;login:

THE MAGAZINE OF USENIX & SAGE

April 2001 • volume 26 • number 2

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

needles in the craystack: when machines get sick

Part 3: The Myth of Computer Control

by Mark Burgess

Mark is an associate professor at Oslo College and is the program chair for LISA 2001.

<Mark.Burgess@iu.hioslo.no>

If apathy could place our technological future in jeopardy, making us slaves rather than masters of our contrivances, then the diametric pitfall is a simple arrogance: the desire to control it all. The absolute control of computers is doomed to failure, for the same reasons that the absolute control of weather and environment are doomed to failure. Such complexity cannot be controlled, it can only be regulated. To understand why and what this means for the administration of our systems, we must leave the relative safety of simplification, and dive into a more turbulent sea, in search of detail.

Imagine opening up a computer and aiming a microscope inside. Imagine turning up the magnification step by step through circuit boards, components, materials, and, finally, atoms and electrons. Clearly the important electrical behavior of a computer takes place at the level of electrons, so we could in principle understand almost everything about a computer in terms of (1) the arrangement of its atoms and (2) the motions of its electrons. Although possible in principle we could never do it in practice. We would never be able to cope with all those details in one go. It would be like trying to explain the trends and goings-on in a city by tracing the movements and interactions of each of its inhabitants over time.

Our brains have limited processing power; we are not good at modeling complex details and extracting long-term trends, so we must use a strategy of divide and conquer, *by scale*. "Scaling" is a common term in physics, where it has long been understood that an understanding of complex phenomena can, in many cases, be separated into the understanding of different partial-phenomena at different scales or sizes. This was how the magician's illusion worked in Part 2 of this series.

If we return to the microscope analogy, it seems reasonable to divide the understanding of computers up into scales, which minimize the number of independent levels compounded to create the illusory magic of computers. For instance, there is the level at which electrons move around electric circuits in controlled ways, leading to microscopic switching technology. The transistor, invented by twice Nobel laureate John Bardeen (his second prize was awarded jointly with Leon Cooper and Robert Schrieffer for the BCS theory of superconductivity), was the key development that paved the way to miniaturization of digital technology. Then there is the level at which transistors are combined to make Boolean logic gates, the level at which logic gates are combined to produce processors and memory, then the level at which these are combined into a working computer with devices attached, the level at which computers are combined into networks, and so on.

These levels are essentially independent. Each scale adds new information, which does not depend significantly on the details of how the lower levels worked. So, it would not matter to the construction of computers if one replaced transistors with optical switches, the operation of the higher levels would be the same. It might be faster or heavier, so specifics might vary, but the evaluation of answers would not be affected by the changes. This is fortunate, otherwise we would have to redesign computers every time a new switching technology came about. Systems are routinely designed in layers

so that components can be exchanged for newer and better ones, without causing turmoil. For instance, think of the OSI network model, or TCP/IP. Telnet works over serial lines, or via TCP/IP over Ethernet, or with TCP/IP over fiber optics, etc. It is like procedural programming. It is the most important principle in computing.

This principle is applicable in many situations in science. For instance, one does not need to know about quarks in order to describe planetary motion, even though planets are almost entirely made up of quarks. The scale at which quark interactions are important is so small that the results do not depend on what the quarks do to a hundred decimal places or more. In other words, low-level differences can be ignored to a very good approximation.

The Most Important Idea in Computing

Paradoxically, we tend to claim that we understand something if we know how to express it in terms of smaller parts. This idea is the essence of all hierarchical thinking: directory structure in file systems, process trees, and so on. It is all about hiding unnecessary detail in Pandora boxes, to simplify and thereby reveal clarity of structure. As long as we know how to open a black box, we are happy to say that we understand it, at least in principle.

Paradoxically, we tend to claim that we understand something if we know how to express it in terms of smaller parts.

Of course, this reduction is not always possible: there are prominent counterexamples. Quantum mechanics is one: it cannot be reduced into anything else we know about. Quantum mechanics consists of a set of rules for calculating the mechanics of particles at the microscopic level, which works with extraordinary precision. Despite this, no one knows how to explain these rules in terms of anything deeper, and for that reason most of us feel that we do not understand quantum mechanics, and rightly so. Another example is nonlinear systems where chaos leads to a mixing of levels, in often unpredictable ways.

