The Future of Open Source Software

Bill Appelbe
Victorian Partnership for Advanced Computing
Carlton South, Victoria 3053, Australia

Open source software is becoming increasingly popular. The open question is how far it will go in displacing traditional proprietary or "closed source" software? Is the whole face of software likely to change over the next decade? Will open source software topple the Microsoft juggernaut? This article attempts to delve into open source from the viewpoint of users, developers, and major software and service vendors.

WHAT IS "OPEN SOURCE?"

There are as many definitions of "open source" and "free software" as there are software lawyers drafting contracts that define them. But the essence of "open source" (a.k.a. "free software") is that:

- when you get the software, you get the source code, not just the binaries
- you can modify the source code to make improvements and fixes to the software.

By and large, the terms "free software" and "open source software" are interchangeable, as the word "free" in "free software" means "free(dom) to modify the program's source code", not "at no cost". This potential ambiguity in the meaning of the phrase "free software" has led to the term "open source software" becoming in more common use.

In general "open source" software is also available at low or no cost – sometimes depending on the class of user (academic or commercial). The variations in open source licenses all revolve around what users of the source code that comes with open source can do with it. Obviously, if open source is "free and open", then the developers of open source do not want others to freeload off their open source code to develop and market software that is not "free and open". A useful analogy is that if you were giving away fruit or vegetables from your garden to your neighbours, you would not want them to sell those to others… but there are lots of grey areas – what if your neighbour made jam with your fruit, can they sell that? Open source licenses range from the Free Software Foundations "copyleft" licenses (that require that any software that is built on open source software is itself open source), to pure open source that has no restrictions (or warranty) on the use or application of the software.

A SHORT HISTORY OF OPEN SOURCE

Open source software's roots go back to the earliest days of computing. The scientific programming community has always tended to "give away" its source code – hoping that others would use, build, and improve on it; just as scientists publish, or give away, their research results. There is an implicit "honour code" in science, which says that all research results and related developments (software) should be shared with the scientific community – scientists view colleagues who are not willing to

Manuscript received: 27 July 2003
Communicating Editor: Sidney A. Morris

share their results and work openly with deep suspicion. So from the late 50s, scientific software was being freely circulated – "open source".

1960–1980: The "Custom Code" Era

In the 60s, almost all computer applications were written from scratch, initially in Assembler, but by the end of the decade in an increasing diaspora of programming languages: Cobol, PL/1, Fortran, etc. Suppliers of computer hardware bundled in increasingly sophisticated software such as operating systems, compilers and related utilities. But the dominant software cost was in the development of custom applications code. Outside of the scientific and academic community, there was little need or even purpose to providing "open source" – most application code was unique, built to a client's exact specifications, and owned by the developers or the client. Computer vendors, such as Burroughs, IBM, Control Data, and NCR invested considerable resources in developing systems software for their hardware, but it was all unique and necessary to use their hardware, which was jealously guarded, and the cost was generally bundled into maintenance contracts. When you bought a computer in the 60s you had to pay for both software and hardware from the hardware vendor, and then write your application from scratch using the hardware vendor's compiler – this was the "custom code era".

In the late 60s and early 70s there was increasing experimentation with operating systems and programming languages in academia and research labs – computer science was being born as an academic discipline. Operating systems such as MTS, the Michigan Terminal System, and Multics, were being developed by groups other than the hardware vendor circle. The key nascent seed to breaking the hardware vendors' control of system software was UNIX, developed at Bell Labs in the early 70s, as a portable operating system, building on earlier multi-user OS's such as Multics, and a new programming language "C", built on earlier open-source languages such as BCPL. Bell Labs made UNIX and C available for academic research users at no cost, with source code – "open source". By the late 70s the University of California at Berkeley (UCB) was using this open source to make major advances and improvements to UNIX – the "BSD" (Berkeley Software Distribution) of UNIX. Berkeley freely distributed their source code to academia and encouraged improvements and fixes. But because BSD UNIX was a derivative work of the original AT&T (Bell Labs) UNIX, there were all sorts of convoluted legal arguments about what access should be provided to the source code, to whom, and at what cost. By the 80s AT&T was commercialising UNIX (as System V).

