Is IT worth it?

If you're a dedicated software developer you probably read a lot. I always think it pays to read widely, get outside the software field and expose yourself to some new ideas. While for some developers outside the field means swapping Sutter's *Exceptional C++* for Hofstedter's *Gödel, Escher and Bach* some of use do venture as far as *The Economist*, and maybe the odd business book. After all, IT is itself a massive business and, if we believe the hype, can benefit real businesses.

If you have ventured into the business press lately you will have found business writers and mangers questioning the true value of IT. This has been going on since the Y2K bug, dot-com bust and telecom's crash but has picked up momentum in the last few months thanks to an article entitled *IT Doesn't matter* in the Harvard Business Review.

Now the HBR (as aficionados call it) is a rather august journal that sees itself at the heart of the business debate. Its detractors may suggest that many more copies are bought than are actually read but it still has the capacity to ignite a debate, which is just what *IT Doesn't matter* did.

What did he say?

Nicholas G. Carr is a business writer and consultant, and a past editor of HBR. So when he writes about IT he is writing from a business perspective. Before you turn the page, remember that it is business that pays us. Unless you flip-burgers by day and write Linux by night the chances are that at some point you need to consider IT from a business perspective too.

Carr's argument runs something like this. IT is really a form of infrastructure, like the telegraph or railways. During the last 20 years or so we have been building this infrastructure, not just the internet but putting PCs on peoples desk, writing word processors, CRM packages, etc. During this time there has been an advantage to getting their first. That is, the company that is first to install word processors would see a business advantage, a *competitive advantage* to use business speak.

However says Carr, that time is past. The infrastructure is now built. There is no point in you upgrading your secretaries to Pentium 19s because (a) it will cost you, and (b) it doesn't offer that much advantage over your competitor where they all have Pentium 18s.

In the process IT has become a commodity. You can buy 500g of IT like you buy 500g of cheese at the local deli. Look at the net services being promoted by Microsoft, IBM and others to see this in action. "Computing on demand" you can switch it on and off like electricity, 100Mips of computing power may come from a Linux server farm in Aberdeen or a mainframe in Mumbia.

Now, since IT is a commodity there is no differentiating factor, there is nothing unique therefore it cannot be the basis of competitive advantage. Even though IT may be essential to our operations it is not something we can use to gain an edge on our competitor. If I need 100Mips of power to run my business then so be it, I can buy it and so can my competitor, therefore it is not something that gives me an advantage. The only difference is how much each of us pay our suppliers for it, I need to drive a harder bargain with IBM than my competitor drove with EDS.

Even if you come up with some revolutionary new idea for using IT you still can't create an advantage because it is easy for anyone to copy.

Carr makes a good argument. He shows how IT has followed the same path of previous infrastructure technologies, complete with investor bubble and bust.

In conclusion he suggests we need to move "from offense to defense". Rather than spending vast sums on IT projects we need to look to manage the risks associated with IT - such as network outages, and manage IT costs more closely. Why buy new PCs when the old ones still work? Why invest in storage when as much as 70% is wasted?

Is IT really a commodity?

Undoubtedly some aspects of IT have become commodities. Hard discs, RAM chips, even PCs are really commodities even if we take a perverse interest in the seek time on a Maxor v. a Shugart drive, or whether we have SIMMs or DIMMs in our box.

It is probably also true that increasingly software is becoming a commodity. Although it is an unusual commodity that is only available for one supplier, can Microsoft Word really be a commodity word process if it is only available from on supplier? So, part of "software as commodity" debate is entwined with the Microsoft monopoly debate.

However, it is true that mail clients, and particularly web-based e-mail are pretty much a commodity. As the ASP software model spreads software starts to look more like a service than a product.

However, there is a dimension to IT that defies the commodity classification, that is intent. This point is expressed by David Fenny:

> "we encounter a unique characteristic of IT, its inherent lack of application purpose. If I explain to someone any of a range of traditional technologies - balance scales, bulldozers, or blast furnaces - the application is obvious. However, if I explain what is meant by a multi-media workstation, who knows what relevance it may have within a bank, a supermarket chain, or a government department." (Willcocks, 1997, p.xxii)

We may all have access to the same commodity hardware and software but what do we choose to do with it? Two companies can buy the same hardware and software. They can each operate in the same mail order businesses but if one company intends to simply make their operation more efficient, while the other intends to identify repeat customers and sell more to them then there is a significant different in the outcome. Of course, this increases the importance of getting your implementation and roll-out right.

The true power of IT is beyond simple automation and efficiency, that is a commodity. If we want to get the most from IT we need to use it for innovate ways and keep innovating. IT becomes a tool to help bring about change, and indeed, learn to do things better than our competition. And most importantly of all, keep innovating, changing and learning. Not that we want to change for change's sake - or for the sake of the IT - but, change for the sake of the businesses.

IT has another agenda

The role of IT has traditionally been seen as one of automation. Sometimes, when you speed things up enough you get something new. For example, you could view Amazon as a very fast form of catalogue shopping, but it is so fast that is has become a new way of shopping.

However, IT has another agenda, one of learning. Imagine being asked by one of your companies clerks, Bert, for a small application. He explains what he wants to you - so you learn something. But at the same time you ask questions "Why do you do it like that?" which forces Bert to think and learn himself.