But how well does anyone understand anything? The illusion of understanding is, in a sense, proportional to the number of times we can reduce a thing into smaller or more fundamental things. At some point that process has to stop, and we end up with pieces which we can no longer break down. So how deep do we need to go in order to claim to understand something? In fact, most of us are not very demanding; if we were, we would probably not be able to cope with the detail required.

The principle of reductionism is simple to state: an efficient way to understand any *thing* is to take it apart, breaking it down into its constituent parts to see how these parts interact to form the whole. Usually this is a complex undertaking, and we never manage to complete this program of inquiry. Often the parts themselves are so numerous and so unexpected that their study tends to dominate the discussion, and the important information about how they fit together is left out. This has certainly been true of system administration.

If we want to understand a car, we begin by describing its main features and then proceed by taking it apart and examining each of those parts. Each part which we remove can be broken down into smaller and smaller parts until, in the end, we come down to atoms and their constituents. The basic Lego pieces. We learn a lot by this reductionist process, but it is crucial to understand that, at every stage of the reduction procedure, we *lose* the information about how to put the parts back together in the right places in the right order – that is, the information at the level above. In the end, the Lego bricks all look the same and tell us little about the original construction.

Understanding the relationship between continuous and digital signals is a step toward understanding the implications of complexity for computer systems.

It is the structural information which distinguishes a car from a refrigerator, or a horse from a block of buildings. When it comes down to it everything is made up of a few basic constituents. What defines unique entities is the information about how to put them together. Once again, it is about levels of details, where structure does not necessarily depend on the exact nature of the building blocks, to some degree of approximation.

Analogy, Approximation, and Cognition

So is this about approximation? In a sense it is. Computers were not always digital by construction. Before digital electronic computers were built, electrical circuits were constructed to model particular calculations with voltages and currents. These continuous signals evaluated continuous solutions to differential equations. They did not work to a fixed digit floating point accuracy, they worked by *analogy* and answers were obtained with voltmeters and galvanometers. They were thus referred to as analog computers, or analogy engines. The word "analog" has since entered our vernacular with the meaning "opposite of digital," which is clearly nothing to do with its root.

Understanding the relationship between continuous and digital signals is a step toward understanding the implications of complexity for computer systems. Everything looks more blurry than it is. There is an inherent uncertainty, not only in our senses, but in any measurement, due to the limits of digitization (the resolution) of a scale of measurement. The same uncertainty lies in any digital or digitized process. There are no truly continuous signals. Even detailed sources are only sampled with limited resolution, so their original nature is neither relevant nor determinable.

When we play a CD we hear continuous music, not digital staccato. The resolution of audio sampling on a CD is greater than the sampling rate of our ears, and thus it is beyond our ability to discern the difference: higher Fourier modes that might otherwise unmask the charade are not available to our perceptual apparatus and thus we are incapable of knowing more without artificial aids. Even artificial aids have limitations. They might extend the ability to detect greater resolution, but no instrument or physical object can provide infinite detail, for the simple reason that the universe itself has finite detail.

A CD recording is but a simulacrum of an original physical (musical) process: wasn't that original process continuous, with infinite detail? (Hi-fi buffs have sometimes used this argument in favor of LPs over CDs.) The answer is no: the bow of the violin is a rough surface which plucks the string very fast, giving only the illusion of continuous play. The grains in the vinyl of an LP imply a maximum resolution that can be represented in that medium, beyond which is unpredictable noise. At some point (for whatever reason) one comes down to a minimum size for representing information: single atoms, for instance, probably represent the smallest units one can use to represent information with any fidelity.

Our sensory-cognitive apparatus has a great influence on the way we go about analyzing information and "understanding" it. So much so that its prejudices are frequently fooled by optical illusions, Rorschach tests, blurry images, color blindness, Martian canals, etc., etc. Our desire to see patterns and to fill in gaps, coupled with the experience that a closer look reveals more detail, builds a tower of assumptions about how things behave. We are seldom asked, by nature, to examine the limits or consequences of digitization, but when analyzing scientifically, we are obliged to explore these consequences.

