UNLOCKING
CONSCIOUSNESS

BRIAN MIND FORUM

Appendix 006

Exploring the

SCIENCE of SOFTWARE

Discovery of Software
Plus six appendices

When Crick and Watson published their ground-breaking discovery of the structure of the DNA molecule, the world gained more than a blueprint for the genetic transference of physical traits in living things. It gained an insight into a code by which life itself is constructed. In addition, it revealed the process whereby biological information is processed in the natural world and so opened the door to nothing less than a new branch of science: far wider in its scope than just biogenetics

Around the same time that Crick and Watson were working on genomics, in the early 1950's, another code was being developed by engineers seeking to make electronic computing devices perform an ever-widening range of tasks for mankind. The physical Hardware had a very limited set of instructions, but, led by Alan Turing, it had become apparent that these basic instructions could be repeatedly carried out in increasingly complex sequences and patterns and so be 'programmed', to carry out relatively complex functions. These functions could then be assembled into patterns that could carry out an apparently unlimited variety of applications. The design of these programs was described as 'Software' to differentiate it from the 'Hardware' label attached to the manufacture of the integrated circuits, memory systems and the array of input and output equipment. Two, quite distinct industries were born.

We are beginning to realise that there are many systems in nature that operate using the same fundamental architecture as both DNA and computing, in particular there is the controlling hub of the central nervous system. The Brain is made up of billions of neurons, connected through their dendrites and axons by synapses to all the organs of the body: not dissimilar, in principle, to the integrated circuits of a computer and the genes in our DNA. Flowing over these networks is a continuous mass of electrochemical signals generating electrochemical fields and waves, which transmit input patterns and sequences of signals from the sensory and other organs, and output patterns and sequences of signals to stimulate the muscles, glands and other organs. This mass of electrochemical activity is what we recognise as the Mind.

We are beginning to realise the significance of this dual process that we have invented for our computers. What we have called Software for our computer programs is remarkably similar to the process by which DNA creates human beings and the Brain and Mind work together.

The triumvirate of Professors Daniel Dennett, Richard Hawkins and Matt Ridley have drawn attention to the very useful and accurate analogy of Genes as words or subroutines (*Intuition Pumps* Chapter 36) " The list of words in *David Copperfield* is almost the same as *The Catcher in the Rye* …. The difference is in the order in which these words are strung together". A relatively small number of words –genes – can generate an unlimited variety of books – humans - according to the sequence and pattern in which they occur; or, better then a sentence is the concept of a toolbox of subroutines. As computing and neuroscience converge we can begin to apply the architecture of software design to further understand this line of reasoning the triumvirate has opened up.

Definition of 'Software'.

The immensely powerful common factor is that Software allows a standard piece of Hardware, organic or inorganic, to operate in a multitude of different ways. Software enables a double helix of genomes to grow unique human beings: a collection of integrated circuit boards to beat chess masters: or a network of neurons to learn to solve problems, think and create.

History of Software

The concept of software is not new. In the 18th century the Jacquard weaving looms used an early version of punched cards to weave an unlimited number of patterns from one standard loom. Babbage's difference engine could solve different equations by adjusting a series of metal disks. However, the real breakthrough was Turings's 1936 paper on 'computable numbers' which described how a 'universal machine' could solve a multitude of problem. Computers can store information based on the coding system invented by Morse to transmit information. By 1948 the first 'stored program' computers were operating with one additional twist. Programs could be stored in the same format as all other information. A list of various programs could be stored in the system and a master program could select which program to run in which sequence: the beginning of the 'operating systems' that today control all our computers, much as the autonomic system controls all the functions of the body. The original 'machine code' of early computers was phenomenally difficult to program so programmers designed programs to help programmers write programs: programming languages were born.

Two versions of Software

There is a veritable Tower of Babel of programming languages, but they broadly divide into two groups. There is much to learn from this, because this dual operation is also reflected in the software of biogenetics, the Mind and other software based systems.

Computer Software programming systems are either 'compiled' or 'interpreted'. Systems designers and programmers using a compiled language plan, organise and code their application, then use a 'compiler' program to convert this to the machine code that will run the application. That application can not be directly amended. If the application program needs amending, editing or updating, the original ‘source code’ has to be changed and the whole program re-complied from scratch. Compiled ‘object code’ is very efficient but inflexible.

Alternatively, systems can be designed to incorporate ‘interpreted’ program. Code is included in the system that is only converted into machine code whenever the application is used. Interpreted object code is slower but is very flexible allowing everyone using the system wide latitude to tailor the application to suit their changing needs.

