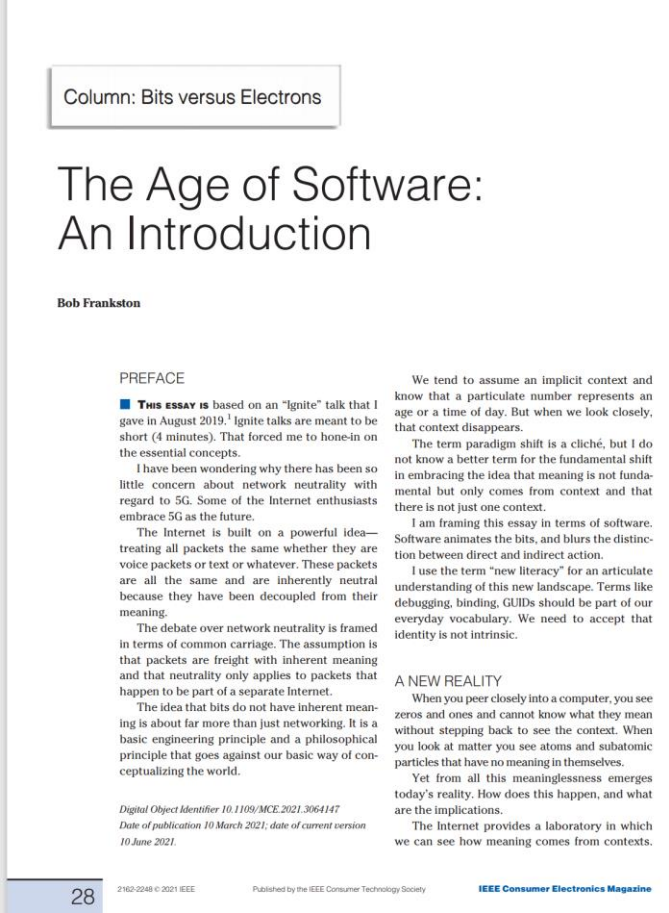
The Age of Software: An Introduction



Column: Bits versus Electrons

The Age of Software: An Introduction

Bob Frankston

PREFACE

This essay is based on an "Ignite" talk that I gave in August 2019.[1] Ignite talks are meant to be short (4 minutes). That forced me to hone-in on the essential concepts.

I have been wondering why there has been so little concern about network neutrality with regard to 5G. Some of the Internet enthusiasts embrace 5G as the future.

The Internet is built on a powerful idea—treating all packets the same whether they are voice packets or text or whatever. These packets are all the same and are inherently neutral because they have been decoupled from their meaning.

The debate over network neutrality is framed in terms of common carriage. The assumption is that packets are freight with inherent meaning and that neutrality only applies to packets that happen to be part of a separate Internet.

The idea that bits do not have inherent meaning is about far more than just networking. It is a basic engineering principle and a philosophical principle that goes against our basic way of conceptualizing the world.

Digital Object Identifier 10.1109/MCE.2021.3064147
Date of publication 10 March 2021; date of current version 10 June 2021.

We tend to assume an implicit context and know that a particulate number represents an age or a time of day. But when we look closely, that context disappears.

The term paradigm shift is a cliché, but I do not know a better term for the fundamental shift in embracing the idea that meaning is not fundamental but only comes from context and that there is not just one context.

I am framing this essay in terms of software. Software animates the bits, and blurs the distinction between direct and indirect action.

I use the term "new literacy" for an articulate understanding of this new landscape. Terms like debugging, binding, GUIDs should be part of our everyday vocabulary. We need to accept that identity is not intrinsic.

A NEW REALITY

When you peer closely into a computer, you see zeros and ones and cannot know what they mean without stepping back to see the context. When you look at matter you see atoms and subatomic particles that have no meaning in themselves.

Yet from all this meaninglessness emerges today's reality. How does this happen, and what are the implications.

The Internet provides a laboratory in which we can see how meaning comes from contexts.

28 2162-2248 © 2021 IEEE Published by the IEEE Consumer Technology Society IEEE Consumer Electronics Magazine

https://doi.org/10.1109/MCE.2021.3064147

Preface

This essay is based on an "Ignite" talk[i] I gave in August 2019. Ignite talks are meant to be short (4 minutes). That forced me to hone-in on the essential concepts.

I've been wondering why there has been so little concern about network neutrality with regard to 5G. Some of the Internet enthusiasts embrace 5G as the future.

