Part I

Blinded by Specs

1 In Search of ~~Excellence~~ the Fundamentals

Every truth passes through three stages before it is recognized. In the first it is ridiculed, in the second it is opposed, in the third it is regarded as self-evident.

(Arthur Schopenhauer, German philosopher, 1788–1860)

The more things change, the more they stay the same

Technological change is usually associated with progress. Airbags, electronics and fuel-efficient engines have made cars safer, more comfortable and more reliable. Jetliners with the latest in engine technology, materials and avionics have made flying safer, cheaper and quieter. The modern world abounds with similar examples, from consumer electronics to mobile telephones.

There is one sector in which the rate of technological change has easily surpassed the above examples by orders of magnitude, and that of course is the computer industry. It has gone from mainframe to minicomputer (remember those?) to PCs, and now to the Internet. In the enterprise, commercial computing moved from the glass house to the desktop, dropping its price-tag a hundred million-fold in the process. And all of this was achieved within the short space of 25 years, as opposed to over 100 years for cars and planes. We've probably all heard about the comparison about a Rolls-Royce costing a few dollars and getting a million miles to the gallon if it had followed the same rate of progress as computers.

So what has this new generation of faster, better and cheaper computers actually brought in terms of progress (in the workplace – this book is not about computers in the home)? Well, to start with, around 80% of employees in the average company in the developed world have a computer on their desks today, as opposed to less than 10% in the early 80s. They also use these computers for the most part of their working day, as opposed to only 1-2 hours previously. But probably most important, unlike our car drivers and plane passengers, who are essentially still performing the same activity they did a century ago, which is driving or flying from point A to point B, only with more modern technology, the vast majority of computer users in the enterprise today are doing things that were not even possible 10–20 years ago, e.g. using spreadsheets, word processors, enterprise software and the Internet to automate and transform their business processes – and even invent new ones, made possible by the new technology.

So, you might say, if that's not progress, what is? In short, where is the problem? Why are you reading this book?

Well, let's go and talk to Marina, Steven and Kevin, who work for Acme, your modern, everyday corporation, to try and find out.

A worldwide phenomenon

Marina is a Regional Sales Director in one of Acme's business units (BUs). Having worked her way up through various positions in sales and marketing, both here at Acme and at other companies, she has been through many IT projects and considers herself a typical user of information technology in the workplace. While she generally recognizes the overall benefits of IT – after all, she wouldn't be able to do her job properly without it – she and her colleagues wouldn't need much prodding to launch into a litany of ills about having to deal with an IT department that: doesn't understand them; is unresponsive; makes them fill out forms to get things done; delivers software solutions that don't correspond to the way they really work; invariably delivers them late and not properly tested – and sometimes actually wants to cross-charge them for it all! What's particularly disturbing for Marina is that based on conversations she's had with friends and colleagues from other companies - and other countries, since Marina works internationally – her own experience is nothing exceptional. She has no idea how so many different IT departments around the world can all be afflicted with the same fundamental problems.

Steven is a Project Manager in the IT department. As part of a team that provides software solutions for people like Marina, he clearly recognizes the benefits of information technology – after all, that's what his job is about. Having worked his way up from programmer analyst to consultant to project manager, he has worked on multiple projects in multiple companies and considers himself a good IT professional. However, in the very next breath he would probably tell you just what he thinks of users who: don't know what they really want; change their minds every week; never come to meetings to sign off on requirements and can't be bothered to perform user validation testing. The IT department he works in has far more work than it can handle (most of it high priority, naturally), and often ends up selecting projects based on 'decibel management', which means catering to those executives who shout the loudest, rather than on any rational decision-making process.

Kevin is the CEO. He doesn't understand or even care much about IT - it wasn't part of his generation when he was growing up, and he considers it an achievement to be even using e-mail. What he does understand though are costs, and his IT budget has been steadily increasing over the past 10 years, and is now sitting at 5% of total revenue and accounts for almost

50% of capital spend (and that's just the visible part owned by the CIO – he shudders to think of the disguised IT budgets sitting in the operational budgets of the various BUs). He's got no idea whether this is normal or not, but he does know that the company cannot function without these systems. And he can't understand why it's so difficult for the CIO to explain his cost base – where does all the money go, and why is it so difficult to calculate a ROI for all these exotic three-letter acronyms? Finally, it doesn't help that there's always one or two board members constantly complaining about IT's inadequacies. Maybe they should just get another CIO – this one doesn't seem much better than the last three ('gosh, have there been that many already?'). Or maybe just get rid of the problem and outsource all or part of IT – after all, others are doing it.

