KEEP IT SIMPLE, STUPID

By Richard Drake

I wouldn't say Tom Gilb is a vain man. But in 1994 I had a feeling that the respected international consultant might be tickled by something in the newly-released book from Microsoft, *Code Complete*. The next time Gilb popped over to Objective, the company I co-founded that he had influenced so greatly, I had my chance. I knew I'd struck exactly the chord I wanted – and that Tom had fewer issues with his dignity than many lesser men – when his first reaction was to ask where the photocopier was.

Tom has stood for a truly simple approach to software development longer than anyone else. His thoughts on the subject since the 70s have recently been subsumed into a trendier bandwagon called *agile methods*. But the important truths stand. Keep development simple. Build systems in the smallest possible steps, allowing customers to use, or failing that to see, each step in turn. So they can tell you what you are getting right and, more importantly, wrong. (And, indeed, tell you things that they had no idea could be better, till they saw what exists now. That's how it always works.)

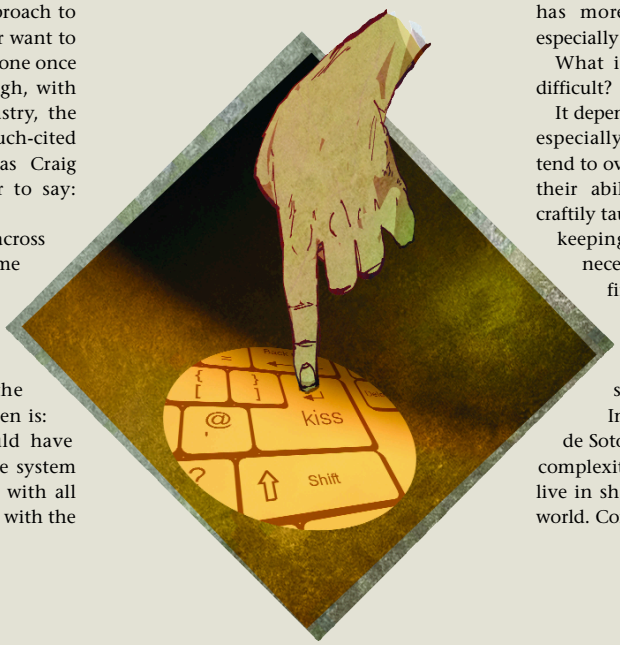
The resultant feedback has never failed to be a tremendous eye-opener, since I first came across Tom's ideas in 1986. Every step. Of every system. For companies as diverse as Reuters and Rio Tinto, on everything from hi-tech simulation for defence to more humble, yet lucrative, linear forecasting for hedge funds. As you might assume, one learns most from the early steps. But once your customers know that this approach to development is possible, they never want to go back. To the old way, what someone once dubbed the *waterfall model*. Though, with massive irony for our whole industry, the original paper that gave us the much-cited waterfall diagram only did so, as Craig Larman delights to tell, in order to say: Never Do It Like This!

What is waterfall? You've come across the main idea in the news, every time a government computer project goes another hundred million pounds over budget or fails to deliver anything useful. Some expert is interviewed about the disaster and the answer always given is: the stupid people involved should have written a specification of what the system should do, in every detail, agreed with all stakeholders, then accurately priced with the developers, before they started.

Nonsense, say Gilb, Larman and fifty years' hard experience with software. That kind of thinking is what caused the problem in the first place. Keep it simple, especially if you are that stupid. Agree the smallest step that will make a positive difference. Do that. And see how you go.

It's barely possible to understate how much taxpayer money would have been saved, in producing systems vastly superior to the white elephants littering the scene today, if the agile rubric had been employed for government. Vastly superior for users, for the often disadvantaged people they exist to serve, and for the elected officials whose careers are so often on the line when the waterfall chickens come home to roost.