1980–2000: The "Proprietary Package" Era

The early 80s saw the introduction of the PC, and along with it increasing complexity of computer applications software – online databases, networks, and transaction processing. Commercial applications developers increasingly relied on a rapidly expanding smorgasbord of proprietary software packages and tools on mainframes, minicomputers, workstations, and PCs. Proprietary software packages and software add-ons were marketed by companies that specialised in software as well as the traditional hardware vendors. Small independent software companies founded in the late 70s, such as Microsoft and Oracle, were growing rapidly. Development and maintenance costs of software were no longer completely dominated by custom applications development costs. By the late 80s proprietary but often portable software packages and their maintenance costs started to dominate the cost of computer ownership. But PCs were putting computing power into the hands of the masses; the market for open source was emerging.

The next big breakthrough in "open source" came in 1984 with Richard Stallman, from MIT who coined the term "free software". Richard, and others, were becoming increasingly dissatisfied with the restrictions of proprietary software vendors, and the tendency for more and more necessary software to be proprietary (closed source). Richard's vision was twofold: encourage more software developers to make their source code "open", and to create a completely open source software platform (operating system, tools, compilers, etc.) – based on the UNIX design and philosophy. The organisation that Richard founded was the "Free Software Foundation" (see *www.gnu.org*). Richard and a coterie of very clever software developers proceeded to develop an open source licensing contract (GPL) and suite of software tools that were "open source" – gcc, bison, gawk, flex, mailx, etc. By the early 90s there was a suite of open source software that was all-you-need to run a small business or home office, except:

1. The operating system kernel
2. WYSIWYG (What You See Is What You Get, or "visual") applications such as word processing and spreadsheets.

Building an operating system kernel itself is not too difficult – the problem is making it robustly portable (to the myriad of different hardware configurations – processors, discs, external devices). A portable operating system kernel almost requires a "community effort", as no one developer is going to have access to all the platforms needed to test and deploy a robust and portable operating system.

In 1991, Linus Torvalds found the first missing piece – he created an open source UNIX-type kernel from scratch, Linux, (UNIX-type means supporting all UNIX applications, or exporting the same Applications Programming Interface (API) as commercial UNIX). The key to Linux's success is that Linus Torvalds "opened it up" for community improvement and development, and the software community jumped on the bandwagon. Since the early 90s Linux has gone from strength to strength. Other free UNIX-style kernels appeared, such as FreeBSD (*www.freebsd.org*). The key to the success of Linux has been the open source developer community throwing their weight behind it, to include advanced features such as:

- ***Autoconfiguration*** – the ability of the operating system to figure out what hardware the machine it is being installed on has, and allow for easy upgrades of hardware. Early versions of Linux required expert configuration, but now Linux autoconfigures about as well as Windows. This means that a novice can install Linux as easily as he or she can install Windows.
- ***Active desktops*** – software that manages the desktop and makes it easy to launch new applications, find files, etc. Early versions of UNIX were all text and command-line oriented, and UNIX users scorned "point and click" users. Early UNIX active desktops, such as HPs and SGIs, were clunky compared to Microsoft's ever improving desktops. In the late 90s the open source community counterattacked by developing several active desktops that were as full-featured, or more full-featured, than Windows – notably KDE and Gnome. Both of these desktops use "windows style" menus and task-bars, making it easy for Windows users to migrate to them.

Another key factor in the growth of Linux, and open source generally, was the rise of the internet as a tool for software developers to communicate, collaborate, and distribute software. By the late 90s there was an explosive growth in internet-enabled and international open source software development.

Aside from an operating system, the major missing piece of open source software in the mid 90s was desktop applications – the equivalent of Microsoft's Office suite – Word, PowerPoint, and Excel. In the mid 90s SUN Microsystems got behind an effort to build such a suite, called "Star Office", to try and take back some of the office desktop market. Early versions of Star Office were disappointing – OK for developing your own documents, but often unable to make sense of a Microsoft Word attachment. Such attachments were the "killer app" that tied desktops to Microsoft.

By the late 90s everyone and their dog was mailing Word attachments back and forth. If you could not open and read such attachments you were a hermit.

Creating a WYSIWYG word processor that can read and write Word documents is a monumental task. For starters, Microsoft does not openly publicise the "format" and "markups" of Word documents. How do all the myriad features such as font styles, pagination, anchors, tables of content, revisions, and diagrams all interact? Unlike other WYSIWYG document processors, such as WordPerfect, there is no easy way to see all the embedded 'codes' in a Word document. Even if you could see them, how would you make sense of all the rules that governed how they interact?

