A Short History of Software

Graeme Philipson

This document is the first draft of a chapter commissioned for a book on software development, to be published in 2004 by Routledge (more details as they come to hand)..

A Short History of Software

Introduction

The two key technologies in computing, hardware and software, exist side by side. Improvements in one drive improvements in the other. Both are full of major advances and technological dead ends, and both are replete with colourful characters and dynamic start-up companies.

But there are key differences between the hardware and the software industries. Hardware design and manufacture is a comparatively costly exercise, with a consequently high cost of entry. Nowadays only large, or largish, companies can do hardware. But many of the major software advances have been the results of individual effort. Anybody can start a software company, and many of the largest and most successful of therm have come from nowhere, the result of one or a few individual's genius and determination.

There are many different types of software. There is applications software, such as financial programs, word processors and spreadsheets, that let us do the sort of work we buy computers for. There is systems software, such as operating systems and utilities, that sit behind the scenes and make computers work. There are applications development tools, such as programming languages and query tools, that help as develop applications. Some types of software are mixtures of these – database management systems (DBMSs), for example, are a combination of applications, systems, and applications development software.

The software industry has made thousands of millionaires and not a few billionaires. Its glamour, its rate of change, its low cost of entry, and the speed at which a good idea can breed commercial success have attracted many of the brightest technical minds and sharpest business brains of two generations. Hardware is important, but in a very real sense the history of information technology is the history of software.

Software Before Computers

The first computer, in the modern sense of the term, is generally agreed to be the ENIAC, developed in the USA in the final years of World War II (see below). But the concept of software was developed well over 100 years earlier, in 19th century England.

Charles Babbage (1791-1871) was the son of a wealthy London banker. He was a brilliant mathematician and one of the most original thinkers of his day. His privileged background gave him the means to pursue his obsession, mechanical devices to take the drudgery out of mathematical computation. His last and most magnificent obsession, the Analytical Engine, can lay claim to being the world's first computer, if only in concept (Augarten, 1985: 44).

By the time of Babbage's birth, mechanical calculators were in common use throughout the world, but they were calculators, not computers – they could not be programmed. Nor could Babbage's first conception, which he called the Difference Engine. This remarkable device was designed to produce mathematical tables. It was based on the principle that any differential equation can be reduced to a set of differences between certain numbers, which could in turn be reproduced by mechanical means.

The Difference Engine was a far more complex machine than anything previously conceived. It was partially funded by the British government, and partially by Babbage's sizeable inheritance. He laboured on it for nearly twenty years, constantly coming up against technical problems. But the device too complex to be made by the machine tools of the day. He persevered, and was eventually able to construct a small piece of it that worked perfectly and could solve second-level differential equations.

The whole machine, had it been completed, would have weighed two tonnes and been able to solve differential equations to the sixth level. After battling with money problems, a major dispute with his grasping chief engineer, the death of his wife and two sons, and arguments with the government, the whole project collapsed (Augarten, 1985: 48). Part of the problem was Babbage's perfectionism – he revised the design again and again in a quest to get it absolutely right.

By the time he had nearly done so he had lost interest. He had a far grander idea – the Analytical Engine – which never came close to being built. This remarkable device, which lives on mainly in thousands of pages of sketches and notes that Babbage made in his later years, was designed to solve any mathematical problem, not just differential equations.

The Analytical Engine was a complex device containing dozens of rods and hundreds of wheels. It contained a mill and a barrel, and an ingress axle and egress axle. Each of these components bears some relationship to the parts of a modern computer. And, most importantly, it could be programmed, by the use of punched cards, an idea Babbage got from the Jacquard loom. The first programmer was Ada, Countess of Lovelace, daughter of the famously dissolute English poet Lord Byron.

Augusta Ada Byron was born in 1815 and brought up by her mother, who threw Byron out, disgusted by his philandering (O'Connor and Robertson, 2002). She was named after Byron's half sister, who had also been his mistress. Her mother was terrified that she would become a poet like her father, so Ada was schooled in mathematics, which was very unusual for a woman in that era.

She met Charles Babbage in 1833 and became fascinated with the man and his work (Augarten, 1985: 64). In 1843 she translated from the French a summary of Babbage's ideas which had been written by Luigi Federico Manabrea, an Italian mathematician. At Babbage's request she wrote some "notes" that ended up being three times longer than Manabrea's original.

Ada's notes make fascinating reading. *"The distinctive characteristic of the Analytical Engine … is the introduction into it of the principle which Jacquard devised for regulating, by means of punched cards, the most complicated patterns in the fabrication of brocaded stuffs … we may say most aptly that the Analytical Engine weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves."* (quoted in O'Connor and Robertson, 2002)

Her notes included a way for the Analytical Engine to calculate Bernoulli numbers. That description is now regarded as the world's first computer program.

Ada Lovelace's life was beset by scandalous love affairs, gambling and heavy drinking. Despite her mother's best efforts, she was very much her father's daughter. She considered writing a treatise on the effects of wine and opium, based on her own experiences. She died of cancer in 1852, aged only 37. Ada's name lives on in the Ada programming language, devised by the US Department of Defense.

Alan Turing and the Turing Machine

Alan Turing was a brilliant English mathematician, a homosexual misfit who committed suicide when outed, and one of the fathers of modern computing. He was one of the driving forces behind Britain's remarkable efforts during World War II to break the codes of the German Enigma machines. He is best known for two concepts that bear his name, the Turing Machine and the Turing Test.

He conceived the idea of the Turing Machine (he did not call it that – others adopted the name later) in 1935 while pondering German mathematician David Hilbert's *Entscheidungsproblem*, or Decision Problem, which involved the relationship between mathematical symbols and the quantities they represented (Hodges, 1985: 80). At the time Turing was a young Cambridge graduate, already recognised as one of the brightest of his generation.

The Turing machine, as described in his 1936 paper "*On Computable Numbers, with an application to the* Entscheidungsproblem", was a theoretical construct, not a physical device. At its heart is an infinitely long piece of paper, comprising an infinite number of boxes, within which mathematical symbols and numbers could be written, read and erased. Any mathematical calculation, no matter how complex, could be performed by a series of actions based on the symbols (Hodges, 1985: 100)

The concept is a difficult one, involving number theory and pure mathematics, but it was extremely influential in early thinking on the nature of computation. When the first electronic computers were built they owed an enormous amount to the idea of the Turing Machine. Turing's "symbols" were in essence computer functions (add, subtract, multiply, etc.), and his concept of any complex operation being able to be reduced to a series of simple sequential operations is the essence of computer programming.

Turing's other major contribution to the theory of computing is the Turing Test, used in artificial intelligence. Briefly, it states that if it is impossible for an observer to tell whether questions she asks are being answered by a computer or a human, and they are in fact being answered by a computer, there for all practical purposes that computer may be assumed to have reached a human level of intelligence.

The Birth of Electronic Computing

The first true electronic computer was the ENIAC (Electronic Numerator, Integrator, Analyzer and Computer). In 1942 a 35 year old engineer named John W. Mauchly wrote a memo to the US government outlining his ideas for an "electronic computor" (McCartney, 1999: 49). His ideas were ignored at first, but they were soon taken up with alacrity, for they promised to solve one of the military's most pressing problems.

That was the calculation of ballistics tables, which were needed in enormous quantities to help the artillery fire their weapons at the correct angles. The US government's Ballistics Research Laboratory commissioned a project based on Mauchly's proposal in June 1943. Mauchly led a team of engineers, including a young graduate student called J. Presper Eckert, in the construction of a general purpose computer that could solve any ballistics problem and provide the reams of tables demanded by the military.

The machine used vacuum tubes, a development inspired by Mauchly's contacts with John Atanasoff, who used them as switches instead of mechanical relays in a device he had built in the early 1940s (Augarten, 1985: 114). Atanosoff's machine, the ABC, was the first fully electronic calculator. ENIAC differed significantly from all devices that went before it. It was programmable. Its use of stored memory and electronic components, and the decision to make it a general purpose device, mark it as the first true electronic computer.

But despite Mauchly and Eckert's best efforts ENIAC, with 17,000 vacuum tubes and weighing over 30 tonnes, was not completed before the end of the war. It ran its first program in November 1945, and proved its worth almost immediately in running some of the first calculations in the development of the H-Bomb (a later version, appropriately named MANIAC, was used exclusively for that purpose).

By modern day standards, programming ENIAC was a nightmare. The task was performed by setting switches and knobs, which told different parts of the machine (known as "accumulators") which mathematical function to perform. ENIAC operators had to plug accumulators together in the proper order, and preparing a program to run could take a month or more (McCartney, 1999: 90-94).

ENIAC led to EDVAC (Electronic Discrete Variable Computer), incorporating many of the ideas of John von Neumann, a well-known and respected mathematician who lent a significant amount of credibility to the project (Campbell-Kelly and Aspray, 1996:92). Neumann also bought significant intellectual rigour to the team, and his famous paper "report on EDVAC" properly outlined for the first time exactly what an electronic computer was and how it should work. Von Neumann's report defined five key components to a computer – input and output, memory, and a control unit and arithmetical unit. We still refer to the "Von Neumann architecture" of today's computers.

When the war was over, Mauchly and Eckert decided to commercialise their invention. They developed a machine called the UNIVAC (Universal Automatic Computer), designed for general purpose business use. But they were better engineers than they were businessmen, and after many false starts their small company was bought by office machine giant Remington Rand in 1950. The first commercial machine was installed in the US Census Bureau.

UNIVAC leapt to the forefront of public consciousness in the 1952 US presidential election, where it correctly predicted the results of the election based on just one hour's counting. It was not a particularly impressive machine by today's standards (it still used decimal arithmetic, for a start), but nearly 50 of the original model were sold.

