John C. G. Sturdy M.A., PhD
Marital Status: single
Clean UK full driving licence (Category C)

Cnoc na Gortíní
Maigh Rua
Contae Luimnigh
Eire

Profile

An experienced and versatile Computer Scientist with research background and extensive experience of large-scale commercial and research software development, good knowledge of various computer systems and languages, and experience of teaching small and large groups.

Education

PhD in Computer Science School of Mathematical Sciences, University of Bath, 1985-1988 (full-time) 1988-1991 (part-time)

M.A. in Computer Science (with one year Biological Natural Sciences) Pembroke College, Cambridge University 1982-1985

Employment

2006-7 Cúntóir Taighde agus Dearthóir Cursaí, Roinn Riomheolais, Ollscoil Luimnigh[1]

2003-2006 Postdoctoral Research Fellow, B4STEP programme, CSIS Department, University of Limerick

1990-2002 Part-time Supervisor, Cambridge University Computer Laboratory

2000-2001 Senior Computer Programmer at the Wellcome Trust Sanger Institute

1998-2000 Software Engineer at EDS Unigraphics

1988-1998 Computer Consultant (Software Engineer) at Harlequin Ltd

Previously Various vacation jobs in software engineering, mostly at Acornsoft on BBC Micro; Gap Year job included writing an editor and working on the kernel of a small concurrent Operating System

[1] Research Assistant and Course Developer, Department of Computer Science, University of Limerick; working on Irish-related material

Teaching

University of Limerick, 2003-2005 Lectured "Introduction to Operating Systems" module for the B.Sc. in Computer Systems

Cambridge University, 1990-2002 External Supervisor (small-group tutor) for most aspects of the B.A. in Computer Science and the Diploma in Computer Science[2]

Research Interests

Software editing, and its psychology and cognitive ergonomics How programmers think of changes to programs; how they get the changes into the computer, and new ways of doing so; how the changes propagate from one computer to another; how changes get back to programmers; feedback from the facilities available to the way programmers think

Ergonomics for serious techies User-friendly computer systems for computer-friendly users: How much more use can we make of the bandwidth between human and computer, when catering for users motivated to tackle steeper learning curves? And can people be trained to think like advanced users, by using systems designing for advanced use?

Multilingual software development This is an adjunct to my interests in software editing.

Advanced programming language technology Particularly reflection (self-referential systems); also partial evaluation, lazy evaluation, parallel and distributed computation; the potential for combinations of these

Factors affecting creativity and productivity Solo work and team work; the built environment; controlling distraction; how and why people work "away from it all" or at strange hours; unplugged distributed source control networks

Particular skills and aptitudes

Research in, and development of, software development tools, taking care to make them clearly comprehensible to other engineers who may work on them in the future.

I am an experienced Lisp (Common Lisp and Emacs Lisp, with about 2.4Mb published as Open Source in the latter) and C programmer (with fair C++ experience) mostly on Unix (largely BSD) but also with a couple of years Windows-NT/MFC experience, and have very good PostScript skills. For the past few

[2] I was also invited to do this as a 3rd-year undergraduate

years, I have had a high proportion of my Lisp code work first time. I'm experienced in general Unix work (shell, Perl, make, sed etc). I have done some work in Assembly languages (various) and also worked on CPU microcode (various graphics hacks such as Mandelbrot calculations on the Orion-1). My Unix work has included a major multiprocessing project (a pioneering implementation of PostScript for very-high-resolution printing systems) in which I led the work on the parallelism in it.

I have done some BSD sysadmin work in the past (a normal side-effect of doing a PhD in that era!), and am now picking up on this again as needed.

Coding skills

The number of years mentioned against a language in the list below is the time for which I used it as a major element of my work.