As humans, we think and assign meaning in digital terms: either we coarsely classify shades (blue is still blue, no matter how light or dark, turquoise is still blue-green, no matter what mixture), or we enumerate classes (like red, green, blue, with arbitrary shade boundaries). In spite of this, we still maintain the illusion of continuous change, and we describe it mathematically as the limit of a digital process, abhorring the discontinuous in favor of smoothly changing lines.

Continuous representation of information is a thorn in the side for pattern recognition. True pattern recognition can only be achieved for a digital representation and the same is therefore true of computer control, since the elements of control are digital instructions. When two strings of digits are the same, a match is found. We can match any kind of blue, only because the digital concept of blue represents a whole range of colors. But the range of possible blues means that there is an uncertainty in the meaning of "blue." It is a class of finite width. When a digitization is very detailed, i.e., approaching the continuous, it is not always practically possible to distinguish every digit of data. Similarly, it is not practical to issue very complex instructions at the microscopic level.

Analog and Digital: Some Information Theory

This makes more sense if one knows some information theory. The study of information began in the 1930s and 1940s by mathematicians such as Church, Turing, Von Neumann, and Shannon. What we call information theory today was largely worked out by Claude Shannon (of Bell Labs) and published in 1949 under the title *The Mathematical Theory Of Communication*. He defined the mathematical measure of information (entropy) and devised many of the core theorems used in information theory today.

Information theory is about the *representation* (manner of storage), *interpretation*, and *transmission* of patterns of data, i.e., patterns made up of different kinds of "something" and "nothing." We attach meaning to these patterns and call the result "information." Patterns of data are mainly of interest when they are transmitted from a *source* to a *receiver*: e.g.,

- Text read from page to brain
- Morse code sent by telegraph or by lantern
- Speech transmitted acoustically or by telephony
- Copy data from hard-disk to memory
- Copy data from memory to screen
- Copy DNA using RNA within a cell

In each case, a pattern is transferred from one representation to another. Information is fundamentally about representation. It is the limitations implied by a given representation which form the substance of information theory. Some of the issues one has to contend with are:

- Distinguishing patterns of "something" and "nothing"
- Perhaps distinguishing different kinds of "something" (classification)
- Space-time coordinates and units (is a long beep the same as a short beep? quantization/digitization)
- The meaning of redundancy and repetition in a signal

To build a more precise picture of what information is, we can begin with a signal f(t,x,..), of unknown detail, which is a function or field that changes in time or space. The signal is really just a pattern formed by a disturbance in a physical medium. Our everyday experience leads us to believe that there are two types of signal f(s):

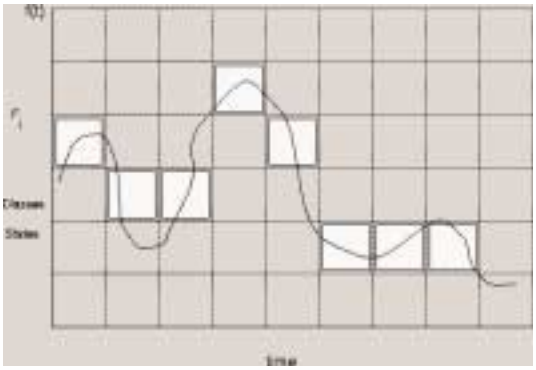

Fig 1. Coarse-graining or digitization is a coordination of the continuous signal.

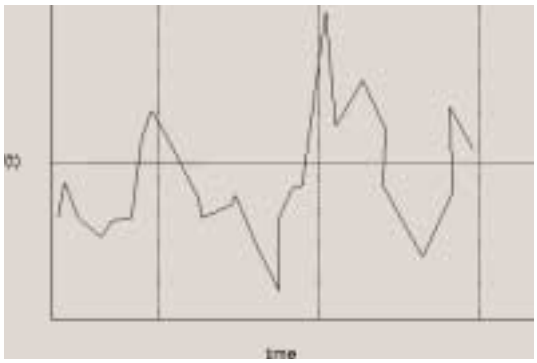

Fig. 2. A poor digitization cannot sensibly determine the value of the signal within the cells.

