UNLOCKING
CONSCIOUSNESS

BRIAN MIND FORUM

Appendix 050

Implications of Computing
Next Generation of Software
Software Design Skills

Designing ever more intelligent systems and robots,

Computer Hardware.

Von Neumann's design and transistors have powered the computing revolution with power doubling consistently every year for forty years or more – known as Moore's law. In the 1980s an attempt was made to design circuits that could adapt – an attempt to copy Hebb's law that neurons that fire together wire together. However, 'field programmable gate arrays' as they were called were not a success. Manchester University's Spinnaker project is building a silicon brain with a million ARM chips. Lausanne University is leading a European initiative with Э I billion. The Human Brain project has a number of initiatives to simulate the brain. One promising project is to build multiple layers of simulated neural networks to facilitate 'learning systems', imitating the concept of engrams, neurules and memes.

Other initiatives have been to build analogue computers, sometimes called neuromorphic engineering, or more colloquially 'fuzzy logic'. These ideas were given a new lease of life when Hewlett Packard announced the Memristor in 2008. One problem with transistors is that when the power is switched off all data is lost. Memory resistors, or memristors hold their charge, or orientation – their data. Developments of the memristor technology in Germany take these components closer to the function of synapses.

The US Defence Advanced Research Projects Agency (DARPA) has invested $200m in Neuromorphic computing and IBM has announced a True North chip which simulates a million neurons and 256 programmable switches. Another US initiative is the Human Connectivity Project designed to plot which parts of the brain primarily process various functions. Taken together these advances will help us learn more about our own brains.

As transistor components get smaller their behaviour ceases to be predictable and becomes 'probabilistic' as quantum physics predicts. This seemed to produce a limitation on miniaturisation, but in the 1980s David Deutsch in Oxford suggested turning this limitation to advantage and so was born the idea of a quantum computer. As a result of the Quantum Computing in Europe Pathfinder Project, managed by the British Computer Society and the Real Time Club, the European Union allocated 30m Ecus to quantum research in 2002. Early prototypes are beginning to be tested. Quantum computers can be massively parallel. Instead of being binary they can be in many states simultaneously. Remarkably, all the entries in a quantum database however large, can be searched simultaneously. In these days of ever larger quantities of data this facility when fully developed could be a game changer.

Stanford University has successfully built a computer powered by DNA, but so far it has proved difficult to manipulate.

There are many ideas in the pipeline to build imaginative new systems using the effects of superconductivity, but all depend on freezing the processors and so none are practical yet.

All these initiatives strive to use our brains to design faster, better computers.

Future of Software

There is a major opportunity opening up to design a new generation of software that is orders of magnitude more sophisticated imitate our brains. The great drive since the famous meeting in Dartmouth in 1956 has been to design intelligence into software, hence the term 'artificial', or machine intelligence. As a result of some stunning successes: playing chess, winning quizzes and learning to play 'go', much research is being invested to build even better systems. The problem, as everyone agrees, is that these programs, brilliant as they are at the one specific task they were designed to solve, are completely unable to do anything else. It is like a child prodigy that can multiply any numbers together instantly, but cannot do anything else. Cannot read or write, find the way to school, cannot dress or even eat, an idiot savant!

Together with our widening understanding of human intelligence and our knowledge of how software is designed, this presents us with an opportunity. All application programs operate on the platform of an operating system which manages the hardware. All tasks we learn grow circuits which make use of the autonomic functions of the central nervous system. We have explored the similarity of the learned skills that are partly 'automatic' and partly conscious; and the programs that are partly compiled (fixed) and partly interpreted (tailored to each application).

An Operating Systems for the age of Intelligence

There is an opportunity to develop the next generation of operating systems, or perhaps a number of generations ahead to imitate the sophistication and flexibility of the human autonomic functions of the central nervous system and develop a system that does far more than manage the computer's capabilities but to hold a massive library of applications – in the millions initially – and select and operate these on demand. Many of these 'applications' would be sub routine applications modules and the meta operating system would be designed to select modules to create innovative new applications to meet novel situations.

A Programming Languages for the age of Intelligence

But this is one half of the opportunity.

Programming languages were designed to help programmers code programs, thus they are inevitably stylised and designed to suit the temperament and skill sets of programmers. Now is the time to start inventing systems design languages that suit the temperament and skills of a much wider community who can specify what they need accurately. Given the advance in language engineering – speech recognition – these new languages could be interactive – a conversation – a discussion, and be similar to learning another language like French or Spanish, or more accurately Latin which does to longer migrate. All teenagers would be encouraged to learn '*computerese*' much as well-educated teenagers learn at least one foreign language in good schools today. Executives will be able to align their systems to take advantage of fast changing business trends and opportunities by discussing how to accommodate these new needs directly with their systems much as today they call in their IT specialist.

