Disclosure Agreement (NDA) or Non Compete Agreement (NCA) will protect them.

We have reviewed contracts which, if we were to sign, would restrict us from developing any 'similar' project for 10 years. Some want us to give them any code we develop for other companies as part of our agreement or tell them about other projects we have done or will do during the designated period. Generally speaking, these contracts are drafted by lawyers who don't understand business; they understand law. We don't sign agreements with this kind of abusive terms. What are the differences and what makes a good or bad contract?

- A very good test for fairness is to ask your attorney *"If the relationship were reversed would you recommend I sign this contract?"* If your attorney says *"NO!"* You probably want less restrictive terms.

- Consider a partner who has the requisite skills to do your project likely learned on some similar projects in order to qualify to work on yours. This goes to qualifications and would be absurd for them to give up all future work to contract to you.

- Consider that your partner earns their living writing code not competing with you or stealing someone else's code and giving it to you. Should they offer you should run!

- Consider that in order to develop your application quickly and effectively they need to rely on libraries of macros. These macros will lie in many applications some similar some not.
- Consider that your partner is in another country and if your NDA or contract is abusive it won't even be enforceable in the US much less than enforceable in another country. And if it were enforceable would you have the resources to enforce it?
- Consider that if you spend the time finding a reliable partner -- a trustworthy partner -- a simple 2 or 3 year NDA is adequate, if for no other reason as the technology will make most any program obsolete in 4 years. Furthermore the

expiration period usually begins on the termination of work. Thus a long term working relationship could extend any NDA forever.

One thing which makes a relationship work is knowing a contract is that it is only a framework for the relationship not the relationship itself.

What every successful partner knows is the relationship will require a bending of the rules from time to time. We know there will be reasons for change, perhaps the specs weren't clear, or the client's responsible person changed. We have seen one client who had us develop the application on their server. This was not a problem for us until the client's server crashed and they didn't have backup. The implications were for reiteration of some code and a great deal of rework. In other words we had to sit down at the table and work with the client to find a compromise. Could we start over for billing? How do we handle the portions completed? Our cooperation in the rebuilding was essential.

The client was understanding and our cooperation was given. We mutually rewrote the terms of the contract specifying the deliverables and time tables we searched our systems for the builds and were able to help them recover a substantial amount of the work.

**Myth: All software belongs to me.** (He who pays -- owns everything.)

This myth goes hand in hand with the myth: the contract is absolute.

A bit of legalese here... Authorship in software is defined in the way the source code is expressed and the look and feel of screen shots. This authorship is protectable as a copyright. If an employee creates software, then the copyrights are automatically owned by the employer as a work made for hire. When source code is created by an independent contractor, then the independent contractor owns the copyright, unless properly assigned or otherwise transferred in a written agreement between the developer and the independent contractor. That transfer should be spelled out.

Use of open source applications and applets that may be imbedded into a program by a careless developer or an independent contractor may negatively impact the rights of the developer or the owner.

Back to lay English.

The use of macros and library functions, open source code, and licenses and sub licenses all interplay when developing an application particularly when it runs on a proprietary system such as Microsoft or employs another application such as Oracle.

Imagine developing an application which runs on Microsoft and utilizes Outlook as the user interface. Microsoft owns Outlook. Yet you own your application. Can you claim rights to Outlook? Can Microsoft lay claim to your product?

Ownership problems expand when utilizing open source software. Utilizing GNU libraries may transfer full development rights to use the software to you but there are very strict rules of what you can and can't do with your application once you have it completed.

One example helpful in explaining the concept of ownership is the analogy of a photograph. Say that someone takes a photograph of your face and sells you a print of the photograph. As the buyer you own the print, but not the copyright in the photograph. Even if the photograph is of your face, you don't own the copyright of the picture, yet you own 100% of the face.

Suppose you buy the photograph to use in your marketing brochures. To do so legally, you would need a license or assignment from the photographer, the photographer. Does this make sense?

If you want to see what the GNU (General Public License) is go to:  
<http://www.gnu.org/copyleft/gpl.html>

The terms of use and transfer rights are clearly spelled out.

If this isn't clear, here is a very simplified explanation of ownership. An employee of Enterra wrote this paper. Because the author is an employee, authorship rights are transferred to Enterra. That is clear. But Enterra does not own the words in the article, they just own the arrangement of the words. By law someone may quote this article and include portions of it with reference to the ownership. But the words go back into the open source lexicon. The words may be used again and again. Someone else may make their own rendition of the myths of outsourcing and reuse all the words in this article with a different arrangement and perhaps a different meaning.

In a very loose way software ownership is similar, but the devil is in the detail. Ownership of a software application is a complex area of law, far too broad for this report. If you are unsure ask your intellectual property attorney. The fact is there are many issues which effect ownership in software, and to assume **all**, and full, ownership in the finished product is transferred to the client automatically is simply a myth.