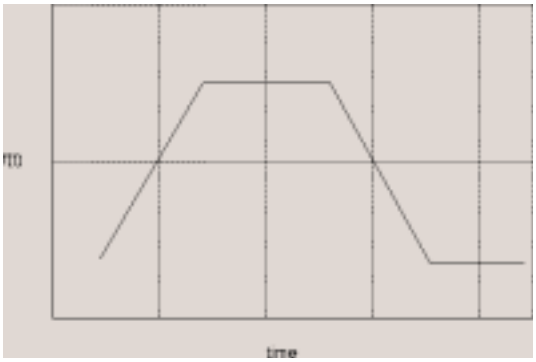

Fig. 3. A well-suited digitization without loss. This signal can be represented with just two classes, i.e., binary digitization.

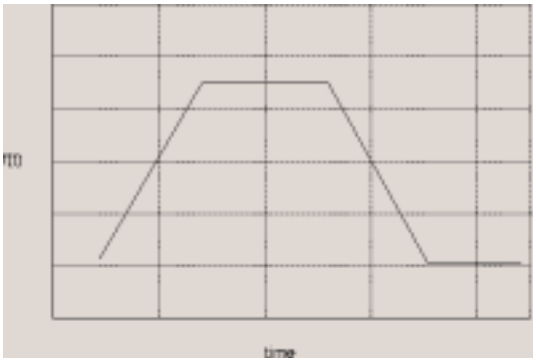

Fig. 4. The same signal, as in Figure 3, this time digitized into six classes.

- Analog or continuous functions f(t)
- Digital or discontinuous (step) functions

It should already be clear that this distinction is artificial but that in this distinction lies the central essence of what information is about. An analog signal is a limiting case of a digital signal.

In order to say anything about the signal, one has to map it out by placing it on a discrete grid of coordinates. At some arbitrary level, one decides not to subdivide space-time any further and one reaches a limit of resolution.

Information can only be defined relative to such a set of coordinates since it is the only means one has of describing change. Let us begin by assuming that the detail in the signal is greater than the resolution of the grid. We do the following:

- Divide up the time axis into steps of equal width dt. Here we shall look at an interval of time from t=0 to t=N dt, for some N.
- Divide up the f-axis into C classes $F_i = [F_i^-, F_i^+]$, which touch such that $F_i^+ = F_{i+1}^-$.

Digitization means that, whenever the function f(t) is mostly inside a cell F_i, its value is simply represented by the cell. We have compressed the detail in the square region F_i into a single *representative* value i, over an interval of time.

There are good digitizations and bad digitizations (see Figures 2, 3, 4). Nyquist's law tells us that the interval widths need to be half the width of the "finest change" in the signal. In Fourier language, the sampling rate must be twice that of the greatest frequency we wish to resolve. This is why CD players sample at 44.1kHz and DAT samples at 48kHz: the limit of human hearing is about 20kHz when we are young, and falls off to about 12kHz as we grow old.

The digits F_i are the "atomic" units (the Lego bricks) of information: they are a strict model for representing change. If C=2, we have binary digits $\{F_1, F_2\} = \{0,1\}$ etc., or *bits*.

God Is a Pile of Bricks but the Devil Is Fractal Soup

The reality of continuous data is a myth, because the idea of precise values is a myth. We sample or measure in finite elements, which have a finite size and thus an intrinsic uncertainty.

You might think that this sounds wrong and look for a way out: after all, mathematical curves have precise values, based on strict relations. If we plot a curve of system behavior, then we can sample with unlimited accuracy. This is true, but it is cheating. Naturally, one can compress data into a few parameters, if the whole story is known for all time, e.g., y=sin(x). Computers are not like this, because they are moved by unpredictable environmental influences. The number of parameters needed to describe their behavior is not known and may not even be constant. The values are hidden in "the environment," which is too big and complex to describe precisely.