APPLE'S RECENT RENOWN FOR BEAUTIFUL SIMPLICITY, NOT LEAST BY MUSIC FANS ACROSS THE WORLD, ALTHOUGH SURPRISING TO THE MANY PUNDITS WHO WROTE THE COMPANY OFF IN THE 90S, IS HARDLY AN ACCIDENT



But ah, would directors of consultancies who take on such massively profitable trips into fantasy land for government be quite so happy with such a radical change? Or senior civil servants who seem to shuffle across to the boards of such companies with consummate ease, whose bulge of budget is all too often seen as the measure of the importance of the man? Such small-minded motives have, I'm sure, nothing to do with the fact that the incremental approach has been spurned for years by fat cats on both sides of large development contracts. But not by ordinary software engineers. Many of us have known the truth for years. And, like any professional, it hurts to be forced to do work that is so utterly second-best.

The little Seattle-based software outfit called Microsoft is not, meanwhile, so stupid. Despite its products not always being as simple as they ought to be, developing more commercial software than anyone else on the planet, as they were doing by the 90s, tends to concentrate the mind. Around that time the men and women under the beady eye of Gates took a close look at where they were going wrong and took advice from people that seemed to have answers. Then Steve McConnell, by that time an outside consultant, wrote the first account of what they felt they had learned: *Code Complete* was published by Microsoft Press in 1993.

The index that my old friend took such delight in showed, under G, that in the much-awaited text Bill Gates was cited two times, Tom Gilb ten. Tom had waited around fifteen years for that kind of recognition. He has more of it today. But old habits, especially in government, die hard.

What is it about simplicity we find so difficult?

It depends who the 'we' is. Software geeks, especially young and badly taught ones, tend to overdesign systems and overestimate their abilities. Elites, especially old and craftily taught ones, have a vested interest in keeping things more complicated than necessary, lest the unwashed masses find out what they are missing. But things they are a-changin'. Technology itself, from cheap mobile phones to the internet, is seeing to that.

In *The Mystery of Capital* Hernando de Soto urgently shows how unnecessary complexity blights the lives of billions who live in shanty towns across the developing world. Contrary to rumour, the vast majority

of such people want to join the official economy. And they have amassed trillions of dollars of assets, mostly through honest hard work. (Gangsters will always be with us but remain a minority.) If such people, perhaps five-sixths of mankind, were able to register such possessions and trade off them, raising loans, starting businesses and the like, their prospects would soar. As would that of the countries in which they suffer such squalor, too often dependent on western handouts that prevent such fundamental issues being addressed.

Where does the problem lie, according to the brilliant Peruvian? The complexity of rules governing entry to the official economy. And the culprits (beyond rich elites that tend to corrupt governments of all kinds)? Lawyers and software suppliers. The lawyers make far too much money from the complex rules for registering property or starting a business to have any motive to simplify. And I was hardly surprised to learn from de Soto how geographic information systems had, by 2000, become the latest complex boondoggle to make money for system suppliers while diverting attention from the real issues. (Complex maps of property rights were being drawn up on such systems, blessed by highly paid international consultants. But they could only reflect and reinforce the official view. What was needed was new policy, on the ground. People power would do the rest.)

One interesting aspect of the story is that nowadays Google Maps and Google Earth are for free. There's even Google Code to store open-source 'mash-ups', combining new data sets with the familiar, easy to use vector-mapping offerings. The competitive pressure caused by the web – and the money to be made through advertising, due to the affluent eyeballs successful sites attract – is freeing up software, democratising all manner of systems. Despite the challenges, we all have much for which to thank Tim Berners-Lee and his brilliant, unselfish design in 1990.

Brilliant and unselfish because simple, of course. And if you asked Tim himself he'd say that it was essential to bring the web to reality in small steps, server and browser, thus refining his ideas and making them truly elegant. Which he was able to do partly because he was using a system designed with that kind of thing in mind: NextStep on the lovely old NeXT cube.

You may not have heard of Next. You probably have heard of Microsoft, who at the same moment were trying so hard to simplify their development process, with help from my mentor. But one has to admit that Microsoft's end products are seldom loved for their elegance and simplicity (certainly not by Gilb, who has long been a Mac fanatic). Unlike the iPod and the iPhone. They're from the same stable as Next, indeed the iPhone runs a fully-fledged version of OS X, where the brilliant object-oriented programming ideas of Jean-Marie Hullot and team, designed to simplify creation of graphic user interfaces by anyone, now reside. It's a long story. But this beautiful system was what Tim was playing with at the European Particle Physics Laboratory in 1990. Simplicity in one area begets it in the next, at least when the user or programmer concerned is intelligent and creative.