The 1950s was a decade of significant improvements in computing technology. The efforts of Alan Turing and his Bletchley Park codebreakers during World War II led to a burgeoning British computer industry. Before his death, after studying von Neumann's EDVAC paper, Turing designed the ACE (Automatic Computing Engine), which led to the Manchester Mark I, technically a far superior machine to ENIAC or EDVAC (Augarten, 1984: 148). It was commercialised by Ferranti, one of the companies that was later to merge to form ICL, the flag bearer of the British computer industry.

The most significant US developments of the 1950s were the Whirlwind and SAGE projects. MIT's Whirlwind was smaller than ENIAC, but it introduced the concepts of real-time computing and magnetic core memory. It was built by a team lead by Ken Olsen, who later

founded Digital Equipment Corporation, the company that led the minicomputer revolution of the 1970s (Ceruzzi, 1999: 140).

SAGE (Semi-Automatic Ground Environment) was a real-time air defence system built for the US government in the Cold War. The project was accorded top priority, with a virtually unlimited budget. In a momentous decision, the government awarded the contract to a company that had only just decided to enter the computer industry. That company's name was IBM.

SAGE broke new ground on a number of fronts. The first was its sheer size. There were 26 data centres, each with a 250 tonne SAGE mainframe. It was built from a number of modules that could be swapped in and out. It was the world's first computer network, using the world's first fault-tolerant computers and the world's first graphical displays. And it gave IBM a head start in the computer industry that it has retained ever since (Augarten, 1985: 204).

By the end of the 1950s there were dozens of players in the computer industry. Remington Rand had become Sperry Rand, and others like RCA, Honeywell, General Electric, Control Data and Burroughs had entered the field. The UK saw the likes of Ferranti and International Computers and Singer, and continental Europe Bull and Siemens and Olivetti. In Japan, a 40 year old company called Fujitsu moved into computers.

All these machines, of course, ran software, but there was no software industry as we understand it today. Early commercial machines were programmed mechanically, or by the use of machine language. In the early days there was little understanding of the distinction between hardware and software. That was to change with the development of the first programming languages.

Programming Languages and Operating Systems

The term "software" did not come into use until 1958. It is probable that it was coined by Princeton University professor John W. Tukey in an article in The American Mathematical Monthly in January of that year (Peterson, 2000).

The word "computer" was originally applied to humans who worked out mathematical problems. ENIAC was designed to take over the work of hundreds of human "computers" who were working on ballistics tables. Most of them were women, recruited from the best and brightest college graduates when the men went off to war. Thus, the first computer programmers were, like Ada Lovelace, women.

The most famous and influential of them was Grace Murray Hopper, a mathematician who joined the US naval reserve during the war and who rose to become an Admiral. She died in 1992. In 1951 Hopper joined Eckert and Mauchly's fledgling UNIVAC company to develop an instruction code for the machine. She devised the term "automatic programming" to describe her work (Campbell-Kelly and Asprey, 1996: 187).

She also used the word "compiler" to describe "a program-making routine, which produces a specific program for a particular problem" (quoted in Cerruzi, 1999: 85). Today the term "compiler" means a program that translates English-like instructions into binary code, but Hopper used the term to describe a way of handling predefined subroutines, such as those hard-wired into the ENIAC. The first compiler in the modern sense of the word was devised for the MIT Whirlwind project in 1954 by J.H Laning and N. Zierler (Ceruzzi, 1999: 86).

Hopper became a tireless proselytiser for the concept of automatic programming. Her work led directly to the development of FORTRAN ("FORmula TRANslator"), the world's first true computer language. FORTRAN was developed in the mid-50s by an IBM development team led by a young researcher named John Backus.

The first version of FORTRAN was released in 1954. There were many skeptics who believed that it would be impossible to develop a high-level language with anything like the efficiency of machine language or assembler, but Backus argued for FORTRAN on economic grounds. He estimated that half the cost of running a computer centre was the programming staff (Cambell-Kelly and Asprey, 1996: 188), and he saw in FORTRAN a way of vastly improving programming productivity.

He was right. FORTRAN enabled people to program computers using simple English-like instructions and mathematical formulas. It led to a number of other languages, the most successful of which was COBOL (Common Business-Oriented Language), initially developed by the US government Committee on Data Systems and Languages (CODASYL) and strongly promoted by Grace Hopper. COBOL and FORTRAN dominated programming until the late 1970s. Other languages, such as ALGOL (Algorithmic Language), PL/1 (programming language 1), RPG (Report Program Generator) and BASIC (Beginner's All-purpose Symbolic Instruction Code) also became popular, inspired by the success of FORTRAN and COBOL.

These languages became known as 3GLs (third generation languages), so called because they were an evolution from the first and second generations of computer language – machine code and assembler. Some people wrote bits of applications in those more efficient but much more cryptic languages, but the English-like syntax of the 3GLs made them easier to learn and much more popular.

3GLs brought a discipline and a standardisation to program design. They enabled programs to be structured, or ordered in a hierarchical fashion, usually comprising modules with a limited number of entry and exit points. Structured programming led to structured design, comparatively simple methodologies which set out ways in which these modules could be strung together. Soon, the term "systems analysis" came to be used to describe the process of collecting information about what a computer system was intended to do, and the codification of that information into a form from which a computer program could be written.

Not only did they not have programming languages, early computers also did not have operating systems. Every function had to be separately programmed, and in the early days there was no distinction between systems and applications software. Programming languages such as FORTRAN and COBOL greatly improved general programming functions, but the task of handling machine-specific functions such as control of peripherals was still left up to individual programmers.

Most of the innovative early work on what we now call operating systems was done by individual users (Ceruzzi, 1999: 96). One such system, designed at General Motors in 1956, evolved into IBM's well-known JCL (Job Control Language), a basic operating system designed for punch card systems that would tell the computer which cards were data and which were instructions.

The first true operating system is generally agreed to be MAD (Michigan Algorithmic Decoder), developed at the University of Michigan in 1959 (Ceruzzi, 1999: 98). MAD was

based on the ALGOL 3GL, and was designed to handle the various details of running a computer that were so tedious to code separately. But the concept of the operating system was still largely unknown, until the momentous development of IBM's S/360.

The IBM System/360 and OS/360

April 1964 marks the beginning of the modern computer industry, and by extension the software industry. In that month IBM released the System/360, its revolutionary mainframe architecture. The 19 models in the S/360 range comprised the first ever family of computers, the first with a consistent architecture across computers of a different size. They could use the same peripherals and software, making it very easy to move to a larger computer within the range, and to move on to new models as they were released.

The idea of a family of computers seems quite normal now, but back in the early 1960s it was revolutionary. Previous IBM machines, such as the 1401, were incompatible with other machines in IBM's range. Every time IBM, or anybody else, brought out a new computer, users had to rewrite their entire applications suite and replace most of their peripherals.

IBM bet the company on the S/360, investing over $US5 billion and 350,000 man-years (Watson, 1990: 340). It was the largest R&D project ever undertaken by a commercial organisation. It was an enormous gamble, but it paid off. The S/360 (the numbers were meant to indicate the points of the compass) was more than twice as successful as IBM had hoped, and it became virtually the standard operating environment for large corporations and government agencies.

The S/360 evolved into the S/370, and then into the S/390, and then into today's zSeries, with many of its features intact. A piece of software written for the first S/360 will still run on today's zSeries machines. Predictions of the death of the mainframe, common 10 to 15 years ago, have proved wildly inaccurate, and the S/360's successors still power most large transaction processing systems today.

But the S/360's success was not pre-ordained. Many within the company argued against it. The machine was rushed into production, and IBM could not handle the demand, its internal accounting and inventory control systems buckling under the strain. IBM's chairman at the time, Thomas J. Watson Jr, recounts the story (Watson, 1990: 349)

"By some miracle hundreds of medium-sized 360s were delivered on time in 1965. But … behind the scenes I could see we were losing ground. The quality and performance … were below the standards we'd set, and we'd actually been skipping some of the most rigorous tests … everything looked black, black, black. Everybody was pessimistic about the program …

"… we were delivering the new machines without the crucial software; customers were forced to use temporary programs much more rudimentary than what we'd promised … with billions of dollars of machines already in our backlog, we were telling people they'd have to wait two or three years for computers they needed … I panicked."

But IBM recovered and the S/360 became the most successful computer in history. It introduced a number of innovations, such as the first transaction processing system and the first use of solid logic technology (SLT), but it was no technical miracle. Its peripherals performed poorly, its processor was slow, its communications capabilities were virtually non-existent.

Most importantly, the S/360 introduced the world's first sophisticated operating system, OS/360. This was both the architecture's biggest advance, and also its biggest problem. OS/360 was by far the largest software project ever undertaken, involving hundreds of programmers and more than a million lines of code (Campbell-Kelly and Asprey, 1996: 197). In charge of the project was Fred Brooks, who was to become the father of the discipline known as software engineering. Brooks's book on the development of OS/360, *The Mythical Man-Month*, remains one of the all-time classic descriptions of software development.

The task facing Brooks and his team was massive. They had to design an operating system that would work on all S/360 models, and which would be able to enable the machines to run multiple jobs simultaneously, a function known as "multitasking". At its peak the OS/360 project had more than 1000 programmers working on it, which led Brooks to his famous conclusion that software cannot be designed by committee, and that there are no necessary economies of scale: "the bearing of a child takes nine months, no matter how many women are assigned" (Brooks, 1995: 17).

OS/360 was eventually delivered, years late and millions of dollar over budget. It was never completely error-free, and bugs kept surfacing throughout its life. IBM eventually spent over half a billion dollars on the project, the biggest single cost of the entire S/360 project (Campbell-Kelly and Asprey, 1996: 200). But it was the archetype of the operating system, the precursor of all that have followed it.