In the early days of program languages in the 1960's some research was done to directly convert flow charts of systems into coding, a sort of visual programming system, but computers were not powerful enough and processing of graphics was in its infancy, however the principle was probably correct.

Clearly these two leaps forward, to design a new generation of software: a comprehensive operating system, and a new super level language would be built to operate in unison.

There is a major opportunity for an entrepreneurial nation to design a sophisticated general intelligent operating system providing a standard platform, designed specifically to operate with a standard specification language to facilitate programming intelligent applications.

Thus, computers could begin to emulate how everyone's brain can learn ever more useful information that can be shared within the community. Rather like Microsoft provided the standard interface – MSDOS – in the 1980s that unleashed the power of computing and made Microsoft the richer corporation on earth.

Interfaces for the age of intelligence

Communication is always high on any list of advances. The computing revolution only got into top gear when processors could be accessed directly from keyboards, and when the contents of the processor could be displayed on a screen. One of the mild disappointments of computing is the time it is taking to design accurate speech recognition systems. Most forecasters expected this problem would be cracked much sooner. Systems that were 60% accurate were operational in the 1960's. Research into digital sound and the development of hidden Markov models in the 1990s enabled systems to reach 95% plus accuracy. The British Computer Society and the Oxford University Press published the first magazine to be typeset from the voice of the Editor in December 1994. Programs attempted to deconstruct syllables and phonemes, and further development stalled. Even 99% accuracy is not enough. Present systems follow a quite different algorithm. Vast volumes of speech are analysed to identify patterns that fit the words and phrases being recognised, with good results and 99+% accuracy. Nearly there.

Prosthetic Links

With this experience to work with, we can start to invent and design the best way to design prosthetic interfaces. Much work has been done to connect prosthetic limbs directly to the nervous system in such a way that the limb can be activated by thought, and signals from the limb can be picked up by the nervous system. We have a long way to go to transfer thought patterns, even speech directly from a prosthetic memory system to the brain, but we know it can be done. Now we have to make it efficient.

Now is the time to invest serious funding into the possible ways of building interfaces and the equipment that we may want to communicate with.

Corporations have built powerful businesses out of establishing a standard interface to some new technology – Hewlett Packard's laser printer interface is a good example. The whole community benefits as this standard provides a solid base for many people to design new systems.

Skills of the Software community

There are three quite distinct groups of people in the software industry with quite distinct and difference tasks and quite different skill sets and abilities.

There are the analysts who ferret out how tasks to be computerised are executed now, and define the problems to be solved.

There are the systems designers who work out how these tasks can be achieved and layout the strategy and specifications, hardware, software, interfaces and psychology of the system that can solve the definitions of the problems.

Then there are the coders who convert the specifications and parameters into products.

They all tend to have very different personalities and characteristics.

Systems Analysts.

At first sight this would seem to be fairly easy task – just observe and describe how the task is done now. This attitude is the usual cause of all the biggest and most expensive disasters. The first problem is to find people who know in the minutest detail how the present system operates.

There are few applications easier than banking, but it took twenty years to build the earliest systems. The problems highlight the difficulties encountered in many other applications. In the 1940's the Banking industry's manual systems of maintaining the ledgers holding customer's accounts were being replaced by accounting machines. The senior staff did not know in detail how these automated systems worked. The very junior staff who operated the machines just new the procedures of their small piece of the jig saw. Staff in the companies who manufactured the machines knew how the whole system worked but it did not occur to anyone to ask them. Anyway a manual automated system was very different to a computerised system. For instance, when A sends B a cheque, B pays it in to his branch, who credits his account and along with all the other cheques that day sends them to, and debits head office. HQ lists A's cheque with all the others to go to A's bank. They go by the central clearing house where all the cheques are relisted on the way to A's head Office then to A's branch and finally A's account is debited. Every time clerks listed every cheque at every stage there was a risk they would make a mistake so the cheques were always listed twice by different clerks. It took three days.

Computers work the other way. Once the value was correctly entered it would always to transferred accurately. The computer problem was to make sure each cheque went to the right place, never a problem for clerks. The problem was eventually solved by encoding the codes of the two banks and the sum in machine readable form along the bottom of each cheque. There have been no clerks in the clearing system since that invention. Nowadays a majority of transfers are on line. For decades the banks kept the three days, and the balances involved, quite unnecessarily.

