The Second Industrial Revolution – Computer Safety and Security

"Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning." Rick Cook

"Measuring programming progress by lines of code is like measuring aircraft building progress by weight." Bill Gates

"The digital revolution is far more significant than the invention of writing or even of printing." Douglas Engelbart

"*Complex systems almost always fail in complex ways*." Columbia Accident Investigation Board Report, August 2003

In the twenty-first century, it is difficult to imagine a world without computers. Almost every aspect of our lives is now tied to computers in some way or other. They are used throughout industry for the control of industrial plant and processes, manufacturing, aircraft, automobiles, washing machines, telephones and telecommunications, etcetera, etcetera. Most domestic electrical hardware now contains some sort of microprocessor control.

Microprocessors are now so completely ubiquitous, and so much of our lives is completely dependent on them, that it is a continuous source of surprise to me that the people and the laboratories that led their invention and development are not household names in the same way as, say, Newton's discovery of gravity or Darwin's theory of evolution. Microprocessors have caused an ongoing revolution in our lives. The first industrial revolution in the nineteenth century led to steam power for factories, ships, and trains, and also to the ready availability of substances like steel, which transformed the lives of our forebears. This new, second, industrial revolution may even be more overwhelming than the first.

The starting point for the history of computers is unclear, and it can be selected almost arbitrarily, because so many separate technical developments were necessary before they could all come together to create the modern computer. Also, unlike many other technical developments, one cannot point to a single 'inventor' to give the credit – it is really much more complicated than that. The following section is intended as a very brief summary of key developments and the names of those principally involved. However, any condensed summary such as this will almost inevitably be invidious, and doubtless some key names have been omitted, for which I apologise.

A VERY, VERY BRIEF HISTORY OF COMPUTING

The history of computing really begins with ideas for mechanical calculating devices and machine programming, long before electronics existed. These ideas originate at the time of the First Industrial Revolution, or even earlier. A very brief and selective historical overview follows. Many will no doubt think this is too selective, or too short; for those people, I apologise in advance.

Blaise Pascal, the French philosopher, mathematician, and inventor who died in 1662 aged only 39, is nowadays remembered mostly for his contributions to mathematics. Pascal achieved some truly amazing things when he was still a teenager, such as a mathematical analysis of conic sections. He also produced, at the age of 19, a mechanical calculator capable of addition and subtraction. It became known as the Pascaline, the only functional mechanical calculator of the 17th century, and some 20 machines were built.

In 1801, Joseph Marie Jacquard demonstrated a means of controlling looms using punched cards, thereby creating the world's first programmable machine. If a different set of punched cards was used, the pattern of the loom's weave was changed.

In 1812, Charles Babbage conceived the idea of what he called a 'difference engine', a mechanical device for computing and printing tables of mathematical functions. He demonstrated a small version in 1822 that enabled him to secure funding from the Royal Society and the Government of the day to develop a much more sophisticated version. He worked on this for twenty years before government support was withdrawn. Part of the problem was that Babbage had lost interest in the Difference Engine and had become interested in a more ambitious scheme to produce a device he called the 'Analytical Engine'. (He had other distractions, too. In 1828 he became the Lucasian Professor of Mathematics at Cambridge University, although it is said he never gave any lectures.) The Analytical Engine was never built, but he conceived it as a 'universal machine', a general purpose device that could perform any sort of calculation whatsoever. His Analytical Engine would have comprised several different parts.

- There would have been a memory, for holding numbers to be used as data in the calculations, and also temporary storage of numbers involved in intermediate stages of the calculations.
- There would have been an arithmetical unit, which performed the arithmetical operations on the numbers.
- There would have been a control unit for ensuring the desired operations occurred in the correct sequence.
- There would have been input devices for supplying numbers and operating instructions to the machine.
- There would have been output devices for displaying the outputs from the calculations.

The Analytical Engine would have comprised a huge number of shafts, gears, and linkages. He expected the memory would hold 1000 numbers of 40 decimal digits each, which equates to about 16 kilobytes. He also expected to be able to supply data to the machine in punched-card format. (A portrait of Joseph Marie Jacquard hung in Babbage's drawing room.)

The whole device was impossibly complex for its time, although Babbage continued to work on it up till his death in 1871. His ideas were too ambitious, and the required precision in manufacture was too great for the nineteenth century. In later life, he apparently became a quarrelsome old man, a misunderstood genius, who was intolerant of criticism and embittered by his own failure.

The real problem, of course, was that Babbage was trying to achieve things with nineteenth century mechanical technology that needed twentieth century electronics to be realisable. However, his ideas were visionary, and the overall architecture of the Analytical Engine sounds very familiar to those who work with electronic computers.

In 1991, the Science Museum in London successfully built a working model of Babbage's earlier Difference Engine. In 2010, an organisation called Plan 28 announced plans to raise funds to build an Analytical Engine.

Lady Ada Lovelace was born in 1815, the only legitimate child of the poet Lord Byron. He abandoned her and her mother Anne Isabella Byron shortly after her birth, and her mother later deliberately encouraged her interest in mathematics to counteract what she thought of as her father's madness. As a young adult, she formed a friendship with Charles Babbage and an interest in the Analytical Engine. In 1843 she published an article on the Engine, translated from an article in French by the Italian Luigi Menabrea which he had written after attending a seminar given by Babbage in Turin. Her own notes were appended to the translation, and these notes included a description of an algorithm which could have been used on the Analytical Engine (if it had ever been built) to calculate Bernoulli numbers. In effect (and, OK, this is maybe stretching the point) this algorithm was the world's first computer program. She also recognised that the Analytical Engine might be used for things other than just calculations. As an example, she suggested that perhaps it could be used to compose music.

In 1936, the British mathematician Alan Turing proposed a machine, now known as the 'Universal Turing Machine', which he described thus: "It is possible to invent a single machine which can be used to compute any computable sequence." This was a piece of mathematical philosophy at that time, but it formed the basis for many developments in World War Two and later. Turing himself went on to wartime work on cryptography at Bletchley Park, Buckinghamshire, in England – he specified the electromechanical (relay) devices, called 'Bombes', used in decoding intercepted German radio messages which had been encrypted using the Enigma devices. In 1946, he proposed one of the first designs for a stored-program computer, the Automatic Computing Engine (ACE), which could carry out subroutine operations, with a memory of some 25 kilobytes and a speed of 1MHz. A prototype was built using 1450 thermionic valves (or 'vacuum tubes' as they were known) and mercury delay lines for memory. It first ran in 1950. Production versions were produced by English Electric Ltd and sold from 1955.

The tragedy of Turing was that he was homosexual at a time when it was illegal in the UK. In 1952, he was prosecuted for "gross indecency" and found guilty, and he was given chemical castration as an alternative to prison. He died from cyanide poisoning in 1954, which the inquest determined was suicide. In 2009, Prime Minister Gordon Brown gave an official government apology for the way he had been treated.