With all its problems the S/360 was an architecture, the first the computer industry had ever seen. It excelled nowhere, except as a concept, but it could fit just about everywhere. For the first time upward and downward compatibility was possible. The phrase "upgrade path" entered the language. With the S/360, IBM also promised compatibility into the future, protecting customers' investment in their applications and peripherals. With the S/360 and OS/360, the operating system became the key distinguishing factor of a computer system.

Soon other companies began making "plug-compatible computers" that would run S/360 software. The best known of these was Amdahl, started by IBM engineer Gene Amdahl, who had led the deign team for the S/60. Amdahl released its first IBM-compatible mainframe in 1975.

Software Comes of Age

Largely as a result of the problems highlighted by the development of OS/360, there was an increased realisation in the late 1960s that software development should be regarded as a science, not an art. A seminal conference held in Garmisch, Germany, in October 1968, entitled "Software Engineering", brought together many of the world's leading software designers. The conference was sponsored by NATO, whose member states were desperate to bring some order into software development for the military, the costs of which were spiralling out of control (Cerruzi, 1999: 105).

The Garmisch conference marked a major cultural shift in perceptions of what software was and how it should be developed (Cambell-Kelly and Asprey, 1996: 201). The term "engineering" was used by adherents of the approach that building software was like building a house or a bridge – there were certain structured techniques and formal design methodologies that could be applied to the complexity of writing software. This approach

marked the beginnings of "structured design methodology", which became enormously popular and influential in the 1970s and 1980s.

At around the same time another major development occurred. In December 1968, just two months after the Garmisch conference, IBM made the decision to "unbundle" its software. The two events are unrelated, but together they constituted a revolution in software and software development.

Until 1969 IBM included all its software with the computer. Buy (or lease) the computer, and the software was thrown in as part of the deal. Hardware was so expensive they could afford to give the software away. But in 1968, under pressure from the US government, which was soon to initiate an anti-trust suit against IBM, the company started to charge separately for its systems software (Ceruzzi, 1999: 106). The first piece of software to be sold separately was CICS (Customer Information Control System), IBM's widely used transaction processing system.

The effect of IBM's decision was to open up the software market to independent software vendors (ISVs), who could compete against IBM. The changed environment, and the vast increase in the use of computers, led to the emergence of software contractors and service houses. By the mid1970s, there were hundreds of software suppliers, many of them developing the first software packages – pre-written software that could be purchased off the shelf for any number of applications. Most of the new companies were formed to supply software for IBM and compatible mainframes. Such was the growth in computing that even a company of IBM's size was not able to come close to keeping up with demand.

Software was moving from being a cottage industry to the mass production model. Many companies came into being to supply applications software, such as financial and manufacturing packages. Leading applications software companies that were formed in this era included McCormack and Dodge, MSA, and Pansophic. By 1972, only three years after the unbundling decision, there were 81 vendors in the US offereing packages in the life insurance industry alone (Campbell-Kelly and Asprey, 1996: 204).

There was also a large group of companies formed to supply systems software. Many of these were one-product companies, formed by individuals who had worked out a better way to perform some operating function. This included tape backup, network management, security, and a host of other features that OS/360 and its successors were not able to do well. Systems software companies from this era included SKK, Compuware, Dusquene, Goal Systems, BMC, Candle, and many more. Most of them eventually merged with each other. Many ended up being acquired by the largest of them all, Computer Associates, started by Chinese immigrant Charles Wang in 1976 (Campbell-Kelly and Asprey, 1996: 205).

But the largest and most successful group of independent software vendors were the database management system (DBMS) suppliers.

Database Management Systems

The development of programming languages and the standardisation of operating systems brought some order to software, but data management remained an important issue. During the 1950s and the early 1960s there was no standardised way of storing and accessing data, and every program had to manage its own. The file structure was often determined by the

physical location of the data, which meant that you had to know where data was, and how the program worked, before you could retrieve or save information.

The mid 1960s saw the first attempts at solving this problem. In 1963 IBM developed a rudimentary data access system for NASA's Apollo missions called GUAM (Generalized Update Access Method), and in 1964 Charles Bachmann developed a data model at Honeywell which became the basis for IDS (Integrated Data Store), the first true database (Crawford and Ziola, 2002). A company called Informatics, one of the first independent software vendors, developed a successful file management system called Mark IV in 1967 (Campbell-Kelly and Asprey, 1996: 204). After IBM unbundled software in 1968, Informatics became the first successful independent software vendor. And IBM developed IMS, still in use today, in the late 1960s.

Boston-based Cullinet, founded by John Cullinane in 1968, bought a data management system called IDMS (Integrated Data Management System) from tyre company BF Goodrich and turned it into a successful mainframe product. In 1969 Germany's Peter Schnell founded a company called Software AG, which developed hierarchical file management system called Adabas.

There were many others. Most of them used a hierarchical system, where data relationships are represented as a tree. This model was codified by CODASYL in 1969. But the biggest revolution in data management came with the concept of the relational database. Like many other developments in the history of computing, it came from IBM.

In 1970 IBM researcher E.F (Ted) Codd wrote a seminal paper called "A Relational Model of Data for Large Shared Data Banks". It became one of the most famous and influential documents in the history of computing. It described how data could be stored in tables of related rows and columns. This became known as an RDBMS – a relational database management system. The idea revolutionised the IT industry. It meant that data structures could be standardised, so that different programs could use the same data, and that applications could take their data from different sources. Back in 1970 it was a revolutionary idea.

Ted Codd was born in England in 1923. He studied mathematics and chemistry at Oxford, and was a captain in the Royal Air Force during World War II. He moved to the USA after the war, briefly teaching maths at the University of Tennessee before joining IBM in 1949 (Philipson, 2003). His first job at IBM was as a programmer on the SSEC (Selective Sequence Electronic Calculator), one of Big Blue's earliest computers. In the early 1950s he was involved in the development of IBM's STRETCH computer, one of the precursors to today's mainframes. He completed a masters and doctorate in computer science at the University of Michigan in the 1960s, on an IBM scholarship.

Codd wrote his paper while working in IBM's San Jose Research Laboratory. IBM saw the potential of the idea, and initiated a project called System R to prove the concept in a real application. Codd expanded his ideas, publishing his famous twelve principals for relational databases in 1974.

RDBMSs were a revolution in IT theory and practice. Codd's work made database management a science, and vastly improved the efficiency, reliability and ease of use of computer systems. Relational databases are today the basis of most computer applications – indeed, it is impossible to imagine computing without them.

Ted Codd retired from IBM in 1984. His career is dotted with awards recognising his achievements. He was made a Fellow of the British Computer Society in 1974, and an IBM Life Fellow in 1976. He received the Turing Award, probably the highest accolade in computing, in 1981. In 1994 he was made a Fellow of the American Academy of Arts and Sciences. He died in 2003 (Philipson, 2003).

There have always been some, such as Software AG's Peter Schnell and Cincom's Tom Nies, who have doubted the validity of the relational model and pursued other directions, but their models have faded, even as the relational model has grown stronger. Codd himself admitted to limits in the capabilities of relational databases, and he became a critic of SQL (Structured Query Language), a standard language for querying RDBMSs which was also developed as part of the IBM System R project. But his was one of the greatest software innovations in the history of IT. It led directly to the foundations of companies like Oracle and Informix, and to the database wars of the 1980s and 1990s.

IBM was not the first to market with an RDBMS. That honour goes to a company called Relational Technology with a product called Ingres. Second was Relational Software with Oracle. Both these companies eventually renamed themselves after their better-known products. Ingres had some modest success before being eventually acquired by Computer Associates, and Oracle under co-founder Larry Ellison went on to become one of the largest and most successful companies in the IT industry (Symonds, 2003: 72). IBM's own development was slow. It released SQL/DS in 1981 and DB2, for mainframes, in 1983. Other companies, such as Informix and Sybase, entered the fray, and the DBMS market became one of the most hotly-contested in the software industry.

Today the DBMS market has changed significantly. Most of the big players of ten years ago are gone or irrelevant, and IBM and Oracle are the only ones left. They have been joined by Microsoft as the only games in town. Microsoft's Access has seen off its competitors in the PC DBMS market, and it has had great success with its high end SQL/Server DBMS. SQL/Server runs only on Microsoft's Windows operating system, but that has not held it back. Windows has been the big operating system success story of the last ten years, halting Unix in its tracks and destroying most proprietary operating systems (see below).

The Rise of the Minicomputer

By the mid 1960s computers were in common use in government and industry throughout the world. Owing to the tremendous success of the S/360, IBM was the industry leader, as large as its major competitors combined. Its main competitors were known collectively called the "Bunch" – a clever acronym for Burroughs, Univac (by this time the company was called Sperry, though it still used Univac as a model name), NCR, Control Data and Honeywell. IBM and the Bunch sold large and expensive mainframe computers to government departments and large corporations. But some people thought that computers need not be that big, nor cost that much.

One of these was Ken Olsen, who had worked on the Whirlwind project. In 1957 he started a small company called Digital Equipment Corporation, better known as DEC, in an old wool mill in Boston. DEC's first products were small transistorised modules that could be used to take the place of the vacuum tubes still used by computers of that era.

These modules proved very popular, and DEC was soon making so many different types that Olsen decided to build his own computers based around them. The first of these, the PDP-1

(the PDP stood for Programmed Data Processor), was released in 1960. It was followed by the PDP-5 in 1963 and the PDP-8 in 1965 (Augarten, 1984: 257).

