From Ada ……to Zuse

…… and onward

- the evolution of computer software

Computers could not exist at all without the silicon chip and all the other hardware; but all the clever and varied things they do are attributable to the software - software which embodies enormous human ingenuity. Who can we thank?

Bill Bryson said that "a computer is a stupid machine with the ability to do incredibly smart things, while computer programmers are smart people with the ability to do incredibly stupid things. They are, in short, a perfect match."

If we asked people to list the names of creators of software, most would produce a list with one name - Bill Gates. In fact there are a large number of people who really deserve to be better known than Bill Gates!

Their contributions have been much greater than his, though he is the one who has made his fortune! Many have been recognised by prestigious awards such as the Turing Award and the Lovelace Medal.

Turing Awards are given by the American Association for Computing Machinery and include an award of $250,000. Some people refer to them as computing's Nobel Prizes. I have picked out six of the recipients here, and I'll be mentioning a few more.

Lovelace Medals, just as prestigious, are given by the British Computer Society - and here are three recipients. Finally on this slide I include three people who I find interesting - all are in Wikipedia. Back to them in a moment.

Of course we can't forget Bill Gates. But he has had no awards of this kind, though he has been recognised by honorary doctorates, an honorary knighthood, the Order of the Aztec Eagle etc.

They should be as famous as Bill Gates !

*… these **Turing Award** Winners*


Wilkes


Dijkstra


Thompson & Ritchie


Wirth


Hoare


Thacker

*..these **Lovelace Medal** Winners*


Linus Torvalds 'BDFL'


Sir Tim Berners-Lee OM KBE FRS


Karen Sparck-Jones

.. and two other interesting people


Sophie Wilson CBE FRS


Alan Black

Before going on with the story, there's a bit of personal stuff. Karen Sparck-Jones observed that "Computing is too important to be left to men." and continued "**I think women bring a different perspective to computing, they are more thoughtful and less inclined to go straight for technical fixes.**"

One of her proteges was our daughter Alison. Many of you know that we lost her to cancer in 2009. Alison co-authored several papers with Karen Sparck-Jones and Karen Sparck-Jones was acknowledged in Alison's first book. The acknowledgements in that book ended by thanking Richard for being Richard! Richard became, and is now our son-in-law. They shared a house in Edinburgh with two other computer scientists who Kay and I got to know personally, and one was Alan Black (here), now a world expert in the generation of speech by computers. Another member of their circle was Robert Dale, another achiever who is now a professor in Australia. I shall return to them all later in the talk.

There are two other people in my "interesting" category. James Ellis was a colleague I knew well, and I'll return to him too, and also to Sophie Wilson. (No-one has asked ?) BDFL is "Benevolent Dictator For Life"! Torvalds isn't the only one who has been given this label. Quirky, but not actually silly. There are a lot of unconventional people in the story I shall tell. In fact we will occasionally visit geekdom!

The early pioneers

Ada Lovelace

Max Newman

Alan Turing

The Eniac Programmers

John von Neumann

Konrad Zuse

Before getting into the main story, I want to spend a short while on the computer pioneers. Karen Sparck Jones must have been happy that a woman is recognised as the very first programmer. That's Ada Lovelace, who worked with Charles Babbage in the 1840s. Babbage's aim was to develop a mechanical computer for generating accurate tables of mathematical functions. Ada Lovelace's program was a step-by-step description of how Babbage's machine could generate mathematical functions called Bernoulli Numbers. But Ada was also a visionary. She said that:

"The Analytical Engine might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations. . . Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent."

The early pioneers

Ada Lovelace

Max Newman

Alan Turing

The Eniac Programmers

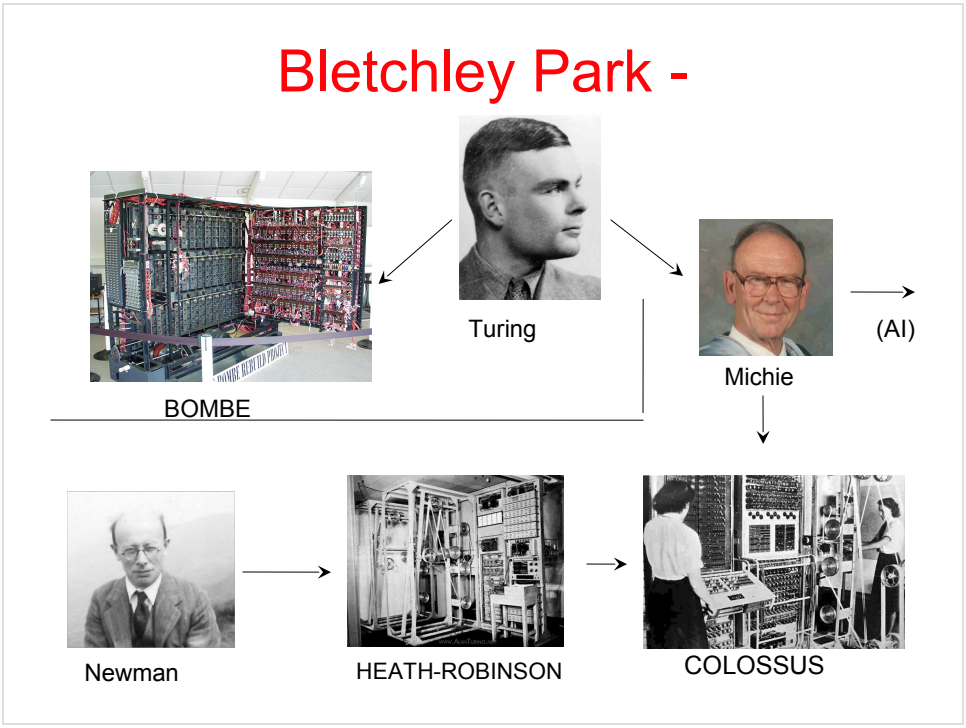
John von Neumann

Konrad Zuse

But the really important computer pioneers emerged in the 1930s. First, there are Alan Turing and Max Newman. I'll talk about them, and their work at Bletchley Park, and then return to Eniac, von Neumann and Zuse.

Turing has had all the acclaim, but Newman should probably be up there with Turing.

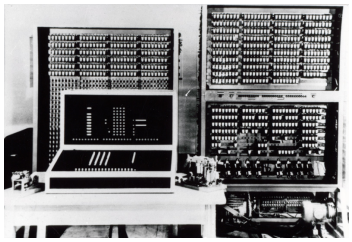
Max Newman lectured to Turing at Cambridge, and provided the trigger for Turing's work. In 1936 Turing produced an idealized mathematical model that reduces the logical structure of any computing device to its essentials. His Turing Machine was the theoretical prototype of the electronic digital computer, Turing machines are one of the key abstractions used in modern computability theory, the study of what computers can and cannot do. Appropriate Turing machines have found application in the study of artificial intelligence, the structure of languages, and pattern recognition.