Also at Bletchley Park during wartime, a semi-programmable electronic digital computer called 'Colossus' was developed using 1600 thermionic valves (with later versions rising to 2400 valves), to help with decryption of German coded messages that used the Lorenz cipher (known in the UK as 'Tunny'), which was more complicated and sophisticated than Enigma. The first Colossus became operational in January 1944 and immediately speeded up the decryption process which had previously been done manually. Ten Colossus machines were built before the end of the war. Colossus was developed by Tommy Flowers, Max Newman and Allen Coombs, although little credit was given to Flowers and his team until several decades later because of wartime and Cold War secrecy.

Among their many other achievements, the Colossus machines were used to decrypt German signal traffic around the time of the D-Day invasion of Normandy on 6th June 1944. Colossus decrypted on 5th June a message from Hitler to Field Marshal Rommel that ordered Rommel not to move any troops away from the Pas de Calais area. Hitler was expecting the real invasion to be near Calais, starting five days after the Normandy landings. This confirmed that the Nazis had been misled by Allied efforts to have them believe that the main invasion would be near Calais and that the Normandy landings were just a feint. General Eisenhower, who was in overall command of the Allied invasion forces, saw this decrypted message on 5th June and said, "We go tomorrow." Much later, Eisenhower said that, without the information Bletchley Park had supplied, the war could have gone on for at least two years longer than it did.

Flowers' key contribution was probably his realisation that high-speed switching processes (as are used in computer calculations) could be carried out much more quickly by circuits using thermionic valves than by circuits using relays. Colossus was the world's first electronic computer but it was not designed as such – it was strictly for decryption. However, the basic technologies used in modern computers – data storage and retrieval, fast processing, variable programming, data output by printer – were all anticipated by Colossus.

Flowers' invention remained secret for decades after the war possibly because the advancing Soviet army had captured Lorenz cipher machines and had begun to use them for their own purposes post-war. Hence, Bletchley Park was able to continue to monitor Russian Lorenz-enciphered messages well after the end of World War Two, so Colossus remained 'Top Secret'. Flowers' very success ensured his lifelong near-anonymity[1].

Flowers was the son of a bricklayer who had studied electrical engineering at the University of London in the 1920's, and had subsequently worked on the design of electronic telephone exchanges. He is almost forgotten but he should be an extremely well-known name indeed, as well-known as other famous British wartime engineers such as Frank Whittle and Barnes Wallis. He developed the world's first semi-programmable electronic computer, under extremely difficult wartime conditions, which enabled some very secret Nazi communications to be decrypted. He was unable to patent his developments. Of the imposed secrecy which lasted until the 1980's, he wrote, "The one thing I lacked was prestige." Flowers died in 1998.

[1] See *Colossus: The secrets of Bletchley Park's code-breaking computers*, B Jack Copeland et al, OUP 2006, which includes some recently-declassified information. Also http://www.bbc.co.uk/news/technology-21384672

After the war, work proceeded in parallel in the UK and the United States, with multiple developments, and a simple chronological order becomes almost impossible.

In 1946, Max Newman (who had worked with Flowers on Colossus) went on to establish the Royal Society Computing Machine Laboratory at Manchester University, where with colleagues he built the world's first electronic stored-program digital computer, called the 'Manchester Baby', which was first operational in June 1948. Their development led to the world's first commercially available general-purpose electronic computer, called the Ferranti Mark 1. The first was delivered in February 1951.

Whereas much of the UK-based wartime work was kept secret, work progressed in the USA in parallel with the UK work, and in a more open fashion. ENIAC (Electronic Numerical Integrator and Calculator) was conceived and designed by John Mauchly (1907-1980) and John Adam Presper "Pres" Eckert (1919-1995), following a 1942 memo from Mauchly proposing a general-purpose electronic computer. A contract was received from the US Army in 1943, and the completed machine was announced to the public in February 1946. It used some 17000 thermionic valves and weighed about 27 tonnes. Vacuum tube failure rate was such that its availability was only about fifty percent. Input was via an IBM card reader.

The successor to ENIAC was called EDVAC (Electronic Discrete Variable Automatic Computer), which was proposed by Mauchly and Eckert in 1944. The Hungarian-American mathematician, polymath and all-round-genius John von Neumann acted as a consultant on the EDVAC project, and in 1945 he wrote a report[2] summarising the project and proposing some improvements. In particular, von Neumann described a design architecture for electronic digital computers, separating the computer into a central processing unit (CPU), a memory, and input and output mechanisms. EDVAC included a magnetic tape reader (or wire recorder), a control unit with an oscilloscope, a dispatcher unit for receiving instructions from the control with memory to enable redirection to other units, a computational unit, a timer, a memory unit comprising two sets of 64 mercury delay lines, and temporary memory. It had some 6000 thermionic valves and consumed 56 kilowatts of power. EDVAC was delivered in 1949 and operated until 1960. Its delivery was delayed because of disputes between Eckert and Mauchly and their then-employers, the University of Pennsylvania; such disputes have been a recurring theme in computer developments right up to the present day.

Back in the UK, Maurice Wilkes (1913-2010) at Cambridge University was able to read a copy of von Neumann's 1945 report which inspired him to develop a working stored-program computer called EDSAC, the Electronic Delay Storage Automatic Calculator, which was operational in May 1949 and actually pre-dated EDVAC.

The first US-produced commercially-available electronic stored-program digital computer was UNIVAC 1, which was developed by Eckert and Mauchly and was delivered to the US Census Bureau in March 1951.

The name of John Von Neumann (1903-1957) appears above in relation to EDVAC and EDSAC; he made contributions to many aspects of science and technology in the twentieth century, including

[2] "First Draft of a Report on the EDVAC", John von Neumann, Moore School of Electrical Engineering, University of Pennsylvania, 30 June 1945

mathematics, game theory, quantum mechanics, linear programming, the Manhattan project that led to the atomic bomb, and the post-war development of the hydrogen bomb. He applied game theory to the Cold War nuclear stand-off, and helped develop the strategy of Mutually Assured Destruction (MAD) which (arguably) helped avoid global destruction during the period from the 1950's until 1990.

Von Neumann was also a founding figure in the development of computer science. With Stanislaw Ulam he developed the Monte Carlo method for statistical modelling of complex problems such as nuclear criticality; these problems can really only be solved by computers, since they require many thousand repeat calculations. He also proposed improvements to ENIAC that enabled it to run stored programs. In 1949, he designed the first self-replicating computer program, thus anticipating computer viruses.

The above paragraphs trivialise John von Neumann's achievements. He did so much, in such a wide variety of topics, that it is difficult to describe his achievements without sounding hyperbolic. His name should be extremely well-known indeed, as much as Einstein or Newton; perhaps he will receive wider recognition at some point in the future[3].