The PDP-8 was revolutionary. It ushered in the minicomputer revolution and brought computing to a whole new class of users. It had just 4K of memory, but it was a real computer, and much less expensive ($US18,000) than any other machine on the market. It was an enormous success, especially with scientists and engineers, who finally had a computer they could afford and easily use.

In 1970 DEC released the equally successful PDP-11. New technology and economies of scale meant that the prices of DEC's minicomputers kept dropping as quickly as their capabilities improved, and soon DEC had many competitors. One of the most successful, Data General, was founded in 1968 by ex-DEC employees. The Data General Nova, announced in 1969, set new benchmarks for price-performance, and by 1970 over 50 companies were making minicomputers (Augarten, 1984: 258).

These included IBM and the Bunch, but they moved slowly and largely missed the act. Only IBM caught up, by the end of the 1980s, with its AS/400. The big winners were Data General and other startups and companies new to the industry like Prime, Hewlett-Packard and Wang. In 1977 DEC released the VAX, the world's most successful minicomputer, still in use today. DEC was acquired by Compaq in 1997, then became part of Hewlett-Packard in 2002.

If smaller was better, smaller still was better still. Some people even believed that computers could be made small enough and cheap enough that they could be bought and used by individuals. For this to happen, computers needed to get much smaller and cheaper. Technology, as always, was to make this possible.

Early computers used vacuum tubes as switches. These were soon replaced by transistors, invented by Bell Labs' William Shockley, John Bardeen and Walter Brattain in 1948. The three received the Novel prize for their achievement. Transistors worked in solid state – the switching depended on the electrical properties of a piece of crystal – and led to further developments in miniaturisation throughout the 1950s and 1960s. The next significant development was the invention of the integrated circuit (IC) by Jack Kilby at Texas Instruments in 1959. ICs combined many transistors onto a single chip of silicon, enabling computer memory, logic circuits and other components to be greatly reduced in size.

Electronics was becoming a big industry. After he left Bell Labs, Shockley started a company to commercialise the transistor. His acerbic personality led to problems with his staff, and eight of them left in 1957 to found Fairchild Semiconductor. Two of those eight, Gordon Moore and Bob Noyce, in turn founded their own company, Intel, in 1968. All these companies were located in the area just north of San Jose, California, an area dubbed "Silicon Valley" in a 1971 Electronic News article by journalist Don Hoefler.

The next step was the development of the microprocessor, conceived by Intel engineer Marcian Hoff in 1970 (Augarten, 1984: 265). Hoff's idea was simple. By putting a few logic circuits onto a single chip, the chip could be programmed to perform different tasks. The first microprocessor, the 4004, was developed as a cheap way of making general purpose calculators. That was to lead directly to the microcomputer revolution of the late 1970s and 1980s.

The Birth of the Microcomputer

Intel's 4004 microprocessor was not an immediate success. No-one really knew what to do with, but sales picked up as its flexibility became apparent. In 1972 Intel released a vastly improved version called the 8008, which evolved in 1974 into the 8080. People started to realise that these devices were powerful enough to run small computers.

In July 1974 Radio-Electronics magazine announced the Mark 8, "Your Personal Minicomputer", designed around the 8008 by Virginia postgraduate student Jonathan Titus (Augarten, 1984: 269). You had to send away for instructions on how to build it, but thousands did. It success inspired rival magazine Popular Electronics to announce the "World's First Minicomputer Kit to Rival Commercial Models" in its January 1975 issue. The device, the Altair 8800, was designed by Ed Roberts, who ran a small electronic company in Albuquerque called MITS. MITS sold Altair kits for less than $US400, at a time when the cheapest DEC PDP-8 cost more than ten times as much. Roberts was swamped with orders, and the microcomputer revolution had begun (Campbell-Kelly and Asprey, 1996: 242).

One of the Altair's great strengths was its open architecture. Roberts deliberately designed it so that others could add to it by developing plug-in cards. Soon hobbyists and small companies all over America soon began developing Altair cards. But, like the early mainframe computers, the Altair had no software. You programmed it by flicking switches on its front panel.

Then one day a Harvard undergraduate called Paul Allen noticed the Altair story in the magazine. He and his friend Bill Gates had been playing with computers since their high school days in Seattle. Allen suggested to Gates that they write a BASIC interpreter for it. Allen rang Roberts in Albuquerque, Gates wrote the compiler in six weeks, and the two drove to New Mexico, Gates finishing off the software in the parking lot before their meeting with Roberts (Freiberger and Swaine, 1984:143). The compiler worked, and Gates and Allen dropped out of Harvard and started a company they called Micro-Soft around the corner from MITS in Albuquerque. Soon after they dropped the hyphen and moved their small company back to their hometown in the Pacific Northwest.

The Altair spawned a host of imitators. Computer clubs sprang up across the world. The most famous, located of course in Silicon Valley, was the Homebrew Computer Club. Two of the club's most active members were Steve Wozniak and Steve Jobs, who teamed up to build a little computer called the Apple I, powered by the 6502 microprocessor from a small Silicon Valley company called MOS.

The Apple I was moderately successful, so the two Steves decided to go into business properly. Jobs sold his VW microbus and Wozniak his HP calculator, they borrowed $5000 off a friend, and they were in business. They soon attracted the attention of venture capitalist Mike Markkula, who believed that microcomputers were the Next Big Thing. He was right.

Apple released the Apple II in the middle of 1977, about the same time as commercial designs from Tandy (the TRS-80) and Commodore (the PET). But the Apple II outsold them both, because of its attractive design and its greater ease of use. In 1980 Apple went public, in the most successful float in Wall Street history (Carlton, 1997: 10)

The Apple II used its own proprietary operating system, called simply Apple DOS, but the most popular microcomputer operating system was call CP/M (Control Program /

Microcomputers). CP/M ran on most microcomputers that used the popular Z80 microprocessor from Zilog.

The low cost and ease of programming microcomputers spawned a software industry to rival that of the mainframe and minicomputer world. Most early applications were amateurish, but microcomputer software came of age with the invention of the spreadsheet in 1979.

The PC Software Industry

The first spreadsheet program, called VisiCalc, was released for the Apple II in November 1979. It was the brainchild of Dan Bricklin, who had the idea while doing an MBA at Harvard (Freiberger and Swaine, 1984: 229). Bricklin's professors had described to him the large blackboards divided into rows and columns that were used for production planning in large companies (Cringely, 1992: 65). Bricklin reproduced the idea electronically, using a borrowed Apple II.

Visicalc and its many imitators revolutionised accounting and financial management. Soon large companies were buying Apple IIs by the dozen, just to run Visicalc. With the release of Mitch Kapor's Lotus 1-2-3 for the IBM PC (see below), the spreadsheet became a standard microcomputer application.

The other key microcomputing application was the word processor. Word processors evolved from typewriters rather than computers, but with the advent of the microcomputer the two technologies merged. The first word processor was IBM's MT/ST (Magnetic Tape/Selectric Typewriter), released in 1964 (Kunde, 1996), which fitted a magnetic tape drive to an IBM Selectric electric typewriter.

In 1972 word processing companies Lexitron and Linolex introduced machines with video displays that allowed text to be composed and edited on-screen. The following year Vydec introduced the first word processor with floppy disk storage. All these early machines, and many that came afterwards from companies like Lanier and NBI (which stood for "Nothing But Initials") were dedicated word processors – the instructions were hardwired into the machines.

The first word processing program for microcomputers was Electric Pencil, developed for the MITS Altair by Michael Shrayler in 1976. It was very rudimentary. The first to be commercially successful was Wordstar in 1979, developed by Seymour Rubinstein and Rob Barnaby (Kunde, 1996). Wordstar used a number of cryptic commands, but it had all the power of a dedicated word processor. By the time the IBM PC was released in 1981, PCs and word processing machines had all but converged in technology and appearance. It took some time before word processing software caught up with dedicated machines in functionality, but they won the battle in price-performance immediately. All the dedicated word processing companies were out of business by 1990.

Spreadsheets and word processors led the way, but there were many other types of PC applications. PC databases became popular, the leading product for most of the decade dBase II and its successors, dBase III and dBase IV. The growth of the microcomputer software industry in the 1980s mirrored the growth of mainframe and minicomputer software in the 1970s (see above). Leading companies of the era included WordPerfect, Lotus (acquired by IBM in 1995), Ashton-Tate, Borland. And, of course, Microsoft.

The IBM PC and the Rise of Microsoft

The success of the Apple II and other early microcomputers persuaded IBM to enter the market. In July 1980 Bill Lowe, head of IBM's entry level systems division made a presentation to IBM senior management about why Big Blue should make a move. More importantly, he suggested how this could be done.

The key to doing it quickly, said Lowe, was to use standard components. This was a major departure for IBM, which normally designed and built everything itself. There was no time for that, argued Lowe. Management agreed, and he was told to go off and do it. The building of the IBM PC was given the name Project Chess, and the machine itself was internally called the Acorn (Campbell-Kelly and Asprey, 1996: 255).

The machine was ready in less than a year. It was a triumph of outsourcing. The microprocessor was an Intel 8088. Microsoft supplied the operating system and a version of the BASIC programming language. Disk drives (just two low capacity floppies) were from Tandon, printers from Epson, and power supplies from Zenith. Applications software included a word processor and a spreadsheet.

Many in IBM were uncomfortable with the idea that the company should become involved in the personal computer market. One famous internal memo warned that it would be "an embarrassment" to IBM. The doubters were quickly proved wrong. Within days of the machine's launch on 12 August 1981, IBM was forced to quadruple production.

Still they could not keep up with demand. People, and especially businesses, who were previously wary of microcomputers were reassured by the IBM logo. A brilliant advertising campaign featuring a Charlie Chaplin look-alike hit just the right balance between quirkiness and quality. The machine was no technological marvel, but it worked, and of course it was from IBM.