The Internet is built on a powerful idea – treating all packets the same whether they are voice packets or text or whatever. These packets are all the same and are inherently neutral because they have been decoupled from their meaning.

The debate over network neutrality is framed in terms of common carriage. The assumption is that packets are freight with inherent meaning and that neutrality only applies to packets that happen to be part of a separate Internet.

The idea that bits don't have inherent meaning is about far more than just networking. It is a basic engineering principle and a philosophical principle that goes against our basic way of conceptualizing the world.

We tend to assume an implicit context and know that a particulate number represents an age or a time of day. But when we look closely, that context disappears.

The term paradigm shift is a cliché, but I don't know a better term for the fundamental shift in embracing the idea that meaning is not fundamental but only comes from context and that there isn't just one context.

I'm framing this essay in terms of software. Software animates the bits, and blurs the distinction between direct and indirect action.

I use the term "new literacy" for an articulate understanding of this new landscape. Terms like debugging, binding, GUIDs should be part of our everyday vocabulary. We need to accept that identity is not intrinsic.

A New Reality

When you peer closely into a computer, you see zeroes and ones and can't know what they mean without stepping back to see the context. When you look at matter you see atoms and subatomic particles that have no meaning in themselves.

Yet from all this meaninglessness emerges today's reality. How does this happen, and what are the implications.

The Internet provides a laboratory in which we can see how meaning comes from contexts. Meaning is not freight carried inside the network. The meaning (as in is 7 a color or a temperature) is only in the interpretation at the endpoints of a connection. This is a fundamental shift from the traditional idea of telecommunication, which treats meaning as valuable cargo that exists at every point along the way.

Understanding this is central to embracing the new reality.

I chose "The Age of Software" as the focus because, as I see it, software has given us something new – a language for talking about concepts that speaks for itself (*res ipsa loquitur*). I can give the instruction "draw a box," and we see a box being drawn as if by magic and, indeed, the idea of automatons that could follow arbitrarily complex rules was magical with mechanical systems giving only a hint of the possibilities.

Yet when we look into the details – the very bits (binary digits) out of which the instructions are built – the magic and meaning disappear. We crafted this magic out of nothing. Where did it come from, and what are the larger implications?

Finding, and Losing, Meaning

Philosophy to Science

Math and science started out as philosophical musings. Natural philosophy became Physics. The conjectures of alchemy gave way to the rigor of chemistry.

This raises the philosophical question of "what is science?". The dictionary definitions don't capture the importance of science as an approach. The popular image of science is as a bunch of facts or a strict methodology.

I see science as an operational approach to "understanding". The heliocentrism wasn't more correct than geocentrism but a choice of reference frames that gives us insight. The term "solar system" gives us a way to organize our thinking and make a very complex system simpler. When we get into the details, we lose some of this simplicity, but for the purpose of general understanding, it is a very useful construct. Software, in a sense, is the art of managing such perspectives towards a purpose (which includes creating new perspectives or abstractions).

In psychology, Skinner's mechanistic approach alienated those with a more abstract view of cognition. It drove people like JCR Licklider, an acoustic psychologist, to go to MIT, putting us on the path to today's Internet. With the Internet, the value is in the whole rather than any particular element.

Zeros and Ones

The zeroes and ones of computing (BITS or Binary Digits) were central to solving a problem faced by the telephone company – how do you preserve a signal over a long distance. Traditional analog systems were plagued by the signal becoming fuzzier the further it went.

This is no surprise to those who played the aptly named game of telephone in which each person whispers a message to the next person. By the end, the message is garbled.

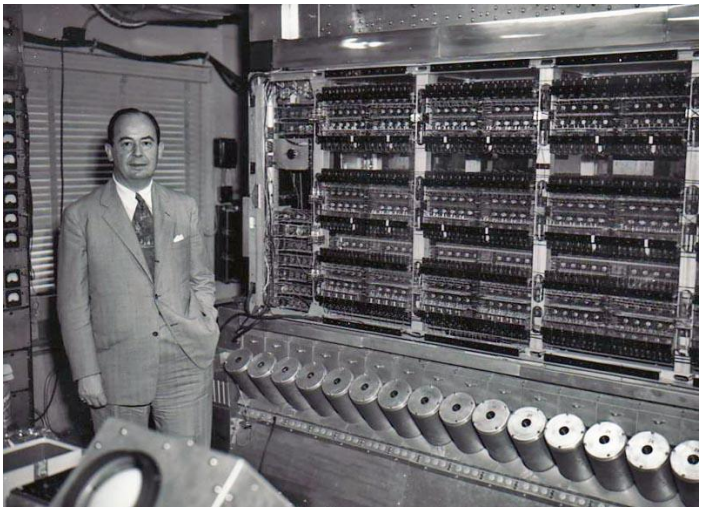
The solution was to encode the signal as discrete zeroes and ones. A signal had to be one or the other. This allows it to be regenerated when it gets a little fuzzy since we know it must be one or the other. This was a very powerful idea but wasn't fully appreciated because, as with any technology, it was initially used to emulate the analog circuits.