Very good Lisp (21 years), C (14 years), PostScript (8 years), Bourne Shell, LaTeX, TeXinfo, HTML

Good Perl, C++ (2 years), Java (1 year), CGI, Make (2 years), XML, BCPL (1 year)

Some experience plain TeX, Assembler (6502, 68000, 370, 8086), Microcode (PERQ, Orion 1), Fortran, SNOBOL, MFC, awk etc

Highlights of Work

In my PhD I constructed a reifying, reflective, mixed-language programming language interpreter providing access to the underlying abstractions of data and process representation, that could regress to an infinite number of infinite levels of abstraction and representation, the resulting interpreter nevertheless running around 40% of the speed of a plain commercial interpreter for the same languages. This system provided systematically the means to examine and extend a running software system dynamically, putting debuggers and modularly extensible systems on a more formal footing. (Reflective facilities are now beginning to appear commercially, and a relatively very primitive form of reflection is regarded as an advanced feature of Java.) The implementation language for this was Common Lisp, running under BSD Unix. The ideas behind this built partly on the work of Brian Cantwell Smith, and partly on an abstract machine for extensible languages, that I had designed and implemented for my undergraduate dissertation project.

At Harlequin, I researched and constructed an experimental dynamic software migration system for Common Lisp (for an EU Esprit project, Chameleon), capable of suspending a process on one system, packing all the data needed to represent it, transferring it over the network to another machine and continuing to run it where it left off on its previous host. While still on the Lisp projects, I also developed a link loader for loading externally compiled code (COFF objects from the C compiler, for example) into the running Lisp image.

After that joined Harlequin's Electronic Publishing division (and was there for most of my time with this employer), where I worked on many parts of their PostScript-compatible ("clone") Raster Image Processor, the highlight of which was designing and implementing the conversion of our existing single-process system to a pipelined parallel system, implemented in C. I also wrote large amounts of PostScript embedded in the implementation. I designed and implemented the computer end of the low-level interface to the laser imagesetter hardware. Later parts of the parallelism work involved working on obscure timing-related problems. I later moved over to various aspects of their advanced colour system, learning about colour science to do this.

At Unigraphics, I worked mostly on a series of projects within a very large CAD/CAM system (about 20 years work by an average of 300 people), including an object-oriented database interface to the system.

At the Wellcome Trust Sanger Institute, I worked on the genomic database system AceDB (an object-oriented genomics database system), working mostly on a new database query language for it (in C), and also on the project's web site and its automatic build and test systems (which were mostly in Perl).

I am experienced in writing web pages, taking care to keep them straightforward and fast-loading, and to make them accessible to all, including those using text-only (or speech or braille) browsers. I did some web work professionally while working at the Sanger Institute.

Between 1990 and 2002 I also supervised (tutored) Computer Science students for all years of the Degree course, and for the post-graduate Diploma course, at Cambridge University (as an evening job), thus both keeping in touch with what is now being taught in computer science and also developing my skills in explaining ideas and in evaluating others' understanding of topics. I have supervised a wide range of topics, but mostly those related to programming languages and operating systems.

In my work in B4STEP, I work mainly on software development tools and the psychology underlying them. However, of necessity, I have also spent considerable time co-ordinating our work on the EU project COSPA (Consortium for Open Source in Public Administration), including managing the construction of a knowledge base system for it. The COSPA work also involved a considerable amount of programming (Java and Perl), unfortunately not leading to publishable research output. I also led our independent evaluation of Prof David Parnas' "Tabular notation" specifications. As well as these, I lectured the "Operating Systems Overview" module of the Computer Systems degree for three years.

The main part of my recent work on software tools resulted in a set of experimental extensions to the GNUemacs editor/working environment, using language-specific plugins to support not only navigation in terms of semantically significant units, but also automated high-level operations for work that programmers have so far done as highly stereotyped sequences of manual actions, such as converting code blocks to procedures (finding automatically what

needs to be passed in as parameters, what types the parameters must be, etc)[3].