A final name-check in this first, pre-transistor stage of computer development is for Harry Huskey. Harry Huskey was an American who had worked on ENIAC, and then worked with Alan Turing at the UK's National Physical Laboratory on development of the ACE computer in 1947. He then moved back to the USA and worked on the EDVAC project and, later, he designed the G15 computer for the Bendix Corporation in Los Angeles. The Bendix G15 used 450 thermionic valves, weighed 950 pounds (430 kg), and was the size of a small cupboard, about 1.5 m^3. It cost $49500 to $60000 and could also be rented for $1500 per month. It went on sale in 1954, and over 400 were manufactured with some remaining in use until 1970. With the Bendix G15, the computer truly entered the marketplace.

Hence, by 1950, the basics of computer science and engineering were in place – logic elements for computation, memory, programming, means of control, and input and output devices. The ability to scale up the early computer designs was limited, however. The logic elements used in computation and processing normally employed thousands of thermionic valves, which were large, consumed a lot of power, and were unreliable. Memory size was limited because of the space requirements and general clumsiness of mercury delay lines. Fundamental improvements in technology had to be made for the computer to increase in capability and reduce in size.

Mercury delay lines and thermionic valves have now gone the way of the dodo, the steam engine, the gramophone and the video cassette recorder. Their replacements came ultimately from the invention of the transistor in the Bell Laboratories (formerly the Bell Telephone Laboratories, originally named after and created by the inventor of the telephone, Alexander Graham Bell), in New Jersey, in the period 1946 to 1951. This was the product of a team including William Shockley, John Bardeen and Walter Brattain, for which they were jointly awarded the Nobel Prize for Physics in

[3] For a greater insight into the extraordinary mind of John von Neumann, see for example *Prisoner's Dilemma*, William Poundstone, Anchor Books, 1992.

1956. John Bardeen is the only person to have been awarded the Nobel Prize for Physics twice (he won it again in 1972 for the theory of superconductivity known as the BCS theory). John Shive was another important team member, although he was not honoured by the Nobel committee. However, it is Shockley who is most often associated with the invention of the transistor.

Shockley had during the war on radar and anti-submarine warfare. In July 1945, he was asked by the US War Department to prepare a report on the likely casualties if Japan were invaded. His report predicted millions of both Japanese and Allied casualties, and influenced the decision to use the atomic bomb.

After the war, Shockley and his team at Bell Labs first developed the point-contact transistor, and then the junction transistor, using germanium, and for which they received the Nobel Prize. However, Shockley was a difficult person to work with – he had a domineering, abrasive style that caused him to fall out with his co-inventors and subsequently to leave Bell Labs. He then set up his own Shockley Semiconductor Laboratory, which became part of what would later be known as Silicon Valley in California. Shockley Semiconductor Laboratory was sold to ITT in 1968.

In later life Shockley was Professor of Engineering and Applied Science at Stanford University. He became interested in eugenics, and made some fairly bizarre statements about intelligence, race and reproduction, including advocating that people of low intelligence should be sterilised. He donated his sperm to a sperm bank that only supplied sperm from high IQ individuals. In 1981 he sued a newspaper for calling him a 'Hitlerite'.

Following the invention of the transistor, some visionary people began almost immediately to conceive of an integrated circuit, where transistors, resistors, diodes and connecting links could be constructed on a single piece of semiconducting silicon. Geoffrey Dummer (1909-2002) was a graduate of Manchester College of Technology who worked at the Royal Radar Establishment during the Second World War. In 1952 he presented a paper at a US conference in which he suggested that "it now seems possible to envisage electronic equipment in a solid block with no interconnecting wires". He is recognised as the "Prophet of the Integrated Circuit" and he went on to propose a simple design for an integrated circuit in 1957, although he never produced a functioning circuit.

A group of Shockley's co-workers left Shockley Semiconductor Laboratory in 1958 to set up Fairchild Semiconductors, because Shockley would not allow work to continue on the silicon transistor. The group included Robert Noyce (1927-1990) and Gordon Moore (b.1929).

Noyce filed a patent for the integrated circuit in July 1959, a few months after Jack Kilby (1923-2005) had produced the world's first integrated circuit while working for Texas Instruments. Kilby received the Nobel Prize for Physics in 2000.

Moore and Noyce together founded the Intel Corporation in 1968, which today is the world's largest and highest valued semiconductor chip manufacturer. Moore's name is probably best known for his proposal of what has become known as 'Moore's Law' which is based on a presentation he gave in 1965 showing that the number of components in integrated circuits had doubled every year and he predicted at that time the trend would continue for at least a further ten years. In fact, the trend has continued to the present day, and Intel expects to be offering memory chips with a 'feature size' of 10 nanometres by 2015. This astonishing rate of technical development has enabled integrated

circuits, microprocessors and memory chips to have the power and the ubiquity they currently enjoy. However, it is becoming clear that the limits of Moore's Law are being reached and any further reduction in feature size will require another radical change in technology, if that is possible.

