Abstraction, Indirection, and Sevareid's Law: Towards Benign Computing

Barath Raghavan
ICSI
barath@icsi.berkeley.edu

ABSTRACT

Computing is one of the primary means by which we solve problems in society today. In this short paper we examine the implications of the primary techniques used in computer systems work—abstraction and indirection—and of Sevareid's Law, an epigram that suggests that our problem-solving instinct may be leading us astray. We explore the context of this dilemma and discuss instances in which this has arisen in the recent past. We then consider a few design options and changes to the normal mode of computer science practice that might enable us to sidestep the implications of Sevareid's Law.

1. INTRODUCTION

The chief source of problems is solutions.

–Eric Sevareid [17]

Sevareid's Law expresses a conundrum at the core of modern industrial society. As we collectively identify and "solve" "problems", we inevitably find that not only are many "problems" not real problems, but that "solutions" applied to them are the causes of new woes. The history of the last century is littered with examples of this conundrum, from chemical solutions like DDT [2] and Tetraethyl lead [6] to agricultural solutions like mono-cropping with synthetic fertilizers and pesticides [15] to financial solutions like leverage [3] and complex derivatives [8]. In recent decades, computing has become central to new solutions and to technological progress.

Seeing the consequences of these "solutions", some have argued that technological progress, as currently understood in industrialized societies, is itself suspect and undesirable, and perhaps even something against which to fight. On the other hand, proponents of technological progress argue that no matter the downsides, technological progress must be pursued [5]. We hope both to sidestep this intractable debate in this paper and to chart what might be a middle course in the context of computing.

Specifically, we introduce the notion of *benign computing* and attempt to draw up a set of principles for computing that is less likely to have unintended, harmful downsides to the global ecosystem and to the subset of the ecosystem that is human society. Today computing is seen as a source of potential solutions in nearly every major sector of society, including energy, agriculture, transportation, health, education, manufacturing, science, and governance. Modern computing, however, is still new enough that its principles and approaches have not withstood the test of time, and so the implications of Sevareid's Law for computing can only be definitively seen with the benefit of hindsight. As we enter a critical period in global society due to the ecological limits industrial society faces today [10, 13, 16, 19], this is an appropriate time to reconsider both the foundations upon which computing is built and consider a course correction if computing is to meet the needs of human society in an age of limits.

LIMITS'15, June 15–16, 2015, Irvine, California, USA

2. ABSTRACTION AND INDIRECTION

Modularity based on abstraction is the way things are done.

–Barbara Liskov [9]

In most fields of computer systems research and engineering, abstraction and indirection are key design principles. Abstraction involves the distillation of data or concepts, creating orderliness and enabling simplification and modularity. Indirection involves interposition on the flow of data or control between two modules within some software or hardware system. Abstraction makes indirection easier, and together the two principles make tractable computer systems of a scale unimaginable a few decades ago—all large-scale computing systems today rely upon layer after layer of indirection built upon an intricate weave of abstractions.

2.1 Computing and Society

Given the ubiquity of computing in wealthy nations today, the impact of computing on society is self evident. However, the nature of that impact—its benefits and drawbacks, its consequences and character—is still far from clear and hotly contested. We do not aim to recap the ongoing debates on this subject, but briefly note below a few points relevant to this paper.

In his role as long-time technology journalist and pundit, Kevin Kelly's attempts to find a middle ground between techno-utopian and techno-phobic arguments are well worth considering [5]. He concludes that while technological solutions do almost always create new problems (per Sevareid), in his view technological progress is an unstoppable force and ultimately should be embraced. Thus Kelly concludes that society should embrace the sisyphean task of solving problems created by a previous generation of technology while creating new ones in the process. While still techno-utopian in many respects, Kelly's view is more moderate than many technology commentators (and indeed many researchers and engineers) who ignore or dismiss technology's downsides.

Hidden in Kelly's discussion are three issues that critics have seized upon. First, technology often solves problems temporarily, if at all, when evaluated holistically. Second, technology often doesn't even solve problems on its own terms—that is, even by the metrics used by a technology's proponents, it often fails. Third, some technological solutions aim to address problems that are in a fundamental sense unsolvable.[1] Examples in each of these three categories are numerous, and are staples of the work of polemicists such as Morozov [11, 12]. Many of these hidden issues are due to abstraction and indirection—consider the ways in which, for example, friendships are cheapened by the abstraction and indirection introduced by social networking, or self-employed freelance entrepreneurs are reduced to replaceable cogs in a task-based economy intermediated by various web services.

