THE UNSUNG HEROES OF SOFTWARE

By Richard Drake

Let's start our whirlwind tour of software ethics with a perceptive anonymous critic.

> What I see, as a non-programmer, is the total lack of accountability of the software industry towards society. But as software becomes vital to many aspects of our lives, this situation seems to become more and more untenable.

I read that on the web. It is part of an intelligent online discussion following a thoughtful interview called 'The Problem with Programming', published by the MIT *Technology Review* last November. What I thought was a fair, well-stated criticism of modern software was posted on a website in Boston and thus relied on a whole lot of programming – good or bad – to become known to me in London. Such are the daily ironies of the modern, techno-savvy life.

We've always said that the phrase 'ethics of software' is really shorthand for any of the myriad ethical issues raised by hardware and software, computers and communications. So let's hear a rather stunning counterpoint from out-and-out hardware man, Henning Sirringhaus, Hitachi Professor of Electron Device Physics at Cambridge, as recorded by *Electronics Weekly* in May 2005:

> It is obvious in conversation that the work, with its potential to affect people through products such as electronic paper, satisfies Sirringhaus for similar reasons to those for which renewable energy once appealed. 'There aren't many fields where you take basic research and very quickly have the chance to affect people's lives,' he admits.

Not that the two are entirely inconsistent, of course. This computing stuff, whatever it is made from or into, affects people's lives enormously. If it didn't, Professor Sirringhaus's start-up company, Plastic Logic, would not have been breezily announcing new funding of $100 million this January. It's a significant moment for the reflective nerd: the first real stirrings of devices and displays with logic no longer built on silicon (or its close sister, gallium) but on plastic – 'semiconducting polymer materials' to their friends. That's our first real change in hardware foundations since vacuum tubes were replaced by silicon-based transistors in the 1950s. If it comes off. At least $100 million – a lot more, if you take into account would-be rivals of the Cambridge hopefuls – is being bet that such an unforeseen thing will come to pass. And, who knows, you may have read it here first. Or at least realised a little bit more how historic it is!

Even before starting on our main theme I wanted to emphasise the complexity and ambiguities of our subject by throwing into sharp contrast:

- partially understood past, radically unpredictable future
- frustrations of software, excitements of hardware
- sceptic or victim, idealist or perpetrator
- cynical American, courageous Brit

The last is a joke. I have no idea where the first poster came from, and I admire his point. But I did want to get in just one mention of a plausible inventor of the future based in the UK, even if the name Sirringhaus suggests other roots than the gentle Fens. We have to admit that not many Brits are in such a position these days, despite our world leadership in wiring those vacuum tubes together in the 1940s, helping the secret code-breakers at Bletchley Park make such a signal contribution – sorry – to victory over fascism. All under unsung hardware genius Tommy Flowers, based on the brilliant, creative mathematics – software, if you will – of Alan Turing.

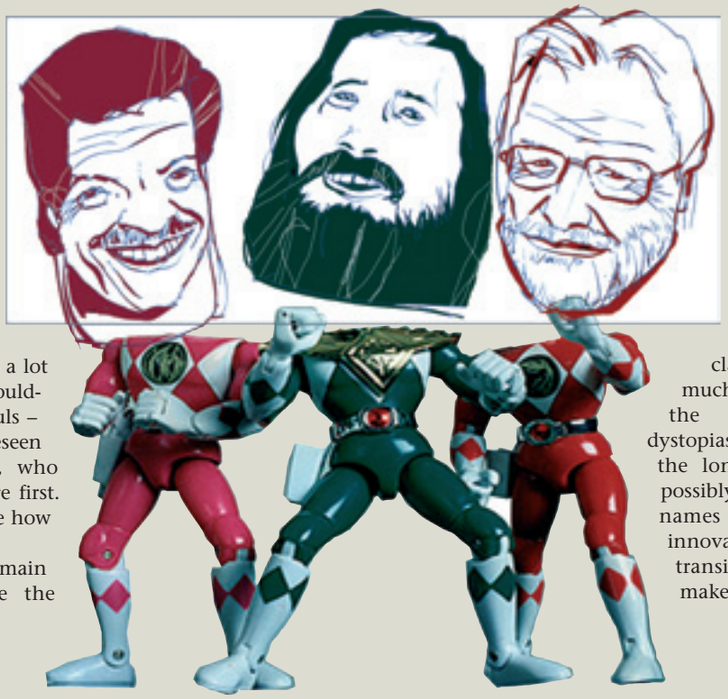
It's an amazing story of a mere 65 years, developments of profound beauty begetting tumultuous social change, all in a time period that's a mere pinprick by world history standards. All of which I now aim to cut through with this charmingly deceptive question: Who are the three wise men of software?