Big Blue's decision to source the components from other manufacturers had far-reaching, if unintended, consequences. It meant that anybody could copy the design. Hundreds of companies did, and the IBM PC became the industry standard. A huge industry grew up in peripherals and software, and for the first few years the big battle in the computer industry was over "degrees of compatibility" with IBM.

But the decision with the most far-reaching consequences was IBM's decision to license the PC's operating system, rather than buy one or develop one itself. It initially called on a company called Digital Research, developers of the CP/M operating system used on many early microcomputers. But Gary Kildall, Digital Research's idiosyncratic founder, broke the appointment because he was out flying his plane. (Freiberger and Swaine, 1984: 272). Irritated, IBM turned to another small company in Seattle they had heard had a suitable operating system. That company's name was Microsoft.

The problem was, Microsoft did not have an operating system. Bill Gates didn't let IBM know that. He quickly bought an operating system called QDOS (which stood for "quick and dirty operating system") from another small company, Seattle Computer Products, for $US30,000. He renamed it MS-DOS and licensed it to IBM. The licensing deal was important – for every IBM PC sold, Microsoft would receive $US40. Microsoft previously just another minor software company, was on its way. Kildall at Digital Research realised his mistake, and his company developed a rival to MS-DOS called DR-DOS. The licensed version of MS-DOS that IBM used it renamed PC-DOS (Campbell-Kelly and Asprey, 1996: 257).

At the end of the 1980s it was far from certain which computer operating system, and which hardware architecture, would win out. Apple was still strong, and in the IBM PC and compatible world there was substantial uncertainty over which operating system would win out. MS-DOS was still dominant, but it faced challenges from Digital Research's DR-DOS and – more significantly – from IBM's OS/2.

After the PC's astounding success, IBM realised it had made a mistake licensing an operating system off Microsoft. The operating system was where the battle for the hearts and minds of users was being won, so it determined to wrest control back from the uppity boys in Seattle. OS/2 was the answer. It was a vastly superior operating system to MS-DOS. It had a microkernel, which meant it was much better at multitasking, and it was built for the new era of microprocessors that were then coming out. It would operate across architectures and across platforms.

In its early days, Microsoft and IBM cooperated on OS/2's development, but Microsoft withdrew to concentrate on something it called Windows NT, which stood for New Technology. Microsoft then outmarketed IBM, and OS/2 died. Apple imploded, Microsoft started bundling its applications, and the battle for the desktop became a one-horse race.

A key development in Microsoft's success was its release of Windows 3.11 in 1992. It was the first Microsoft operating system to successfully employ a graphical user interface (GUI) with its now-ubiquitous WIMP (Windows, Icons, Mouse, Pull-down Menus) interface. Microsoft had released earlier versions of Windows, but they did not work well and were not widely used (Campbell-Kelly and Asprey, 1996: 278).

The GUI was popularised by Apple, which introduced a GUI on the Macintosh, first released in 1984. Apple got the idea of the GUI from Xerox, which developed the first GUI in its famous Palo Alto Research Center (PARC) in Silicon Valley in the late 1970s. Xerox PARC developed many of the key inventions in the history of computers, such as computer networking and laser printers, but has itself has rarely made money from PARC's innovations. Apple's Steve Jobs visited Xerox PARC in December 1979 and saw a prototype of Xerox's Alto computer, the first to use a GUI. He was inspired to develop the Apple Lisa and then the Macintosh, which was the first commercially successful machine to use a GUI (Carlton, 1997: 13).

The genesis of the GUI goes back well before even the establishment of PARC, which opened its doors in 1969. Stanford University's Human Factors Research Center, led by Doug Engelbart, was founded in 1963 to develop better ways of communicating with computers. Engelbart's group developed the first mouse, and in 1968 demonstrated a prototype GUI at the National Computer Conference in San Francisco (Campbell-Kelly and Asprey, 1996: 266).

Getting Users in Touch With Data

As computers become widespread in business the role of programmers, and programming, changed. When computers were first employed on a wide commercial scale in the 1960s, applications development was a comparatively simple, if cumbersome, exercise. Just about everybody wrote their own applications from scratch, using third generation language (3GL) like Cobol (see above).

In the late 1970s a new applications development tool came into existence, as demand for increased computer performance began to far outstrip the capabilities of the limited number of 3GL programmers to write and maintain what was wanted. These were so-called 4GLs, or fourth-generation languages. The first 4GLs, such as Ramis and Focus, were reporting and query tools that allowed end users to make their own enquiries of corporate information, usually in the IBM mainframe environment. They worked well, but only when coupled with a good database. They were useful in this role, freeing up programmers to develop applications (Philipson, 1990: 12).

4GLs produced usable computer code. They were widely used by end users to create applications without going through the IT department. At about the same time PCs became widespread in many corporations, further enhancing the idea of end user computing. 4GLs spread from the mainframe onto PCs. Many of the most popular PC applications, such as the Lotus 1-2-3 spreadsheet and the dBase database products (see above), were in effect 4GLs optimised for particular applications.

But most of them produced code that needed to be interpreted each time they ran, which made them a drain on computing resources. Programs written in these early 4GLs typically ran much slower than programs written in 3GLs, just as 3GL programs run slower than assembly language programs, which in turn run slower than those written in machine language.

This drain on resources caused many companies to re-evaluate their use of 4GLs, and also caused the 4GL suppliers to create more efficient versions of their products, usually by the use of compilers. 4GLs are more accurately called non-procedural languages. They are used to explain what needs to be done, not how it should be done. Non-procedural languages are now used widely, and have been a major factor in the move to increased end user computing.

For all the advantages offered by 4GLs, there has remained a need for query languages, which was what 4GLs were designed for in the first place. This need has grown with the widespread use of relational databases. SQL (Structured query language) has evolved to perform many of the query functions of the early 4GLs. The wheel has turned full circle.

From 4GLs it was a short step to applications generators. Whereas 4GLs produced code in their own specialised languages, applications generators used a user-friendly 4GL-type front end to produce standard 3GL code. Thus, the use of a 4GL was similar to that of an applications generator, and the two types of product competed against each other. But the end result was different. Applications generators had the advantage of producing code which could then be maintained or modified by 3GL programmers unfamiliar with the way in which it was produced.

Applications generators merged with another new technology in the 1980s – CASE. CASE stood for computer-aided software engineering, and it was one of the big IT buzzwords of the late 1980s and early 1990s. It referred to the use of software to write software, and it seemed like the answer to a lot of problems. Building applications from scratch is never an easy job. CASE promised to simplify the task enormously. Instead of cutting all that code yourself, you would tell a smart computer program what you wanted to do, and it would do the work for you.

The term CASE had its origins in the Apollo space program in the 1960s, the most complex computer-based project of that era. Two computer scientists working on Apollo, Margaret Hamilton and Saydeen Zeldin, developed a set of mathematical rules for implementing and

testing complex computer systems (Philipson, 1990: 14). They later formed a company called Higher Order Software, and their methodology evolved into a product called USE.IT.

USE.IT was not a commercial success, despite being promoted by software guru James Martin in his influential book "Systems Development Using Provably Correct Concepts", published in 1979. Martin was at the time one of the industry's leading and most successful theorists, and an important figure in CASE's development.

CASE, in its broadest sense, is any computer-based product, tool or application that helps in the process of software development. With the realisation that software design could be systematised came the inevitable development of standard methodologies, procedures to guide systems analysts through the various stages of software design. Many such systems were designed, such as SSADM (Structured Systems Analysis And Design Methodology), which was mandated by the British Government and became very popular in the United Kingdom. These design methodologies were the precursor to CASE, and a precondition to its effective use, guiding analysts and end users through the design process.

The Rise and Fall of CASE

CASE boomed. IBM released a strategy called AD/Cycle, which was kind of a unified field theory of applications development, an umbrella under which various CASE tools that addressed different stages of the applications development life cycle could fit together, lego-like, allowing developers to mix and match various products to suit their purposes (Philipson, 1990: 24),

The promise of CASE was virtually irresistible. Applications development had long been beset by two major problems: difficulty in getting the developments finished on time, and difficulty in ensuring that the finished software was robust: properly documented, internally consistent, and easy to maintain. CASE promised to solve these problems. No longer did software developers have to design systems then laboriously write the programs with outdated third and fourth generation languages. CASE would (so the theory went) allow a developer to outline the specifications, from which point the CASE software would automatically generate the code, all documented and modular and consistent.

But it did not work like that in theory, and CASE declined, for a number of reasons. First, CASE products demanded a level of discipline that did not come naturally to many developers. They were a great aid to program design, but the real creative work was still left to people, not machines. Many CASE tools were quite advanced, but none of them ever reached the stage where you could simply tell them to develop you a retail banking system, and have it come back the next day with the code all written. Far from it.

The bigger the system being developed, the more could go wrong. The complexities were not so much in the computer system as in the organisational structure it was trying to service. CASE did not address those problems. So most large and small organisations stopped developing their own applications, and most of the CASE vendors withered and died. Some stayed in existence, but they changed their focus and moved into other areas.

In the early 1990s there was a massive move towards packaged software. Packaged applications became much more flexible than ever before, and it simply didn't make sense for many users to write their own software when something they could buy off the shelf would do it just as well, for a fraction of the price.

The action in application development moved to the desktop, where people used products like Microsoft's Visual Basic and Access to build quick and dirty systems that allowed them to download corporate data, often out of these packaged software systems, into desktop systems like Excel where they can be easily manipulated. That became the real battleground – end user access to corporate data.