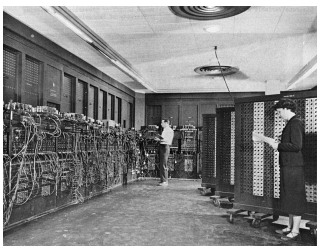
Both Newman and Turing went to Bletchley Park as codebreakers. They worked in different sections, and it is Newman's work, not Turing's which led to Colossus, that first computer. While Turing was working on the Enigma code and breaking it with the electromechanical "Bombe", Newman was attacking the even more difficult Lorenz code. He devised Heath Robinson - and that then led to Colossus.

Also at Bletchley was Donald Michie. He too contributed to Colossus. But it was Turing's ideas which inspired him to make his career in Artificial Intelligence, setting up the Edinburgh group which I have touched on already.

- And elsewhere



ZUSE-3 - 1941



ENIAC - 1946

EDVAC REPORT - 1945

Von Neumann architecture

But they were not the only people developing primitive computers. ENIAC was the early American computer, designed to calculate artillery tables. Less well known is the significant work of Konrad Zuse on programmable calculators in Germany; and Zuse even designed "Plancalcul" a high level computer language.

Machines like Eniac and Colossus were special purpose machines for which the analysis of the specific problem had determined the physical hardware. They were no good for anything else, and 'software' had not yet been born.

In 1945 John von Neumann wrote a paper on the logical design of a digital computer and that led to the term "von Neumann architecture" which is still the underlying basis of our computers. The concept needed some method of electronic data storage, and that did not yet exist
There now began a race to devise data storage methods and to demonstrate the first general-purpose stored program computer. Turing went to the National Physical Laboratory and led a team on the ACE programme. Newman went to Manchester University to lead the programme which eventually led to the Atlas machine

And it was this team which demonstrated the "Manchester Baby" testbed in 1948.

In America, von Neumann and others were working towards EDVAC which eventually became the UNIVAC of the 1950s. But Macarthyism intervened in the American program; Senior people were said to have communistic tendencies, and military contracts were cancelled.

The man who is generally reckoned to have won the race to produce a useful computer was Maurice Wilkes at Cambridge, who died just before Christmas aged 97. EDSAC was switched on in 1949 and ran the first software program to print out a table of squares. Here's the report from "The Star" newspaper

“1949 - a Mechanical Brain”
- from “The Star”

On the top floor of a rather drab building in a narrow Cambridge back street is an apparatus which seems to consist chiefly of a vast number of valves set in grey painted racks……...

This is how it works. First Mr Wilkes fed a strip of paper punched with holes into a "ticker-tape" machine. As the paper ticked through, miniature television screens showed a row of green blobs ... then almost instantaneously a teleprinter nearby began to print rows of figures. That was all……...

There are not enough "brains" to go around at the moment, but a dozen would probably be sufficient for the whole country ... The future? The "brain" may one day come down to our level and help with our income-tax and book-keeping calculations.

1949 was actually the year I went to Cambridge and while I was there I attended Maurice Wilkes lectures on computing and EDSAC.

Even the Americans now recognise the place of Colossus, and then EDSAC in the history of computers. And EDSAC is where the story of software really starts.

The EDSAC processor, like all processors, was told what to do by a succession of machine-code instructions - binary numbers drawn from the memory and obscurely defining the operations and the operands.

About Machine Code Programs

Machine Code - a stream of binary

……10111001 00101111 10001010……..

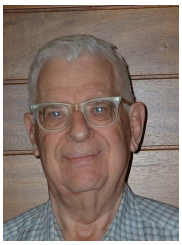
“There are 10 types of people in the world; those who understand Binary and those who don't.”

Here’s a well-known statement about binary!

Programming in machine-code is possible, but Wilkes and his team were the first to introduce mnemonics and produce an 'Assembler' to make the programmers difficult job somewhat easier. Assembler languages remained important for many more years, especially in times when programs had to be designed for speed, or had to be shoe-horned into a restricted amount of computer memory. Nowadays processors are so fast, and memory so cheap, that this is rarely the case.

EDSAC - 1949 and Leo 1 - 1951

Maurice Wilkes & EDSAC



David Wheeler

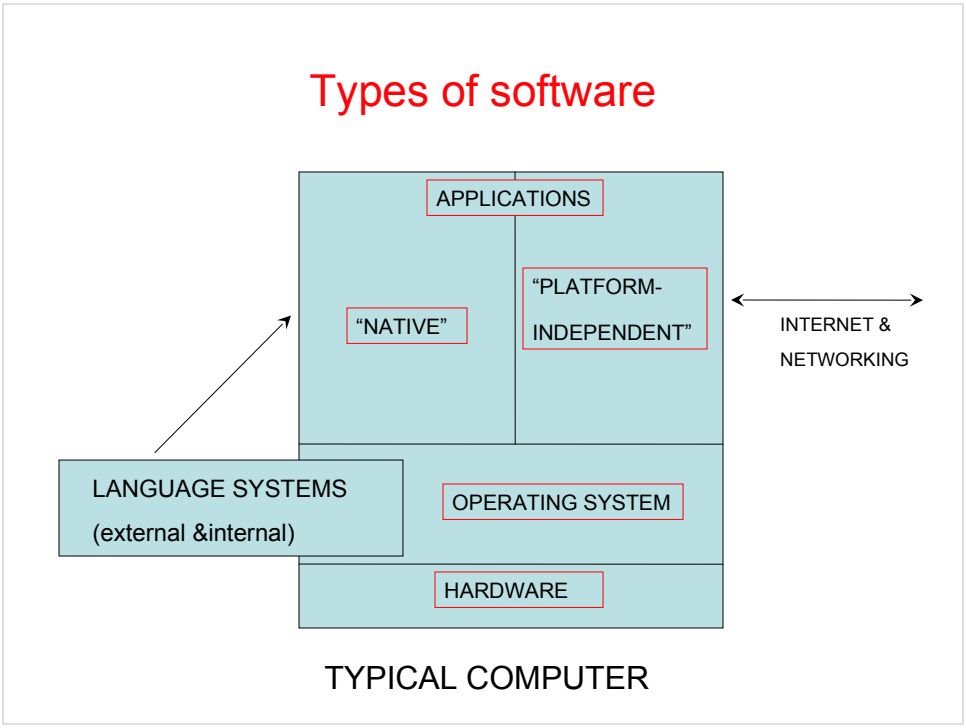
David Caminer

So here is Wilkes with EDSAC. And here is David Wheeler, the post-grad who got the very first computer science PhD for his programming work on EDSAC

The next phase of the EDSAC story was, in many ways, even more remarkable. The Joe Lyons company, the tea shop chain, were already thinking about a business computer. In 1951, LEO 1, based on EDSAC, was built and was successfully used for payroll, inventory and production control tasks. And the met office also had a LEO 1. How did they do it in a machine with only 8000 bytes of memory. My laptop there has more than 100,000 times as much memory! Here's the man who did it. (One leg, lack of recognition apart from an Honorary Doctorate)

I must now move on. What happened in the 1950s? We'll skip over the first few years of EDSAC-like computers with thousands of valves and archaic data storage systems. In the mid 1950s transistors quickly took over from valves, and

magnetic ferrite cores were introduced for data storage. And IBM was becoming a major computer manufacturer delivering their first mainframe computer in 1954. Software was now being seen as highly important, and the hardware manufacturers like IBM and Univac saw it as their job to provide complete proprietary systems - hardware and software. Specialised software companies like Microsoft did not arrive for another 20 years.