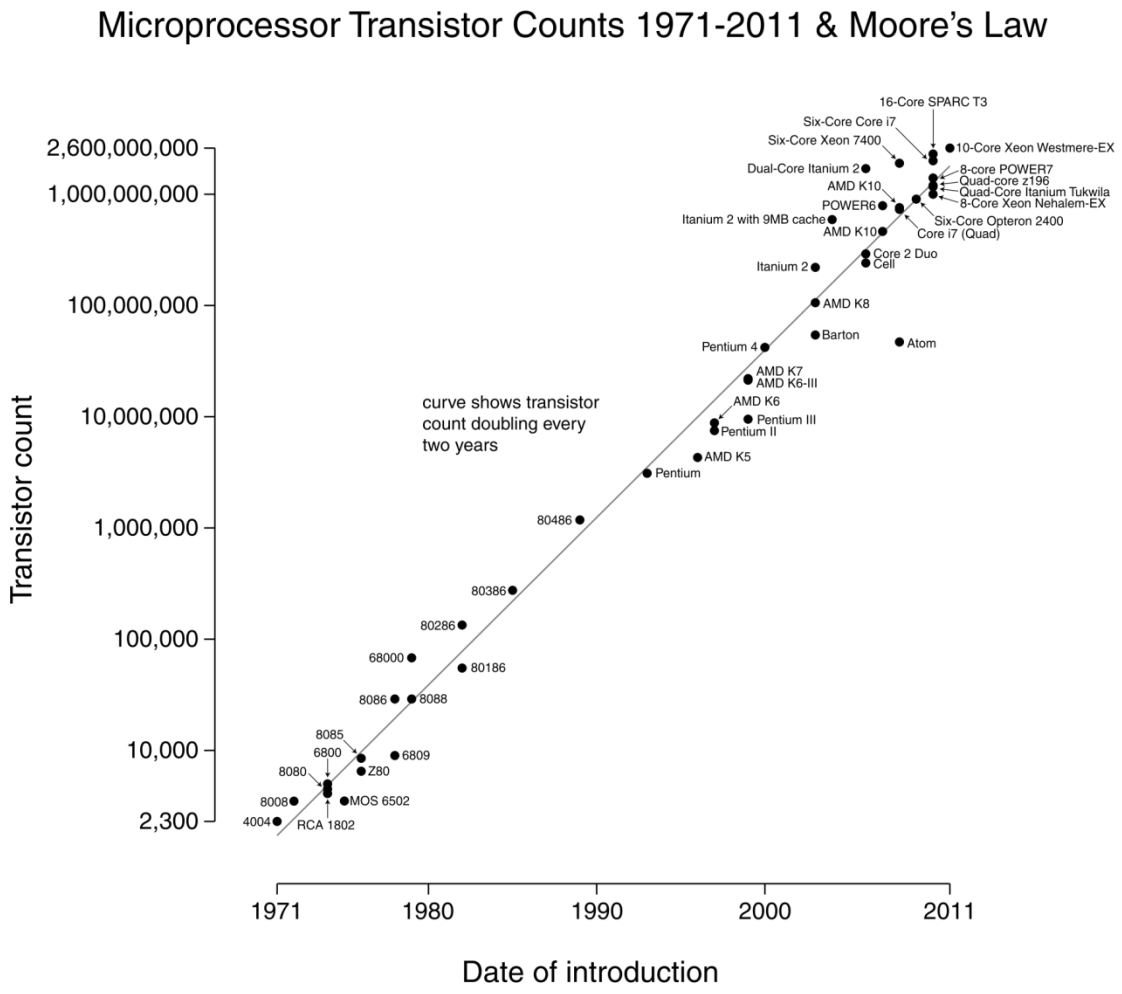

Fig 1: Moore's Law 1971-2011

Source: WG Simon, Wikimedia Commons 2011.

The Development of the Computer – A Highly Selective Roll of Honour

Unlike many other technical developments, the computer cannot be said to have any single inventor. Here is a personal selection of some of the key innovators whose work led to computers as we know them today. In some instances the work will have been the efforts of large teams; assignment of credit to individuals may be invidious.

Blaise Pascal	1623-1662	France	*Mechanical calculator*
Joseph Marie Jacquard	1752-1834	France	*Programmable loom*
Charles Babbage	1791-1871	UK	'*Difference Engine' and 'Analytical Engine'*
Lady Ada Lovelace	1815-1852	UK	*First 'program'*
Alan Turing	1912-1954	UK	'*Universal Turing Machine' concept, ACE*
Tommy Flowers	1905-1998	UK	*Colossus semi-programmable electronic computer*
Max Newman	1897-1984	UK	*'Manchester Baby', Ferranti Mark 1*
John Mauchly	1907-1980	USA	*ENIAC/EDVAC/UNIVAC*
Pres Eckett	1919-1985	USA	*ENIAC/EDVAC/UNIVAC*
John von Neumann	1903-1957	Hungary	*'Von Neumann architecture'*
Maurice Wilkes	1913-2010	UK	*EDSAC*
Harry Huskey	1916-	USA	*Bendix G15*
William Shockley	1910-1989	USA	*Transistor (Bell Labs)*
John Bardeen	1908-1991	USA	*Transistor (Bell Labs)*
Walter Brattain	1902-1987	USA	*Transistor (Bell Labs)*
John Shive	1913-1984	USA	*Transistor (Bell Labs)*
Geoffrey Dummer	1909-2002	UK	'*Prophet of the integrated circuit'*
Robert Noyce	1927-1990	USA	*Integrated circuit (Intel)*
Gordon Moore	1929-	USA	*Integrated circuit (Intel)*
Jack Kilby	1923-2005	USA	*Integrated circuit (Texas Instruments)*
Ted Hoff	1937-	USA	*Microprocessor (Intel)*
Federico Faggin	1941-	Italy	*Microprocessor (Intel)*
Chuck Thacker	1943-	USA	*Personal Computer (Xerox PARC, project leader)*
Steve Jobs	1955-2011	USA	*Apple*
Bill Gates	1955-	USA	*Microsoft*

Ted Hoff (b. 1937) joined Intel in 1967 and is credited with the insight that led to development of microprocessors during the nineteen-seventies. Federico Faggin (b.1941) also worked for Intel and, in 1970-1971 developed the methodology and chip design which led to the first microprocessor. Faggin had previously worked for Fairchild Semiconductor where he led the project that developed silicon gate MOS (Metal Oxide Silicon) technology that enabled the production of semiconductor memories and microprocessors.

In the early nineteen-seventies, the Xerox Corporation's Palo Alto Research Centre (PARC) developed what could probably be called the world's first personal computer as a research project. Chuck Thacker was the project leader. It was never a commercial product although some two thousand were built. Called the Xerox Alto, it featured a Graphical User Interface (GUI) on a television screen and a mouse with point-click functionality. It used a Texas Instuments 74181 chip and had 128 kilobytes of main memory, with 2.5 megabyte hard disk. In short, it had many of the features we recognise in today's personal computers. It was undoubtedly highly influential in the later developments of Apple and IBM.

Like many others, I worked on mainframe computers in the nineteen-seventies, cursing their inflexibility on a frequent basis. The first personal computer I used (in 1979) was a Commodore PET, which used audio cassettes for data storage. The IBM PC was introduced in 1981 but required the user to be fluent in the Disk Operating System (DOS) computer language. DOS was an early Microsoft development, but it was very clumsy.

I remember the first time I used an Apple Macintosh in 1984. You switched it on and it just *worked*. I remember thinking, "This is what computers should be like!"

COMPUTERS IN INDUSTRIAL CONTROL APPLICATIONS

One early use of computers in industrial control systems was the Apollo Guidance Computer, used in the Apollo spacecraft for the moon landings between 1969 and 1972. The astronauts communicated with it via a keypad, and the computer gave messages to the astronauts using a numerical display. The computer carried out computation for guidance, navigation and control. It was designed by MIT Instrumentation Laboratory and manufactured by Raytheon, using Fairchild Semiconductor integrated circuits. During the first moon landing, the computer famously issued '1202' alarm messages during the latter stages; Mission Control advised Neil Armstrong to ignore them.

In the UK, the Atomic Energy Authority's Prototype Fast Reactor at Dounreay in Northern Scotland was an early adopter of computer control systems in safety-related industrial applications. The PFR was designed in the early nineteen-seventies and used dual-redundant Ferranti Argus 500 computers for plant indications, alarm messages and some control functions.