Innovative technology needs new systems

New equipment often requires new systems. An apocryphal tale makes the point still relevant today.

A tea planter visited his estates and was shocked to find his staff carrying heavy baskets on their heads. Back in England he despatched a consignment of wheel barrows. On his next visit he was shocked to find the baskets still in use. The supervisor said the staff much appreciated his concern for them, but the wooden wheelbarrows hurt their heads.

This problem has happened in a majority of cases where new equipment has been installed. Sometimes the installation of both new equipment and new systems has proved too much for staff and it has been necessary to design a half-way house allowing the operatives to get used first to the new equipment and later the new system.

Innovative technology in teaching

Relevant to the educational world today, the Massive Open On-line Courses, or Moocs, effectively uses the technology of cameras and transmission technology to deliver lessons to every computer, merely replicating the classroom. Much to the relief, and in many cases delight of the teaching profession that has not worked very well, because the classroom experience has not always travelled well.

It is too soon for the Luddite tendency to celebrate. The second and later generations of distance learning systems will develop quite different learning experiences, probably learning from the film and animation industries, or something quite novel.

The more general problem is that all systems are usually much more complex than we realise. All people using any system use their common sense in a multitude of small ways. Computer systems have no common sense whatever. Instruct them by accident to step left instead of right over a cliff and they will obey instantly without demur. There are many systems in broken heaps at the bottom of many cliffs.

The psychology of the way any system operates in often extremely important and needs to be given a high priority in the design of new systems. Another problem, routinely ignored, especially in big nationwide systems is that the speed of hardware development is so fast and the implementation of new big systems inevitably so much slower, by the time a new system is ready to be rolled out the hardware is out of date. Does the Minister install an antiquated system to much ribaldry, or ask the Treasury for massive additional new funding to update the system, and be embarrassingly late?

Systems Designers

The design of new systems is one of the most exciting tasks. It closely resembles the architecture of new buildings. Both architects and systems designers start with a clean sheet and an outline of what is wanted. Buildings from the Acropolis to the Guggenheim museum and the Sydney Opera House enthral, encourage and illuminate the lives of our communities. Living immersed in the architecture of Oxford and Cambridge educates its alumni almost as much as the lectures of the professors.

Architects see the result of their work. Systems designers never see their programs, only the output. Systems design, and to a considerable extent coding or programming is almost completely abstract. Architects can draw pictures, but the designer can only produce a definition, a specification, lists of inputs and outputs and structures of memory systems, processes and interfaces. Yet a computer system can be an artefact of great beauty.

A great system needs to fulfil the functions for which it is being produced, but it must also be resilient in many ways: power supply, communications failures, external hacking, operator mistakes on purpose and by accident. Systems need to be intuitively obvious to operate, especially if used by staff, or particularly clients, customers and the community randomly, or only infrequently. Outputs need to display specific information that is easy to find and easy to assimilate. In particular, systems need to be transparent and build the confidence of users. It is doubtful if a perfect system has been designed yet, so it must be possible to be able to work out how to bend the system to solve rare and unusual requirements. Testing new systems is crucial. Very rarely do design errors slip through, however the Achilles heel is nearly always when two or more problems interact, and where nobody foresaw this combination. Major systems need to be designed in discreet sections so that each segment can be updated to cope with evolving needs, new ideas, or new hardware. Thus, the whole application can be continuously updated, segment by segment independently, causing minimal disruption. Discarding and replacing whole systems at one moment is a hostage to fortune. The design of the graphical user interfaces is usually a crucial element of most systems. Fitting the necessary information into the limited capacity of a screen so that it is clear, concise and easy to assimilate is a work of art in itself.

Last but not least, systems should be designed in modules so that programmers can code them separately to facilitate testing and later updates as mentioned above.

And the whole must be held in the designer's head, from the outline framework, right down to the minutest detail.

Systems designers take nothing for granted and need to have an open mind to every possibility and a willingness to question everything. No one has yet come up with any test of aptitude. A vibrant imagination and lots of self-confidence are very useful. Most are extravert, risk takers, curious, explorers and leaders. The facility to simplify the task in hand, focus the essence of an objective, description, definition, or specification into a sound-bite is of rare value.

