A moral and ethical dilemma, systems that fail

by
Dr James A Robertson PrEng

"19 out of 20 ERP implementations fail to deliver what was promised" according to a Financial Mail survey published some years ago (McCleod 2003). An article published in Computer Business Review Africa a few years later quoted a Gartner executive as saying that *"Most organisations are not making better decisions than they did five years go"* (Technews 2005).

Professor Richard Nolan of Harvard University is reported as saying that *"Information Technology is the next corporate disaster waiting to happen"* (Alter 2006). The indications are that as many as 70% of corporate business system investments fail to deliver anything material at all when viewed from a commercial and strategic standpoint (Robertson 2004).

For those who are listening attentively there are a significant number of reports of massive I.T. write off's by big name corporations yet silence and the "Delete" key have become the most effective tools in disposing of failed investments that the world has ever known.

If buildings, bridges, factories or large ships failed at the rate that business information systems fail to function to specification, or fail to function outright, technologically based society in the shape that we know it today would never have reached anything approaching the level of sophistication that we know today.

One can argue that without computers we would not have reached the current levels of technological sophistication that are currently experienced, however, much of the computer software that has enabled this sophistication has been developed by specialist software engineers operating to a different set of standards to those that are applied to general commercial software.

Furthermore, much of the core technical functionality that exists in business software is fundamentally unchanged from that which existed two or even three decades ago. The average user of word processing software today probably uses the same or less functionality than that which was used a decade or more ago. The software may be considerably more functionally obese than it was then but the basic capturing of text on paper as a more sophisticated replacement for the typewriter has not changed much and

most users are making much less use of macro's and other productivity aids than they did a decade or two ago.

Compounding the dramatic failure rate of business information system investments we find ourselves confronted with wholesale forced obsolescence and apparent failure of software that is functioning entirely satisfactorily. The software industry is the only industry in the world that has successfully granted itself a license to demolish and destroy what it sells when there is no technical basis to do so. Most software is forcibly discarded after a service life of about three to seven years or less unless customers pay extortion money (license fees) in order to be entitled to purchase upgrades they often do not need that frequently are costly and time consuming to install. While many of these upgrades may include added features and benefits which some users may find of value, many contain significant repairs to latent defects in the original product.

Marketing hype suggests that old versions of software may not run on current computers yet it is a fundamental reality that last year's software will ALWAYS run on this year's hardware because this year's software cannot exist until the hardware to run it has been manufactured. So software creation always lags hardware. By logical extension of this principle, it is theoretically technically possible to run software that is many years old on new processors and, in practice, this will be found to be the case. I regularly write documents (including this article) using word processing software that is 16 years old notwithstanding the fact that I have been told by computer sales people for close to a decade that this DOS character based software will not run under Windows.

In this particular case there are some features that no longer function for valid practical reasons and which I choose to live with and there are some foibles that result from conscious choices that some human being has made to deviate from an established standard at the expense of backward compatibility. But the most interesting thing about this software is that the most recent version of this software offers a compatibility mode and keyboard option that emulates the 16 year old software a reflection of a commitment to customers that has cost the company concerned the market leadership position that it once held.

As a matter of interest I continue to use the old software because I am so familiar with the function keys that I do not have to think in order to use them and they are faster to use. The inconvenience and time wasted to learn new software is simply not warranted and the content is the same whatever tool I use.

And so we find that old software fails under new versions of the best marketed operating system not through a fundamental tendency towards

mechanical failure but through a lack of interest in maintaining consistency of standards which has the interesting effect of making the perpetrators much more profitable than they would otherwise be.

The harsh reality is that individuals and corporations who would not think twice about contacting their attorney's if confronted by failure of this magnitude in any other area of business and life generally meekly pay up to replace software that is dysfunctional at best through negligence and at worst through intentional design while the application software is actually still fully functional. In the process of doing this organizations accept that huge investments in the training of personnel are summarily trashed in a manner that is demoralizing and demotivating to any staff member who has made a real effort to develop more than minimal skills with the current version.

As a consequence with the next version staff make less effort to learn how to use advanced functionality and become more inclined to dabble.

Returning to the subject of forced obsolescence there is an interesting legal principle termed *"the right to maintain and repair"* which originated in the motor industry many years ago and which, as far as I can ascertain, forms the basis of the so-called "pirate parts" industry in the motor trade. The essence of this principle is that when one purchases a product one has the right to maintain and repair and therefore use that product for as long as one chooses to do so.

The practical implication of this principle is that if a motor manufacturer ceases to manufacture exhaust pipes for your motor car they axiomatically void their intellectual property rights to the extent that is required for you to have a third party manufacture an exhaust pipe to the original specification or such other specification as you deem appropriate. The principle says nothing about the economic or mechanical practicality of doing this, it simply states that you have the right to do it.

What is also interesting about this principle is that as far as my casual inquiries have revealed, most attorney's know of the principle but it appears that there is only one case on record relating to its application. It would appear that the principle is so intuitively sound that no one has ever seen fit to contest it since it was first adjudicated in the House of Lords many decades ago in the context of motor vehicle exhaust pipes.