In the nineteen-seventies and -eighties, computers such as PDP-8, PDP-11, and VAX from the Digital Equipment Corporation (DEC), or other manufacturers such as Honeywell, became widespread in industry. These were generally known as minicomputers (to differentiate them from the older, larger

mainframe computers) although they were still each the size of a large cupboard. (DEC was acquired in 1998 by Compaq. Compaq merged in 2002 with Hewlett-Packard.)

However, by the late nineteen-seventies, some concerns were appearing about software safety and reliability. Academic work to try to analyse or predict software reliability was not very productive. Software was not amenable to the approach used in hardware systems for reliability calculations, where failure rate data for each type of component could be used to produce an overall estimate of failure rate for a system or sub-system. The problem with software is that it is generally unique to each application.

During the nineteen-eighties and -nineties, many companies began to produce microprocessor-based industrial control equipment for use in factories, oil platforms, refineries, and power plants. Through the normal process of mergers and acquisitions, this sector of industry has 'shaken down' until today most of the main players in microprocessor-based industrial digital control systems can be identified in a single table:

Some major suppliers of microprocessor-based industrial digital control systems (2013)

Supplier	**Digital control system brand names**
Siemens	Simatic SPPA-T2000
Westinghouse	Common Q
ABB	System 800xA Control Advant AC450
Emerson	DeltaV Ovation
Yokogawa	Centum
Invensys	Foxboro
Rockwell	Regent ICS Triplex AADvance
Honeywell	Experion/TDC 3000
Doosan (S Korea)	HFC 6000
CTEC (China)	HolliAs
Mitsubishi	Meltac

COMPUTERS AND SAFETY

The point of the above short history is that we are still really just at the beginning of the Second Industrial Revolution, i.e. the revolution brought about by computers. The pace of change is still incredibly rapid, and we are all still learning fast. It is just this sort of environment where mistakes are made, and when computers are used in safety-related applications we need to proceed with caution.

In the nineteen-eighties, concerns about the application of software-based systems for the control of hazardous industrial processes began to be expressed. A first attempt to produce guidelines on the design and verification of software for high-integrity applications, the Programmable Electronic Systems (PES) Guidelines, was produced by a cross-industry committee. The PES Guidelines subsequently formed a contribution to the development of international standards such as IEC 61508 "FunctionaI safety of electrical /electronic /programmable electronic safety-related systems" IEC 61508 was first formally issued in 1998 although it had been in gestation for many years[4]. (IEC stands for the *International Electro-technical Committee*.)

The approach adopted by international standards for the production of safety-related software is generally risk-based. In summary, this means that, firstly, there must be *rigorous specification*, to ensure that the software programmers (who may not understand the industrial process for which they are writing software, where the software may be protecting against major accidents) produce software which delivers the correct functions. Next, there must be *risk targets* which can be used to determine how reliable the software-based system should be. Then, *graded quality assurance* is applied – the greater the reliability claim, the more elaborate and extreme the efforts that must be made to check the software does exactly what it is supposed to do.

So, why are we concerned? What is wrong with software that we are so worried about it, and why is it so difficult to confirm its fitness-for-purpose? There are particular reasons which mean that microprocessor-based industrial control systems (including aircraft control systems), with instructions encoded in software, require extra care:

1. Software is invisible. It exists inside a computer and the only assurances that anyone has that the software has been written to the correct specification, verified properly, tested properly, or even that the right version has been installed, are because of quality assurances checks – which means signatures on pieces of paper. You just cannot go up to a computer and look at the software inside it.

2. In a typical large industrial application, there may be thousands of plant inputs going into the microprocessor-based control system. This means that, ultimately, each microprocessor may receive thousands of inputs. The problem this causes is that, although you can test to ensure that an input (or combination of inputs) will generate the right output, you can't test to ensure that *all possible* combinations of inputs can *never* generate an incorrect output – there will just be too many combinations.

[4] See, for example, *Safety-Critical Computer Systems*, Neil Storey, Prentice Hall, 1996

3. A microprocessor-based industrial control system may have millions of lines of software code – the *application software*. Checking this code becomes a huge project in its own right.

4. The microprocessor works in a cyclic way at extremely high speed, repeating all its routines millions of times per second. (See Figure 2.) It does not process each input in parallel – if any input changes state, there will be a miniscule delay before the microprocessor notices. We say the microprocessor is carrying out *serial processing* of its input data. Hence it is actually impossible to know the signal flow path in the same way that you would know for an old-fashioned 'hard-wired' electronic control system. The microprocessor decides its own signal flow path on a moment-by-moment basis. The decisions about the signal flow path (and other key decisions) are made by the microprocessor's own *operating system software* – which may also consist of millions of lines of code (in addition to the application software). Thoroughly checking the operating system, and any possible dynamic interactions between the application software, the operating system software, and the hardware, becomes a huge and virtually impossible task. It is frankly impossible to carry out full, comprehensive checks on millions of lines of software, which is why software like Microsoft Windows sometimes behaves strangely.

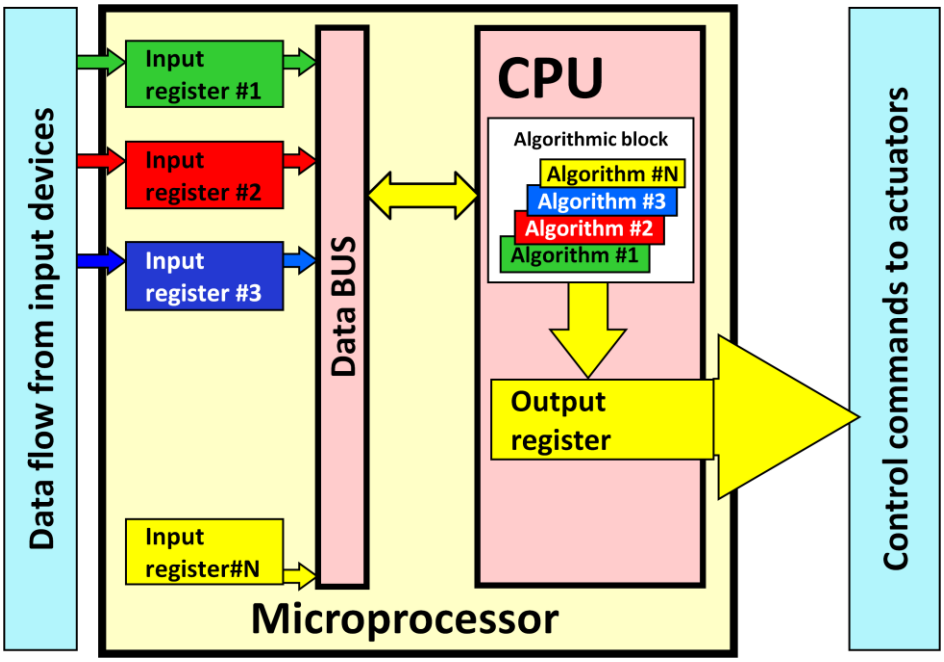

Fig 2: This shows the microprocessor in an industrial control application. This Central Processing Unit (CPU) does its activities one at a time ('serial processing') - but at extremely high speed.