Next you then code the application and show it to Bert. As a result you are both forced to think, together, on what it is doing, to remove incorrect assumptions and even improve the entire process. You have both learned.

Now the application is deployed Bert has some spare time on his hands. So he can follow up some of those complaints (yes, the ones he was throwing in the trash). As a result he talks to Doris and together they realise that if we could just extend the application a bit, it could help Doris and cut down on some of the mistakes.

(Unfortunately, with all this done your boss notices that he doesn't need both Bert and Doris so fires on of them. The next week he is told to cut his IT budget so he fires you too.)

The point is: IT has the power to help people learn about their business not because it provides some neat little training package, but because it helps us reflect on what we are doing and why. If handled correctly The process of introducing IT helps us remove obstacles which block our vision and encourages us to think about the bigger picture.

Not only is it bringing about learning but it brings about change. Sometime for the better and sometimes for the worse but if we simply automate what already exists then we don't see the full benefit of IT.

This is where we leave the realm of IT and consider management. If management don't want change then fine, things can stay as they are, but ultimately someone else will adopt the changes we reject and beat us in business.

If management accept change then there are two ways to go about it. One if the top-down, mandated change that we saw with the business process re-engineering (BPR) movement. This is the change that says "The consultant knows best, there shall be an IT system and this is what you shall do." This has the capacity to destroy businesses.

Alternatively, there is the more compassionate management who want to harness this change and learning for the benefit of the business. Giving Bert and Doris their new system improves the quality of their work, recognising that the people who work with the existing system probably know more about the subtleties in the process than a BPR consultant ever will.

So far, I've describes this from a business perspective. The flip side, the IT perspective is also interesting. If you try to introduce a new system without properly considering those who will use it then you will encounter problems. And if you've ever wondered why people tell you something today, and come to you tomorrow to contradict themselves it may be that in telling you they too where learning.

In both perspectives we are uncovering knowledge. Such knowledge can lie unrecognised until IT is applied to the problem, but, while the IT may be a commodity and offer no competitive advantage this is not true of knowledge. Indeed, knowledge, offers a very special resources that can be used to give companies a unique competitive advantage.

Move on up

There is another reason why software need not become a commodity. As we complete software and hardware technologies we raise out sights, we tackle bigger problems.

There was a time when developing a new computer meant developing a new OS. We still develop new OS's for but if we are building a new machine we won't need an army of developers to write an operating system. We can buy Windows of the self, or port Linux. This frees the resources (money and developers) to concentrate on new applications.

A good example of this is XML. Ten years ago all file formats where proprietary, getting data from 1-2-3 into my applications was painful. The idea of getting it in nicely tagged mark-up language would have been a joy. Problem solved.

But now we have XML, and the file format problem is solved we don't stop. EDI (electronic data interchange) is being re-invented, web-services are appearing, SOAP is being used in applications where we would never have dreamed of using CORBA or DCOM.

XML may have solved one immediate problem, but in doing so we created a thousand other opportunities for using it. Indeed, we are still learning of new applications for XML.

In short, as we commoditise one part of the software market it serves as a base to move onto the next. Again, we are learning, always trying something just beyond our reach. It doesn't matter if today's advantages are tomorrows commodities, we will have moved on to some new advantage.

What does this mean for software development teams?

On the one hand, if Carr is right there doesn't look like there is much future for software development teams. However, if we view software as the medium in which we embed our knowledge then there is a brighter future.

In this future software developers codify company knowledge. They become what Nonaka calls Knowledge Engineers. Of course, the software is not itself knowledge but it is the result of knowledge work. By changing, or not changing, the software developers become the gatekeepers of change. Choosing to accept a change request will spread one person's insights to many, refusing the same request is limiting our capacity to change.

There is a very difficult line to define here between what changes should be accepted and what should not. This is nothing new in software development but it does mean we need to move away from the myth that we can limit change. Forget the idea that if we had enough time, enough analysis and good enough people we could write down a complete specification. Forget it because the very process of writing it down will change it.

Your best analyst could work with Bert for six months and produce the perfect specification. However, as soon as people, and especially Bert, see it coded they will learn and see room for improvement.

Fortunately, software people are used to change. We love learning new languages, operating systems, and application domains. Unfortunately, we don't always recognise that other people don't relish change in quite the same way, in truth most people don't like change and feel threatened by it.

I increasingly suspect that reason IT people have a reputation for lacking social skills is simply that they are placed in the position of introducing change where people don't want it. My suspicion is that the social skills of IT people are at least average, but introducing change requires more understanding and empathy than average. Not only this, but we often work under time constraints that don't leave us time to talk through someone's problems, or sympathise with them.

Conclusion

If Carr is right, and we want to stay in the software business we have two choices. We either need to get into writing commodity packages, or we need to accept a life in maintenance.

However, I can't accept Carr's argument. I think he is exposing an over simplistic view of IT, for all the reason's I've outlined above I think he's wrong. And for all the same reasons I increasingly believe we need to view software less as IT and more as business.

References

Carr, N.G., 2003, *IT Doesn't Matter*, Harvard Business Review, May 2003. Available at www.hbr.com, also check his website http://www.nicholasgcarr.com where you will find some other responses to his ideas.

Nonaka, I., and Takeuchi, H., 1995, *The Knowledge Creating Company*, Oxford University Press

Willcocks, L., Feeny, D., and Islei, G., 1997, *Managing IT as a Strategic Resource*, McGraw-Hill.