Before putting my own fingers to keyboard and mouse I asked three close friends in London what their answers might be. My local experts should have keen insights on the subject, I felt, from quite different perspectives. One has made a significant amount of money through software but has never written programs. The other two have, like me, been programmers for much of their working lives, the younger now playing an active role leading one of the most respected agile development groups in the UK, of over 20 people. (Remember that hopeful-sounding adjective. It's going to appear a lot more before our ethical explorations are finished.)

The punchline, of course, is that adding the ten people nominated by my advisers – one insisting on both founders of Google as a single choice – to three of my own, we had not one in common. Mind you, my oldest collaborator had the temerity to choose himself. When asked if he could, just once, be serious, he deferred, independently, to one of mine, Tom Gilb. All three of my temporary mentors then grunted approvingly as I mentioned Alan Kay. Oh yes, they should have thought of him. But at least two made a face – quite a feat on a phone without webcam – when I declared that my last selection was Richard Stallman.

As the wags would say, you could tell we were experts because we disagreed so much with each other.

But I'm happy. First, because I seem to have invented an excellent new parlour game for anyone with the slightest interest in the future of software. And second, because my three choices were not just straight from the heart, they clearly provide rich seams of software ethics, controversy and paradox to mine for the rest of the article.



So who are these wise men, so unsung that they made hardly a ripple in the mainstream media for 30 years and – not unrelated – collected vastly less loot than those clever chaps from Google in the last ten? Men whose ideas have shaped us but, I claim, we need to listen to much more, if we are to avoid the worst of technosocial dystopias in the future. Cutting out the long words, what tags can possibly be attached to such old names to cut any ice in the innovative, young, cool, brash, transient social networks that make up Web 2.0? Well:

- Richard Stallman began open source
- Tom Gilb pioneered agile development
- Alan Kay invented … well, it is difficult in Alan's case, for his contributions have been both deep and wide. But the phrase I'm going to use in the rest of the article is *constant fun*

So the future of software in my opinion should look like this: open-source, agile and constant fun.

Interestingly, in saying so, all three men have now been tagged with terms that others invented.

What Richard Stallman originally proposed, in 1986, and has always been his preferred name, is *free software*. With the sad but inevitable rider: free as in speech, not beer. Stallman urges all software, used by anybody, to be provided, without exception, with full source code – the painstaking descriptions that programmers type to produce it and that others need to enhance and fix it. His case has always been about ethics, justice and the well-being of society. Many people, not least those with a strong vested commercial interest, will still say that he is an extremist who goes too far. But without him the open-source movement would not be the way it is today, transforming much of the software landscape. Stallman is its founding father and it's right to pay him respect for that, in my view, and to learn as much as possible from his arguments, as well as those of good faith who respectfully beg to differ. More of that anon.

Tom Gilb is a personal friend, with homes in London and his native Norway. He too spent a long time in the wilderness from the 1970s onwards, sent out to chew on the technology world's equivalent of wild honey and locusts for the temerity of saying things that were not only different to everyone else but often exactly opposite to them. But Tom was right. And his core thesis was profoundly simple. Though there is more to the ideas and practices by now, a large part of the agile methods movement that has emerged so strongly in the last eight years, both independent of and influenced by Tom, boils down to two old phrases of his: *small steps* and *juicy bits first*. That's the way all systems should be built and, wherever possible, delivered to their users. As Tom and more recently Craig Larman have shown, it's the only way that has ever worked, in the whole history of our industry. The government of the UK, among others, needs to know about that. Very badly. So do many other institutions and corporations. That's a key ethical issue of our time. Again, some major vested commercial interests come into play when the radical nature of the required change becomes clear.

