Invited talk to Queens College Computer Science Student Society, 14 May 2023, by Don Syme

Invitation by Conall Moss, Lochlann Baker, Dan Wendon-Blixrud, Andy Zhou

---

A long, long time ago I wrote in assembler
and those opcodes used to make me smile
I wrote my hello world program in 16kb of RAM
No function call no do or for or while

I'd sit all night in rapt frustration
Trying new optimisations
Forget to eat or sleep
I could not admit defeat

Every day brought new confusion
Surrounded by my own delusion
I finally came to the conclusion
To do
The Big Rewrite

That is the opening verse to "The Big Rewrite", a song to the Tune of American Pie by Dylan Beattie and the Linebreakers, a band I used to play bass in, but have recently hung up my bass and passed those duties on to another. Starting with this experience of the 80s the song continues a journey through computing history along the theme of “The Big Rewrite”. That is, whether it's the 1980s, 90s, 00s or era of the hyperscale cloud and AI, we see the same recurring themes of our tendency to create overly complex systems, and yet we try to rewrite it all, to achieve simplicity and perfection, again and again.

Now Dylan Beattie Is an absolute genius at bringing out these existential, deep themes with humour, lightness, and happiness, and I can only wish I could match his ability to talk about these things – the business of software and the experience of being involved in it – with these qualities.

Of course, I chose that song because, when invited to give a talk like this, you begin to reminisce, you think back to your university days, when the world was bright and fresh, the sunlight on the grass at Trinity Hall, the punts on the river, and every new idea seemed to sparkle and radiate.

The song looks back further than that, however, back to the 80s and 70s. For me I was a kid with an Apple II. In the song we see language coming in - *no function calls, no do, no for or while*. These things had been invented of course but weren't standard kit on the programming tools for the Spectrum, Commodore or Apple, so right from the start language, compilation, interpretation, expressivity are central to the child's experience. Soundness too – when we made a mistake copying a program key by key from a listing in a magazine, we'd pay the cost in days of debugging, and the Big Rewrite was impossible as we lacked the tools to do it.

Now I understand I'm the one keeping you from dessert and I'll try not take too much of your time. I've decided to structure this talk around three quickfire themes, mainly addressed to the students in the audience, undergrads and postgrads, especially those who might be interested in a career in applied research, perhaps inside a big company, or who at least want to work at the more innovative end of the computing spectrum.

So I've been invited mainly because, back in 2000s, I took the chance to make a programming language – a language very much inspired by OCaml – my aim was to make typed functional programming work in practice in industry. Now this agenda has had some success - if you want to experience the joys of programming daily in operationally sound functional-first programming in practical, applied settings – whether making money or for pleasure – you can certainly do that. And some of you may continue on to get jobs in London or Cambridge where you do precisely that in industry.

Probably more likely, however, is you'll be using one of the truly massive languages, languages with six zeroes in their user count, but in truth heavily influenced by the agenda of OCaml, F# and related languages – I mean the mammoth languages such as C#, Typescript, Java, Python and C++. I've written

and spoken about the history of F#, and its influence on these languages both direct and indirect, with some fun reminiscing along the way, in the HOPL conference in 2021.

That all brings me to my first theme, which is this: **if you see a chance, take it**. What I mean by this is that effecting change in the foundations of technology stacks is difficult, and requires many stars to align. In the 2000s I was at a research lab - MSR - a wonderful place with an incredibly diverse range of talent - and a chance opened up to do a new language. When unique chances come in life, take them.

Before that, another chance had come up. A group of us at MSR had been present at the birth of what is now C#, and we saw a possibility to effect change early on in the history of C#, early enough to systematically infuse C# with the kind of design ideals that would help shift the industry away from the nonsense of pure OO towards more rigorous foundation.

One chance leads to another, and by opening the door with C# and F# we did many things subsequently in later versions and related settings and brought a little light into the very core of what is really the most conservative, reluctant part of the software industry.

This brings me to my second theme, which I call **dancing on the edge of the vortex,** or you might call **sitting in a cage with a gorilla,** or **engaging with the monolith**. What I mean by this is that as researchers we try to effect change - but the software industry is truly vast, and the core technical artifacts in our technology stacks such as Linux, a modern microprocessor, the implementation of GitHub, or Windows, or a modern runtime, or Excel, or an AI stack, or whatever are vast and feel intractable. They also have a vortex-like power of their own - two of these things can't sit too close to each other, one tends suck up the other and combine their power.