Unfortunately, Claude Shannon used the term "information" as the measure of entropy or the limit on the ability to regenerate such bit patterns. It's an abstract measure of entropy that has nothing to do with our day-to-day use of the term. He understood that the complexity of the system was not inherent – he was just measuring one property. He was solving a very constrained problem of measuring the number of symbols that could be transmitted in an abstract channel.

Speech is not contained in contained inside channels. So the measure is irrelevant. And since packets are each tagged with a destination, there is no need for channels. The information is in the whole and not inherent in each packet, so we don't even need reliable delivery. Our entire public policy, in the guise of the Federal Communications Commission, is based on a mistranslation. Speech and the particulars of technology are not congruent.

The Von Neumann Architecture

John Von Neumann's general-purpose computing engine seemed clever. Instead of hardwiring each instruction into a circuit, he represented each value as a symbol consisting of zeroes and ones. There was no need to rewire the computer for each new computation. Instead, one simply loaded new values into the machine registers.



The digital nature of the zeroes and ones are the key to enabling reliable hardware and for that hardware to scale from simple components to very large systems. As an aside, Jay Forester, in the late 1990s, gave a talk on the big problem facing such computers – tube burnout. By simply keeping the filaments on, one could avert the burnout that came from thermal cycling. We could've gotten very far with computing, even without transistors. It's about the concepts and not the particular technology.

One remarkable part of this is that there is no distinction between data and instructions. The ones and zeros are just numbers and are not intrinsic but rather come from the context. If the hardware interprets 1011 as an add

instruction, it would do addition. It could also treat it as a number, and adding 1 would yield 1100. Later we would go meta with instructions interpreting numbers as other instructions.

I don't know how much was understood at the time, but powerful ideas are discovered, and the agility of software allows us to learn as we build. We can learn from mistakes rather than having to avoid them.

Going Autonomous

The ability for the computers to operate autonomously without our direct control while performing arbitrarily complex operations takes this building block and gives it life. Perhaps literally, as we'll see in the discussion on DNA.

The ability to create an intelligence that performs actions on its own blurs the line between direct and indirect action and between intent and consequences. Even a simple program can take on a life of its own beyond our ability to understand. Today's Machine Learning gives up any pretense of the ability to explain its actions. Since we only see what is on the surface, we don't know how its actions reflect what we may presume about intent in the same way that a single value isn't predictive if we don't know the derivatives behind it.

The distinction between human cognition and machine intelligence is blurred by using anthropomorphic language to describe computing. It isn't the same as human intelligence. Not because we understand any fundamental distinction but because it happens to use different hardware and different interfaces to the world.

We need a nuanced understanding of the commonality as well as the differences if we are to come to terms with machine learning (AKA artificial intelligence).

We can debate philosophy, or we can simply take an operational approach and try to understand what we've wrought and what computers have wrought.