There are plenty of good designers with double firsts in Maths and equally many who have no GCSE's, GCE's, or 'O' levels, or whatever they have been called from time to time. Maths is essential for designing mathematical systems but for no one else. Few business systems use more than the two times table. What are the best subjects to learn? A bit of everything helps. The history of ideas is illuminating. How does one learn to think? Orienteering and the humanities are a help, but the best of all groundwork is philosophy.

Arguably computer systems design is increasingly the most exciting and creative occupation in the whole community.

Coding and Screen Design.

Writing computer programs, coding, is a completely different task. Almost the opposite. Coding needs long periods of intense concentration at a level few can achieve until they are thirty or more. One needs to hold in one's head a wealth of detail, which applies to and influences every group of code or sub routine as it is written. Coders can get very irritated if they are interrupted. Many programmers are introvert. Again, they come from every background, but their most valuable attributes are a ferocious capacity for rigorous logic, demanding precision, and an almost obsessional attention to detail. There are no known tests for aptitude. Writing a piece of program is the only test of ability that is consistently accurate.

Interestingly, some exceptionally good coders are not particularly good at designing screen layouts, which requires yet another group of skills. Systems designers are not much better. Both tend to try and cram as much as possible into the limited space. Artists can put a lot of information onto a screen, yet it appears to be easy to read and the information the user is looking for seems to be clearly visible, and almost jumps out at the user.

Like the architectural and entrepreneurial worlds, dyslexics often make excellent systems designers, while some on the autistic spectrums make good programmers.

All Software professionals have one attribute in common: meticulous attention to intricate detail. In most professions and life generally getting more than half the decisions right is often a good record. In computing a relatively insignificant error may mean the whole system is useless. Often when a system is 95% built the whole installation is often only half way to completion.

Commissioning & Managing Installations

From the decision to install a computer system, or update an existing system the choices about the strategy for implementing that decision are crucial. Experience has shown that the appointment of a leader to own the project from start to finish in very important. Similarly, it is essential that leader and group thoroughly understand the objectives and functions of the organisation the computer system is to serve.

A lot can be learned from the experience of the government. Not one single major national system has been installed to specification, on time, and within budget. The principle reason is that governments have tried to make computing contracts fit into tried and tested civil service procedures rather than make civil service systems fit the requirements of computing.

Team leaders are swopped around regularly, and civil servants are not selected or trained to have any expertise in technology. Consultants are employed at every stage, but not for advice. They are employed as principles. Initially to write the commissioning documents, as the leader is not expected to have any particular expertise in the detailed workings of his government department or agency. Further consultants write the specifications, then negotiate the implementation contracts, and so on. The most difficult example was over the national medical records debacle where £12billion had to be written off in 2010. The NHS does not have a records system to this day. In the seventeenth year of the twenty first century a local hospital consultant has to type the results of a patient's test and post it by snail mail to his local surgery, where the letter will be scanned into their system some days later. No one has calculated the cost of this system. Surprisingly, there was no appetite to learn from this experience, as subsequent installations have not improved. The government are not alone. The stock exchange got into a similar situation in the 1980's and had to be bailed out by the Bank of England.

Coordination of the various aspects of the Computing Revolution

Computing is changing almost every aspect of life. People and communities can accommodate change if it is reasonably slow in arriving. Computing is not only hurtling towards us, but its effects are magnified by the multiple changes that exacerbate each individual aspect.

The whole concept of work, the basis of life for five thousand years is in flux. The economic principles that drove the industrial revolution are being reversed, huge ethical and moral problems confront us. The principles of education that we are familiar with are suddenly open to question. The whole financial structure of nations is moving into uncharted territory of which we have no experience from which to learn. Some people are even suggesting we might be able to operate without money. We have in prospect robotic partners, and the components to grow and developed cyborgs, and learn to edit our basic heredity. The political settlement we pride ourselves upon seems to be suddenly vulnerable. Yet no one in government in the widest sense seems to be aware, nor seems to wish to know about this multiple tsunami that is well on the way from the horizon.

There is a vague feeling that technological changes sort themselves out. This complacency is very dangerous.

It would seem wise for every department of state to set up departments to identify their problems and opportunities and start planning solutions. They need to start selecting, recruiting and training staff with the necessary skills, then experiment with pilot schemes, build up expertise and implement new ideas, different concepts and thinking. The biggest problem may be the knock-on effect of multiple changes occurring simultaneously. It would seem wise to coordinating these multiple changes.

The United Kingdom has led the world in almost every field of endeavour. It is our responsibility to start to reach out to meet the future once again.

2018 // Book Final // Appendices NEW // 050 Next Software 17 10 Oct