Regardless of where you work, from America to New Zealand, and regardless of what your company makes, from cars to cosmetics, you will probably have no trouble identifying yourself with Marina, Steven or Kevin above. Which is pretty scary when you come to think of it: either a goblin has cast a spell on a whole profession – or that profession is doing something fundamentally wrong.

If IT doesn't work in one company, you could justifiably say that you might have a company problem. If the same symptoms are now visible in most companies in a given sector, then you could say you've got an industry-specific problem. If IT generally still doesn't work in a whole country, then you might get away with saying – at a stretch – that there might be some dominant cultural characteristics at play which might explain why the Americans or the French or the Japanese or whoever just don't get it. But when you end up with identical symptoms in companies of all sizes and all sectors in countries across five continents, then maybe it's time to step back and re-examine conventional wisdom.

In summary, despite the amazing technological advances in IT and the proliferation of computers in the workplace, the IT department is still perceived as a combination of one or more of the following: a bunch of technical people incapable of communicating in business terms, profligate in its ways, unable to cost-justify its spending, almost always delivering late and over-budget, and finally providing unsatisfactory service to generally dissatisfied users.

In the rest of this chapter, we will show how this situation came about, and why it still exists in the twenty-first century.

How the traditional IT model started

In any new field, people naturally turn to similar or analogous activities for guidance on how the new one should work. Then over time, through trial and error, the fundamentals begin to emerge, and previously held assumptions are correspondingly validated, adjusted or rejected.

For example, the very first television shows were essentially modelled on radio, with the main novelty being that you could now see people in addition to hearing them speak or sing. Of course, it didn't take long for the industry to figure out that TV allowed you to do much more than simply move a camera into the radio studio. Another example is the auto industry, whose first cars were essentially horse-drawn carriages with the horse replaced by an engine (hence the term horsepower). Again, it didn't take long for the automobile to take on a shape of its own, driven as much by technological progress as by the ability for designers and engineers to move away from the horse-drawn paradigm.

For IT, building systems was initially modelled on the construction industry, with the IT department the equivalent of a contractor who is supposed to deliver systems on schedule, within budget and to spec. Unfortunately, since building software has little to do with building houses, despite appearances, this analogy led us into a trap, which we will now examine.

The construction industry trap

Drawing on the construction industry, the IT department would develop systems for the rest of the business through a standard client–vendor relationship, based on a contractually signed-off requirements specifications document. This would then drive a sequential 'waterfall' method, with its strict linear approach from analysis, design, development and testing through to implementation, with each phase performed by different teams of specialists.

This proved to be a non-starter, and generally remains so to this day. You can specify requirements for a house because the desired outcome is relatively easy to conceive and visualize. You can then have it built 'to spec' by a vendor because the corresponding specifications (weights, dimensions and forces) cover standard mechanical components (beams, widgets, tiles) and are applied to the hard sciences (physics, engineering and mathematics) to produce relatively predictable results. In the construction industry, you can therefore separate the design phase (which constitutes on average less than 20% of the total effort) from the construction phase (which accounts for at least 80%) and have them done by different teams. You are also spared the burden of testing – after all, once you've calculated the maximum allowable stress for a beam based on the force of gravity and the strength of the materials you are using, then you can rest assured that it's not going to collapse.

Human behaviour however, which is what most business processes are about, is another matter altogether. You cannot come even close to fully and accurately specifying requirements because it is not easy to imagine or visualize the final outcome, since you are usually trying to do something you haven't done before. Plus, the business is constantly changing, which makes it a moving target anyway. A team of programmers will then try and convert these imperfect specifications into a workable product using the 'soft sciences' of programming logic and software configuration. When building software, it is therefore very difficult to separate the design phase (which can account for up to 80% of the total effort) from the construction phase (which only accounts for 20%). Not only that, but you now also have to do a lot of testing to ensure it all hangs together.

For software development therefore, the final product will necessarily be based on interpretation and assumptions, and while it might correspond to documented requirements, it stands very little chance of corresponding to actual requirements.

The free lunch trap

In parallel to the above, organizations fell into another trap concerning the economics of supply and demand, and trying to establish who the payer for all these new systems would be. In the beginning, the regulation of supply and demand was not a big issue because the technology was so expensive and clunky, and the concept of computers in the enterprise so new anyway, that its initial use was limited to things like accounting and payroll.

