



# Havenshire Limited

## Newsletter

July 2009 From Our  
Founder

Technical and Business Consulting You Can Trust

### Let's Keep in Touch!

**Newsletter Subscription**  
*Subscribe to our free email  
newsletter*

### Send An Inquiry or Comment

*Ask a question, send us  
feedback, tell us what your  
business needs*

### Information to Help You Prosper

**Information and  
Referrals to Bargains**  
*Information products for  
sale, and also free reports  
and referrals to special  
discounts we negotiated for  
you with our favourite  
vendors*

### Get Paid! As Much As You Deserve, And On Time

*Affordable updated ebook –  
how to get your customers to  
pay on time and in full,  
without needing collection  
agents or lawsuits (USA  
edition, very helpful if you  
sell into the States)*

### 2003 Book (USA)

*Amazon.com listing for the  
original book **Make Sure You  
Get Paid (And Other Business  
Basics)***

### Contact Us

[www.havenshire.com](http://www.havenshire.com)  
[service@havenshire.com](mailto:service@havenshire.com)

*4 Knight Street  
St. Johns  
Worcester WR2 5DB  
England (UK)*

*+44 (0)121 288 1440 phone  
+44 (0)870 123 6161 fax*

*Director: Bonnie D. Huval*

## The Wisdom of Paying Enough

Recently a manager I know at a large company complained that his company is not paying enough to its contract labor. Top management has dictated that contract labor can only be paid perhaps one third to one half as much as before the recession. People are so desperate, they sign on as contractors at those rates on the grounds that a pittance is better than nothing.

Why is my friend complaining about getting such a bargain?

***Because it is so darned expensive.***

My manager friend's group designs, builds, maintains and enhances software systems. It isn't software you could buy for a few bucks and load onto your PC. It's large, complex, custom software, running very fast to carry out transactions for which his company makes money. His group needs to be highly skilled.

It takes months to train a new worker to do this type of work properly. People willing to work for peanuts are not necessarily the sharpest tacks in the box, so they may take longer to train than brighter people who cost more. Besides, their minds aren't committed to it. They are being paid so little that they never stop looking for something with better pay.

My friend spends too much of his budget training people, because they leave as soon as they can. That's why he is complaining about the cost of the bargain top management forces onto him.

If you think about it, that bargain puts his entire company at risk by keeping an undertrained staff in charge of a revenue generating system. Some bargain!

What my friend understands, and what the top brass ignores, is the wisdom of paying enough.

When the recession ends, my friend's company will still be training new hires, over and over again, and getting by with a lot of workers who don't quite know what they're doing. But a few other companies are taking the opposite approach. Some of them are even getting good publicity for it.

In a recession, more superb talent is available in the market than usual. You can hire the very best—and you no longer have to offer sky high pay. If you pay reasonable rates instead of rock bottom and treat the new hires well, you can build loyalty to keep them as times improve. Is there a big project you've wanted to do, but couldn't find even midlevel people with time to take it on? If you need contractors or consultants, the cream of the crop are available. You can start your project now.

Guess which approach will have the better edge when the recession ends!

---

## Choose the Right Platform

Last month we had an article about choosing the right computer system. It was an introductory example.

What if you are dealing with something much larger? What if you need to build a major system to handle highly proprietary revenue generating work for your business—and you have to start from nothing?

You'll need to extend the concepts from last month's article to all your decisions. Choosing the right foundations and tools are among your most crucial decisions.

As with my parents in last month's example, hardware is probably not your starting point. Think about what your system must do.

What operating system is best suited? If downtime is intolerable, you may need a special disaster-tolerant platform. But try not to limit your thinking to the newest products.

When the World Trade Center came down in 2001, CommerzBank's headquarters were located less than a hundred yards away. CommerzBank was running its most essential banking applications on a properly configured OpenVMS wide-area cluster. It could lose a data center without a noticeable blip in service. OpenVMS is about 30 years old. With appropriate hardware and communication lines, the computers in a cluster can be up to 500 miles apart—great for disaster redundancy.

## Universities and Experience

That's an example of why you should doublecheck the opinions of your most recent hires about the operating system, programming language, database engine, and other tools to be used in building your system.

There are distinct limits to how much anyone can learn in a specific period of time, and computing is a vast field. No university can teach its computer science graduates everything about computing. For that matter, even the most experienced top notch computer expert can't know it all, either—but the experienced expert has been learning a lot longer.

(Fair warning: Some nerd talk is next, but stick with me.)

At one of my projects, the group I worked with tangled itself up in knots trying to take care of a procedural application written in C. To make matters worse, a portion of it had been an attempt to make the procedural language C behave like the object oriented language C++. It would have been clean, easy code to maintain in a higher level procedural language such as FORTRAN.

Eventually, the boss of the group told me that he was a programmer analyst not long out of college when that application was conceived. He knew C and C++ from his degree studies, so that was what he recommended. He thought C++ was great, but the company was only willing to provide C, so he personally wrote the especially ugly portion that tried to mimic C++.

He admitted those were mistakes, especially the C++ imitation. It should have been done in a higher level procedural language. Now he had to live with the budget and resource strains that came from his decision. What you need to know is that he made choices which were harder to maintain than other choices that were available.

For well designed, well built software, you can reasonably expect about 20% of its lifetime cost to go into creating it. About 80% of

the cost is maintenance and enhancement. By making a less maintainable choice, he increased the lifetime cost of the system substantially.

He shook his head. "I didn't know any better. That was what I knew from college. All of us were right out of school. Nobody else knew any better, either."

### **A Little Outside Opinion Goes A Long Way**

You can still use a young team to build your system. There are some good reasons to want youth on the team—energy, creativity, comfort with some of the latest software. Besides, if your team is young when they build the system, you have the potential to keep original expertise on hand for a long time.

Don't skip the doublechecking, though. It's worth your while to bring in a consultant to look at what you want to do and give you some advice. Even if you don't keep the consultant involved for the whole project, at least the consultant's experience can help you make sure you're going to use the right foundation.

When you're tempted to skip that step, just think about how much that shortcut can cost in the long run.