



## Plenty of Room Outside the Firm

Charles Petrie • *Stanford University*

**T**here are many ways to think about the future of computing, but to do so successfully requires looking at previous experience and predictions. My focus here is on computing rather than computers, though the increasing power of the latter facilitates the former. For example, I have nothing to say about the future of multicore or parallel computing, except that beyond the Internet, the predictions for this technology have been rather optimistic for the past 30 years. Multicore, and other near-term developments such as the Internet of Things, a smarter planet, and cloud computing, are well-described elsewhere. Rather I'm going to argue for a future not often addressed.

### Envisioning Computing Applications

When considering computing's future, it's important to keep in mind that its basic functions are numerical calculation and bookkeeping. Looking back at how people have envisioned computing in the past, it's instructive to note that it helps to have some historical knowledge but also that it's important to see how basic functions could prove useful beyond current applications.

It's widely alleged, but without support, that the founder of IBM said in 1943 that the world only needed five computers. As Gordon Bell has pointed out, in a 1997 ACM anniversary talk,<sup>1</sup> were that quote actually made then, it would have been a good prediction for a decade (Bell was technically wrong that no computers existed in 1943, but the only general-purpose programmable digital computer in existence was the binary electromechanical Z-series built by Konrad Zuse in Berlin). More telling, British computer scientist Douglas R. Hartree said in 1951 that the world would only need three computers.<sup>2</sup>

Doug Engelbart had already invented modern

computing when commercial computers were still primitive: he started working in 1962 on the system that his team demonstrated in 1968 (<http://sloan.stanford.edu/MouseSite/1968Demo.html>). He didn't just envision what more computer power might enable, he and his team demonstrated a prototype. Some of his ideas are still waiting to be implemented, such as computing with concepts. It took until the early '90's to get graphics with hypertext links, which he had demonstrated almost three decades earlier.

Engelbart had the benefit of 10 years of seeing fast computer growth, which let him abstract away the still primitive state and expense of computers and foresee the potential of computers that were many times more powerful and cheap.

More importantly, Engelbart was much better at imagining applications. Hartree limited his vision to applications that involved differential equations. Engelbart envisioned an office work prosthesis, mostly based upon bookkeeping of types of things and links between them, with some graphics thrown in for ease of human interaction.

Even today, computers are still good at performing numerical calculations, which are as useful for graphics as they were for ballistics in the early days of computing. Throw in some maps, and you can compute paths from one place to another. Perhaps more importantly in the long run, computers are good at bookkeeping, which is, in essence, sorting and tracking things and making decisions based on the results. It's a very powerful idea. So, the first success of computing, after ballistics, was in counting, sorting, and then financial systems.

Bookkeeping turned out to work for more than just accounting. If you network some computers together, you get modern airline reser-

vation and credit-card systems. If you're really clever, you can use a kind of bookkeeping to prove math theorems. Lots of people have come up with all sorts of clever ideas, some with good business models, for using computers to calculate and keep books for all kinds of crazy things that were difficult to foresee.

So, we must imagine new uses for numerical calculation and bookkeeping. We know from past experience that we shouldn't limit our imagination to current applications, and we especially shouldn't underestimate the potential to sort and track: bookkeeping.

### The Fall of the Computing Politburo

There's something else going on that combines sociology and economics with both computer and human networking, and even has a political analogy.

Before I talk about network computing, let me be the first to admit that I never thought personal computers would catch on. When Apple came out with those, I was very familiar with mainframe operating systems. I knew that, for the foreseeable future, personal computer users would spend an unreasonable amount of their waking time managing the software on these little machines. Why would reasonable people do this when they could just let the mainframe folks do it for them?

I failed to understand how much people didn't want to be controlled by a central authority, how much they'd enjoy personalizing their own system, and how much more functionality would develop once systems were freed from the control of a few people, much the way that the breakup of the phone company allowed new functionality to flourish (cloud computing might be a better answer than personal computers when customized for personal computing).