DNA like a Word Processing Application

Let us return to the analogy of ‘genes as words or subroutines’ and study for a moment the architecture of word processing applications.

This is not the place to describe the history of computing. Suffice it to say that early computers did not involve much text. Software and printers could only process capital letters all the same width. Of all the banking systems only Coutts & Co printed any descriptive text narratives on their statements, and that was thanks to a paper presented to the Chairman by a 22 year old clerk in 1958. In the early 1960’s computers were programmed to control ‘hot lead’ linecasting machines which were used to set newspapers, but the problem of inputting text made applications like book production by this process uneconomic. Early screens attached to the first personal computers could only display fixed size and width characters. Then laser printers arrived that could create any shapes from a matrix of dots.

The designers of the first word processing applications were faced with a problem. Programmers could easily insert the control codes to drive printers, but that suggested typists would have to learn programming. The solution was to combine the use of compiled and interpreted programming into one language. We would submit that is very similar to the architecture of the construction of cells from the sequences of DNA.

It works like this. The underlying robust word processing (WP) application is compiled. This program allows the operator to key the text stream, differentiating only between capital and ‘lower case’ letters, plus numerals, punctuation marks and symbols.

Initially the typist could insert into this text stream codes to ‘change to bold face’, change to ‘italic’. ‘Change to a larger typesize’ and so on. In conventional typesetting since Guttenberg this was known as ‘marking up’ text. So the first standard typesetting mark up languages (STML) were designed. [They have evolved today to control the whole internet and word wide web. HTTP is hyper text transfer protocol].

Nowadays nobody inserts codes, we all click on ‘bold’ or ‘italic’ icons and the WP application inserts the codes into the text stream for us. How this then actually works is illuminating.

When we hit the ‘print’ icon, the underlying WP compiled application program starts processing the sequence of text, character by characters, sending a stream of dots to the screen or printer. When this program encounters an STML code it interprets that piece of program to ‘change all subsequent text to bold face’, for instance. The underlying program then resets the typeface table to output the stream of dots that will display or print, thicker, blacker ‘bold’ characters, at the same time adjusting the width of all subsequent ‘bolder’ character so the program can continue to fit the text onto the screen or page.

This software architecture makes every keyboard operator an expert typesetter. Every document ever printer on a computer has just 256 characters. It is vanishingly unlikely that any two documents are the same!

Polymerase

This process has an uncanny similarity the way polymerase travels along a length of the DNA of a gene to produce a replica. Along the way some codons of base pairs signal the structure of enzymes, amino acids all the way to the proteins of cells, while others indicate a wide variety of sequencing, patterning and 'on-off' instructions that we are gradually beginning to understand.

We are becoming aware that there are a number of other biological processes that are of a similar architecture – that differentiate between information and the application of that information. There is an interesting twist. If programmers look behind the screen at the text stream in the memory of the processor, they will just see one long succession of codes. Only a very few software designers would be able to differentiate between the codes for the characters that eventually appear on the screen or printer and the mass of codes that determine the shape, size and position of those characters. Programs that process formulae can have as many as a hundred control codes for every one that appears on the screen or printer! To have any hope of interpreting these code streams we have to use another word processor to translate the eight bit bytes into characters we recognise because, actually, all that exists in the memory of the processor is just a stream of 'I's and '0's!

DNA and its subsidiary RNA are streams of the four base pairs, guanine, cytosine thymine and adenine.

Deoxyribo Nucleic Acid (DNA)

There is increasing evidence that the biogenetic system that uses DNA to create a living being works in much the same dichotic way. If we use computing language to describe the process, we can say that the stream of codons of base pairs form genes which are 'compiled' into enzymes, amino acids and in due course protein stem cells and finally into all the cells that make one living organism. But there is another process at play that determines whether genes are switched on or off, which sequences are selected, how they are expressed and a variety of other epigenetic functions that we are just beginning to learn to understand. These second order codes appear to be interpreted individually for every person, thus identical twins, who may have identical genes, can develop to have quite different characters. Transgenerational epigenetics is a very contentious subject. However, it seems increasingly apparent that DNA has double level processing Software: the compiled main stream that determines the overall organism, and the interpreted second stream that tailors that organism to suit the prevailing circumstances.

The Brain

There is a similar double dichotomy in the Brain. Our present level of knowledge suggests that the foetus grows a mass of some two billion neurons which connects up all the organs of the body to the brain – the Hardware. In particular the foetus links the ears and vocal chords together through the brain. Over this network flow streams of electrochemical signals - Software. The hardware in the ears convert sound waves into signal patterns and transmits these along the dendrites to the neural networks in the brain. Equally these networks can send streams of signals along their axons to the muscles of the lungs, vocal chords, mouth, tongue and lips which enables new born babies to make sounds. This is remarkably similar to a compiled system. Curiosity, imitation, and endless repetition then interprets particular sounds that we learn to hear and speak as words, tailored to the circumstances and environment of each individual baby. We learn to speak the language of our parents and peerhood.