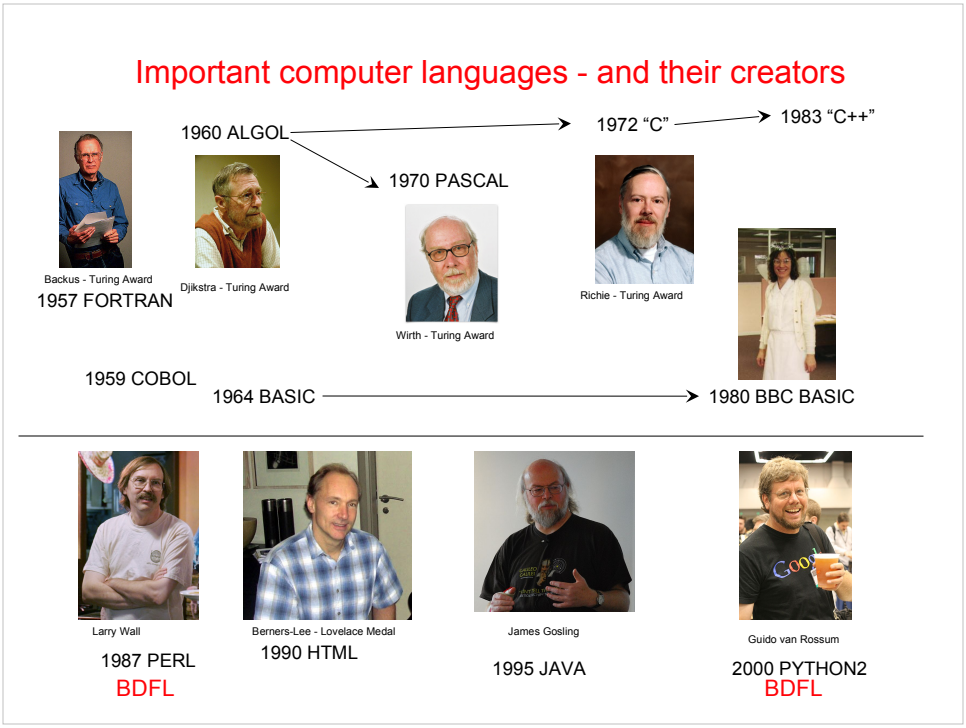


I'm going to deal with software under 3 main categories. There's the 'Applications Software' - things like word-processors, desktop publishers, internet browsers, email programs and many more. Then at the next level down there are the operating systems, like Windows. And the development of both of these requires the use of computer languages to write the extremely complex programs.

Let's start with computer languages. As the size of computer memories got bigger it rapidly became apparent that assembler language programming was just too tedious, and that the computer itself must do some of the work of turning a user-friendly description of a problem into the machine-code.

Going back to EDSACs first 'squares' program, this was inserted into the machine as a string of gobbledygook. That program was tailored to EDSAC and could not run on any other machine. Just to explain what I mean by a high level language program, here is a version in BASIC

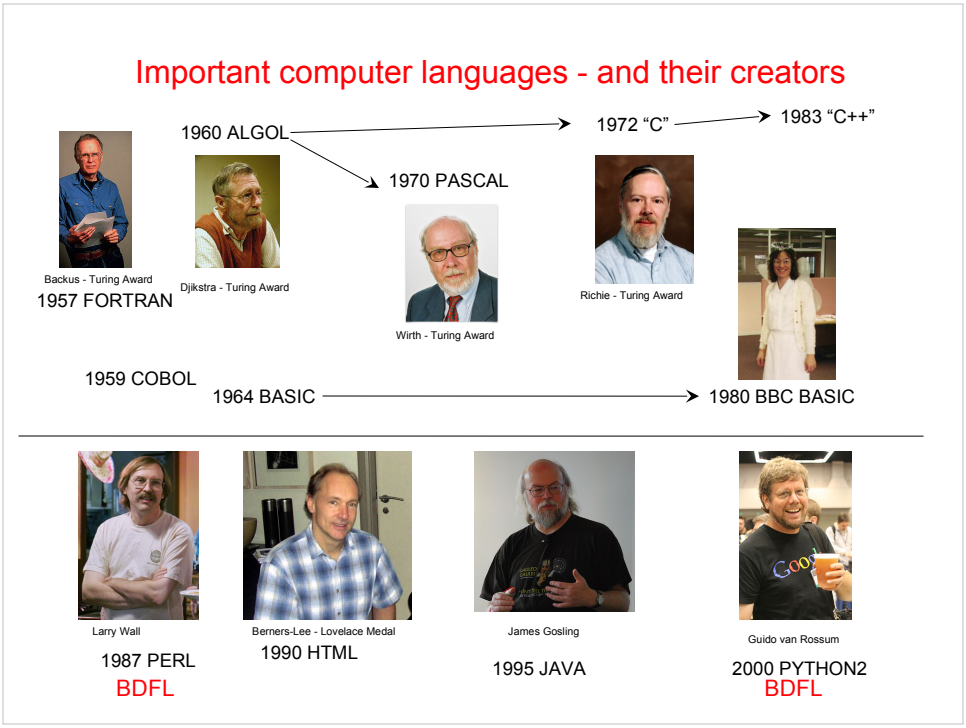
Unlike the EDSAC program, this program will in principle run on any machine. That is one of the advantages of high level languages as long as they conform to standards.



The first high-level language was Fortran developed by John Backus at IBM. It was first delivered to users in 1957. Cobol came next, a language designed by committee in 1959 in the USA. Both these languages were very successful in themselves because they were the languages backed by the large vendors like IBM; but they were dead ends. A far more technically significant early language was Algol, which resulted from a joint European/American committee which first met in 1958. Djikstra was one of the main contributors and received a Turing Award for this and other work This led to the language called Algol60. Tony Hoare was another major contributor which earned him a Turing Award. He created the version called Elliott Algol which I learnt and used in 1965. Hoare said of Algol60
"Here is a language so far ahead of its time, that it was not only an improvement on its predecessors, but also on nearly all its successors"

Not many people used Algol, compared with Fortran and Cobol. But Algol led first to Coral66, Pascal (1970), then C (1972) and C++ (1983) .