SOME SAFETY PROBLEMS WITH MICROPROCESSOR-BASED CONTROL SYSTEMS

Case studies are probably the best - and most interesting - way to explain the problems with microprocessor control systems. However, remember that before computers, control systems failed also. There is no point in trying to say we should turn back the clock and stop using computers in industrial control systems – they are here to stay. Also, old control systems were not as versatile as microprocessor-based systems, nor did they have diagnostic systems anything like modern systems. However, we have to be alert to the difficulties of microprocessor-based systems and learn from past mistakes.

I will start with a trivial example which nevertheless illustrates the point that *software is invisible*. This example was reported by my friend and colleague Tom Nobes at a conference in 2007. The supplier's name is omitted to protect the guilty.

Industrial plant all over the world has for many decades used chart recorders, where a parameter of interest – say a temperature in a particular part of process plant – is monitored using a pen on a roll of paper. The pen moves in proportion to the temperature signal, and the paper moves forward at a constant speed. A small section of paper is visible, so the last few minutes of recording are always available for inspection. Multi-pen chart recorders were available, using different colours of ink, so that a number of parameters could be recorded on a single roll of paper; a single roll of paper could last for days. You could then use the roll of paper to go back and review plant changes, say after an incident, so you could investigate what happened.

The problem with pen recorders was that, whenever you wanted to investigate something, it always seemed that the ink had run out or the paper had run out. So, with the advent of microprocessors and memory chips, suppliers started to provide 'paperless' chart recorders, where the section of 'paper' that would normally be visible was replaced by a small LCD display, and the memory inside the instrument stored the signals for later replay if required. They were made to be the same size as old paper chart recorders so they were 'like-for-like' replacements of the older units. Brilliant. They have now replaced the older paper chart recorders almost everywhere, and everyone is delighted.

However, in about 2005, operators at Sellafield nuclear reprocessing plant began to notice problems with one particular make of paperless chart recorder. They began to 'go to sleep' or require constant re-booting. Chart recorders were swapped for spares, and sent back to the manufacturer, but the problem kept recurring.

Eventually, it was discovered that the operating system software in the paperless chart recorder contained what programmers call an 'Easter egg'. In the factory-installed software, there was included a submarine-warfare computer game based on the movie 'The Hunt for Red October'. If a particular combination of buttons on its front panel were operated, the chart recorder changed to a game console. Plant operators had somehow found this – probably by chance - and were playing the game on quiet night shifts. The problem was that the game caused the memory to overflow, and dumped old stored data, and also caused the whole chart recorder to start operating slowly and temperamentally.

This 'Easter egg' is one of numerous examples. I have been told that an old version of Microsoft Excel (Excel 95) had a similar hidden computer game, if one knew where to find it, called 'The Hall of

Tortured Souls'. Normally, this sort of thing should not matter (although it is certainly not good practice by the programmers.) In the case of the paperless chart recorder, it caused the recorder to malfunction because it flooded the memory.

The next examples of failures relate to *the difficulty of testing software-based systems* in all possible scenarios. The problem with these examples is that is always possible, with hindsight, to say "Why didn't they know that?" It should be the case that, especially for safety-related software systems, the development project includes thorough verification of the software and validation testing[5]. For high-integrity applications, where the software could cause serious loss or injury if it were to fail, the verification and validation should be done completely independently – so called Independent Verification and Validation or IV&V.

The reality is that people are fallible, and development projects can be immensely complicated, and the interactions between application software and operating system software and hardware can be extremely tricky. Sometimes, even with good planning and conscientious people, things can be overlooked.

Automobile recalls by the manufacturers to fix software problems have now become commonplace. To my knowledge, the list of affected manufacturers includes Cadillac, Ford, General Motors, Honda, Jaguar, Lexus, Nissan, Pontiac, and Toyota – but there may be others. Some of these software faults will be non-safety issues, but undoubtedly some have been safety-related, and some may even have cost lives. Of course, manufacturers do their best to keep matters which could generate adverse publicity out of the news as much as possible. Some examples of which I am aware are as follows:

> "Jaguar has recalled nearly 18,000 X-type cars after it discovered a major software fault, which meant drivers might not be able to turn off cruise control. The problem lies with engine management control software developed in-house by Jaguar. The problematic software is only installed on diesel engine X-Types, which were all produced between 2006 and 2010. Some 17,678 vehicles have been recalled, as a result of the potentially dangerous problem. If the fault occurs, cruise control can only be disabled by turning off the ignition while driving – which would mean a loss of some control and in many cars also disables power steering. Braking or pressing the cancel button will not work. "Jaguar has identified that should an error with certain interfacing systems be detected the cruise control system will be disabled and an error message displayed to the driver on the instrument cluster," the company said in a statement."
>
> *Source: Computer World UK, 24 October 2011*

[5] *Verification* means the software is reviewed against its specification, line-by-line, by someone other than the author. *Validation* means the software is tested as thoroughly and realistically as possible.

> “The Prius, Toyota's technological flagship, may have a brake problem. Under certain conditions, according to the company, there may be slight interruption in the car's brake response. The fix, says Toyota, is a software update that dealers will download to all of the 133,000 2010 Prius and 14,500 of the similar 2010 Lexus HS250h models on the road. The brake system on these hybrids is complicated and different from what's on conventional cars. The system combines hydraulic brakes with brake regeneration and an anti-lock function. Under light braking, the calipers don't squeeze the rotors. Instead, the resistance of the electric motors provides the deceleration. This is how the Prius captures the moving car's energy, charges the batteries, and later electrically boosts acceleration. Most hybrids work this way to keep fuel economy numbers high. Harder braking engages the calipers in a normal fashion. And finally threshold, or maximum braking engages the ABS system to keep the tires from skidding. The computer choreographs the various functions with inputs from several sources like the wheel-speed sensors, battery charge meter, and brake-pedal stroke. Toyota says that under certain conditions, like on an especially bumpy or slippery road, there may be brief momentary delay in the brakes response. The car will still stop--we're not talking about brake failure here--but the distance required could increase slightly. The software fix tweaks the software to better deal with those rare conditions. There are no new parts with this recall. The technician will simply hook a laptop to the car and download the new firmware.”
>
> *Source: Popular Mechanics, 9 Feb 2010*