However, as computer technology progressed and became more affordable, corporate IT became more heterogeneous, with mainframes in the 1960s being joined by minicomputers in the 1970s. Departmental IT became possible, with cheaper minicomputers beginning to take root and software vendors appearing, thus increasing the options available to the business. By the time we get to the 1980s, decentralization was in full swing and the first wave of – often evangelical – microcomputer users started sensing that their time had come.

With this evolution from the sixties to the eighties came vastly increased possibilities for the use of information technology, and the initial driver for developing systems moved from mere cost savings to increasing revenue and even competitive differentiation. In short, more and more parts of the business were beginning to ask for more and more things from IT.

The upshot of all this was the inversion of the supply and demand curves, i.e. the demand for IT products and services had by now significantly outstripped the IT department's ability to supply more than a fraction of it. In any case, the IT department was no longer the sole supplier – there were software package vendors, plus lots of consulting companies more than willing to jump in and build software for departments with money to spend.

The regulation of supply and demand therefore became a key requirement. The economy has interest rates and pricing to help regulate supply and demand, but IT was not to be so lucky. Most IT budgets are centrally funded as a corporate cost centre. For example, a Forrester survey (*The State of IT Governance in Europe*, 28th Sept 2005) of 518 European senior IT and business decision-makers found that two-thirds of companies (40% of which had revenues greater than €1b) fund their IT as a cost centre out of the corporate budget. The same is true of similarly sized American companies, though slightly less so, with 60% of companies funding their IT centrally, as opposed to two-thirds for the Europeans. In another survey, *The State of the CIO 2006* on www.cio.com, 60% of all IT organizations control their spending centrally.

In the absence of an adequate pricing mechanism, IT became essentially ‘free’ for users, who could ask for almost anything (inevitably high priority, and with or without a serious cost/benefit analysis), which IT would then have to pay for and deliver – with expectations of 100% perfect service levels to boot.

Even when some sort of chargeback mechanism existed (usually in the form of annual cost allocations), it rarely had the desired effect of regulating demand, for a number of reasons. Firstly, because allocations are usually buried in annual overhead, they are not adequately communicated to the actual users whose behaviour they are supposed to influence - indeed, they might not even be aware that they are paying for IT. Secondly, they were often calculated based on some ‘voodoo formula’, which meant that clients usually didn’t know what they were really being billed for. For example, the IT department of a major financial services company which was implementing a time-tracking system actively resisted cost allocations based on actual costs – which it could now readily obtain from the new system – presumably because it feared that such transparency would reveal the shaky foundation on which its voodoo formula was based. Finally, it was very difficult for IT to explain to their clients what business benefits they were getting in return, so the net result was inevitably conflictual. In the worst-case scenario, the law of unintended consequences applied and chargebacks actually had the effect of *generating* demand, the reasoning being that ‘since we’re going to pay for it anyway, we might as well ask for as much as possible’.

IT therefore ended up as a black hole in which the rest of the business could pour work requests at will, and whose volume would always exceed its capacity to deliver. In *The State of the CIO 2006* survey on www.cio.com, CIOs rated ‘the overwhelming backlog of requests and proposals’ as their biggest barrier to job effectiveness. Like government-funded healthcare in most of the Western world, when the users are not the payers, it becomes difficult to regulate supply and demand.

Houses of ill repute

At this stage let us introduce the notion of a business model, which at its simplest describes how a company builds and sells its products, at what costs and margins, and how it interacts with its customers. The construction industry trap and the free lunch trap above had far-reaching consequences on the IT business model. The first one ensured that IT would rarely be able to deliver systems that really met business requirements, thereby setting itself up for one or more cycles of 'corrective maintenance' (an oxymoron really). The second one ensured that users would always be fundamentally dissatisfied with IT, because if something desirable is essentially free, then by definition they will end up asking for more than what can be physically and economically delivered.

Any one of these two factors by itself was a big enough challenge. But the two combined was to prove extremely damaging over time, and gave rise to a number of other ills which, though at first sight seem unrelated, can ultimately all be traced back to one or both of these factors.

One of the most visible results was an adversarial relationship between IT and the rest of the business (Users – 'not only was it delivered late, it's not really what we asked for!'. IT – 'we delivered to spec – they don't know what they want!'). Needless to say, this provided a fertile ground for vendors and consultants to hype new concepts and technologies, with the implicit message that they could probably do it better and quicker than the IT department – thereby further driving a wedge between the two.