WHAT WAS THE OTHER FACTOR THAT MADE FACEBOOK THE FORCE IT IS TODAY? A CULTURE, RIGHT FROM THE START, THAT USERS … SHOULD USE REAL NAMES, NOT PSEUDONYMS. IN A WORD, SIMPLICITY

Apple's recent renown for beautiful simplicity, not least by music fans across the world, although surprising to the many pundits who wrote the company off in the 90s, is hardly an accident. Here's a classic insight from Steve Jobs, interviewed by *Byte* magazine in February 1984 on the launch of the original Macintosh:

> When you first attack a problem it seems really simple because you don't understand it. Then, when you start to really understand it, you come up with these really complicated solutions because it's really hairy. Most people stop there. But a few people keep burning the midnight oil and finally understand the underlying principles of the problem and come up with an elegantly simple solution for it. But very few people go the distance to get there.

Note again, as well as Jobs's characteristic enthusiasm for his 'insanely great' people at Apple, the place of dogged, stepwise refinement of design. Another name for this is humility. Or, if that sounds too weedy for those that want to dream big dreams, let's try honesty. Here's what software gurus David Parnas and Paul Clements wrote in 1986:

> The picture of the software designer deriving his design in a rational, error-free way from a statement of requirements is quite unrealistic. No system has ever been developed in that way, and probably none ever will. Even the small program developments shown in textbooks and papers are unreal. They have been revised and polished until the author has shown what he wishes he had done, not what actually did happen.

Another way to say it is that software developers, being human, are dumb. At least faced with the systems we'd like to build. Keep it simple, stupid. It's knowing that you are that is the key.

Let's finish by comparing two young developers who are not normally called stupid. Mark Zuckerberg is, at 23, founder and chief executive of Facebook, and David Heinemeier Hansson is, at 27, award-winning creator of Ruby on Rails, a web development framework.

One thing on which the two men differ is programming language. And that can easily get nasty. Facebook is written in PHP, widely used to build websites. Rails, as its full name suggests, uses Ruby. At the age of 49 I'll tell you this for free: Ruby is the better language. But it is not, in some surface ways, quite as simple as PHP. Instead, it allows you to develop simpler systems. Or, more accurately, given the same system to build, the Ruby version should be much simpler underneath. Fewer lines of code, easier to understand. It matters a lot. Trust me.

This simplicity thing is, as Jobs implied, a bit complicated. As Einstein once said, things should be as simple as possible. But no simpler!

More striking is how Zuckerberg and Hansson represent an articulate regard for simplicity among young developers who are role models for others. Hansson goes on about the issue the whole time, and rightly so. And almost anyone who has dipped into both MySpace and Facebook comes away saying that Facebook feels simpler. Not unrelated, it's pretty much the hottest web property there is right now.

Other similarities include deep reliance on the designs of Tim Berners-Lee. If it ain't broke, don't redesign it. For the technically minded, I'm talking of REST interfaces built from simple HTTP and URL. Hansson has been an advocate for years. Zuckerberg endorsed REST in his 29 May opening up of Facebook to third-party developers – an amazing media event about which *Fortune* magazine, the *Washington Post*, Uncle Tom Cobbley (on his blog, obviously) and all wrote vast amounts of glowing copy.

One more thing about the genius of Zuckerberg. (It's an overused word, but the young man is fabulously influential and I'm feeling generous!) What was the other factor that made Facebook the force it is today? A culture, right from the start, that users, initially students, should use real names, not pseudonyms. In a word, simplicity. It helps. Ask those Wall Street financiers and Fortune 500 chief executives who are so keen to help the young fellow, as and when he wants to cash in on the many billions of value that he and his team are already deemed to have created.

Not many would dare to call Zuckerberg or Heinemeier Hansson stupid. But where they've become great, I'd argue it's largely because they kept it simple. As for the rest of us … we know the real reason.