But by the early 00s, there was a credible attack on the last bastion of closed-source software – the "Open Office" effort (*www.openoffice.org*). It leveraged off the earlier StarOffice effort. It is about 99.9% Word and PowerPoint compatible – it gets diagrams in the wrong place sometimes and bullets change their style, but you can read almost any Word attachment with Open Office. Building a Microsoft-compatible word processor is the "Everest" of Open Source – the sheer difficulty can be gauged by the size of the document that defines the Open Office definition of document "markups" in XML – over a thousand pages. Open Office even has some nice features that are not in Word, like "word completion" (optionally guessing what the rest of the word you are typing is). The Open Office folks learnt from the Gnome and KDE efforts – it is not enough to deliver cut-down or minimalist open source. To capture the desktop market, you need to be better and still compatible.

On other fronts, more and more "backend" applications and software was open-source. Much of this was driven by the "World Wide Web" which was an open-source project from the outset. The earliest browsers, such as Mosaic, were open source, and later browsers, such as Netscape navigator, became open source. While Netscape has lost out in market share to Microsoft Explorer, there are plenty of open-source browsers around (such as KDE's Konquerer or Gnome's Mozilla). In other relevant areas, open source was king – Apache still dominates the Web server market, and Java tools and libraries are all open source. On the database side, mySQL is gaining more and more ground as an open-source enterprise database – not all the bells and whistles or performance of either DB2 or Oracle, but not far off either. In addition there is a growing collection of Java middleware libraries for building distributed applications – JSP, J2EE and others. The entire standard Java libraries developed by SUN and others, and many Java tools, are all open source.

Today, there is almost no arena of software for which there is not an open source alternative available to proprietary software. The growth in open source has led to websites such as *www.bkbits.net* and *www.sourceforge.net* devoted to making it easier to find, download, and install and maintain open source software.

An obvious nagging question is why are tens of thousands of individual software developers and companies contributing so much of their time and effort for free to develop open source and contribute it to the community? What is it that drives individuals and companies to do work for free! For individuals, the motivation is usually the honour and kudos that comes from being recognised as authors and contributors of open source software that is widely used and of value to the community. For companies, providing open source software is driven by market pressure.

OPEN SOURCE TODAY

The rising use of open source has hardly been unnoticed in the IT industry. The reaction of different major players in the industry has been at times schizophrenic, but by and large most major players in the IT industry (such as IBM, SUN, HP, and SGI) are supporting open source in one way or another. Both SUN and SGI are between a "rock and a hard place" though, as both derive much of their revenues from proprietary hardware and/or software (Solaris and Irix respectively). Open source and commodity hardware threatens all that. Other vendors, such as IBM and HP have moved with the times to the realisation that in the coming decades IT revenue will derive from services, support, and expertise, not just "boxes and closed software". Most IT tenders are a combination of hardware, software, and support costs and margins. When a tenderer puts "free" open source into the mix, the response to tender can deliver more hardware or support for the same fixed price (as opposed to "expensive" closed software with high annual maintenance costs). Thus a tenderer who bundles in open source often has a competitive advantage over a tenderer who relies on closed source software.

The company that has most to lose from open source is clearly Microsoft. One by one it has vanquished its desktop opponents, such as OS/2. But now Linux and Open Source are rising phoenix-like from the ashes. The obvious question then is to what extent Microsoft can slow or stop the advance of open source? In the late 90s Microsoft was largely insulated from the rise of open source by two factors – the immaturity of Linux (lack of autoconfiguration or office applications), and the "arms race" with its Office Suite. But the evidence is that both of these defensive cloaks against open source are falling away.

In the 90s more and more features were being added to office applications such as Word and PowerPoint. Developers of WYSIWYG competing tools such as Star Office and WordPerfect could hardly keep up. Indeed, users of older versions of the office Suite were forced to upgrade (the famous debacle where Word98 could not read Word95 documents). But by the 00s the wind was running out of the sails of the Word "feature creep" – Redmond developers had run out of ideas for new features to add to Word or PowerPoint; and Open Office was catching up.

So in the early 00s, open source has almost caught up to proprietary source across a whole range of platforms and applications. Several large government organisations worldwide have started legislative moves to give preferential treatment to open source in tenders and software procurement, notably Belgium, Spain, Bulgaria, Costa Rica and others, (see *http://global.bsa.org/usa/press/newsreleases//2003-06-12.1653.phtml*) and the state of Oregon in the USA (see *http://linuxtoday.com/infrastructure/2003041801626OSLLPB*). The rationale for the preferential treatment for open source includes providing some leverage to overcome preference that users and IT procurers have for "safe, tried, and true" IT solutions such as Microsoft Windows and Office. Microsoft and other organisations such as the Business Software Alliance (*www.bsa.org*) have responded aggressively, lobbying hard to oppose preferential treatment for the adoption of open source.