Now, unless you are living off venture capital, then probably anyone paying or funding you for innovative work has one of these platforms or systems or products. This especially applies if you're doing research within existing companies.

This is surprisingly hard. The best way to effect big change is to change something big. So we walk up to these systems, or the industry as a whole, and try to change things. Simon PJ used to say researchers were like mosquitoes

slamming into an aircraft carrier trying to get it to change course. Now Simon's work has had influence on many big things, including a host of languages and Excel, and many more to come.

So if you plan a career in R&D, this is part of the decision matrix you'll have to consider: which big systems to try to influence, where to locate your innovation and energy, who to team up with, what kind of change to try to make, what are the design ideals that drive this change, what kind of setup do you need to be able to wave the magic wand of research and watch as the industry adopts new ideas the way you want it to.

But what kind of change? My final theme is about **technical communities** and **breaking the rules of the tribe.**

Participating in a programming language has brought me in touch with almost every conceivable part of the software industry

- People making the software for hi tech devices used in wheat silos in the fields in Australia near where I grew up,
- German device makers making software for cars, trains and listening devices on oil pipelines
- Commodities trading in London
- The world's first 1h delivery service in Dhaka in Bangladesh and the software that powers it.
- Or perhaps you'll get the chance to visit the Louvre Museum in Abu Dhabi - the second Louvre - and look up to the roof, as the rain of light dapples through from the hot desert sun, and contemplate that every piece of that remarkable roof has had its precise measurements and machining calculated by an F# script, written by an Austrian engineering company

So these kinds of anecdotes. And the people behind them, have given me one of my technical families.

We need our technical families, and in today's hyper-partisan world it's possible that technical cooperation can allow positive interaction with others outside our cultural echo chambers, and while there are limits to that there’s still huge value there too.

Now MSR Cambridge brought together four of the extended technical families - or traditions - of computer science - the same ones that stretch back through to the dawn of time to the early history of computing and the Cambridge computer laboratory, groups I fondly remember from the Cambridge CL tea room at the Old Museums site.

At MSR we had the systems tradition, the logic tradition, the HCI/UX tradition and the engineering department provided the machine learning folk.

Each of these came with very strong orthodoxies and beliefs. Technical communities are crucial, and so are beliefs and principles, but when beliefs become inviolable, totemic, unchanging, fixed, unquestionable then you have a tribe. And a tribe can be both terrible and immensely boring.

At MSR what I really remember most is the people who broke with the orthodoxies of their local groups, repeatedly and habitually. The ML researchers who rejected the local Bayesian orthodoxy and went into RL and DL, and used FP along the way. The systems researchers who threw away the fashionable attempts to implement software isolation and helped invent containers. Antonio Criminisi who'd look at a Renaissance painting and say "I can reconstruct that ". Or look at a tissue image and think "I can automate the detection of cancer in that "

To give one example on the UX side, I've fond memories of Lyndsay Williams at MSR. I think she won't mind my saying crazy, amazing Lyndsay Williams. Smart pens. Cameras that take photos of your entire life. Devices that measure your pulse – today's wearables. Touch input. Each one pushing boundaries.

Lyndsay was - and still is - breaking the orthodoxies repeatedly. She is also a reminder that breaking the rules needs true diversity - mixed teams - along many dimensions of diverse. For a research lab, there's nothing worse than group-think.

In any case, if you're planning a life of research, then be ready to break with the orthodoxies of your tradition - don't hold back. Perhaps break them one at a time, rather than all at once, and be aware of what you're breaking. And breaking orthodoxies doesn't mean breaking the law or breaking morality. But still don't hold back from breaking with the orthodoxies of your tradition.

Now, with that I'll finish up, with the final observation that the three themes I've mentioned

- if you see a chance, take it
- dancing on the edge of the vortex, and
- breaking the orthodoxies of the tribe

are all quite risky activities, and you are going to need a good degree of stability and security along the way, both personal and as a group.

I can see that here at Queens, in this CS society, you might not realise it but you're part of a family, a cross-cutting community here at Queens. That community, if you treat each other well, will support you and allow you to take risks, of many kinds. This is not the room where it happens – those rooms are open source repositories, or conferences, or technical leadership in companies, or emails sent to CEOs. But it is a room you can come back to and consider, support. So stick with each other through your careers, and also contribute to the other technical families you'll find along the way, and where possible always keep the door open to newcomers and keep your groups diverse. And don't stop doing The Big Rewrite of technology and computer science - that attempt to make everything right and perfect - in whatever way you choose to do it.

And with that on to dessert!