2.2 Benefits and Drawbacks

While we are concerned primarily with computing technology here, the tangle of these three issues and of abstraction and indirection are common in other disciplines as well. For example, the woes of industrial agriculture could be seen as a consequence of the abstraction of plants as machines that take water, N-P-K, and sunlight and turn it into food; the abstraction of home mortgages and bundling into complex financial instruments through many layers of indirection was key to the financial crisis of 2008. It would be worthwhile to examine whether abstraction and indirection are central to solutions and their subsequent problems in many fields; indeed, given the complexity of today's computing systems and their importance to today's society, understanding the benefits and drawbacks of these principles when applied elsewhere is crucial.

By and large, abstraction and indirection have shown clear benefits in solving problems in the design and implementation of computing systems, so much so that they are sometimes jokingly viewed as the only two ideas in systems research. When applied within a system for its own purposes, it is relatively easy to identify whether the indirection introduced yields a benefit or adds unnecessary complexity. Indeed, unnecessary complexity is one of the key ways that indirection can go awry and cause drawbacks in excess of benefits; this is true along the entire scale of systems of all types, from a small piece of software to a civilization itself [18].

As Toyama notes, however, technology is only an amplifier of human intent, and thus it is important in many of these instances—both those examples hailed by Kelly and scorned by Morozov—to note the benefits that a technology's purveyors receive [20] (and in many instances the drawbacks that they ignore). Even when benefits and drawbacks are heeded, a technology's amplification can quickly get out of control if its power and subtle implications are not understood in advance.

3. BENIGN COMPUTING

Many small things breed a kind of stability; a few big things endanger it—better the Fortune 500,000 than the Fortune 500 (unless you want to be an eight-figure CEO).

–Bill McKibben [10]

There are many possible responses to the above issues, and in this section we propose one possibility: *benign computing*, a general design framework for building computing systems that are less likely to produce harmful impacts to the ecosystem (and thus to human society) and are less likely to become trapped by Sevareid's Law. Here we only offer a vision of what benign computing might become, in the form of design principles.

A key aspect of benign computing is a rejection of the utopian notion of creating new technology that is strictly "beneficial" or that advances "development". Such efforts suffer from a number of problems. First, benefit is always relative. Second, benefit, even when broad-based, is often difficult to measure. Third, the temporal profile of benefits and drawbacks can be complex for many technologies—benefits can occur before drawbacks, or vice versa, and worse still, even once drawbacks (or benefits) arrive they can be hidden. Instead, the aim of benign computing is computing that is of a scale and structure such that even if its downsides dominate, its overall harm is small because they are made apparent.

3.1 Inspirations

Setting aside the proposed responses of ardent boosters of any and all technological development and critics who suggest to throw it all away, there are some healthy trends in computing research we first consider for inspiration.

A positive trend is work in the field of Information and Communication Technology for Development (ICTD), which aims to use computing to address urgent and practical needs in countries and regions with fewer resources and less infrastructure, often employing thinking from the older field of appropriate technology. ICTD has advanced significantly in the past decade, and more importantly has established the practice of defining clearly both the societal problems being solved and the measures used to evaluate impact. In doing so, ICTD work has been more likely to yield intended social impact, though the resulting "benefit" and "development" often remains fuzzy.

While ICTD work often is careful in the means of implementation, focusing on "appropriateness" of the interventions applied and systems built, it begins with an assumption of good intentions and a judicious researcher employing the system for what is believed to be a worthy end. However much of computing work, both in academia and industry, is not done with social ends in mind; instead, intellectual and financial ends dominate. Thus we must consider how the notion of benign computing, which we detail below, might co-exist with these motives.

Another promising but preliminary area of work is in biomimicry, the design of systems that are modeled after nature. The challenge for biomimicry is to integrate true ecological understanding into the mimicry being attempted, rather than decomposing natural systems to identify pieces that serve specific needs. For example, work on stigmergy in computing systems has the potential to increase system resilience.