Alan Kay is a true genius. One of his many memorable goals for software is that 'simple things should be simple, complex things should be possible'. In order to achieve this for all computer users, but especially for children, Alan and the brilliant research team he built at Xerox in the 1970s ended up inventing many things. All very consistent with another famous motto: 'The best way to predict the future is to invent it.' The nearest thing we have to Kay's own web page today starts by saying that he was 'one of the earliest pioneers of object-oriented programming, personal computing and graphical user interfaces'. Not the pioneer, because one of the man's greatest strengths has been to pay generous tribute to those that came before and alongside. But there's no question in my mind, and in many others, that Kay's passionate vision of the uniformity of both user and programmer experience needed to achieve his goals was, in the 1970s, both startlingly prescient and unique in its profound understanding of humanity. The best way I can find to describe the kind of system Alan believes in, for the layman, is, as I've mentioned, constant fun. The only unexpected things that should happen with software should be positive, in other words. There should be a consistency, a constancy about everything we do and interact with that encourages fun and thus enables the full range of human creativity. As you may have noticed, it isn't always that way.

I SEEM TO HAVE INVENTED AN EXCELLENT NEW PARLOUR GAME FOR ANYONE WITH THE SLIGHTEST INTEREST IN THE FUTURE OF SOFTWARE

When Xerox didn't know what to do with the richness they had funded, some of Kay's ideas were picked up in the 1980s by a talented and ambitious entrepreneur called Steve Jobs, co-founder of Apple Computer. Not all, not by any means. That's where the frustration meets the excitement for those of us who have followed such matters for the best part of three decades. But it sure seemed appropriate to hear Jobs pay tribute to Kay during his recent, widely praised introduction of the Apple iPhone at MacExpo:

> One of the pioneers of our industry, Alan Kay, has had a lot of great quotes throughout the years and I ran across one of them recently that explains … why we go about doing things the way we do. Because we love software. Here's the quote: 'People who are really serious about software should make their own hardware.' [Applause] Alan said this thirty years ago and this is how we feel about it.

There have been some dire compromises, even at Apple, in implementing Kay's vision. But, as we aim to review in future, the sparks from an honest attempt to follow his insights, and highly synergistic ones from the other two thinkers, have gone on to light fires leading to some of the most important software breakthroughs of the last two decades:

- Tim Berners-Lee and the world wide web
- Ward Cunningham and the Wiki idea, leading to Wikipedia
- Matz and DHH's open-source work leading to Ruby on Rails

OK, that last one probably isn't going to make much sense right now. (Just savour the inspiring alliteration of the outcome, for the moment.) In any case, how come I'm writing as if the computing, mobile phone and popular electronics industries still haven't properly learned things first propounded by my wise men two and three decades ago? Surely we can all see that technology has come on leaps and bounds since then?

Well, yes and no. The hardware has, remarkably so. Intel's Gordon Moore, surely one of the wise men of hardware, famously predicted such a wondrous, continuous process of miniaturisation and increase in power, based on the known properties of silicon, way back in 1965. At that point nobody knew for sure about the feasibility of many complementary developments, such as fibre optics for high-speed communications, wireless networking, the many new options for fast – or vast – digital storage, or tiny liquid-crystal displays leading to today's stunning flat panel screens. Let alone plastic semiconductors and the nifty fold-up screens they may soon make possible. But what of the software driving all this amazing, entrepreneur-driven gadgetry? Bogged down in some very old ruts, in many places. Except for the green shoots of hope we've begun to identify.

Don't believe me? Here's how my first quoted critic, in November 2006, went on:

> My super high-tech cell phone crashes fairly often, and it takes two minutes to reboot. Sometimes I wonder, what if this ever happens in an emergency situation? In an emergency, two minutes is an eternity, and it can easily mean the difference between life and death. The list of our everyday software dependence could go on and on …

Let's hope the beautiful-looking Apple iPhone does better than that. For old times' sake, as well as to save some high-tech, 21st-century lives. If others follow suit we may have more time to think about the ethics and global relationships engendered by our strange but wonderful world of new technology. ■

***Richard Drake** has been a software developer since 1980. Today he consults in software strategy and web development. He welcomes feedback at richard@sublimemagazine.com*