Another argument that is sometimes advanced is that open source is not really useful for business or enterprise applications. However, the growing popularity of mySQL for commercial database projects and Apache's domination of the web server market shows that businesses are already embracing open source. Recently, even an open source ERP/CRM product has become available, Compiere, from *www.compiere.org*, to compete with proprietary products from companies such as Peoplesoft.

The stakes are huge. The question is who will win and why? Our answer appears below. But first we need to examine the Pros and Cons of open source in all its myriad variations.

OPEN SOURCE PROS AND CONS

One's perspective on the relative value of open source depends almost entirely on one's position in the industry. The two extremes are academia and companies such as Microsoft who make a living by selling software that competes with open source. IT and Computer Science academia is almost universally in favour of open source, as it reduces the cost of IT infrastructure for teaching IT, removes the problems of software licence management, enables IT experts to adapt software, and provides an open research environment for students (they can build and experiment with software changes on production software). It is probably fair to say that neither academics nor companies like Microsoft would have an unbiased view of Open Source.

Based on current trends, it is probably fair to say that within five years Open Source products will be about as good as, and compatible with, most key Closed Source products from major "commodity" software vendors such as Microsoft. So the choice of Open Source or not will not be based on functionality, or compatibility with existing software, but rather issues such as price, support, and the Total Cost of Ownership (TCO).

The key question is not whether home users or computer hackers will use open source or not, but rather whether corporations and businesses will adopt open source wholesale. In some market sectors, such as biotechnology R&D, open source already has a majority stake.

The two key stakeholders in the open source battle are two groups that have no strong vested interest in the success or failure of open source:

1. IT procurers – typically managers of IT departments
2. IT Solutions Providers

Each of these sees open source from a different viewpoint.

OPEN SOURCE – THE IT PROCURER PERSPECTIVE

Every IT procurer uses a different set of metrics to evaluate tenders and IT solutions alternatives. But in most procurement the following issues commonly arise:

A. *Cost* – what is the cost of the IT system (hardware, software, and maintenance)?
B. *Support* – what downtime of the system and response time to hardware/software problems from the vendor is acceptable?
C. *Risk* – what level of risk (hardware, software, and support) is there in the proposed solution? Is it compatible with existing infrastructure (software/hardware)?
D. *Usability* – will the software meet user requirements and be acceptable to users?

An open source solution generally comes in at a lower cost, but also generally comes in at a lower level of support and at a higher level of risk. Organisations such as banks and others dependent on real-time OLTP (OnLine Transaction Processing) that have a strategic business dependency on a very high level of uptime, and 24/7 response time to problems are unlikely to move to open source en masse in the foreseeable future. Bastions of conservatism, such as banks, have propped up obsolete but highly reliable hardware/software, such as Tandem systems, long after they have been abandoned by the rest of the market. The key question is what level of support/risk is associated with open source software, and what is the trend?

One key concern with open source software is that there is almost no "guarantees" that come with it. Most IT procurers would like some sort of guarantee that the software that is provided is free of major defects, or if a defect is found it will be quickly fixed. However, recent history and common experience is that purchasing software from a major vendor such as Microsoft is no guarantee of freedom from bugs, reliability, or support. Arguably, most versions of UNIX, including Linux, are far less likely to crash at random than Windows. But for many IT procurers, Windows is a "safer bet". As in the 60s and 70s, when IBM ruled the roost, an IT manager has a strong defence against criticism if a Windows solution is proposed – "that is the 'standard', if it is not working Microsoft will fix it". By contrast, open source is a leap of faith – someone out there must have written this code and be able to fix it.

So the major disincentive to growth in open source is the (perceived) lower level of support and higher level of risk, coupled with conservatism of IT managers.

However the level of support for open source software is increasing. Two factors influence software support:

A. Availability of support – Companies and contractors that offer services in configuring, installing, and maintaining open source systems
B. Simplicity of support – for individuals and companies who wish to maintain open source software themselves

Proprietary closed-source software has the advantage that there is a single first point of contact for support – the vendor company that owns and maintains the software. The cost and level and quality of support from the vendor company might not be great, but at least an IT procurer had a first point of contact. With proprietary software, maintaining the software oneself is not feasible (as the customer does not have the source code). Instead the vendor company maintains and upgrades the software. The customer may pay an often-substantial annual maintenance or licence fee, or may need to purchase upgrades at regular intervals. All this certainty in support comes at a cost – both monetary and flexibility (a customer has little or no influence on what improvements and fixes are made by the vendor).