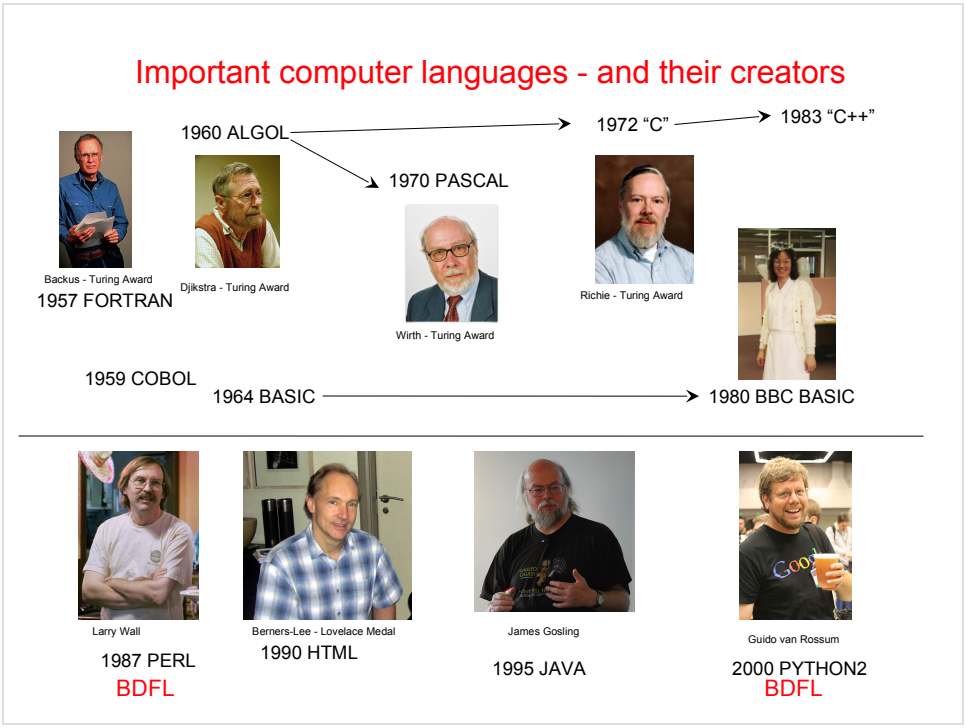
In the late 1960s most programs were still fed to mainframe machines via cardreaders, and run in succession; it was all pretty tedious. But it was becoming possible for a user at a terminal to work directly more nearly as we do now. This was the stimulus for BASIC (1964) a very simple "interpreted" language. This later led to BBC BASIC, usually regarded as the best implementation of BASIC



On now to modern times. The days have gone when computers in schools and at home had to have BASIC and were provided with a manual on BASIC. But in fact programming facilities are there, either tucked away, or down-loadable, often free of charge. Versions of C are still pre-eminent for professionals.

But there are now hundreds of other languages - and there are reasons why many new ones have been developed - some in response to changing circumstances. One thing that has happened is the enormous reduction in the cost of computer hardware. In earlier times it was vital to create compact, efficient, software, so the emphasis was on languages like C generating very efficient code for efficient use of expensive computer hardware.

In contrast, Perl (1987), a recent language, is designed to make efficient use of expensive computer programmers.
Perl's creator, Larry Wall said that "Computer language design is just like a stroll in the park. Jurassic Park, that is."
The next thing is the increased importance of software which can run on any computer. A web page for example must be rendered in essentially the same way on any computer. Two particular languages are associated with web pages - HTML (1990) and other "mark-up" languages, and Java (1995).



And my last example is Python. This could be said to be a successor to BASIC, though much more powerful. It is very popular and easy to run on PCs, Macs, Linux etc - and it is free.

There's another thing about Python. The vast majority of computer languages are in English. Unusually, Python has a chinese version. Even more unusually, the language Perl has versions in Latin and in Klingon !!! These are curiosities, but the Chinese one is serious.

Both Perl and Python are now very popular and are argued about. Python may well be winning, because it is simpler, while still powerful. As you see, both Wall and Van Rossum are referred to as "BDFL". They are regarded as the final authorities on the subject.

So, just to remind you, I've been talking about computer languages which are used to build software or interpret data.
I'll now turn to operating systems, working towards the modern systems like Windows.

What do operating systems do?

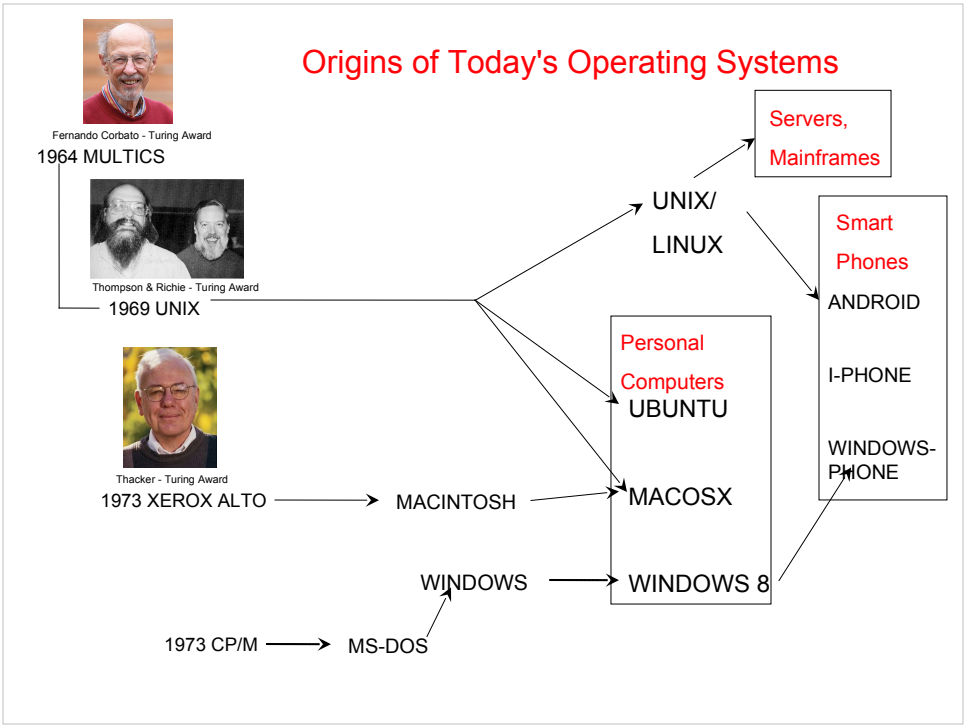
Some functions of Operating Systems
(in computers, smart-phones, tablets ……..)

- User interface - e.g. GUI (Windows)
- Filing system
- Memory management
- Managing peripherals - printers etc.
- Multi-tasking, multi-user

1. WINDOWS

2. UNIX-BASED MACOS X IOS LINUX ANDROID

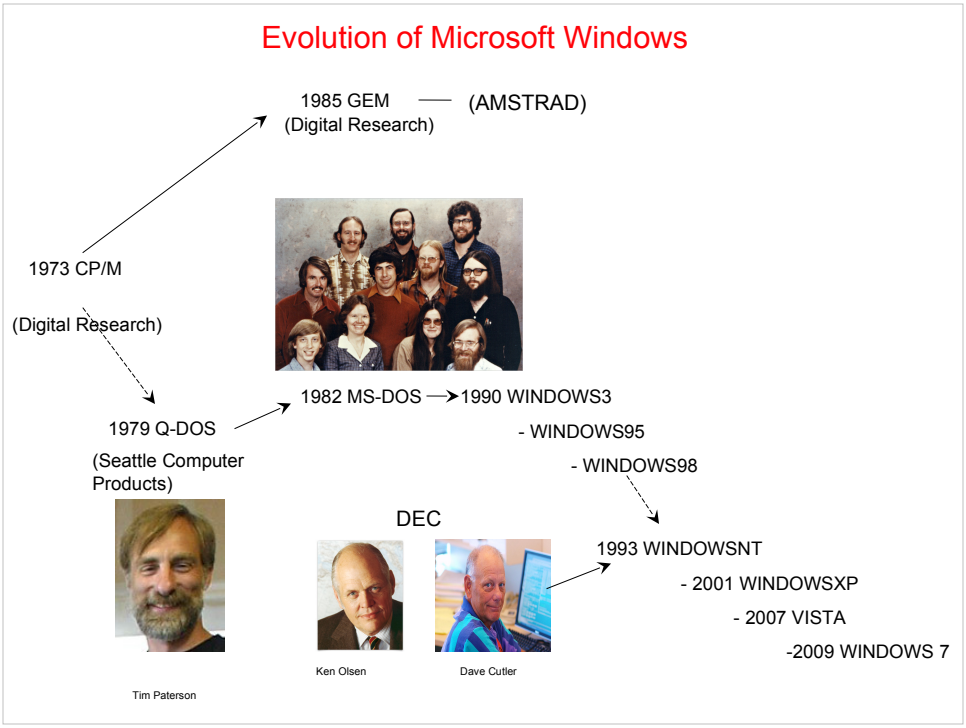
Early operating systems were created by the computer manufacturer, like IBM. They were written in assembler language.