What has this to do with software failure?

As far as I can see a huge amount.

Essentially, the right to maintain and repair says that if a software developer ceases to support a version of their software which you have legally purchased then you are entirely within your rights to have a third party maintain and repair that software for as long as you consider it appropriate and are willing to pay what it costs to procure this maintenance. Since the precedent appears to clearly indicate that intellectual property rights are void to the extent necessary to maintain and repair the software this seems to me to indicate that you are entirely entitled to demand the source code for no more than the reasonable cost of making a copy for your use. The only real question would appear to be whether anyone has the will and legal resources to consider taking the necessary measures to establish a precedent in the software business.

Turning to the high level of business information system implementation failure and sub-optimal outcomes we find ourselves faced with another interesting phenomenon. One of the reasons why *"19 out of 20"* implementations do not deliver what was promised, that is fail to deliver, is that much of what is promised is in the realms of human ability or even in the realms of the super human. We are promised that software will do things that only people can do and in many cases that not even people can do. I.T. marketing hype propagates beliefs in the most remarkable outcomes and creates the impression that companies that implement the software concerned will achieve levels of efficiency and effectiveness that can be achieved with no other software.

The harsh reality is that computer systems, like guns, are value inert. A gun held by someone you consider to be good and pointed at someone you consider to be bad is "good" while a gun that is pointed at you is "bad". computer software is much the same. Well designed and well implemented software used by well trained and well motivated staff is "good", otherwise it is bad (fails).

This problem is compounded by the names we use. Careful inspection of the software commonly referred to as "Enterprise Resource Planning" (E.R.P.) software will reveal that in most organizations where software with this label is deployed the software being applied in practice comprises basically the same modules as were employed in "Accounting Systems" in the late 1980's and "Management Information Systems" in the 1990's. There has been an exponential increase in the functionality that exists and accordingly there has been an exponential increase in the complexity and cost associated with implementing this software but in essence orders still get placed, suppliers still get paid, products get sold and paid for and all these transactions end up with one leg in the General Ledger.

In all cases the name of the product, supplier, customer, staff member, etc comprises the identical pattern of binary 0's and 1's that has applied for

decades to the words in question in electronic form and the numeric values of the data have also been expressed in exactly the same way for decades. So, in practical terms, there is a limit on how much more value one can add to the data and therefore one is faced with a huge conundrum as to the basis on which one can justify the real cost of trashing systems that are five or ten years old or even fifteen or twenty years old and which are getting the job done.

Yes one may well find ways of extracting value by adding new modules and new functionality that work with the existing data. And in most cases one can almost certainly release substantial value by improving the quality and the classification of the data and reimplementing the existing software but telling a customer with five year old software that they have to replace it on the grounds of obsolescence requires careful consideration.

Then there is the question of head count reduction which is frequently used to justify large system investments. The challenge is that frequently the head count reduction does not materialize and, in fact, an increase in headcount and reduction in overall efficiency is experienced. Added to this there is the moral dilemma of the ethics of eliminating jobs in a nation where unemployment is severe and job creation is supposedly a priority.

So where does this leave us?

By now you may well be thinking that I am opposed to all things that relate to computers and that what I am advocating constitutes a threat to all that any worthy computer "geek" holds dear.

Far from it :)

I have devoted most of my career to the application of computers in business. In fact I have spent the last nineteen years seeking to find ways of achieving high levels of reliability and sustainability in the design and implementation of business computer systems. Something that I refer to as *"an engineering approach"* -- an approach which aspires to have the same level of success as we take for granted from engineering structures.

You may recall the bridge failure in the USA last year which made headlines around the world within minutes of its occurrence?

Why?

Because the failure of bridges is so extremely infrequent.

Why? Because engineers do NOT design bridges to stand up -- they design them NOT to fall down.

What is the difference?

There is a fundamental difference in mind set required to design a bridge NOT to fall down as opposed to designing a bridge. The first step in this change in attitude is to recognize that "falling down" is the natural state of things generally and bridges in particular. Just like failure is the natural state of business software investments.

So, the first thing we need to do in order to achieve software success is to prevent the epidemic level of software failure and the first thing we need to do in order to achieve this objective is to acknowledge that failure is indeed epidemic and then to CHOOSE to consciously and actively work to PREVENT failure.

This requires that we understand all the factors that cause failure and work to eliminate them from our projects. This is not a mystical or magical endeavour, the factors that cause failure are mundane and mostly quite easy to recognize. They are:

1. Information technology mythology (30%)
2. Lack of executive custody and inappropriate policies (20%)
3. Lack of strategic alignment (15%)
4. Lack of an engineering approach (12%)
5. Poor data engineering (10%)
6. People / soft issues (8%)
7. Technology issues (5%)

(Robertson 2004)

What you will notice is that only 5% of what causes failure relates to technology, 95% relates to factors which have to do with things that people do and think and 30% relates to the tendency to ascribe human and super-human attributes to binary adding machines which only approximate human ability to the extent that human beings are able to interpret and anticipate applications that simulate what human beings are able to do.