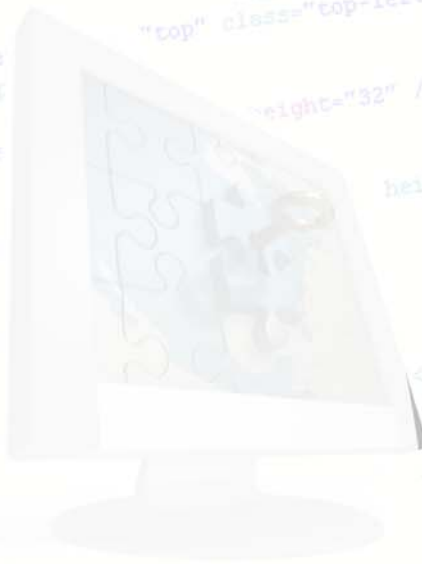
The problem of what is owned by who might seem easily remedied with a restrictive contract or NDA. But that would return us to 'the contract is absolute' myth. Actually it is helpful to understand you may own the finished product, and the employed macros and tools used in the development may be owned by another or live in libraries where they are reused.

This is a law of business as strong as the unifying force of physics:

**It is always less expensive to consult than litigate!**

This is particularly true when you ask someone to employ their libraries and tools in order that they may expedite the development of your product. Please be careful, but more importantly, be reasonable and check with your attorney.

Again, when you are concerned and in doubt, do not assume, do not bluff, consult an intellectual property attorney.



## Myth: You can control costs by tightly managing.

This myth is again prevalent in neophytes to outsourcing software development. The mistake is embodied in actions where the client demands hourly reports. The *“What did you do for 5 hours on Friday? That should only have taken 1 hour!”* are useless and demoralizing ventures.

The myth is wrong. You simply can't tightly manage the project if it is resident in someone else's shop and expect to have a successful relationship. But you **can** manage the relationship. What does this mean? Perhaps an example will help.

A client came in with a project which was partially developed by another vendor. They had perhaps \$50,000 invested in it and, though there were warning signs of danger, we ignored those warnings and agreed to help him finish his project. It was a straight forward hourly billing.

A month into it he wanted weekly reports of everyone's productivity. He wanted reports on what they worked on and what was accomplished. Then after two months he upped the requirement to daily reports.

The problem is this: clients want the best code possible. Yet many are afraid vendors will unnecessarily pad their bill. The logic is flawed. The best way to explain it is by analogy. When the plumber comes in to fix your pipes you know right up front that the plumber will buy his parts at wholesale and charge retail plus a markup for that service. It comes with the territory. You know the markup on a small fitting will be a higher percentage than on a water heater. Do you believe it is important for him to do a good job than a cheap job? Logical reasoning is simple: do you want your plumber back again to re do their work? If you pay the price for a qualified plumber and he does a good job it should be OK for him to make a couple dollars profit off the parts.

I expect my lawyer will bill me for some modest time increments in excess of the actual time. The bottom line is not whether either the lawyer, plumber or software coder overcharges a little. The real question is whether there is inherent value in the price to performance. Do you get your money's worth?

Having a performance standard is fine. Having an hourly standard is fine, but micro managing the staff on hours and

holding them to a performance standard without telling is simply demoralizing and destructive to a relationship. In this example the client was managing the code and ignoring the relationship.

The point is to have a successful outsourcing experience it is best to manage the relationship, not the project.

On the other hand, a lack of oversight management is just as bad. One client for example, wanted to structure the relationship with themselves as the project manager (PM). That is fine with us so long as they have the skills. A PM is a manager and will task and review the work of their team.

This client called us one day irate because some critical task had not been completed. It hadn't been handled because it had never been assigned. The problem was a derivative of responsibility. Who was to assign the tasks? Well it was for this client and his instructions it was clearly the PM's responsibility. He was the PM and it was identified as a line item in the contract. Apologies followed and we were back at work the next day.



## Conclusion

The magic in successful software development outsourcing is relatively simple:

- Communicate with your outsourcing vendor (partner) regularly.
- Communicate with your outsourcing vendor clearly.
- Treat your outsourcing vendor with the respect you would if they were in the next room.
- Make sure you both agree on the scope of relationship before you begin.
- Make sure you both agree on the scope of work before you begin.
- If the relationship doesn't work leave it with dignity not a fight. This extends past the current relationship, meaning don't punish your new vendor for mistakes made by the previous vendor. Rather find the right one and be a good partner and always pay your bills on time.

From a vendor's side -- we may not always want to admit it -- the reality is outsourcing software development arena is much like baseball arena. At any given time there are only a handful of 'pro teams'. There are more AAA players than pro players, more AA than AAA and more A players than any of the others. Though Offshore Experts represents only pro teams, we are not the only one. We know that not every client should team with every vendor.

From a client side -- they may not always admit it -- the reality is they may not always want to pay the price for a pro game, they may find AAA or AA more exciting in a smaller stadium. That is why we partner with emerging vendors as well as established power vendors.

There are many opportunities to succeed and many more to fail. Picking your team should be done intellectually, but if your gut says "no," you should consider listening. And don't let greed (price) be the decision maker. If it looks too good to be true it usually is.

Knowing what you want and articulating it will help you past the myths and traps. But unquestionably, the most important rule for successful partnering is to understand the software vendor can be an asset. And like all HR assets you should treat them as you would want to be treated.

We at Software Experts believe we have made a case for doing diligence in picking a partner to develop your project. We have done much of the legwork. Give us a call and let us help.

Having it done right the first time!

**That is the Zen...**