The Brain then does something no computer can do, or is likely to be able to do in the foreseeable future. Electrochemical activity over the neurons - Software - causes those neurons that are active to form links "neurons that fire together wire together". These new links grow into new neural structures forming permanent memory. This is the basis of learning. Thus, the Software Mind, both compiled and interpreted, grows new structures in the Hardware Brain.

Software Primer

Interestingly there is a system that we have invented that makes this complex concept much easier to comprehend: Language. Somewhere around one hundred thousand years ago humans began to make noises that everyone could learn to make and hear, that by common agreement described things and actions. Thus, they added to the neural structures that represented the sight, taste, smell and feel of familiar things around them, and learned, ie grew the neural networks, to say and hear the sound of words for things like, trees, or activities like running. Around five thousand years ago we started to invent a way of recording words: writing and more recently, printing.

Letters and words are a very good example of Hardware and the two versions of Software.

We are all familiar with the structure of words. They are made up of some 26 letters (in English) which we can write, print and read. At the same time words are made up of syllables that help us enunciate and hear them.

With just 26 symbols we can express the entire body of human knowledge. With just ten symbols we can express any number, measurement or value. There are some half million words in the Oxford English Dictionary and that number is rising fast as more and more ethnic groups use the language adding their own contributions, and the scientists proliferating new ideas, concepts and inventions. Most people use a subset of about 20,000 words in daily life.

That is the basic Hardware we work with, but just writing out the alphabet or looking at every word on the page of a dictionary tells us vanishingly little. The immense value of language is that in addition to being able to label things and activities (nouns and verbs) we can express complex ideas, concepts and emotions. We can write whole books to describe some scientific theory of our universe, or a whole novel to explain some human emotion. Their meaning is an emergent property of the sequence of the words. The value of this sequence is vastly greater than the sum of their individual words. We can program words to describe pretty much everything, although we are not very good yet at adequately describing qualia. So we can compile a text book or play.

But we can go further. A playwright can insert into his text instructions on how the actor to say the following words – 'jokingly, sneeringly, in a rage, pathetically' and any number of actors could say those lines in a multitude of different ways completely changing their meaning. Orators can use ordinary words in sequences and patterns to rouse nations to war. Thus, we can readily see there is the Hardware of the individual words, the Software of the sequence and patterns in which they are Compiled into book, treatise or speech and finally the way in which that word stream is Interpreted.

Brain Mind

The same trilogy is apparent in the brain. We are learning a great deal about the neuron networks and other physical properties, and increasingly which neuron networks appear to be active where, in the Hardware of our Brain and nervous system, as we go about our daily lives. Similarly, we are learning ever more about the patterns and sequences of electrochemical messages and the fields and waves of electromagnetic activity this signally system generates, which we recognise as our Mind. We are also increasingly aware of effects that the complex flow of hormones has over this whole system.

Our neuron hardware may facilitate sending a program of Software signals to a group of muscles, but our ultimate behaviour will be interpreted by the emotional ambience generated by those hormones. In identical circumstances, we can react quite differently if we are in a rage, or if we are frightened or elated.

Sub Conclusion

Putting aside the compiled / interpreted discussion for a moment, there is another attribute of hardware / software architecture worth noting. The three examples described above, biogenetics, computing and neuroscience have something else in common. The Hardware components, the double helix of the genome; the neuron networks; and the integrated circuits, are used again and again by the Software components, the RNA and polymerase; the electrochemical signal patterns; and the computer programs. Employing these components does not change them, and in principle does not degrade, damaged or wear them out. There is no limit to the times they can be used. We need to enter a word of caution. DNA, Neurons and transistors do degrade, but not generally as a result of their use by programs. Most problems arise from their energy supply: the nutrient path in the case of DNA and neurons, the electricity supply to computers.

Chemistry

We are well aware that many chemical reactions involve catalysts. They bear a remarkable resemblance to the underlying hardware of the genome, the neurons and the semiconductors.

They are used again and again by the other chemicals involved, but largely they are apparently un-effected. However, no catalysts: no chemical reaction!

Biologists have for centuries been fascinated by the way bees swarm, organise their hives, find food and direct their peers to the source, and as a group migrate to a new site. Is there another comparison between the hardware of the individual bees and the software of the communication and control systems.