Fortran – a Better Calculator

```
C AREA OF A TRIANGLE WITH A STANDARD SQUARE ROOT FUNCTION
C INPUT - TAPE READER UNIT 5, INTEGER INPUT
C OUTPUT - LINE PRINTER UNIT 6, REAL OUTPUT
C INPUT ERROR DISPLAY ERROR OUTPUT CODE 1 IN JOB CONTROL LISTING
      READ INPUT TAPE 5, 501, IA, IB, IC
  501 FORMAT (3I5)
C IA, IB, AND IC MAY NOT BE NEGATIVE OR ZERO
C FURTHERMORE, THE SUM OF TWO SIDES OF A TRIANGLE
C MUST BE GREATER THAN THE THIRD SIDE, SO WE CHECK FOR THAT, TOO
      IF (IA) 777, 777, 701
  701 IF (IB) 777, 777, 702
  702 IF (IC) 777, 777, 703
  703 IF (IA+IB-IC) 777, 777, 704
  704 IF (IA+IC-IB) 777, 777, 705
  705 IF (IB+IC-IA) 777, 777, 799
  777 STOP 1
C USING HERON'S FORMULA WE CALCULATE THE
C AREA OF THE TRIANGLE
  799 S = FLOATF (IA + IB + IC) / 2.0
      AREA = SQRTF( S * (S - FLOATF(IA)) * (S - FLOATF(IB)) *
     +     (S - FLOATF(IC)))
      WRITE OUTPUT TAPE 6, 601, IA, IB, IC, AREA
  601 FORMAT (4H A= ,I5,5H  B= ,I5,5H  C= ,I5,8H  AREA= ,F10.2,
     +        13H SQUARE UNITS)
      STOP
      END
```

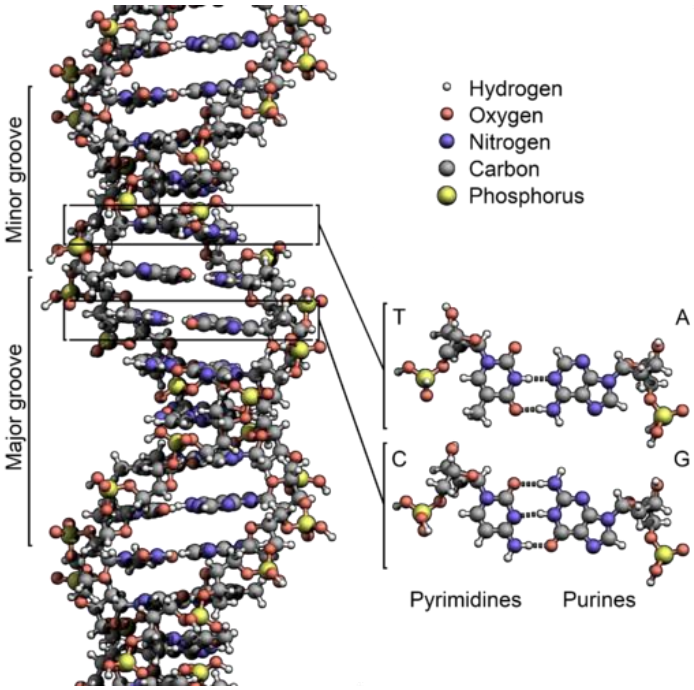
The military saw computers as a way to perform numeric calculations, and the early computers were programmed, using plugboards, to perform tasks like computing ballistics tables for the military.

Reduced costs made such computing available to scientists and business users who needed to perform calculations for such purposes as computing payroll and taxes.

Lisp – Symbolic Computing

In 1958 Jon McCarthy proposed Lisp, a List Processing Language suitable for the nascent field of artificial intelligence.

This is a powerful idea – the ability to perform operations on strings of bits is interesting, but that's only one way to use them. We needn't interpret them as numeric values but merely abstract symbols. It liberates us from being limited to any intrinsic property. The meaning comes only from the context – what meaning we assign to those symbols.




```
(apply speak (cat dog)) [1958 Lisp]

Apply(speak, [cat, dog]) [JavaScript 1995]
```

It's telling that this idea is in the center of today's modern programming languages as in JavaScript. JavaScript may have been developed in a few days, but it has helped shift computing back from a more limited vision based on optimizing for limited resources in the computing devices and has allowed us to focus on concepts.

The rigid rules of languages like C++ and Java were based on a sharp distinction between preparing a program and its execution. One aspect, describing the types and forms of data, was central to optimizing for a particular purpose. Today TypeScript provides a way to describe types, but it's a conversation with a computer as an assistant helping us "explain" intent while allowing for ambiguity.

Note that our very language works by using words as symbols without inherent meaning. As we say, the symbol "information" has very different meanings depending on the context. We fail to communicate when we share the symbol but not the context.

The Nature of Bits

Nature is based on the ability for bits to be regenerated and, as a consequence, preserve nuance. Think about DNA. It's similar to any computer program in being built on zeros and ones (in this case, 00, 01, 10, and 11 for ACTG).

Genes are interpreted, or, as biologists would say, expressed in the context of the chemical bath around them. It then creates its own chemicals, thus modifying its context.