3.2 Industry

A key challenge is that computing today has a thriving industry, one that is naturally driven by profit motives.[2] This motive is not a problem in itself, but the manner in which computing startups aim to "scale" rapidly as an individual organization is a fundamental source of trouble. Indeed, computing startups (unlike in most other industries) are expected to demonstrate hyper-growth. A startup can in a matter of years have a direct impact on billions of people, and profit handsomely doing so (usually via the application of

[1]Greer has advocated differentiating between problems that have solutions and predicaments that have responses but no solutions [4].

[2]This is less the case in many other engineering disciplines, and far less the case in most scientific disciplines.

abstraction and indirection). The amplifying power of technology today is at such a level that the power needs to be used wisely, and there appears to be little understanding of the downside risks to society that such power creates.

Ironically, this structure is at odds with a principle at the core of modern distributed systems: horizontal scalability [1,14]. Horizontal scalability—"scale out"—is an approach in which a system is made faster and/or more resilient by adding more small units (e.g., individual computing nodes in a system) and is broadly favored today over vertical scalability—"scale up"—in which the power of a single machine is increased. However, when systems have been scaled horizontally, they are then cloaked in an abstraction that presents them as a single unified (large) system.

3.3 Principles

We contend that a different paradigm of computing research and practice is possible, which we term *benign computing*. The core aim of this paradigm is to make a technology's benefits and drawbacks more apparent to its designers, researchers, and implementers, enabling a proper evaluation that might otherwise prove difficult or inconvenient. In doing so, the aim is to help anticipate drawbacks of a technology and to help preserve its potential benefits, and to ensure that those benefits are more broad-based (i.e., that they are reflective of more than one perspective of "benefit").

Here we describe several design principles that we believe should be at the center of any work on benign computing. While only the test of time will show whether this is truly possible and whether these principles are the right ones, there is some evidence, both anecdotal and in other fields, that these principles lead to good results.

Scale-out. As we noted above, horizontal scalability—scale-out—is already a common principle in distributed systems work, but this is seldom then applied to the macro systems that they are supporting. There has been work on so-called *federated* systems, which are of the flavor we advocate here—systems where the scaling out is done by autonomous parties (i.e., under diverse administrative control) and the system as a whole is a federation of these parties' systems. An advantage of scale-out in traditional settings is that the failure of a few does not threaten the functioning of the whole, and that to increase scale the whole system need not be re-engineered. Beyond those settings, scale-out has the advantage that a coalition of parties can decide to create an independent offering. Indeed the systems built upon many of the early Internet protocols (e.g., SMTP, NNTP) had this structure, though they have lost it over the years.[3]

Fails well. Natural systems are complex; the weave of interdependencies in the global ecosystem is far beyond our understanding today. Yet this complexity does not yield vulnerability of the sort exhibited by complex human-made systems. A primary reason is that complex human-made systems have only apparent complexity—they seem complex, but have far fewer stabilizing backup systems to ensure resilience, often because resilience requires sacrificing system efficiency and leads to higher short-term costs. Natural systems, on the other hand, have inherent complexity; while not efficient in the way of many human technologies, they have significant resilience to failure. Thus the evaluation of computing systems should look beyond apparent complexity; that is, nature should be mimicked in the ways it handles failure, not just in the ways it succeeds in normal operation.[4]

Open design. While open source software and hardware is important, open designs are far more important. Open designs enable the creation of a diversity of implementations, written by different authors with different motivations but a common goal. This common goal can be codified in the form of an RFC or similar design document. In a scale-out system it is important that each independent party can build upon a base design to create new, differentiated offerings.