By contrast, open source offers extreme flexibility at little or no cost. The customer can choose when and where to make upgrades and install new versions, and often has a choice among several different versions of the open source (such as the multiple versions of supported Linux). Early versions of Linux however were quite difficult for a non-expert to install and maintain. By contrast, now the major Linux versions are about as easy to maintain and install as Microsoft Windows, especially with the introduction of packages (like rpm's), that make it easy to customise and maintain and install.

Any internet search will quickly reveal a wide range of expertise and support services for open source, ranging from larger companies such as Red Hat (*www.redhat.com*) to local smaller support companies in Australia (e.g., *www.linsup.com* in Sydney).

So open source is not just becoming increasingly sophisticated and available, but is becoming better supported. This narrows the gap in support and certainty that leads many conservative and risk-adverse IT procurers to go with proprietary closed-source software.

OPEN SOURCE – THE IT SOLUTIONS PROVIDER PERSPECTIVE

While there are several different definitions of what open source is, in viewing it from an IT solution provider's perspective, we will adopt the most liberal or radical model, which assumes that open source is free of any charge or warranty, and the end user is free to modify or resell it.

Historically, software companies have always preferred to market their own proprietary closed-source software products. The reasons are obvious: the high costs of converting from one software product to another often meant that customers are locked in to proprietary solutions, even if a better or cheaper solution from another vendor comes along. Software maintenance contracts on proprietary software provide a steady revenue stream to fund the development of the next generation of proprietary software. Obviously this is a self-sustaining system, which feeds off the tremendous inertia to change in large software companies. The revenue model of most software companies is based around licence fees and charges to fund development of new software products and maintenance of existing products. Open source software provides a tremendous threat to this whole economic model.

The economic model for companies marketing open source based IT solutions is quite different. Essentially it says that any development or maintenance costs for open source products must be recovered from configuration, installation, and support charges. Open source software development is thus a “cost centre” not a “profit centre” as in a proprietary company. Revenue of an open source based contract comes from services, support, and hardware margins.

Market pressure is forcing many, if not most IT solutions providers to offer open source solutions – either because the customer demands it, or because offering open source offers a competitive advantage (more services, support, and hardware can be offered at a lower price than a competing vendor that has had to bundle in proprietary software and recover its costs). The rise of Linux has forced some IT vendors into a strange split strategy of offering both proprietary (e.g., AIX, HP-UX, or IRIX) and open source solutions (Linux), often on the same bid. The rise of Linux has led to a general merger and decline in investment in proprietary UNIX versions.

Contracts with open source software components are not without problems for IT solutions providers. One of the major problems is managing liability and risk. With traditional proprietary software, the owner or supplier of the software assumed some of the risk and could be legally liable if some major flaw was found in the software. But if an IT solutions provider bundles open source into the solution, the last thing that the solution provider wants to be responsible for is any “warranty” for the open source. So such contracts typically have a disclaimer that the IT solution provider is not responsible for, or warranting, the open source software. In time a market may emerge for certification and warranty services for open source (e.g., for security and reliability).

IT vendors that benefit most from open source software are those who have a strong track record and market share in consulting and services; they stand to gain market share from open source contracts. Correspondingly, IT vendors that rely heavily on revenues from proprietary software that have strong competitors in the open source marketplace are most at risk.

Microsoft is the IT vendor that is most threatened by open source. Microsoft has adopted several defensive strategies. The first is to keep the API for Microsoft Office documents proprietary – the developers of Open Office have essentially had to reverse-engineer the markup strategy used in binary Office documents. While laborious, the developers of Open Office have been able to do this. Microsoft could counterattack by coming up with a new Office products and upgrades (e.g., Word2005 that was not fully upward and downward compatible with current Word versions), but this would cause a market backlash. The second defensive strategy has been to diversify into other products and proprietary APIs, such as .NET, C#, and Managed C++. IT developers and users that adopt these standards are locking themselves into Microsoft products. This strategy does not appear to have been a great success to date. While C# and .NET have been out in the market for a couple of years, they do not appear to be gaining major market share – the jury is still out on these, versus competing open standards and source such as J2EE. Microsoft’s final strategy is lobbying and political pressure – certainly any decline in Microsoft’s fortunes would have a huge economic impact.