Equally visible, especially for the CFO, was the inability to track costs and benefits:

- Instead of properly quantifying and tracking the life-cycle costs for each of the systems it delivers, IT usually gets away with simply measuring aggregate costs for infrastructure (hardware, networking, etc.) and activities (development, maintenance, help desk, etc.). And this doesn't even include the non-IT costs on the business side from the people who manage, coordinate and support applications and their underlying data.

- On the business side, users are either unable or unwilling to properly quantify and track the business benefits of the systems they use, e.g. order cycle time, sales cost per order or first call resolution rate. Amazingly, some business users even expect IT to do this – which would be the equivalent of an airline asking the maintenance teams in the hangars to track the business benefits of the aircraft it uses. The main focus of business benefits is to get a project launched – usually based on a subjective business case – after which there is little or no incentive to check that the benefits actually materialize as planned. Which in a way confirms the subjectivity of the business case – nobody bothers verifying the numbers because they know them to be suspect anyway.

Being unable to track costs and benefits means that we cannot calculate ROI (though as we'll see later, the notion of ROI in the strictly financial sense of the term is not particularly well-suited to IT, and needs to be replaced with other criteria).

From an internal operational perspective, IT is characterized by a general inability to properly manage – when managed at all – basic processes like capturing and prioritizing demand, allocating resources based on business objectives, monitoring the performance of work delivery, and providing general operational reporting on what's going in at one end, what's coming out at the other, and what's happening in between. Any properly run business has to be able to do this to manage customers, orders, production and delivery, and to generally anticipate events rather than be driven by them. IT generally has very little understanding of its demand and supply chains, and would have a hard time being able to answer fundamental questions like 'what is currently in the pipe?' or 'what do we have to deliver over the next 6 months?' or 'what is our current and projected resource utilization rate?'

And finally, the deadly combination of all the above, which is an inability for a department, whose budget can be anywhere from 2-10% of total revenue (and even more for IT-intensive sectors like banking, insurance and telecommunications) and account for up to 50% of capital spend, to actually demonstrate how it is contributing to business objectives.

A business problem rather than an IT problem

That companies turned to the construction industry as a starting point for the IT business model is quite understandable, given the many similarities on the surface ('So you want to automate your claims processing? Tell me exactly what you want and I'll build it for you'). That companies failed to anticipate how decentralized computing would result in demand outstripping supply and how it would impact the underlying economics is also understandable – it would be unreasonable to expect a mainly technical organization to have that level of financial and economic foresight.

What will, however, remain one of the great mysteries of the corporate era is why the many years – nay, decades – of painful experience did not lead to a questioning of both the previously held assumptions. As explained in the introduction, if success is the logical result of experience, then given the above list of ills, IT should be one of the most successful sectors out there! But instead, some sort of organizational insanity took over, as companies ended up doing essentially the same things over and over again and expecting different results.

One of the most common explanations put forward to explain IT's difficulties is the sheer rate of technological change. Whereas other industries with longer innovation cycles have lots of time to get used to new technologies, IT, as the argument goes, has to cope with product cycles of 18–24 months, and by definition will always be behind the quality curve. However, a closer look will reveal this argument to be flawed. Before the arrival of the PC in the eighties, technological change in the IT department was limited to better operating systems running on ever faster and smaller hardware. But ultimately your average IT department was characterized by technological stability in the form of a particular hardware vendor (IBM, DEC, Data General...) whose computers ran a particular language (COBOL, PL/I, RPG...). And yet this technological stability did not result in more successful IT – the problems were essentially the same. Even today, if some benevolent technological dictator could somehow wave a magic wand and freeze all technological change, it would not change the fundamental problems which concern IT's ability to deliver reliable solutions in acceptable time frames, at acceptable costs and with clear business benefits.

In the absence of a valid technology argument, this situation is ultimately attributed to some sort of professional deficiency on the part of the IT department, the usual suspects being lack of mastery of some new tool or technology, poor project management or non-compliance with one of many so-called best-practice methodologies.

This professional deficiency view is sometimes taken one step further by business users who wonder why the products and services they get from IT can't be as sexy and reliable as the 'consumer IT' they get from the likes of Google and Microsoft. The comparison is unfair. Firstly, Microsoft, Google *et al.* can produce slick, reliable products from a user perspective, and can afford to spend millions (billions?) in R&D and product development because their market is the whole world. Our humble IT departments have limited budgets and build systems for a market of one. If only on this basis we shouldn't be comparing consumer IT and corporate IT. Secondly, the commercial business model used by consumer IT is built on a solid foundation of a few centuries and is working pretty well, while the same cannot be said of the IT business model, which is barely 50 years old.