Self similar/fractal. Systems should be scale-out, open in design, and fail well at every level of their structure. While many of today's large-scale distributed systems have these properties at some level of their operation, few exhibit them at all levels. This fractal structure ensures that decoupling can occur at the level most appropriate for it to occur.[5]

4. CONCLUSIONS

The principles we offer above are not in themselves new or deep. Our only aim is to identify those approaches that can lead to a minimization of social harm should a system be recognized to be primarily harmful. Ultimately these principles aim to limit structural power—say, of the sort that large companies like Apple, Google, Amazon, Microsoft, and Facebook wield today—by creating computing systems that have a far greater underlying diversity, as in nature. Crucially, systems that have greater underlying structural diversity—and diffusion of power and control—can be more responsive to the needs of the local human and natural community. As McKibben suggests, there is a resilience—not just resilience to failure, but to unforeseen societal harms—that comes from a diversity of smaller parties providing a service instead of a smaller number of big players. While adopting these principles in commercial settings will be difficult, as they are likely to conflict with profit motives, computer science researchers are not similarly constrained. Indeed these principles appear to be better aligned with the role of research in society, as they enable a greater diversity and debate of ideas and approaches. Whether they can be adopted and supplant industry-driven "normal science" as it is practiced today in computer science remains to be seen [7].

[3]Consider the structure of Facebook vs. Craigslist: Facebook is monolithic—while it uses scale-out in its datacenters, the service offering as a whole is scale-up. Craigslist, on the other hand, is scale-out in many ways, as the sites for each community could (in theory) be spun off and run independently and diverge in its offerings. For a service like Craigslist to fully become scale-out in the manner we propose here, the service as a whole would be a federation of autonomous, regional Craigslists that are loosely tied by APIs, protocols, and links.

[4]To be more concrete, consider the Internet's routing system. While it is designed to be resilient to failure, its resilience is only in one dimension—alternative paths on the data plane—and not in control plane systems (e.g., backup alternatives for BGP), management systems (e.g., the administrative control of large ASes), and physical systems (e.g., IXPs, long-distance fiber bundles, etc.).

[5]For example, A federated alternative to Facebook might require divergence/decoupling at the level of datacenter structure in one region where electric power availability is intermittent, while in another might require it at the level of inter-region connections where different privacy standards require different data retention behavior for transient data. If the system were not fractal, some of these decouplings would not be possible.

5. REFERENCES

[1] L. A. Barroso and U. Hölzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-scale Machines. *Synthesis lectures on computer architecture*, 2009.

[2] R. Carson. *Silent Spring*. Houghton Mifflin, 1962.

[3] J. K. Galbraith. *The Great Crash, 1929*. Houghton Mifflin, 1954.

[4] J. M. Greer. *The Long Descent*. New Society Publishers, 2008.

[5] K. Kelly. *What Technology Wants*. Penguin, 2010.

[6] Kevin Drum. America's Real Criminal Element: Lead. *Mother Jones*, January/February 2013.

[7] T. S. Kuhn. *The Structure of Scientific Revolutions*. University of Chicago Press, 1962.

[8] M. Lewis. *The Big Short: Inside the Doomsday Machine*. WW Norton & Company, 2011.

[9] B. Liskov. *The Power of Abstraction*. Turing Award Lecture, 2009.

[10] B. McKibben. *Eaarth: Making a Life on a Tough New Planet*. Henry Holt and Company, 2010.

[11] E. Morozov. *The Net Delusion: The Dark Side of Internet Freedom*. PublicAffairs, 2011.

[12] E. Morozov. *To Save Everything, Click Here: The Folly of Technological Solutionism*. PublicAffairs, 2013.

[13] D. Pargman and B. Raghavan. Rethinking Sustainability in Computing: From Buzzword to Non-negotiable Limits. In *Proceedings of ACM NordiCHI*, 2014.

[14] D. A. Patterson, G. Gibson, and R. H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *Proceedings of ACM SIGMOD*, 1988.

[15] M. Pollan. *The Omnivore's Dilemma: A Natural History of Four Meals*. Penguin, 2006.

[16] B. Raghavan and J. Ma. Networking in the Long Emergency. In *Proceedings of the ACM SIGCOMM Workshop on Green Networking*, 2011.

[17] E. Sevareid. *CBS News*, December 29, 1970.

[18] J. Tainter. *The Collapse of Complex Societies*. Cambridge University Press, 1990.

[19] B. Tomlinson, E. Blevis, B. Nardi, D. J. Patterson, M. Silberman, and Y. Pan. Collapse Informatics and Practice: Theory, Method, and Design. *ACM Transactions on Computer-Human Interaction*, 2013.

[20] K. Toyama. Technology as Amplifier in International Development. In *Proceedings of iConference*, 2011.