But in 1964, Fernando Corbato and others began a very ambitious system called Multics. Multics itself was not very successful, but it was the seed for modern systems. The Bell Labs people withdrew from Multics and developed Unix the name being a play on words. (Multics has too many ickses, so lets have one ix.) Unix is still the basis for many modern systems, large and small - from mainframes to today's phones, with all their vast arrays of features.

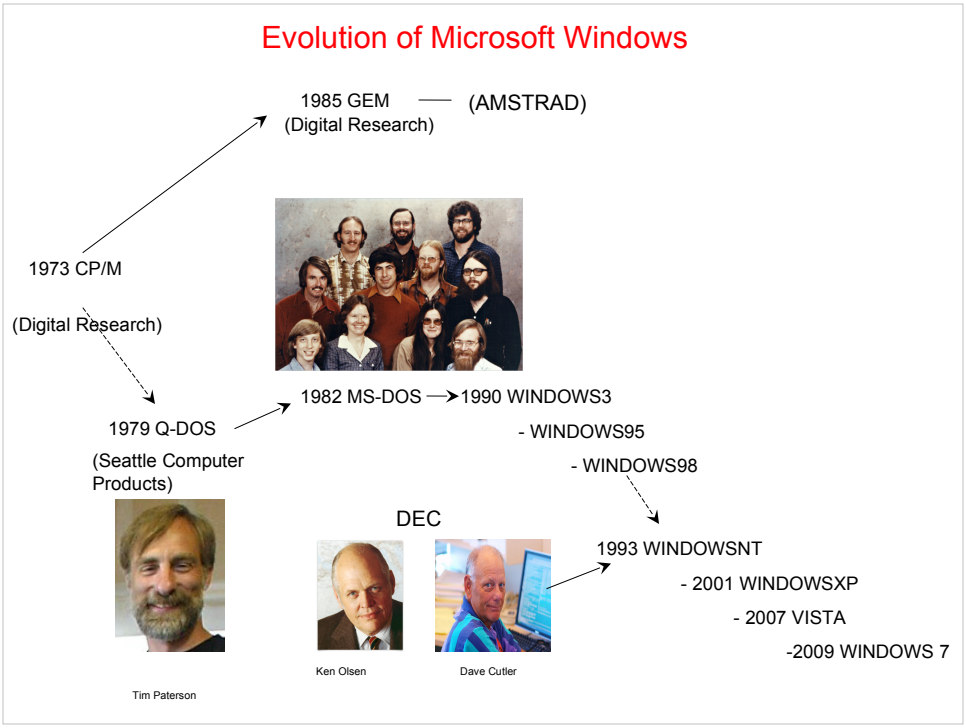
There’s an old story about the person who wished his computer were as easy to use as his telephone. That wish has come true, since we now find it just as difficult to use our phones”

Let's now look in more detail at the evolution of Windows. But just note the inclusion of the Xerox Alto on this slide. That’s where all the Windows style systems began.



In the early days unix and other operating systems could only run on large systems, and microprocessors were arriving - and CP/M Control Program for Microprocessors was developed in 1973 by Digital Research. This was very successful for some years. Moving on to 1982, IBM were looking for an operating system for the forthcoming IBM PC and talks with Digital Research began - and then collapsed. IBM then gave the contract to the fast-talking Microsoft. Here they are – Bill Gates right there.

Microsoft then had to conjure up an operating system from somewhere - and bought one from Seattle Computer Products for some $50k. They didn't mention the IBM connection, and later Seattle Products sued them, and got a $million settlement.



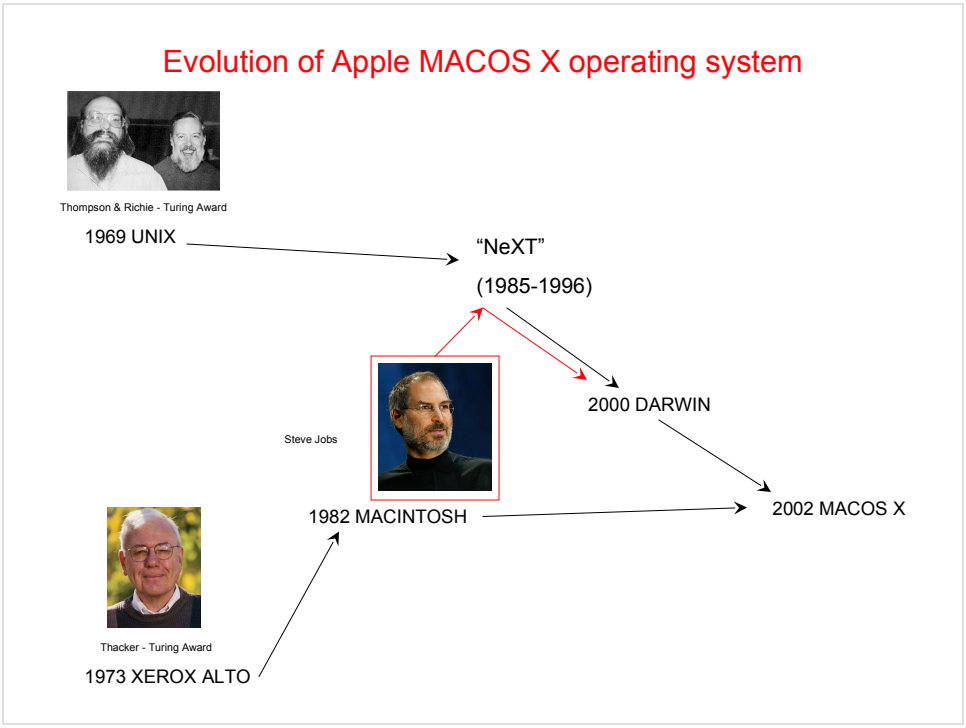
So the IBM PC arrived. The IBM PC world stayed with MS-DOS for a long time. But elsewhere there were glimpses of the future. Right back in 1973, Chuck Thacker, who eventually received a Turing Award, inspired the Xerox Alto, which had the first GUI - Windows-like interface. But it was too far ahead of its time, and we had to wait until the 1980s for the first Apple Macintosh, then the excellent Acorn Archimedes, the fairly horrid GEM system from Digital Research, and last but not least, Microsoft Windows. For 10 years, the Microsoft Windows systems were based on DOS and were very flaky! DOS was eventually abandoned with the arrival of Windows XP and its successors.