Fish move in shoals and birds fly in formations. Are we seeing yet more examples of software systems controlling the behaviour of groups like this?

Obita Dicta

A computer driven analysis of the neural architecture of language yields another intriguing, even startling theory.

Scientists have puzzled at the apparent speed at which speech and particularly writing evolved. It is a slightly uncomfortable aspect of Darwin's theory of evolution that it has progressed seemingly by a vast number of chance mutations, one could almost say by an accumulation of mistakes, that fortuitously generated advances of staggering sophistication.

Our current best guess suggests that the beginnings of speech developed around less than a hundred thousand years ago. That is roughly four thousand generations: almost a blink in evolutionary terms. Writing developed around three thousand years ago: not much more than a hundred generations.

Speech requires the most meticulously exquisite manipulations of the muscles of the lungs to provide the exhalation of air, the vocal chords, the mouth, lips and tongue in exactly the right sequence. Hearing requires the near instant decoding of the ephemeral passage of airwaves over the ear drums. If that were not enough these decoded sounds are neurally cross referenced to all the image patterns from the other sensory systems. By the time the Egyptians, Chinese and no doubt others began to draw pictures to represent words, there were numbers of hundreds of these words for the brain to accommodate. When the Phoenicians and Greeks began to develop the alphabet of sounds the way was open to create orders of magnitude more words. The brain had to cope with this huge expansion of its memory capability. Most children have a more than adequate vocabulary by around six years old, yet no one *inherits* one single word.

Roughly half the adult population are literate and have learned to manipulate the muscles of their arms, hands and fingers to a degree of exactitude no other animals can match.

So, we have five skills of remarkable sophistication. To speak, to hear, to write and to read and to store large volumes of information and all the cross references that give them meaning.

Learning the language of one's community involves imitation and endless repetition and practice. These are processes of the software system as we struggle to compile dictionaries, and add the capability to interpret the nuances of expression.

What we have built into the Hardware of the neurons is the propensity to manipulate the muscles to the required degree of perfection, first to control the voice system then the writing system – not the individual letters or words, but the ability to say them, draw them, store them, recall them, manipulate them.

All these skills take place before reproduction so the modifications learned by each generation to achieve this degree of neural muscle precision, and eye and ear coordination could influence the reproduction transmission system. Thus, that part of the genome that influences the expression and fine tuning of the production and manufacture of the appropriate control systems has been continuously updated so that a twenty first century child can learn to speak almost automatically, and write without too big a struggle. Thus, the neural machinery of *learning* to speak, write, read and recall words has evolved at break neck speed. The particular words each individual learns depends of the language of their community, but the machinery to do this is in place. The same general purpose neural machinery is in place to master multiple languages, verbal, written and sign. Steven Pinker coined the phrase 'the language instinct' for this underlying skill. The Science of Software has shown us a path for how this has occurred. It seems we have a clear example to support Lamarck's theory of transgenerational epigenetics.

Appendix 1.

Other examples of Software in biological systems.

Over the last half century we have learned a lot about the physical structure of Deoxyribo Nucleic Acid (DNA), its derivative RNA, and the enzymes and proteins they construct.
However, it is clear we are missing a piece of the jig-saw. All the processes along the chain appear to be influenced by electrochemical patterns that control the construction of stem cells which then morph into individual cells that execute specific functions. Where do these patterns come from, how do they influence the genome, not just to create the original cells with which we are born, but from an identical set in each cell nevertheless reproduce individual specific replacement cells throughout our lives? In particular, polymerase is an exact example of a Turing universal machine. Polymerase is a protein that travels along a gene transcribing the DNA by inputting one base at a time changing its state according to the chemical information algorithm and transcribing the result, and so in the process generating a perfect replica.

We marvel at the behaviour of flocks of birds and shoals of fish as large numbers fly or swim together, changing direction and formation in perfect symmetry. Much work is being done to understand how a hive of bees reconnoitres and moves to a new hive, selects a Queen, forages for honey and describes to other where nectar can be found. Very clearly the abilities of the swarm far exceed the sum of the skills of all the individual bees. People have speculated that this is some form of group intelligence. It seems likely this is built on some expression of software, which provides the process of coordinating the individual bees to behave as one cohesive cooperative hive.

Appendix 2.

Computers and their Software	The Brain
Digital, Binary, Sequential	Analogue, Parallel
Processes artificial, machine information	Processes biological information
Rule based: Ultimately predictable	Probabilistic, Unpredictable Emotional
Driven by a Processor executing an inserted rule based program	Driven by information. The whole body is the processor
Computers execute algorithms and exhibit artificial intelligence	Brains execute inherited or learned responses and exhibit human intelligence
Programmers devise new algorithms	Brains can think, invent

and programs	imagine, prepare and grow new neural solutions
Computers can ‘play’ chess, because there is a fixed field of play, clearly defined rules and an easily identified winner .	Real World Problems vaguely defined problem balance of probabilities contingent solutions.