An important part of the applications software industry grew out of this need. A class of software developed in the 1980s called the Decision Support System (DSS), which quickly evolved into the Executive Information Systems (EIS), which took operational and transactional data from corporate databases and reformatted it in such a way that it could easily be understood by end users. This usually involved a range of graphical comparisons, of sales data by region by time for example. EISs became necessary because traditional computer systems were not designed to make it easy to extract information. They were optimised for operational purposes – to record transactions and to compute balances and the like. (Power, 2003).

In the 1990s EISs evolved into a whole new class of PC software, generically called Business Intelligence (BI). BI software works in much the same way as EIS software, displaying information in attractive graphical formats, and allowing information from different sources to be easily juxtaposed. The key to this sort of analysis is the ability to display data in multiple dimensions, often called multidimensionality, or online analytical processing (OLAP). OLAP tools are often used as front ends to data warehouses, which are systems in which operational data has been downloaded and optimised for such retrieval (Power, 2003).

Unix, Windows, and the Death of Proprietary Operating Systems

The success of IBM's OS/360 spawned many imitators. Most early minicomputers used proprietary operating systems designed just for that computer, but all of them were mini-architectures that were capable of being run on different models in the range. Data General had AOS, Digital had VMS, Bull had GCOS, Wang had VS, Hewlett-Packard had MPE. There were dozens more. But one operating system gradually replaced them all. Its name was Unix.

Unix came into being in 1969, the brainchild of Ken Thompson and Dennis Ritchie in AT&T's Bell Labs. It was an attempt to build a small and elegant general purpose operating system that would be independent of the hardware platform it ran on. Thompson and Ritchie succeeded in all these aims, and Unix filled a gap at a time each manufacturer was developing its own proprietary operating system (Ceruzzi, 1999: 106).

Unix prospered, largely because AT&T adopted a policy of giving it away to universities. A generation of programmers learnt about the basics on Unix, taking their expertise with them into the workforce. The inevitable happened, and different versions of Unix began to appear, but there remained an identifiable core Unix.

In the 1980s many new hardware suppliers entered the industry, lured by the massive growth in commercial computing and the lower cost of entry afforded by the new breed of microprocessors and cheap off-the-shelf peripherals. The leading vendors stayed with their proprietary operating systems, but most of them could not afford to develop their own. So they used Unix, which was cheap, functional and readily available.

They each tweaked Unix to their own specifications, and at one stage there were dozens of different varieties. The momentum grew, and the major suppliers all jumped on the Unix bandwagon. Hewlett-Packard was the first, in 1984, and the others followed, with IBM finally legitimising Unix with the announcement of its RS/6000 Unix computer in 1990.

Unix was virtually given away to universities, which meant that a generation of computer science graduates joined the industry familiar with its arcane workings. But Unix was not perfect. It was difficult to learn, and a nightmare for end users. But its existence as the only alternative to the maze of proprietary systems saw it prosper, even if all the vendors built their own little proprietary extensions to it to differentiate themselves from their competitors.

There were many attempts in the early 1990s to unify the various strands of Unix. Two major consortiums emerged, Unix International (essentially the AT&T camp) and the so-called Open Software Foundation, which was not open, was not a foundation, and which had a lot more to do with politics and marketing than software. These groups and other splinter bodies conducted an unedifying fight over standards and directions now loosely referred to as the "Unix Wars", but users were supremely uninterested in their petty squabbling and voted with their feet. As so often happens in the computer industry, standards emerged through market forces rather than industry agreement, as the user community coalesced around three major varieties of Unix: those from Sun Microsystems, Hewlett-Packard and IBM.

Other varieties faltered, including the Unix that should have been most successful, that from Digital. Digital, which led the minicomputer revolution and on one of whose machines Unix was originally developed, could never make up its mind about Unix. Its ambivalence become doctrine when founder and CEO Ken Olsen referred to Unix as "snake oil" (Rifkin and Harrar, 1988: 305). Digital was acquired by Compaq in 1999, and the combined company was acquired by Hewlett-Packard in 2002.

Another unifying force was the growth of Novell's Netware networking operating system, which enjoyed a great boom in the early to mid 1990s as people started to network PCs together in earnest. But Netware was a flash in the pan, now relegated to the dustbin of history along with the other proprietary systems. Netware's demise occurred largely at the hand of Microsoft. Flush from success at the desktop level, Microsoft decided to move into operating systems for larger computers. In 1993 it released the first version of Windows NT (for "new technology"), which unlike had just one version across all architectures.

NT did many of the things Unix did, and many of the things Netware did. At first it was slow and underpowered, but it gradually improved and was able to compete at the technical level. There is a great deal of anti-Microsoft sentiment in the computer industry, mostly from the Unix camp, but real live users moved to Windows NT in droves. They were sick of the lack of interoperability of proprietary systems, and of Unix's internecine squabbling. NT was proprietary, but because it was from Microsoft it was seen by many to be the way of the future.

Linux and Open Source

For a while it looked as if Windows would sweep all before it. But in 1992 a Finnish university student named Linus Torvalds developed a version of Unix, called Linux, which used the "open source" model. Under this model software is not developed by any one company, but any developer can improve it, and submit those improvements to a committee

for acceptance. Linux is essentially free, while at the same time having an army of developers improving it all the time.

Linux thus represents the antithesis of the proprietary Microsoft development method, just has it represents the antithesis of the Microsoft marketing model. The battle between the two has become religious. The Linux development process of continuous improvement has now seen it move up the ladder of scalability and respectability, to the point where we are starting to see it in the data centre, and where most software suppliers are now readily supporting it.

One of the key reasons for this shift is the wholehearted endorsement given to Linux by IBM, the company that understands enterprise computing like no other. IBM picked up on the move to Linux in the late 1990s, and is now spending $US1 billion a year on Linux development, mainly in what used to be called in the mainframe world "RAS" – reliability, availability, scalability – the very things that make an operating system suitable for enterprise computing.

Microsoft, and many others, see in the open source movement a threat to capitalism. There is little doubt that the open source software movement has anti-capitalist elements. It has its own manifesto, the well known tract "The Cathedral and the Bazaar", written by the movement's very own Karl Marx, Eric Raymond (Raymond, 1995). Its first three words throw down the gauntlet: "Linux is subversive." The title of the document comes from Raymond's idea that open source software is like a medieval bazaar, an untidy agglomeration of merchants and stalls and thronging people, compared to which the standard method of building software is like a stately cathedral, planned out well in advance and built over time to exacting specifications.

There is much to open source than Linux. All variety of software is available in open source. Websites such as Sourceforge (www.sourcegorge.net) and Freshmeat (www.freshmeat.net) list thousands of open source utilities and applications that anyone can download at no cost. Open source databases like MySQL and PostgreSQL are widely used around the world, often in production applications, and the Apache web server is more widely used than its commercial competitors. There is even an open source ERP package, called Compiere (www.compiere.com). Clearly, open source software is here to stay.

The software establishment is worried. In October 2003 Microsoft president Steve Ballmer cut short a holiday in Europe to try to convince the Munich city council to rescind a decision to move to open source. Microsoft sees it as the thin edge of the wedge. And in early 2004 software company SCO, which claims the ownership of Unix after a convoluted series of deal in the late 1990s, sued IBM for $US3 million over allegations that Big Blue stole Unix code and put it into Linux.

SCO says that IBM's continued shipment of AIX, IBM's version of Unix, is illegal because it relies on a license from SCO. IBM says its Unix license is irrevocable. Novell, which sold Unix to SCO in 1995, says that sale excluded all "copyrights" and "patents". Microsoft has bought a SCO license, though it does plan to ship any Unix. The open systems people say SCO doesn't own Unix anyway.

The basis of SCO's argument is that it is now the owner of the intellectual property that is Unix, and that IBM has stolen that intellectual property and incorporated it into Linux, where all can benefit. It has allowed a select group of analysts to view the pieces of code it says were stolen. They agree there are major similarities. But Unix has evolved in such a complex fashion that it is ultimately impossible to tell where the source code comes from. At the time of writing (April 2004) the matter remains unresolved.

The Internet and the World Wide Web

In the late 1960s the world's biggest computer user was the US Department of Defense. It had many machines of its own, and it used many more at universities and research institutions. Bob Taylor, a manager at ARPA (Advanced Research Projects Agency) proposed that ARPA's computers should be connected in some way, and he eventually persuaded ARPA to call for tenders. A small company in Boston called BBN wrote a proposal for "interface message processors for the ARPA network". The company got the job, and BBN's Frank Heart and his team started work, using a new technology called packet switching (Segaller, 1998: 45).

Packet switching sent data in discrete packets, rather than all at once. BBN was the first company to implement the technology, but the concept was also used picked up by a young man in Hearst's group called Bob Metcalfe, who four years later used it to devise Ethernet, the technology underlying most local area networks (LANs). Metcalfe is famous for Metcalfe's Law – "the utility of a network expands by the square of the number of users" (Segaller, 1998: 283).

On 1 October 1969 the first Internet message was sent, from UCLA to Stanford Research Institute. Before the end of the year the University of California at Santa Barbara and the University of Utah were connected, in a network called ARPAnet. Growth was slow. A year later the network had expanded to 15 nodes, but it took a further seven years to reach 100. Usage was restricted to academia and the military, and it remained very difficult to use until a networking standard called TCP/IP (Transmission Control Protocol/Internet Protocol) was developed by ARPA in 1982 (Segaller, 1998: 111).

Gradually people began calling the new network the Internet. Its most widespread application became email, and things improved substantially in 1983 with the introduction of domain names, like .com and .org. But it was not until Tim Berners-Lee conceived the idea of the World Wide Web in 1989 that it began to resemble the Internet we know today.