Just as with DOS, Microsoft had to bring in talent from outside in creating Windows NT. That talent came from Dave Cutler and his software team from DEC. DEC had been extraordinarily successful in the earlier days of minicomputers, but they were swept away when PCs arrived. But DEC had great operating systems, created by Cutler; so he and they were the foundation for the new generations of Windows.

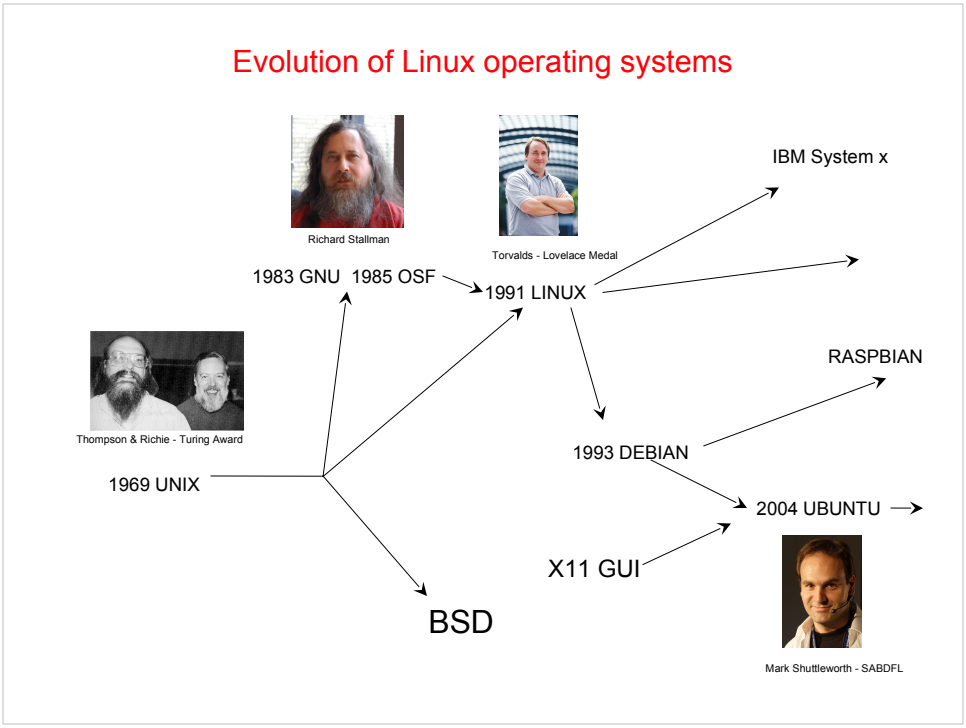
Note Ken Olsen – founder of DEC. He died just last week.

"In a world without walls, who needs Gates and Windows ?"

Marketing slogan! -> other systems

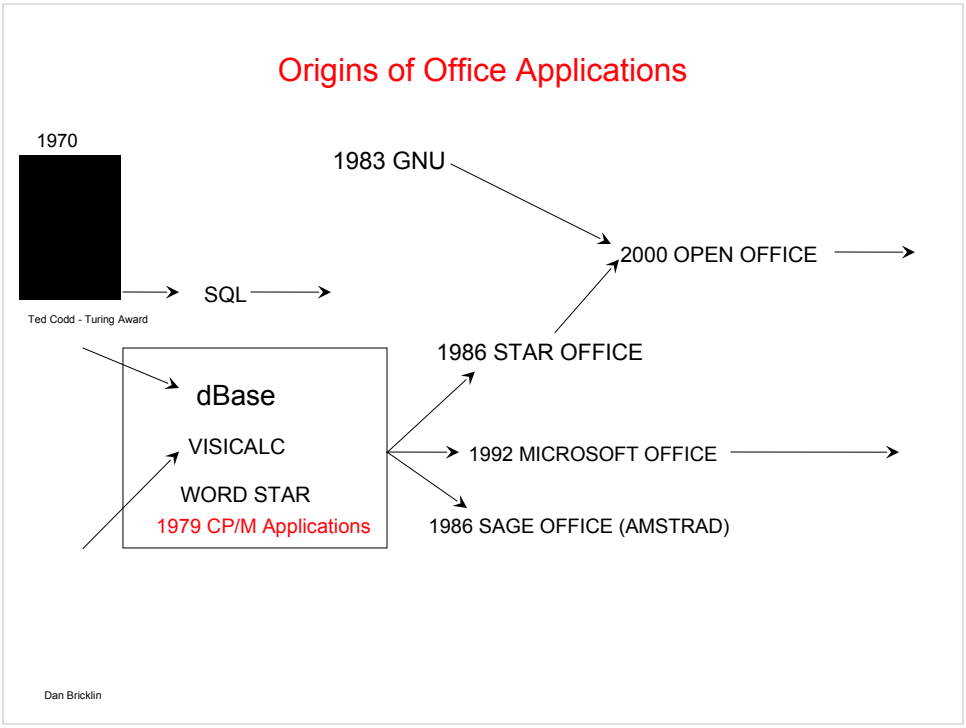


First Apple. They were the first to pick up Thacker's ideas, with the original Macintosh. But it was still too early, and Apple were in the doldrums in the late 1980s, failing to capitalise on the original Macintosh. So Steve Jobs was pushed out in 1985. He formed a company called NeXT with ambitious plans. By 1996 he had a Unix-based operating system called Darwin. He then rejoined the still-struggling Apple and used Darwin to create the MacOSX operating system which has been a major ingredient of Apple's success.