> “Honda stated that it is also recalling 5626 CR-Z vehicles from the 2011 model year in the United States that are equipped with manual transmissions, to update the software that controls the hybrid electric motor. Under certain circumstances, it is possible, according to the company, "...for the electric motor to rotate in the direction opposite to that selected by the transmission.” Honda also recalled 2.5 million 2005–2010 4-cylinder Accord, 2007–2010 CR-V, and 2005–2008 Element vehicles— 1.5 million in the United States—to update the software that controls their automatic transmissions. According to Honda, "Without the updated software, the automatic transmission secondary shaft bearing in the affected vehicles can be damaged if the transmission is quickly shifted between each of the reverse, neutral and drive positions, as may be done in an attempt to dislodge a vehicle stuck in mud or snow. If the bearing is damaged in this unusual scenario, it can cause the engine to stall or lead to difficulty engaging the parking gear. The update to the vehicle’s automatic transmission control module software will ease the transition between gears to reduce the possibility of damage.”
>
> *Source: IEEE Spectrum 7 Sept 2011*

It is difficult to be sure how many accidents (if any, fatal or otherwise) occurred before the above recalls were made. Unless criminal charges are brought, motor manufacturers will try, wherever possible, to settle out of court in order to minimise publicity.

QANTAS FLIGHT 72

Fatal accidents or serious incidents in civil aviation or in industry generally lead to some form of independent enquiry and the subsequent report is openly published. Many such accidents and incidents have now occurred where software could be held to blame for the incident.

An interesting example of a non-fatal accident, which nevertheless led to a large number of injuries, and which must also have scared witless everyone else, was the Qantas Airbus A330 "in-flight upset" of 7 October 2008. The published report is a fascinating read[6]. The aircraft was carrying 303 passengers and 12 crew members on scheduled flight Qantas 72 from Singapore to Perth, Australia in a steady cruise at 37000 feet, flying on auto-pilot. The events as experienced by the passengers and crew can be summarised quite briefly as follows:

At 0440:26 UTC (local time 1240:26) one of the Air Data Inertial Reference Units started producing multiple intermittent spike signals, and the crew received a number of warning messages many of which were spurious and unhelpful.

At 0441:27 UTC (only one minute later and before any meaningful analysis or interpretation of the alarms could possibly be carried out) the aircraft suddenly and violently pitched down for less than two seconds. The acceleration experienced in the cabin exceeded -1g, or -10 meters per second per second, and passengers and crew who were not strapped in at the time were physically thrown against the roof of the cabin.

At 0445:08 UTC a second less severe pitch down occurred.

The captain declared Mayday and the flight was diverted to Learmonth, Western Australia where it landed successfully at 0532 UTC.

At least 110 of the 303 passengers and nine of the 12 crew members were injured; 12 of the occupants were seriously injured and another 39 received hospital medical treatment. Damage to the cabin roof, from people impacting it due to the high negative 'g' force, was significant (see photo).

The circumstances leading to above events are quite complicated and need some detailed introduction, and there are a lot of acronyms for the different parts of the control systems.

> Like all modern Boeing and Airbus civil airliners, the Airbus A330 has a digital Electronic Flight Control System (EFCS), a 'fly-by-wire' system which monitors and controls flight parameters continuously and enables the pilot and co-pilot to take a 'supervisory' role during normal flight. The EFCS uses three Flight Control Primary Computers (FCPC's), one of is selected to be 'master'.Among the many parameters that are monitored is the Angle of Attack (AOA), the slight nose-up attitude which is maintained in normal flight to generate lift but which can lead to a stall if AOA is too great. AOA is a measurement of the angle of the

[6] ATSB Transport Safety Report, Aviation Occurrence Investigation AO-2008-070 Final, 2011

wing (using a nominal reference line known as the 'chord line') relative to the airflow. Angle Of Attack is a therefore a critical safety parameter for the EFCS, and the Flight Control Primary Computers use three independent AOA signals to check their consistency, signals AOA1, AOA2 and AOA3. The AOA signals are created by Air Data Inertial Reference Units (ADIRUs).

The AOA value is then fed into the flight control system and used, in particular, to drive signals to the elevators in the tailplane which control aircraft pitch.

"Inertial Reference" means knowing your position, speed, direction and orientation in space, without reference to external points, by measurement of accelerations in three dimensions and integrating those accelerations twice to calculate position. Modern Inertial Reference Units, such as ADIRUs, use *ring laser gyroscopes*, together with accelerometers and GPS (like an in-car satellite navigation system) to provide raw data. Ring lasers are better than older mechanical gyroscopes because there are no moving parts and they are rugged, lightweight and frictionless.

A ring laser gyroscope consists of a ring laser having two counter-propagating modes over the same path in order to detect rotation. An internal standing wave pattern is set up in response to angular rotation, known as the Sagnac effect. Interference between the counter-propagating beams, observed externally, causes shifts in that standing wave pattern, and thus rotation can be inferred.

Normally in the Airbus A330, the master EFPC used the average of AOA 1 and AOA 2 to determine the angle of attack. If either AOA1 or AOA2 deviated (or generated a signal spike) the master FPCP used a memorised value for up to 1.2 seconds, that is, it remembered the last value before the signal deviation occurred. This was to prevent the control system being too sensitive and over-reacting – which of course is exactly what happened in this incident.

The incident investigation showed that the algorithm (that is, the calculation done) in the FPCP software had a serious limitation, which meant that in a very specific situation multiple spikes on the AOA signal from only one of the ADIRUs could generate a nose-down elevator command.

The FCPC algorithm could not correctly manage a scenario where there were multiple spikes in either AOA1 or AOA2 that were 1.2 seconds apart (exactly the length of time that the FCPC would memorise old data in the event of a signal spike). The Australian Transport Safety Board report notes that "it is very unlikely that (this) FCPC design imitation could have been associated with a more adverse outcome", and that "(this was the) only known case of the design limitation affecting an aircraft's flight-path in over 28 million flight hours."

The report also notes that "……the development of the A330/A340 flight control system during 1991 and 1992 had many elements to minimise the risk of design error (and) none of these activities identified the design limitation in the FCPC's AOA algorithm……Overall, the design verification and validation processes used by the aircraft manufacturer did not fully consider the potential effects of frequent spikes in data from the ADIRU."

There had been two other known occurrences of the data-spike failure mode (presumably without inducing auto-pilot anomalies), on 12th Sept 2006 and 27th Dec 2008.

I think the main point of this incident - in which, fortunately, no-one died - is that it shows how difficult it has become to analyse and review complex software-based control systems, to identify all potential dangerous failure modes, and to ensure that sub-system failures like the ADIRU signal spikes are identified and properly addressed. The identification of the ADIRU-spike failure mode *before* the Flight 72 incident would have required some very clever thinking indeed.