Berners-Lee, an English scientist working at CERN, the European particle physics laboratory, came up with some simple specifications that made navigation around the Internet much easier. He devised a language called HTML (HyperText Markup Language), and a communications protocol called HTTP (HyperText Transfer Protocol) that used the concept of "hypertext" to allow people to jump easily between locations on the Internet (Berners-Lee, 1999: 36).

But the Internet was still not a place for beginners. Addresses and locations were standardised, but you still had to know where you where going, and you needed a range of different software tools to get there. Enter the browser. The browser was the brainchild of Marc Andreessen, a 21 year old student at the University of Illinois' National Center for Supercomputing Applications (NCSA). He was working for $US7 an hour cutting code for the NCSA's Unix computers, and became frustrated with how difficult it was to use the Internet. He enlisted the aid of a colleague, Eric Bina, and set out to change that (Segaller, 1998: 296).

Over three months in the winter of 1992-93 Andreessen and Bina developed the first browser. It ran only on Unix, but it was revolutionary. Their aim was a single piece of single software that could navigate through hypertext links with the push of a button or the click of a mouse,

that could display graphics as well as text, and which had an attractive and easy to use interface.

The result was Mosaic. They completed their first version in February 1993, and in April they released it for widespread use. Immediately there were 10,000 users, and by the end of the year, by which time they had developed Windows and Apple versions, there were over a million users.

A good indication of Mosaic's impact can be seen from the growth in the number of web sites. In January 1993 there were about 50 commercial web sites on the Internet (the US Congress had only authorised commercial usage the previous year). A year later, there were more than 10,000. Amazon.com became the archetype of a new e-business model, and once Pizza Hut let people order pizzas over the Internet in 1994, things started to happen very quickly (Segaller, 1998: 348).

The browser has developed substantially in the last ten years, but it is still recognisably the same piece of software it was ten years ago. Much has happened in that time – Andreessen left the NCSA, which to this day downplays his role in history, saying that development was a collective effort. He became one of the founders of Netscape, which briefly became one of the most successful companies in history before Microsoft entered the browser market and started the acrimonious "browser wars" of the late 1990s, which led directly to Microsoft's legal problems, and also to its unrivalled success.

Tim Berners-Lee, inventor of the World Wide Web, has not been idle in recent years. He is now director of the Word Wide Web Consortium (W3C), the non-profit coordinating body for Web development. His work still involves conceptualising where the Web is headed, and how to get it there. Berners-Lee believes the next big thing will be the "Semantic Web", which he describes as an extension of the current Web where information is given meaning and where computers can not only process information but understand it (Berners-Lee, 1999:157).

The Web as it is currently constituted is optimised for human beings – to allow people easy access to documents. In the Semantic Web, data contained in Web pages will be coded with an extra dimension of information that will enable computers to make sense of it. XML (Extensible Markup Language – an extension of HTML) is a step in that direction, as are emerging Web Services protocols, but the Semantic Web will contain much more meaning. It will enable intelligent software agents to perform many of the searches and conduct many of the transactions that can currently only be undertaken by humans.

ERP and E-Business

The Internet received substantial publicity as the biggest IT story of the 1990s, but the second most important trend was the growth in enterprise resource planning (ERP) systems. ERP basically means the integration of high end applications, usually based around manufacturing or accounting systems. In the early days of commercial computing applications didn't talk much to each other. As they grew more sophisticated they started to interact, with data from purchasing feeding straight into the manufacturing system, and sales information flowing directly to the general ledger.

Back in the 1980s it was still common for large organisations to write their own applications software, hence the CASE boom (see above). The rise of packaged ERP solutions put an end to all that, and today just about every large organisation uses an off-the-shelf ERP system of

one kind or another. The growth of the ERP vendors in the 1990s was phenomenal. The largest of them all is German company SAP, but many others blossomed. These included Peoplesoft and Baan. Database company Oracle moved strongly into ERP, becoming number two after SAP. Many older manufacturing software companies like JD Edwards also made a successful move into ERP, though others like SSA and QAD had less success in breaking out of that niche.

At the end of the 1990s the ERP market went into something of a slump, causing many commentators to write of the "death of ERP". There was nothing of the kind. ERP sales declined, but only because so many organisations had bought ERP systems during the 1990s that the market was close to saturation point.

The Y2K scare also brought a lot of activity forward, meaning there was a slowdown after 2000. The Y2K issue, otherwise known as the "Millennium Bug", came about because of the habit of many programmers writing the date as just the last two digits of the year. That meant that as the years rolled through from 1999 to 2000, the date counters would go back 99 years instead of forward one year. An enormous amount of time and money was spent fixing the problem, which ended up not being much of a problem at all. Whether that was because appropriate remedial action had been taken, or because it was never going to be a problem anyway, will never be known.

Another reason for the apparent decline in ERP was the greater publicity given to applications like electronic commerce and customer relationship management (CRM). These were often spoken of as if they were separate applications, but they were not. They were simply different aspects of ERP. Many commentators try to make a distinction between "traditional" ERP and some of these newer applications. There is no great distinction – it is all part of the increased integration of large scale computer applications.

ERP has now evolved into something variously called eBusiness or eCommerce. With the massive amount of business now being transacted over the Internet, the distinction between conventional commerce and electronic commerce has all but disappeared. eBusiness became inevitable as soon as the Internet was freed from its academic and scientific limitations in 1992. The business world has always been an early user of any new communications medium, because communications is the lifeblood of commerce. Amazon.com and eBay were both formed in 1995, and one of the most amazing booms in history occurred. By 1999 the worldwide eBusiness market was worth $US30 billion (Philipson, 2002: 18).

eBusiness of a kind was possible without the web. The advantages of using computer networks for business purposes became apparent to many people in the 1980s, and a number of different systems were devised to enable commercial transactions via computer. These were generally known as EDI (electronic document interchange). EDI was standardised by the mid-1980s and adopted by a number of organisations, but it was not a widespread success. It relied on closed systems with specialised protocols and on real-time communications – the two computers conducting the transaction had to be connected directly to each other.

EDI was successful in specific industries with regularised transactions. EDI's success in these areas meant that its users were ready to embrace the use of a better and cheaper medium a decade later. EDI systems are still in place, carried via the Internet. The rapid growth of the Internet and eBusiness in the late 1990s made many believe that the old rules of business no longer applied. There was much talk of the "new economy", and many new companies were formed on the basis that the Internet boom would transform overnight the way consumers and businesses behaved. Many of these new dot.com companies were based on unrealistic and

unsustainable business models, but a feeding frenzy of investors eager to cash in on the speculative growth of these organisations led to a spectacular stock market boom, which led to an equally spectacular crash in early 2000.

The crash led many people to erroneously believe that the promise of the Internet was a fallacy. But, in fact, it was essentially a minor correction after a burst of over-enthusiasm. There are more people online today than there were at the height of the technology bubble. More things are being purchased by consumers over the Internet, and more business-to-business commerce is being done over the Internet, than at any previous time.

The Internet, and eBusiness, is here to stay. Innovation continues apace on many fronts, constantly pushing the boundaries of what is possible. The history of information technology is in many ways the history of improved end-user access. Every major development in computers has had to do with making it easier to get at, manipulate and report on the information contained within the systems. The PC revolution, the rise of networking, the development of the graphical user interface, the growth in distributed computing, the rise of the Internet and eBusiness: all are part of this trend.

Outsourcing, Offshoring and the "Skills Shortage"

During the 1990s much attention was paid in the IT industry to outsourcing, or the phenomenon of contracting out all or part of the IT function. The arguments were essentially those of make versus buy, familiar to any manufacturing concern. Is it cheaper to make or do something yourself, or get someone else to do it? Increasingly, as the IT industry evolved, it made more sense to get someone else to do it. The movement from in-house development to packaged software, described above, was part of this trend.

But towards the end of the 1990s and into the new millennium, outsourcing took on a whole new dimension, known as "offshoring", which described the outsourcing of various business functions, including software development, to other countries. Very often "other countries" meant India, whose software industry grew enormously in the 1990s. The removal of import restrictions, improved data communications, and the sheer number and quality of Indian software professionals, combined with their very low cost compared to their Western equivalents, made India an attractive software development centre for many Western vendors and users. There was also substantial immigration of Indian programmers to Western countries, particularly the UK, the US and Australia (Philipson, 2002).

This caused a major backlash in these countries. Even as the governments and the IT industry proclaimed an IT skills shortage, employment figures and salaries in the IT industry fell. A major debate over the reasons ensued, based mainly around IT immigration and offshoring. What most people missed was a fundamental structural change occurring in the IT industry – the shift of IT jobs from inside user organisations to outside. The outsourcing boom of the 1990s meant that all manner of organisations were contracting out all manner of activity, including all manner of IT.

This was largely in an attempt to lower the costs of IT. By far the majority of IT costs are internal – salaries, data centre management, consumables, etc. But the proportion has dropped throughout the entire history of the computer industry. More and more activity has moved from internal (do-it-yourself) to external (product). People moved from write their own applications to buying packages. They moved from building their own networks to plugging into the Internet. Training, programming, help desks, and a whole range of activities now

occur outside of the organisation, with the expenditure transferred from internal to external. The external service providers have economies of scale and greater efficiencies, and so employ fewer people to do the same amount of work.

Vendors became increasingly aware of this trend. IBM has totally reinvented itself over the last ten years as a service company. Oracle, itself now a major services company, is pushing to run its clients' applications for them. Compaq bought DEC, and HP bought Compaq, largely for their consultancy capabilities. Web services, the hottest trend of the new millennium, is all about facilitating the outsourcing of business functions.