Minor projects

In my gap year job, I worked in a group proving theoretically the kernel of a small concurrent operating system (CrestOS, now long-lost; written in BCPL with an 800-line assembler kernel). This gave me a solid foundation in real-world concurrent programming.

In my B.A. dissertation, I designed a low-level reflective programming language (with a view to it being implementable in hardware), and implemented it on an emulator. Also while an undergraduate I was regarded by computing service staff as being the first person to write readable programs in the local editor scripting language (including a toy FORTRAN subset compiler).

Having discovered Koza's work on Genetic Programming, I started an experimental Lisp-based system for looking at the interaction of communicable information between individuals and the evolved characteristics in the population (for example, whether predators in an ecosystem would evolve differently if hunting strategies were communicable), but did not take this to a publishable stage.

As a spare-time activity, I once developed a user-contributable streetmap system (using technology that would now probably be called "Wiki", but before that term was used); I am now re-awakening this as a public project on SourceForge.

Another of my spare-time projects is a multi-lingual vocabulary learning tool (again within the Emacs editor / environment) which displays translations of words as you type them.

Very early work

At primary school, I invented the resistor, but suspected that other people would have thought of it too, and so did not develop this project beyond basic experiments.

When bored at secondary school, I designed my own microcoded CPU architecture, but lacked the resources to implement it.

Also at school I was one of the small group who interfaced the school computer (an Apple II) to an ASR33 teletype, building our own hardware and writing the machine code software to drive it.

Peer-reviewed Publications

- The LispWorks Manual, Edition 1 (editor), Harlequin Ltd, 1991

[3] I have been contacted by RMS about the possibility of including this in future versions of GNUemacs

- Donnellan, B, Fitzgerald, B, Lake, B and Sturdy, J (2005), Implementing an Open-Source Knowledge-Base, IEEE Software, Vol. 20, No. 6, pp. 92-96.
- Sturdy, J (2005), Sidebrain: a sidekick for the programmer's brain, Proceedings of the 17th workshop on the Psychology of Computer Programming (PPIG), 2005
- Sturdy, J (2006), Abstraction levels in editing programs, to appear in Proceedings of the 18th workshop on the Psychology of Computer Programming (PPIG), 2006

Other writing

- A Lisp through the Looking Glass (PhD thesis), University of Bath, 1991[4]
- A Structured Language Architectured Processor (BA dissertation), Cambridge University, 1985

Other abilities, activities and memberships

Languages I am a keen amateur learner of human languages, including the following

- moderate conversational ability in Irish
- fair reading ability in Norwegian, French, Middle German
- basic reading ability in Swedish, Modern German, Occitan, French
- very rusty Polish (now re-learning), Dutch
- very slowly learning several other languages including Mandarin and Finnish

Attended Part III[5] Set Theory, and Mathematics for Postgraduate Computer Scientists (Proof Theory, Transfinite Number and Set Theory, Category Theory) at Cambridge University (but I have no real claim to be a Mathematician)

Category C LGV licence previously called "Class II" HGV (Rigid Large Goods Vehicles)[6]

Change-ringing [7] Member of Society of Cambridge Youths, and of Cambridge University Guild of Change Ringers; Member, and previously Secretary of the Southern District, of the Irish Association of Change Ringers

[4] I am now looking into publishing from this; the main parts of it are still novel
[5] Fourth year Mathematics
[6] Used during a career break
[7] To "Standard Eight" level

Calligraphy Recommended for training for professional lettering design

Business skills Basic training in skills for small businesses: bookkeeping, marketing, office administration etc

Standard English Braille Grade I and most of Grade II (with some awareness of non-visual computing tools and requirements)

Cycling including cycle building and maintenance, on fixed-wheel, trikes and recumbents

Mediaeval music including designing and building reconstruction instruments

Green laning and Land-Rover maintenance and modification

Trained in leading discussion groups and workshops

Other programming languages including FORTRAN, the first programming language I learnt; BCPL