Appendix 3

The solution to the ‘*Entscheidungsproblem*’ or ‘halting problem*’.*

Alan Turing showed that the ‘halting problem’ is a difficulty for man made ‘machines’; both their physical hardware and their abstract software, because, where they encounter an undecidable proposition, the universal machine will continue processing for ever seeking a solution that does not exist.

However, the ‘halting problem’ is not an obstacle to biological systems like the brain. Because the brain has evolved to cope with uncertainty, it is capable of ignoring past experience and any rules that have been learned, and take the probabilistic path that this is a lost cause, and halt processing.

This is the difference between artificial intelligence and biological intelligence. The former is rule bound and therefore, ultimately predictable, and so for this reason can always be outsmarted by better brains than the brains that designed the programs.

This argument also provides some useful evidence to help understand metaphysical questions about ‘thinking’ and the whole subject of human creativity.

There is some evidence to suggest that Alan Turing had begun to think about mathematical biological software (see Appendix 6), when he was effectively handed a cup of hemlock like Socrates two thousand three hundred before. Thus, two of the greatest brains in history were destroyed by a bigoted establishment in an atmosphere of religious intolerance.

Appendix 4

How new neural links and structures are grown

Professor Donald Hebb coined the expression “neurons that fire together, wire together”. This is accepted as the most likely position. The electrochemical activity of existing neurons *firing together* automatically generates the energy to stimulate the growth of new structures. There are a number of theories describing the physical process of how that electrochemical stimulation is converted into physical form: of *how* the neurons *wire together*. Electrochemical activity generates electromagnetic fields. Where adjacent neurons ‘fire together’ the resulting overlapping electromagnetic fields attract glia cells to form a temporary speculative ‘bridge’ generating a short-term memory. Every time these ‘bridges are activated they are strengthened and in due course mature into long term memory. The electromagnetic fields of distant neurons ‘firing together’ attract roaming messenger molecules which attract glia cells to form bridges. Some glia cells exhibit synaptic attributes, so some, if not all 'glia bridges' may be strings of glia cells each connected together by synapses: in effect 'synaptic bridges'.

Appendix 5.

Professor David Hilbert.

David Hilbert (1862 – 1943) was a German mathematician, and is recognized as one of the most influential and universal mathematicians of the 19th and early 20th centuries. Hilbert put forth a list of 23 unsolved questions at the International Congress of Mathematicians in Paris in 1900. This is generally reckoned the most successful and deeply considered compilation of open problems ever to be produced by an individual mathematician and it led, *inter alia,* to the invention of computing. Hilbert's 10th question asked for an algorithm to decide whether diophantine equations have a solution. In 1928 he expanded this to what was called the 'Decision Problem' or 'Halting Problem' the 'Entscheidungsproblem'.

The full title of Alan Turing's famous paper published in 1936 proposing a 'universal computing machine' is *'On computable numbers with an application to the Entscheidungsproblem'.*

Appendix 6.

Alan Turing designed the first ever program to run on the first ever fully programmable computer – the Baby – in Manchester in early 1948.

Appendix 7.

Pattern formation and mathematical biology

Turing worked from 1952 until his death in 1954 on mathematical biology, specifically morphogenesis. He published one paper on the subject called '*The Chemical Basis of Morphogenesis'* in 1952, putting forth the Turing hypothesis of pattern formation. His central interest in the field was understanding Fibonacci phyllotaxis, the existence of Fibonacci numbers in plant structures. He used reaction diffusion equations which are central to the field of pattern formation. Later papers went unpublished until 1992 when '*Collected Works of A.M. Turing'* was published. His contribution is considered a seminal piece of work in this field.

Appendix 8.

The series of ideas about consciousness developed by Daniel Dennett has similarities to the Turing concept of a *'universal machine'*. Dennett uses different language in '*Consciousness Explained*' and subsequent publications. He argues that our illusions are not what they seem, because they are built of still more illusions which give rise to the concept that consciousness is the product of a '*virtual mach*ine' running in the brain'. Substitute Turing's hierarchies of 'algorithms' for Dennett's hierarchies of 'illusions' both contributing to the operation of an 'abstract processor' and the two concepts are indistinguishable. The use of the word 'machine' to describe an abstract concept is probably coincidence but it has served in both cases to put everyone off the trail, 2018 // Book Final // Appendices NEW // 006 Software