Floating point representations can only calculate with limited accuracy; analog computers were only true representations of the calculations they were built to model, within certain tolerances. Can we ever achieve complete accuracy? The answer is no. No non-integer value can be measured with unlimited accuracy; thus no non-integer value can be fed into a computer with total accuracy; hence no exact

answer can be computed, even if the computer has infinite information resolution (which no computer has). In short, once we stray from the solid world of integers, everything is an approximation. This is the core realization of experimental science, and it is the fact which has implications for control of computers.

Digitization has consequences for *classification* and thus for system *state*. A digital representation is good enough to represent any state, but only within a certain tolerance. A digitization of fixed resolution implies an intrinsic uncertainty. Trying to control computers with simple push-button commands is like trying to counteract noise with single-tone filters. The amount of information is incommensurate.

In the mythological realm of truth and uncertainty, one might say that God is digital and the devil is continuous. Absolute truth and clarity can only be determined for discrete sets, at a fixed scale. Truth is not about infinite detail, it is about infinite clarity: or being able to see every digital grain. Truth is thus only obtainable in a low-resolution picture of the world. If one is forced to deal with continuously varying quantities with infinite detail, such as averages, then hellfire abounds in the uncertainties of never knowing how closely to look, never knowing what scale to choose. At some point one must give up and choose an arbitrary scheme of measurement (the measuring stick for comparisons). That is why anomaly detection and pattern recognition are hard. One cannot fail to have awesome respect for the brain as a processing device: it is fantastically successful at pattern recognition, better than any simple man-made algorithm.

There is one more twist which may turn out to be important in the quest to understand computer system behavior. Mathematically, a separation of scales is often possible only because the scales couple linearly. Nonlinearity is a concept which has been adopted from physics into common language, where it is loosely used to convey complexity. It occurs through dependency; a dependency is a strong coupling between parts of the system.

In a nonlinear environment, even small changes can resonate with large consequences. This idea was made popular by the notion of the butterfly effect, where the fantastical flapping of the wings of a butterfly in Japan could lead to equally legendary hurricanes in Florida (weather is nonlinear fluid dynamics). In other words, in nonlinear systems, it is difficult to predict the outcome of small perturbations. Nonlinear amplification tends to mix scales together, which is why nonlinear systems are difficult to understand (one often speaks of "chaos").

Digitization is clearly at the heart of computer behavior and measurement. We can be more precise about describing it and measuring its effects. The ability to discern state is clearly dependent on arbitrary choices. There is an inevitable nondeterminism (probability rather than certainty) in any system immersed in an environment. We need to recognize this and adopt strategies appropriately.

Computer Ecology Means Computer Immunology

Why is it not possible to control computers? The reason is simply that they are not predictable. We grow them in the laboratory, with simple push buttons, but we raise them in the wild, where environment pokes and prods them with far greater resolution. Unless we place systems in straightjackets, the influences of environment take their toll. Computer behavior is more intricate than any set of controlling commands we can practically build. This is an imbalance which provides the explanation for the tendency

In the mythological realm of truth and uncertainty, one might say that God is digital and the devil is continuous.

Competition is all that is left of strategy once digitization is suppressed.

of systems to disobey their controls. In order to compete with the complexity of the environment, one needs continuous measurement and regulative forces. *Competition is all that is left of strategy once digitization is suppressed.*

Machines get sick when this regulation fails. Sometimes regulation fails because a foreign element (virus) invades the system. Actually, it is not really important whether the element is foreign or not: the invading element might be "self" or "non-self." What is important is whether its influence is predictable in relation to the average state of the system. Will it lead to an instability? Will it drive the autonomous system in a new direction, different from the intended one? Sickness is not about right and wrong. Bacteria and viruses are not failures of a system, they are simply unusual input which alters its direction.

Inherent weakness is a precarious strategy for building technology, yet we build simple things for complex jobs. Fragility is no strategy for success. What is it that makes a system resilient and fit for its purpose? That it is under held rigid control, that it contributes to a larger whole, or perhaps that it endures under tough conditions? There are many answers. Much of the work in system administration has been about looking for simple answers to this complex problem, but we need to think of computers more as wildlife in a changing ecology. We need to embrace the intrinsic uncertainties faced when placing coarse digital animals into complex, intricate environments.

In the next part, we turn to entropy and what it means in the context of computer systems.