Preserving nuance is key to enabling systems to evolve by capturing powerful "ideas". If a jawbone can detect vibration, the concept of hearing is born, and a jawbone can be reinterpreted as an ear bone. If we can hear, then generating sounds becomes a thing. This isn't necessarily how these mechanisms evolved or, more to the point, co-evolved, but it is the kind of mechanism.

This process is simply math. It's similar to wrapping a string around a pole and seeing that it's 2π times the pole's radius. It's not a result of calculation but rather an observation based on basic constraints.

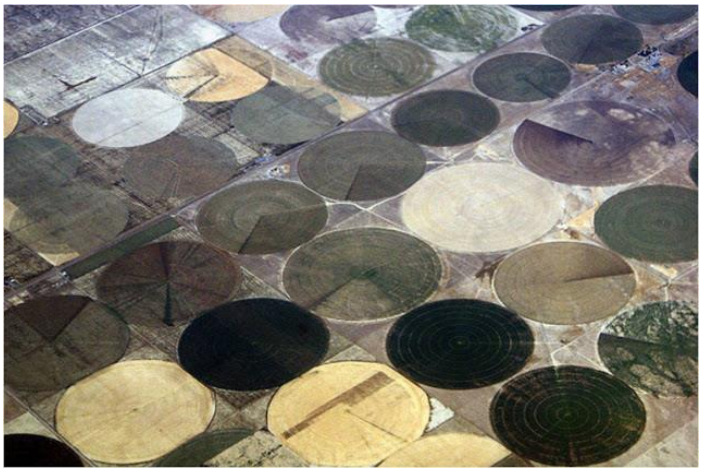
We see the digital pattern repeat at scale, with organs being discrete entities. In geology, we have discrete formations as when silica forms a distinct stone.

Counter-intuitive?

We're used to looking at systems through a fuzzy lens at a large scale. Instead of seeing discrete molecules, we may see a surface or a liquid. In biology, we see the whole as a single entity. The boundaries between two landscapes or two individuals or two clouds are not sharp.

Thus, we tend to view nature in terms of continuous changes and proportionality. If we hit a ball harder, it goes proportionally further.

Bits don't work like that. Change a single gene, and the consequence might be to completely change a forest. A simple idea like center pivot irrigation can drain the aquifer of an entire continent.



We like to think we can predict the future, and good ideas are fated to win. But if nature is fundamentally discontinuous without intrinsic meaning, then that doesn't work.

I was struck by a particular version of this in reading Lee Smolin's book, *Time Reborn*. He presented lots of intriguing examples of the physics of time, but, in the end, he laments that despite all this insight and understanding, he can't predict the stock market. For me, that is the central issue – how do things work if we can't predict the macro from the details. Maybe physics has it backward or is simply out of its league.

Looking Ahead

In my essay, Purpose vs. Discovery[ii], I explore some of the implications of these concepts.

In my upcoming columns, I will continue to explore how I apply these concepts. Part of this is understanding the Internet in perspective. For example, in my house, I build on the same ideas as the Internet at large, but my home applications are not a layer on the Internet. They use the concepts as a resource. And I use these concepts to design my systems.

These concepts need to be understood as basic engineering. Why run a wire when you can simply bind symbols together. A true Internet of Things would take advantage of common connectivity rather than requiring a special infrastructure. We're poised to take advantage of opportunity, but we need to first understand these new concepts.

A first step is to recognize that today's telecommunications policies are framed in the failed idea that we need an intelligent network to transport meaning. The meaning (and intelligence) is in our devices. We need to take an infrastructure-based approach rather than purposely limiting our ability to communicate, limiting our ability to access education and health care, and limiting our ability to innovate.

It is useful to close with a real example. I was able to get my COVID vaccination scheduled because one individual wrote a simple application that tracked availability and sent me an alert. Software has given individuals the leverage to do what the billion-dollar healthcare system could not. We should embrace such innovation rather than accepting the failed idea that intelligence is inside networks rather than in ourselves.

Bob Frankston is best known for writing VisiCalc—the first electronic spreadsheet. While at Microsoft, he was instrumental in enabling home networking. Today, he is addressing the issues associated with coming to terms with a world being transformed by software. Contact him at IEEECEMagazine2021@bob.ma.

[i] https://1drv.ms/p/s!AglBpbKyzKIQj4Ed6OS1Yi42iN-HIQA?e=21uva8

[ii] https://rmf.vc/PurposeVsDiscovery