The argument advanced against open source most commonly is that it will lead to a decline in investment in software development and maintenance. But in practice almost the opposite effect seems to be happening – an explosion of open source projects on sites such as sourceforge and bitkeeper. As of the end of June, sourceforge alone hosted over 65,000 projects! Instead of revenue being derived from licence fees to recover development costs, open source developers derive revenue from support and services. The open source community is utterly global and open; any entrepreneur in any country or village can “hang out their shingle” and let the market decide if their skills or products are competitive. If a product catches on, more and more developers will join that open source product bandwagon.

A VIEW FROM THE TRENCHES

At the Victorian Partnership for Advanced Computing (*www.vpac.org*), we operate an R&D facility with several hundred CPUs, running most versions on UNIX (Linux on multiple platforms; AIX, HP-UX, IRIX, Mac OS-X) and a mix of Windows (XP) and UNIX desktops. Some staff use Open Office (1.0.1), and some use Microsoft Office. Our experience is that Open Office is fine for workgroups that do not need to exchange documents back and forth with users of Microsoft Office. But Open Office Writer documents opened from Word often need a bit of a "cleanup" – bullet styles and indenting may be off a bit, and Word cannot edit Writer diagrams, and vice-versa. The incompatibility is just a little irritating for occasional exchange of documents, but more than irritating for users who need to exchange and share documents on a regular basis with Word users.

CONCLUSION

The open source versus proprietary software battle will reach its nexus in the next few years. The key battle is going to be fought over the office and home desktop – Linux and Open Office versus Microsoft Windows and Office. Right now, there is nothing to stop organisations making a move en masse to open source on the desktop. It suits workgroups who have limited or controlled document exchange with outside groups that use Word. The likely move to open source on the desktop will come from Academia (student access), followed by government departments and not-for-profit organisations, particularly in nations that do not have deep pockets. As more and more users and groups move to Linux and Open Office, and these products become more mature, Microsoft may find the tables turned on it, as it turned them a decade ago on Netscape and others. If a majority of users in a given market move to Linux and Open Office, Microsoft will be forced to either drop prices or make its products compatible with Linux and Open Office! To some extent this is happening now, as there is an increasing use of portable open standards for document interchange, such as PDF from Adobe (*www.adobe.com*) or RTF (Rich Text Format).

The issue is not so much whether open source will win or not, but rather how long it will take and what niches and markets will remain proprietary source. The momentum behind open source and the sheer international size of the open source developer community dwarfs even Microsoft's resources. Proprietary software niches and markets will always be there, and always be significant – either for commercial, security, or market size reasons. But they will no longer be the mainstay of the software market.

In a sense, we are at a turning point in the evolution of software – from the 60s when all software was custom crafted, to the 80s when all software was built on proprietary packages, till the "open source era" when software revenue comes from services and support, not proprietary packages. When will software historians announce that the "open source era" had arrived? That is the open question, along with who the winners and losers will be, and what the brave new world of IT providers will look like.

BIOGRAPHICAL NOTES

Professor Bill Appelbe is the Chief Executive Officer of the Victorian Partnership for Advanced Computing (VPAC) and formerly the Head of the Computer Science Department at RMIT.

Bill graduated with a BSc(Hons) from Monash in 1974, and a PhD in Computer Science and Electrical Engineering from UBC, Canada in 1978. He was on the faculty of several universities, including the University of California and San Diego and Georgia Institute of Technology (1986-1998).

His area of expertise includes both software engineering and programming tools. He led a team of programmers who developed a series of commercially and government funded tools for parallel programming and micro architecture simulation in the 1980s and 1990s, in the USA, including START/PAT and SuperDLX. Bill has been a developer and consultant on a number of large software projects, including design of satellite tracking and control software for Intelsat, object-oriented database software for Hewlett-Packard, and compilers for IBM.

He has authored advanced programming courses that have been taught to thousands of students in many countries worldwide since 1975 and has given presentations and talks in a dozen countries.

Bill has been a consultant and employee for numerous companies and organisations including the US Army, IBM, Hewlett-Packard, Sun Micro-systems, UNISYS, and Los Alamos National Laboratory. He has authored over fifty refereed conference papers and journal articles, primarily in the area of applied high-performance computing.

Bill Appelbe