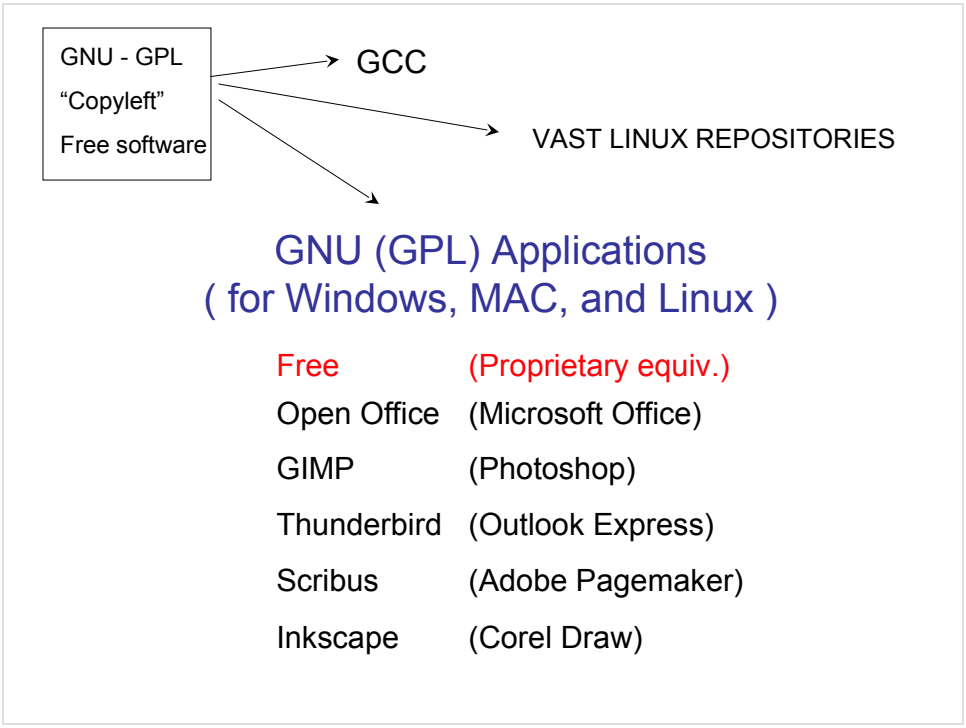


Lets go back to the Unix story in a bit more detail. Unix started off as a proprietary, paid-for system. But in 1983 Richard Stallman came on the scene, crusading against software copyrights. He started GNU, and then the open-software foundation. His original objective was to create a free GNU operating system - Unux-like but not Unix. That has sort-of happened, but not quite as he envisaged. While the GNU people were struggling, Torvalds came along with Linux in 1991. A number of different groups took up GNU/Linux. One was Debian which led to Ubuntu.
Prior to this, Linux was, with some justification, regarded as for geeks. It had been held back by a certain amount of turmoil in the development of a windows-style interface. But by 2004, X11 was very firmly established. Ubuntu has been driven forward by Mark Shuttleworth (SABDFL) and is now more or less on a par with the proprietary systems.

That's all I'm saying about operating systems. Now to Applications - far too big a subject to be covered comprehensively. So let's just look at Office Applications.

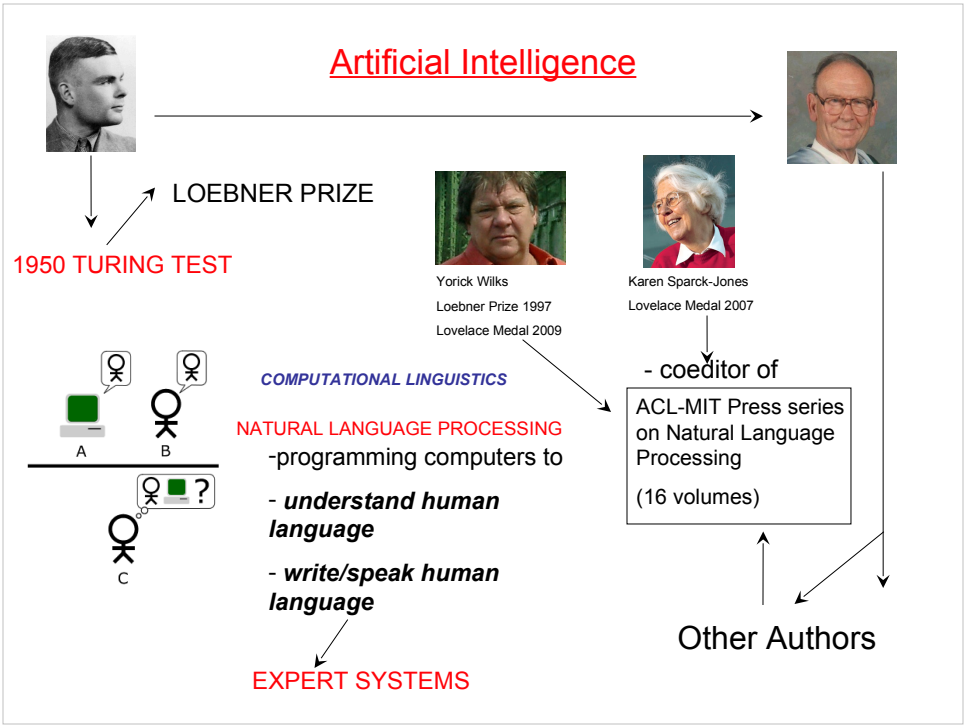


Microsoft Office has a very dominant position - such that it is fairly standard on Macs as well as PCs.



But in the last few years the position has been reached where all the most popular kinds of applications are covered exceedingly well by free versions. And furthermore, there is a wealth of free software available on Linux, and very easily installed, especially in Ubuntu. In general this is also available on Macs because they are essentially Unix systems.

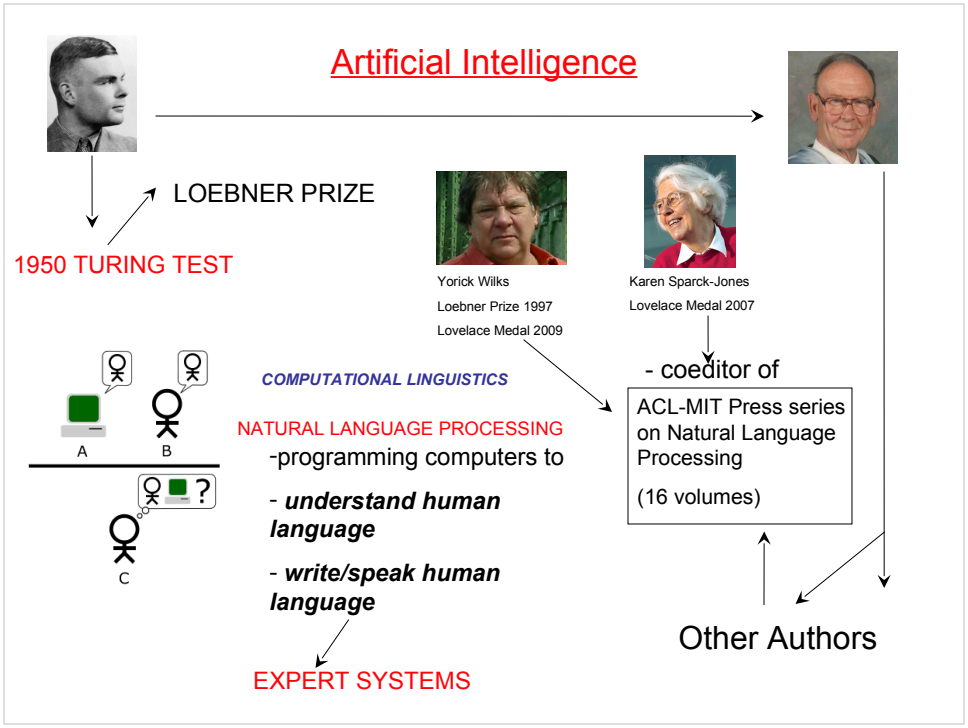
Let's just refresh our memory about our people who should be famous. Most of the ones I've talked about gained their awards from work on languages and operating systems, and those are probably not the areas for the awards of the future. So lets look at Artificial Intelligence and closely related fields.



It all starts with Turing at Bletchley Park. Talking to Turing inspired the 20-year-old Donald Michie to enter the field. Turing himself proposed the "Turing Test" for machine-intelligence, and since 1990 there has been an annual competition, the Loebner Prize, based on restrictedversions of the Turing Test. One winner of the Loebner Prize was Yorick Wilks of Sheffield University, who went on to win a Lovelace Medal, two years after Karen Sparck Jones

Donald Michie went on to establish and direct the Edinburgh University Department of Machine Intelligence, in 1965, and Edinburgh has remained at the forefront of such work.