At the end of the day, therefore, it is none of the above. Rather, it is the logical result of a poor business model, supported by the twin pillars of an unworkable client–vendor relationship, which assumes that building systems is like building houses, and a pricing mechanism (or lack thereof), which can only be described as an economic aberration. This ultimately explains why the monumental technological progress of the past 50 years has had little real impact on user–IT relationships and on the ability of IT to demonstrate its contribution to the business.

Consider the following. If a marketing director spends a million dollars on an unsuccessful marketing campaign, he wouldn't dream of complaining to the CEO that his people didn't deliver, or that the outside agency was unable to give him what he wanted – with a bit of luck, the CEO wouldn't even know about it. If, however, the same marketing director spends a million dollars on a new CRM system that fails to deliver, he would have no qualms about complaining to the CEO about the deficiencies of the IT Department, which was unable to give him what he wanted. And the CEO would probably commiserate with him and pick up the phone to summon the CIO, or have it as an agenda item at the next board meeting. And the CIO, when eventually confronted with the subject, would probably defend his organization's performance by demonstrating his compliance to the same flawed business model responsible for the mess in the first place.

IT and original sin

When you have a house built, whatever gets delivered will be pretty much as specified in the architectural plans. And once you start living in it, it's hardly going to collapse, or part of it become unusable. Nor will you end up paying five times the initial cost of your house in maintenance and repair over the first 5–10 years. The technical and business risks associated with having a house built are therefore close to zero, which explains why we all have no problem taking out a mortgage, and don't lose any sleep while the house is being built.

Unfortunately, the same is not true of an IT application, which carries a high degree of business and technological risk right from the word go, whose usefulness only really becomes apparent once you start using it, and whose costs and benefits continue to evolve significantly over time. Likening IT to the construction industry and building a business model around it therefore got a whole profession off to a wrong start.

Like Christian theology, which holds that the descendants of Adam and Eve were born into original sin as a result of eating the forbidden fruit, this unworkable business model can be likened to a sort of 'original sin', which condemned all successive generations of IT professionals to committing the same mistakes. (Note that whereas Adam and Eve's descendants were at least supposedly aware of the original sin they were born into and could therefore hope to do something about it, the descendants of IT don't seem to be aware of theirs...)

In the final analysis, for IT to work, it's not about technology. It's about a sound business model supported by (i) basic economics and (ii) an understanding of the realities of building systems, which bears little relationship to other areas of economic activity, despite the apparent similarities which lead to frequent but erroneous comparisons. If I had a dollar or a euro for every time I heard or read about the analogy between IT and building houses (or building bridges, another common comparison), I'd already be a rich man and wouldn't have to be writing this book...

No sacred cows

The fundamental error of reasoning in the traditional IT business model is to assume that software can be conceived upfront like a house, and subsequently scoped, spec'd and signed off for commitment by both client and vendor – and that the documented business benefits will start flowing once the solution has been delivered.

If you're able to accept this fundamental error, then suddenly you can view a whole lot of things which we normally take for granted in an entirely different light. For example, the following statements would no longer appear outlandish (for many readers some of them probably never were outlandish – just difficult to justify under the traditional model):

- You can't ask users to define requirements and specifications to be contractually signed off and cast in stone.

- You can't give such requirements and specifications to an IT department (or a vendor) and ask it to define a budget and a project plan to be cast in stone, and expect it to contractually meet them.

- For an IT project, nothing is cast in stone – the business environment, costs, benefits, schedules and risk can and will change, both during the project and after delivery.

- Whatever is initially delivered on day one can never totally correspond to actual requirements, and will have to be continuously reworked and enhanced over many years in the form of new releases and versions. There is therefore is no such thing as a 'project end date'. Rather, the delivery date of an application represents the *first* major milestone, after which a lot of work still remains to be done.

- Strictly financial ROI is not a good criterion for selecting IT projects and later evaluating their performance. IT funding should not be subjected to the same financial considerations as other business investments like plant, property and equipment.

- While any IT department would stand to gain by emulating certain practices of external service providers (ESPs), running IT like a business from a financial P&L perspective will not address IT's fundamental problems.

- Outsourcing (a standard client–vendor relationship) has its limits, and is only really applicable to running mature production applications. Additionally outsourcing applications development will not address IT's fundamental problems; this can only be successfully achieved with an internal IT department.

In order to understand why the above statements are not as provocative as they might appear at first sight, we first need to understand how the current, 'wrong' business model works, and where it falls down. This will then enable us to lay the foundations for the 'right' business model and propose ways of implementing it.