The trend is being greatly accelerated by the increased globalisation of the IT industry. Much of the debate over IT skills worldwide has centred around the immigration of cheap labour, but the real action is in the wholesale export of IT jobs to low cost countries. The chief beneficiary of this has been India, because of its sheer size. With over a billion people, and a much higher birthrate, India will overtake China in population this decade, and it has over half a million IT graduates, with over 50,000 more entering the workforce each year. And they all speak English.

Data communication and voice telephony costs are now so low, and bandwidth so broad, and the Internet so ubiquitous, that it is a simple matter to run an applications development centre offshore. An increasing number of the world's call centres are now in India, or the Philippines, or southern Africa. We are now witnessing, on a global scale, the kind of disruption that occurred in the English countryside in the industrial revolution 200 years ago. We have long witnessed the movement of blue-collar jobs to low-cost countries, now we are seeing white-collar jobs move offshore, at an even faster rate. The dark satanic mills of the information millennium are in the suburbs of Bangalore, Shenzhen and St Petersburg. And it is not just IT jobs – it is architects, accountants, engineers, indeed any type of knowledge worker. If the work can be digitised, it can be exported. Microsoft's second largest development centre is in Beijing, and Oracle plans to have 4000 software designers in India by the end of 2004. Large and small IT shops are moving software development and other functions to India and elsewhere in the world at an increasing rate.

The Future of Software: The Information Utility

On 10 March 2000 the NASDAQ index of US technology stocks hit a peak of 5049, after doubling in the previous year. Two years later it was hovering in the low 1100s, a drop of more than 75 per cent. In the three years following the top of the boom nearly 5000 computer companies in the US, and many more internationally, went broke or were acquired. 200,000 jobs were lost in Silicon Valley alone. In early 2004 the market capitalisation of all the world's computer companies was about 20 per cent of what it was three and half years earlier (Philipson, 2003).

After every boom there is a bust. Rarely has this fact been more graphically demonstrated than in the Great Tech Bust of 2001-2003. The 1990s saw the information technology industry change beyond recognition. Large and proud companies like DEC, Wang, Compaq, Prime, Data General, Amdahl and many others are no more. PCs have become so cheap that they are commodities, and so powerful that no-one takes much notice of their technical specifications.

The Internet is a fact of life. It is now almost impossible to do business without it. Most airline tickets are bought over the Internet, email is the standard method of commercial

communication, and all media are digital. Mobile phones and other wireless devices are commonplace, removing physical location as a constraint on communication. There is more processing power in the average car than in the largest data centre of the 1970s.

Oracle's Larry Ellison famously said in 2003 that 90 per cent of the software companies in Silicon Valley don't deserve to exist (Philipson, 2003), and leading consultancy Gartner says that half them will go out of business in the next five years. The software industry, after more than 50 years of fabulous growth, has grown up. It has matured. As its products have commoditised its margins have reduced. The offerings from one player look much like those from another. In most parts of the industry the number of significant companies has fallen to just three or four.

One company, Microsoft, has a virtual monopoly on desktop software – operating systems and applications. Its abuse of its position has led to legal action by the US Department of Justice, which went after IBM in the 1970s for just the same reason. The IBM case eventually fell apart, not because IBM had not abused its position (it had), but because the industry had changed so much that the circumstances had become irrelevant. The same is happening in Microsoft's case. The software giant has been exposed as a serial bully and a master of IBM's old FUD (fear, uncertainty and doubt) tactics, but it is all becoming meaningless. Microsoft's dominance will decline, because of the very market forces it claims to uphold.

The ubiquity of the Internet will ensure that. Software, and the functions it performs, are increasingly traded as Web-delivered services. Web services standards, which allow business processes to be shared between organisations, are one of the key software building blocks of the new millennium.

The open source movement, which is already making significant inroads with Linux, is another force of the future. The very concept of intellectual property, be it software, or music, or film, is under threat from the new technology. That will be the battleground of the future. We are moving towards the era of the information utility, with IT delivered on demand via an invisible grid that encircles the globe. It is also called grid computing, or fabric computing, or cluster computing.

Like most things in computing, there is no standard definition of what utility computing is, but the idea has been around for some time. The rationale is that most of the services we use every day, like power and water, are utilities. You flick a switch, you get electricity. You turn on a tap, you get water. The telephone is pretty much a utility nowadays. The Internet is getting there. (Sun even tried to popularise the term "web tone" a few years ago). Indeed, the Internet has done a lot to promote the idea of an "information utility", where all the information you want is available right there in front of you.

Utility computing takes this a major step further. You switch on your computer, and you have as much or as little computing power as you need. Like other utilities, you pay only for what you use, plus a standing charge for being connected to the system. Computers are all connected together in a grid, so they can all share the load of every application being run across the whole network.

The idea is an attractive one, and it is becoming pervasive. Just about every major computer supplier, with the notable exception of Microsoft, is talking about utility computing, though few of them call it that. IBM is heavily promoting what it calls "on demand computing". It does not like the term "utility computing" because it wants to be the provider. It sees the "utility" idea as being a little impersonal, and suggestive of the idea that users can seamlessly

shift from one supplier to another. That is also why we are not hearing much from Microsoft about the subject.

But IBM is talking at length about grid computing. So is Oracle, and many other suppliers. Sun has an initiative called N1. HP has something called the Utility Data Center. And Dell plans to populate the planet with low-cost Intel-based servers that will perform any and every computing task imaginable.

The software industry has changed forever, and more in the last five years than in the previous twenty. As hardware becomes ubiquitous and commoditised, software will become more pervasive and more important. It is now nearly 60 years since ENIAC's first switches were flicked and the first program ran. Sixty years from now, today's software will look as primitive as ENIAC's does to us.

Graeme Philipson, Wollongong, 18 April 2004

– o – O – o –

References

Augarten, S. (1985) *Bit by Bit: An Illustrated History of Computers*. London, Unwin Paperbacks.

Berners-Lee, T (1999). *Weaving the Web*. San Francisco, HarperCollins.

Brooks, F.P. (1975, Anniversary edition 1995) *The Mythical Man-Month*. New York, Addison-Wesley.

Cambell-Kelly, M and Aspray, W. (1996) *Computer: A History of the Information Machine*. New York, HarperCollins.

Carlton, J. (1997). *Apple: The Inside Story of Intrigue, Egomania and Business Blunders*. New York, Random House.

Ceruzzi, P.E. (1999) *A History of Modern Computing*. Cambridge, MA, MIT Press.

Crawford, A. and Ziola, B. (2002) *Oracle for Molecular Informatics*. Online: www.managedventures.com/images/OraMolInfoBW6.pdf. Accessed 3 February 2003.

Cringely, R.X. (1992) *Accidental Empires*. London, Viking.

DeLamarter, R. (1986). *Big Blue: IBM's Use and Abuse of Power*. New York. Dodd, Mead & Company.

Freiberger, P. and Swaine, M. (1984) *Fire in the Valley: The Making of the Personal Computer*. Berkeley, CA, Osborne/McGraw Hill.

Hodges, A. (1983, Unwin Paperbacks edition 1985) *Alan Turing: The Enigma of Intelligence*. London, Unwin Paperbacks.

Kunde, B (1996) *A Brief History of Word Processing (Through 1986)*. Online: <www.stanford.edu/~bkunde/fb-press/articles/wdprhist.html> (accessed 12 April 2004)

McCartney, S. (1999) *ENIAC: The Triumphs and Tragedies of the World's First Computer*. New York, Walker and Company.

O'Connor, J.J. and Robertson, E.F. (2002). *Augusta Ada King, countess of Lovelace*. Online: <www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Lovelace.html> (accessed October 2003).

Peterson, I. (2000). *Software's Origin*. Online: <www.maa.org/mathland/mathtrek_7_31_00.html> (accessed 20 March 2004)

Philipson, G. (1990). *Implementing CASE Technology*. Charleston, SC, CTR Corporation.

Philipson, G. (1991). *Mainframe Wars*. Charleston, SC, CTR Corporation.

Philipson, G. ed. (2002, 2nd edition). *Australian eBusiness Guide*. Sydney, CCH Australia.

Philipson, G. “Skills Glut Prompts Clampdown on Indian Visas”. *Butler Group Review*, November 2002 p.30.

Philipson, G. “Database Pioneer Dies”. *“Next” Supplement, The Sydney Morning Herald*, 20 May 2003, p.3

Philipson, G. “A Brief History of Computing”. *Australian Personal Computer*, December 2003, p.84.

Power, D.J. (2003, version 2.8). *A Brief History of Decision Support Systems.* Online: http://dssresources.com/history/dsshistory.html. Accessed 12 March 2004.

Raymond, E. (1995 with continued updates). *The Cathedral and the Bazaar*. Online: <http://catb.org/~esr/>. Accessed 15 October 2003.

Rifkin, G. and Harrar, G. (1988). *The Ultimate Entrepreneur: The Story of Ken Olsen and Digital Equipment Corporation*. Chicago, Contemporary Book.

Segaller, S. (1998). *Nerds 2.0.1: A Brief History of the Internet*. New York, TV Books.

Symonds, M. 2003. S*oftwar: An Initimate Portrait of Larry Ellison and Oracle*. New York, Simon & Schuster.

Watson, T.J. (1990), *Father Son & Co: My Life at IBM and Beyond*. New York, Bantam Books.

Yourdan, E. (1992) *Decline & Fall of the American Programmer*, Englewood Cliffs, NJ, Yourdan Press.

Email graeme@philipson.info
Web www.philipson.info
Ph +61 2 4226 2200
Fax +61 2 4226 2201
Mobile +61 418 609 397
Post PO Box 290, Figtree NSW 2525
Street 147 Koloona Ave, Mt Keira NSW 2500
AUSTRALIA