Important areas within Artificial Intelligence are Computational Linguistics, Natural Language Processing, and Expert Systems. We can relate aspects of NLP to the Turing Test.



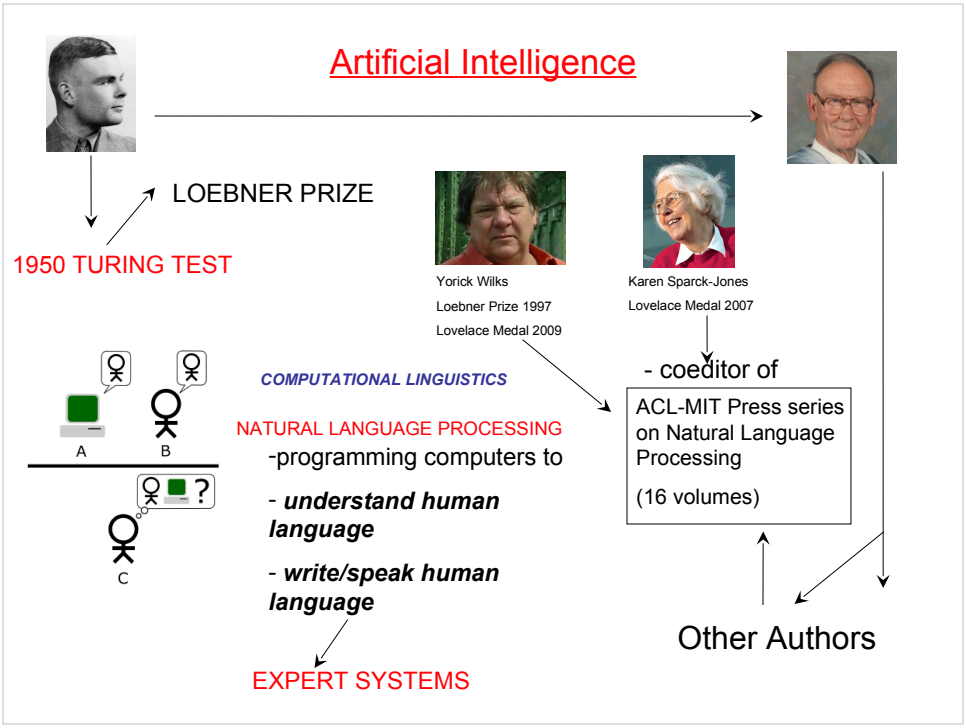
First there's getting computers to understand human language sufficiently well that they can respond sensibly. It is very difficult to write software which will reliably analyse human language, because of all the ambiguities in a language like English, ambiguities which our brains resolve from contextual clues verbal stress, and common sense. For example,

"Following Mary, I saw her duck-under the tree" means one thing; "Following Mary, I saw her duck under the tree" means another!

And sentences like "The British left waffles on the Falklands" could easily be misunderstood.

Next, there's the ability to write or speak human language. Back to that in a moment.

All this comes under NLP



At Cambridge, Karen Sparck Jones led work on Natural Language Processing. And she was the co-editor of a 16-volume series on the subject, each written by an expert or experts. Our Alison wrote this one. Robert Dale, who she knew well at Edinburgh, wrote another and Alan Black was co-author of another. And Yorick Wilks, the Lovelace prize-winner was another contributor.

Expert Systems are an end-product for this kind of work. For example, in the medical field, to take just one example, human specialist experts cannot be everywhere. Can one encapsulate their knowledge and diagnostic skills, so that it is available to practitioners everywhere. There's tremendous interest and there has been some success.

It's a fascinating field. For anyone wanting to read further on Artificial Intelligence, there's another of our Alison's books – a very readable one, which is also available in German and Hungarian.

Now just a little now on Alan Black's work, which relates to the second aspect of NLP.

Alan Black

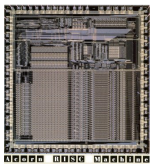
-- “Festival” Speech Synthesis System (Edinburgh)

-- Carnegie-Mellon

-- Founder of Cepstral

We've all heard the Dalek-like voice of Stephen Hawking. And we've heard the realistic voices from gadgets such as a car satnav. It's easy with the satnav, because the very small number of alternative phrases can stored and replayed. It's far far more difficult to take any piece of text and convert it to realistic speech. But there's been huge progress. As Chris knows, his Kindle will speak text.
Here’s a little demonstration of Alan Black’s system.
I’ve two more people to talk about – nothing to do with AI.

Sophie Wilson



1981 Created BBC BASIC

(still in widespread use)

1983 Designed the ARM Instruction Set; emulated the ARM1 on a BBC Micro prior to the making of a chip which worked first time.

(leading to the huge success of ARM processors)

"Sophie nee Roger Wilson, the one-woman once-man whirlwind"

(now "Chief Architect", Broadcom DSL)

There's Sophie Wilson born Roger

Roger/Sophie Wilson and Steve Furber were instrumental in gaining the BBC micro contract for Acorn. They are said to have worked for 5 days and nights turning a paper idea into a computer, the Acorn Proton, which could be demonstrated to the BBC. Then it was Sophie Wilson who created the BBC Basic interpreter. Next came her work on the ARM prccessor. This is what Herman Hauser said, looking back on the early work.

"When we decided to do a microprocessor on our own, I gave Steve Furber and Sophie Wilson two things which National, Intel and Motorola had never given their design teams: the first was no money; the second was no people. The only way they could do it was to keep it really simple."

Of course he was making a virtue out of a necessity. But that simplicity is why ARM processors have triumphed even though Acorn itself was to fail.

Sophie Wilson's part was to design the Instruction Set on which all software is based.

She's now at Broadcom - she's helped to make them hugely successful with the processors which are used in broadband routers. See share prices

James Ellis

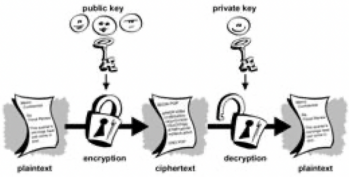
1970 to 1973 James Ellis devised "Public Key Cryptography", which was then secretly developed by Cocks and Williamson at GCHQ.

1977 "re-invented" by Rivest, Shamir and Adleman and publicly described.

1997 GCHQ reveals the work, one month after Ellis died.

2002 Turing Award to R, S, and A.

2010 IEEE 100th Milestone Award to Ellis, Cocks and Williamson.



Next there is James Ellis. I knew him well from the early 1970s when we were cooperating on some work; but I was not then aware of his work on public-key-cryptography. So how's this related to computers and software. Well, the security of information passed over the internet – or by any other communication system, depends on cryptographic protection, and that protection is provided by clever mathematics implemented in complex software. James Ellis had the original idea. A secret is protected by a pair of keys. The so-called "public key" will close the lock, protecting the secret, but it will not reopen it. Only the "private key" will do that. In practice, the two keys are two huge, unrelated, different prime numbers.

From Ada ……to Zuse

…… and onward

- the evolution of computer software