Reaching to the stars with IT projects

Managing out of sight complexity…

Computers solve problems brilliantly; you only have to look at the phenomenal success of Amazon, eBay, Google, Facebook, Twitter or email to know this. These successes make us think: if we have a problem, computers will make the problem disappear.

Yet all is not well. The aborted UK National Health Service computerisation project was the world's largest civilian computer project, yet even with its unheard of level of investment it wasn't successful. A moment's thought brings to mind many large IT project failures, delays and abandonments in every country.

What's going wrong? What can be done? The problems we're trying to solve are complex, and computer programs are enormously complex, easily the most complicated things ever made. All of that complexity is out of sight.

We don't know what we don't know, and while we're planning a system we want to use, our imagination is very narrow – it only sees what it sees. Yet if the system is to work well, it must handle even cases we can't see. Imagine watching a film, say Star Trek, and being impressed with the technology – but if the camera pans back, you'd immediately notice it is a film set, not a real spaceship. It may look impressive, but it can't cope with any deviation from the film's script.

We are often over-confident that our plans are going to work as well as they do in our imaginations. It is therefore very important to use designers with the formal skills to 'pan back' to design programs to work reliably.

Unfortunately, though, it is far easier to make things look right superficially: like producing a Star Trek film without understanding how spaceships work, let alone being able to fly one. You can get away with it in cinemas, and it plays to our blind spots, but reality isn't fooled: the spaceship in Star Trek will never fly, no matter how convincing it looks.

It is particularly hard to think through how one planned feature interacts with another. It seems easy to add features, because computers can do all sorts of things. But for a reliable system, it is important to carefully work through the interactions of each feature with the others. Each feature multiplies the number of interactions, and makes it important to use mathematical design techniques, not imagination.

Suppose your organisation has some rules. However complex those rules are, a computer can automate them. More rules are easily added. Yet your rules may not make perfect sense without human interpretation. Sorting out an organisation's operating procedures is more important than automating them. Indeed, computerising just makes organisational problems visible faster; it is misdirection to blame 'IT failure' for poor organisational (or national) planning.

Consider forms. People make hidden fixes that seem to follow the rules. But as soon as a computer is in charge, it becomes obvious that people don't follow the rules exactly. In fact, nobody filled in forms properly before, but people knew what was meant: bad rules were easy to work around. (I've just filled in a paper form, and 'back dated' it – a white lie a computer might not allow.)

Your mind's eye can easily imagine beautiful things that won't work properly in the real world

When computerised, many essential workarounds no longer work.

We get excited with things like iPads and Facebook because they are fun. But pause. Very few businesses need to run like this; there is a difference between fun and effectiveness. The things Twitter is good at didn't exist before Twitter. Twitter didn't solve any known organisational problem; it created a new way of doing things, and it is very good at that. Just because we love computers doesn't mean they can solve existing problems reliably. iPad is a consumer product and it is seductive; that doesn't mean it is any good at anything else. Thus, we continually confuse 'new' with 'effective', and the newer the idea is, the less experience anybody has with how it will work in practice.

Many decisions to use computers aim to reduce costs. For example, if we have fewer administrators in the finance department, we save money. However, the work has to be done somewhere, and now it has moved from finance (where it had a clear cost) into additional work for everybody (where nobody knows what its cost is). If everybody across an

organisation spends an extra few minutes a day using a computer to enter data, how does that cost compare with the saved salaries in finance? Management thinks computers are saving costs, yet the organisation is less efficient, because the costs are now externalised from the centre that used to monitor them.

Human error is an unavoidable fact of life. Everybody makes slips sooner or later, and many slips go unnoticed, or the teams we work in correct them. Crucially, then, management thinks we don't make slips. So when a system is specified, it is on the mistaken basis that errors never happen. They do, but now they have immediate consequences, possibly disastrously.

It is tempting to bring in computer solutions all at once, the so-called 'big bang'. At 06:00 Monday morning the new system is brought live to do everything. At 09:05 after people have started using it...guess what? It is already showing problems. At 09:10 the helpdesk is overwhelmed. At 09:15 the organisation grinds to a halt. At 10:00 the problems are on the national news.

Suppliers tend to hide behind 'warranties' that absolve them of responsibility for the system working well. They shift blame for problems onto users (read any software warranty). You will be on the news, not your suppliers. And since most suppliers work like this, organisations are forced to accept restrictive contracts. Thus, suppliers run a market where there is no economic pressure for better products.

It is easy to ignore tried-and-tested technology, such as pencil and paper, which are mobile, cheap, and reliable. Where pencil and paper are good, they outclass computers. Don't overlook other solutions, particularly ones where computers work with other technologies rather than ousting them. Don't computerise 100% when parts of the job are better done other ways. Not being over-ambitious means that if the computer fails (and it will), the old ways are still working and people can fall back on them.

A new system should be brought in gradually by triage, and each feature tested to see how it works with people really doing their jobs. This is called user centred design (UCD); it is recognised as best practice (there is an international standard, ISO13047, on it). As people use the system, issues are then continually fixed. Real users will have powerful insights into how to make it even better (very successful systems, such as Excel and Facebook, can be extended directly by their users). As a bonus, these early users learn that the organisation is interested in making the system better – so when it goes live, their first calls will be to the developers, not to the national media.

Let us return to the UK health service system. Computers are very good at automating predictable work, as happens in telecoms, inventory, bank accounts. But in healthcare, every patient is different, records are inaccurate, every speciality is different, every clinician is a law to themselves, and hospital procedures like 'handover' add layers of complexity and opportunities for error. The magic doesn't work. When you go to a hospital now, half the nurses are working behind computers. Whatever other problems computers are causing, the health system has effectively lost a good fraction of its qualified staff.

Summary

Although we landed man on the moon, of course we don't say all transport should be done by rocket; we are far more discerning. Clearly, bicycles, cars, lorries, trains, aircraft, have different niches, even though all are powered by engines (bicycles are powered by human engines). We rarely say, 'we need an engine' (let alone a rocket); we say we need a car or a bus, because we understand the differences between them. With computers, we are years behind, and we haven't yet got any such discernment.

We never know exactly what we are doing when we build and introduce new systems. We don't even know what the problems are. We need to be humble: start new projects with a sense that we don't know what's best, and as the system is introduced gradually, we should do experiments continually to improve it. After all, a system like Facebook wasn't delivered working beautifully first time; it was iteratively improved over years. Today's Facebook would be unrecognisable to the first adopters. We should expect the same of anything else: the first systems should be designed to have a small impact, and real experience should be planned to help make systems better and better. The correct phrase for this process is user-centred design (UCD); it is part of the science of human-computer interaction (HCI). Look out for it, and make good use of HCI experts in your next project.

Further reading

Heffernan M, 'Wilful Blindness: Why We Ignore the Obvious at Our Peril', Simon & Schuster, 2011

Landauer T, 'The Trouble with Computers: Usefulness, Usability and Productivity', MIT Press, 1996

Tavris C and Aronson E, 'Mistakes Were Made (But Not by Me): Why We Justify Foolish Beliefs, Bad Decisions and Hurtful Acts', Pinter & Martin, 2008

Thimbleby H, 'Press On', MIT Press, 2007

Wastell D, 'Managers as Designers in the Public Services', Triarchy Press, 2011

Swansea University
Prifysgol Abertawe

Professor Harold Thimbleby
Director

Future Interaction Technology Lab
Swansea University
Swansea SA2 8PP

Tel: +44 (0)1792 602299

harold@thimbleby.net
www.fitlab.eu
www.harold.thimbleby.net