Software-based systems have grown and grown in recent years, and systems with many millions of lines of application software are now common. With software of this size and complexity, it becomes very difficult indeed for any one person to have a complete overview and profound understanding of the system. In that case, it is necessary to break the system down into modules, with precise specifications for each module and the interfaces with other modules. The role of engineering management then becomes, first, to ensure that all the interfaces are properly identified and verified, then to ensure that all the safety and other functional requirements are properly identified and verified, and finally to ensure the interface requirements are identified and verified. Ensuring that “nothing falls between the cracks” still remains a huge task which requires great care and attention.

Fig 3: This photo shows some of the damage to the cabin roof caused by passengers being thrown upwards by the sudden pitch down of Qantas flight 72.

ARIANE 5 LAUNCH FAILURE

A major source of software errors in industrial systems is *specification error*, that is, the software programmer writes the correct code but the initial specification was actually incorrect. Probably about half of significant software faults are due to specification error. Again the best way to explain how this can happen is by example. It just so happens that this example, like the example of Qantas flight 72, also relates to Inertial Guidance systems.

The Ariane 5 launch on 4th June 1996 was the first of the European Space Agency's new Ariane 5 rocket launcher, which was a much bigger launcher than its predecessor Ariane 4. Thirty-seven seconds after launch from French Guiana, both the duty and the back-up Inertial Reference Systems (IRS's) failed, which led to launcher disintegration[7].

The cause was a software fault in equipment which was unchanged from Ariane 4, but which was unsuitable for the changed launch trajectory of Ariane 5. There had been inadequate analysis and simulation of the systems in Ariane 5, so the unsuitability of the old IRS for use in Ariane 5 was not revealed.

There were 2 IRS's with identical hardware and software. One was active and one was on hot stand-by. Each IRS was programmed to expect the flight trajectory of the older Ariane 4 launcher. Because the flight trajectory was different for Ariane 5, the active IRS declared a failure due to a software exception (that is, an error message). The stand-by IRS then failed for the same reason.

Actually, the affected software module only performed alignment of the inertial platform before launch – it had no function after launch. So, the error message was inappropriately handled, although the supplier was only following the contract specification.

Hence the dual-redundant inertial reference system suffered common-mode failure because the software in the IRS's had not been reviewed thoroughly for its use in Ariane 5. Furthermore, the software module that generated the error messages had no function after launch. The software, which was written by a supplier in accordance with the specification from Arianespace, had a completely unnecessary function to make itself unavailable if the flight trajectory deviated from what it had been programmed to expect.

[7] A video of the launch failure can be seen at
http://www.youtube.com/watch?v=kYUrqdUyEpI&feature=related

10-4: The Ariane 5 launcher and the launch failure of June 1996

CYBER-ATTACK

We now live in an era where the potential for software problems is well understood, and design engineers do their very best to ensure that software faults are either designed out, or else the design mitigates against the worst effects of software faults. Another problem remains however - that of *malicious interference with software-based systems*. Computer hackers first entered global consciousness during the nineteen-nineties, but it was not until 2010 that 'cyber-warfare' became real. Before 2010, most people would imagine that a computer hacker would be an individual who probably had little social life, worked on a PC in his basement, and only washed occasionally.

Then, in June 2010, the Stuxnet computer worm was discovered by a Belarus-based antivirus software vendor called Virus BlokAda. Stuxnet had infected Iranian equipment, including gas centrifuges used in the Iranian uranium enrichment plant at Natanz, which was apparently being used as part of the Iranian nuclear weapons development programme.

Iran had been embargoed from buying much western advanced technology, but the Iranians had nonetheless acquired German Siemens Simatic S7 variable-frequency drive control equipment[8], apparently without Siemens knowledge, and probably via a third country. The equipment was being used to control the speed of electric motors driving gas centrifuges in uranium enrichment plant.

Naturally-occurring uranium contains only 0.7 per cent uranium-235, the rest being uranium-238. The concentration of uranium-235 must be increased to about 90 per cent to make it 'bomb-grade'. Gas centrifuges are required for this 'enrichment' process. Uranium hexafluoride is gaseous, and the centrifugal action in the gas centrifuge causes a slight increase in the concentration of uranium-235 hexafluoride at the inside of the centrifuge, since it is slighter lighter than uranium-238. Thus, a large plant with many gas centrifuges connected in series can be used to gradually increase the uranium-235 concentration.

Siemens Simatic S7 equipment uses Microsoft Windows software. The designers of the Stuxnet worm had identified four so-called 'zero-day' weaknesses in Windows, which were vulnerable to attack. Zero-day weaknesses are weaknesses in software code that were previously unknown, so the attack occurs on the first day that the weakness is known about, and the software company have had 'zero days' to be able to identify the weakness and develop a software patch to fix it. The identification of these weaknesses alone must have been a major undertaking, since Windows has many millions of lines of code.

Stuxnet was apparently designed *only* to target Simatic S7 equipment which controls variable-frequency drives. Thereafter, Stuxnet was really quite fiendish in what it did: It made the gas centrifuge motors run at the wrong speed, while simultaneously sending a signal to the control systems that the motors were running at the right speed. Hence it is possible that the Iranians were held up for some considerable time because they would not have been able to work out what was wrong with their enrichment plant, which will have been unable to produce enriched uranium of the quality they were expecting.

[8] A 'variable-frequency' drive changes the frequency of the electricity supply to an electric motor to control the speed at which the motor rotates.

At least several dozen professional and highly-knowledgeable software engineers must have worked on Stuxnet, so the development project must have been financed at government level. It appears almost certain that the United States and Israeli governments were responsible. The Stuxnet worm was transmitted by USB memory stick.

From personal experience, I can confirm that the Stuxnet worm caused a real shock in the fraternity of industrial control systems. Suddenly, we had to imagine that almost anything was possible – for all we knew, there could be computer viruses which were dormant, waiting for a signal to shut down critical parts of national infrastructure such as electricity supply or air traffic control. Briefly, paranoia was rife, across not just national boundaries but also across industrial competitors – perhaps another company had viruses in your company's software? The paranoia was encouraged by the lack of clear, definitive information about Stuxnet, what it had done, and who was behind it. Even now, when there have been many, many news articles about Stuxnet, it is difficult to know exactly what to believe and what to ignore.

As for the effect on the Iranian uranium enrichment programme, I can only surmise that, for security personnel and for the engineers involved, paranoia will have been out of control. All computers will have been considered suspicious, and any email or USB memory stick will have been considered a possible means for virus transmission.

Design engineers have meanwhile set to work to produce improved systems which should be more resistant to virus infection, or else detect existing infection. There is little doubt that an arms race between those software engineers who seek to make industry work efficiently, and those who seek to disrupt it, will continue indefinitely. Variants of Stuxnet, called Duqu and Flame, have been reported in 2011 and 2012, respectively. Duqu was identified by Budapest University and is said to gather information on industrial control systems. Flame is said to be used for cyber-espionage in Middle Eastern countries, mostly Iran.