To return to the position that I am advocating. Not only am I advocating preventing failure, I am advocating preventing failure in a manner that delivers high sustainable strategic value, that is value that facilitates and supports the organization to thrive.

If we look at the real cost of failed projects coupled with the high ongoing cost of sub-optimal projects, those projects that failed to deliver on a real valid and valuable business case that delivers lasting economic benefit, I submit that the real cost of the few projects that succeed is far higher than is recognized and that accordingly we should recognize the real cost of

success AND the high value of truly successful investments and then act appropriately.

Once we are clear on the real cost and real value of real success we can decide either to stabilize and expand our existing systems or to invest in new systems. Once we are clear on the real cost of a new system we may well be willing to invest heavily in our existing systems and once we do this in concert with a robust requirement for vendors to honour the principle of our right to maintain and repair our systems for as long as we desire we may well see a very different perspective on the way we operate our business systems investments be it our office automation software or be it our accounting systems or extended accounting systems (E.R.P. systems).

The essential prerequisite for this approach is a change in attitude on the part of the client and the I.T. practitioner. I would like to see the change in attitude originate from the profession, in practice it will probably be a combination of change from both sides. In its simplest and harshest form this change might look like a meeting that I facilitated on behalf of a client some years ago after they had been informed that the supplier was going to discontinue support for their version (let us call it xx) of the software and that they should accordingly prepare to spend a considerable sum of money and incur considerable business disruption upgrading to a new version which it appeared there was no strong business case to recommend.

The conversation went something like this:

Supplier: "We are discontinuing support for version xx from the end of next year so we need to start making plans for you to upgrade."

Client: "We do not want to upgrade, we want the source code for version xx and a five year support contract for version xx renewable for five years."

Supplier: "Sorry, perhaps we did not make ourselves clear, we are DISCONTINUING version xx."

Client: "There is a legal principle called *"the right to maintain and repair"* and in terms of this we want the source code and a renewable support contract.

"And we are willing to take this to court."

Supplier: "In that case we will have to speak to our principals."

A few weeks later the supplier returned with a quote to supply the source code and a draft five year support contract renewable for five years. We did

not particularly like the dollar amounts for either item but they were a lot more attractive than the real cost of scrapping the current version and provided a basis for negotiation.

I would prefer there to have been NO NEED for the conversation in the first place and for the supplier to have automatically offered to continue supporting version xx on an ongoing and open ended basis.

Your response to this suggestion may be that it is "impractical" and unreasonable to expect vendors to maintain support for "obsolete" versions at numerous sites indefinitely to which I suggest that the obvious reply is that this is done for buildings, oil refineries and numerous other large and expensive systems of greater tangible complexity and greater overall cost than business computer systems and that if we considered the true cost of trashing thousands of person hours of work implementing new systems every few years we might find that the economics would in fact favour the approach that I am advocating here very considerably.

Taking this full circle to where this discussion started, the unacceptably high level of software failure, I suggest that we now find that there is a perfectly sound economic basis to do the job properly.

In actual fact, in broad terms we know how to do the job properly.

The biggest problem is that too many people lie about the real cost of doing the job because of sales targets and short terms budgets and because there is not an awareness of the real cost of failure.

One option is for the customer to become a lot tougher, which, incidentally, requires the customer to become a lot more thorough and a lot more realistic about the real cost of doing the job right first time.

The other option is for the industry to take voluntary measures to clean up its act.

I recently encountered an outstanding example of the tougher customer approach. The customer spent a year compiling a set of three hundred reference documents together with a tightly structured contract and appointed the attorney who drafted the contract to sit on the project team. The project came in on time and under budget and met all business expectations. In fact, it came in so far under budget that the company paid material cash bonuses to all the staff involved with the project!

The thought that I would like to leave you with at this conference on I.T. Improvement is that it is time for all those who practice in the field of the art and science of business information systems and who aspire to complete

projects on time and on budget that meet customer expectations and who are willing to bite the bullet of being honest about the real costs and willing to take stands with regard to dealing with integrity and in the best long term interests of the customer to join forces to establish a peer moderated professional body that will eventually seek statutory powers to licence and moderate the practice of professional information technology practitioners in the various major fields of I.T endeavour.

REFERENCES

Alter, Allan (2006) Richard Nolan: A Committee of One's Own
Ziff Davis CIO: Insight, http://www.cioinsight.com/print_article2/0,1217,a=119427,00.asp, 13 September 2006

McLeod, Duncan (2003) The I.T. Industry, Time for a Reboot
Financial Mail, Johannesburg, 28 March 2003

Robertson, James A (2004), The Critical Factors For Information Technology Investment Success, Johannesburg, 2004

Technews (2005) Business Intelligence
net.work, The Way Business is Moving, http://www.networktimes.co.za/news.aspx?pklNewsId=17372&pklCategoryID=204, May 2005

The Author

Dr James Robertson is the founder and CEO of James A Robertson and Associates an independent specialist I.T. advisory service provider and can be contacted on ++27-(0)83-251-6644 or James@JamesARobertson.com