This is analogous to the failed

communist models of planned economies. The plans benefited the implementers, not the customers. Mainframe systems were often built to suit the guys who programmed them. When people are empowered to some extent to become programmers, making their own choices, the central planning model gets left in the dust. That lesson gets extended to networked computers.

Few people foresaw how the open Web would change the world. Engelbart demonstrated sophisticated hypertext in 1968. It became openly available in 1992 along with the public Internet and email. It took only a few more years for advanced software companies to abandon their investment in interactive TV as the wave of the future. Still, some engineers thought TCP/IP was too flaky and that everything would be based on the seemingly more reliable ATM protocol. That notion died an early death. In the middle of 1995, Bill Gates was forced to adapt Internet standards for Microsoft systems.<sup>3</sup> That's a relatively short time to change conventional thinking and business models, once the Internet went public.

Tim Berners-Lee saw not only how useful an open Web would be (unlike his co-inventor, Robert Cailiau),<sup>4</sup> he also saw how it wasn't just a medium for broadcasting but also for mass information sharing, which didn't become a reality until this century with the so-called Web 2.0. But then it exploded as fast as the original Web did. Nowadays, only old-fashioned publications, such as this one, don't publish blogs that make it easy for users to give immediate feedback.

This insight about the value of open networks can be further characterized. An informal theory I published in 2005 is that emergent collectives (ECs) are a kind of network technology that can explode and disrupt existing business models.<sup>5</sup>

### Emergent Collectives

ECs are comprised of a network of interlinked valued nodes and are an easy way to add nodes by distributed actors so that the network will scale. Both node content and links are governed by a set of standards and protocols that provides these properties. You don't have to ask someone to provide novel value, you just follow the specifications and link a new node according to the protocols.

There might be a central platform supporting the network, but it's the protocols that govern the system, perhaps with some central censorship, preferably following node addition.

Second, such networks should be very error-tolerant if not homeostatic so that people can be both sloppy and inventive about what nodes they add. Local errors will have minimal network effects. Third, and most important, people should want to contribute nodes.

In the history of Internet technologies – not just the Web but also various types of servers and the Web's predecessors such as Gopher – people have added nodes more often than not for absolutely no economic benefit. In fact, at least one study shows that economic incentives have a disincentive effect.<sup>6</sup> Rather, people have other motivations, perhaps to be famous for a minute, or to be part of something bigger. Certainly there must be some noneconomic incentive for people to do so much in building a Web of information that's so useful and on which so much else has now been built.

The World Wide Web is an obvious example of an EC, as are most Internet technologies. I know a good nonexample as well.

I was the original editor of the World Wide Web Virtual Library (WWW VL; <http://vlib.org>) for *Mechanical Engineering* (which seems now sadly defunct). I did this work because I wanted to contribute to something larger. I lost interest when

it seemed that too much focus on human governance made it unlikely to catch on. Contributors would submit sites to a human editor who had to manually edit the site, which he or she owned. This alone prevented the WWW VL from being an EC. And the internal politics of the WWW VL detracted from the feeling of being part of something larger.

Years later, Wikipedia became an EC built on the Wiki protocols that achieved better functionality with less editorial oversight but much more openness. Even though it's easy to critique the content, Wikipedia has in fact been successful because it's an error-tolerant network to which people find it easy to add nodes.

The original Napster network was another example of an EC. It was pushed underground by the old business model it disrupted, but the old business model never fully recovered, and there are lots of peer-to-peer (P2P) networks still running out there, also obviously ECs. If you think of APIs as standards and links to internal functions as protocols, then smart phone apps can also be seen as an EC. In any case, the lesson is that the easier it is to add nodes, the bigger the network will be.

MySpace is a good example of an EC that lets nontechnical teenagers customize their home pages by providing good ways for them to cut and paste functionality that would have otherwise required sophisticated user interface programming. MySpace makes it easy for people to add highly customized nodes and links to the network.

This tells us what's going to happen in the next 10–20 years. Clearly there will be a trend of networked applications that let people essentially program the systems in ways that they care about. Customization will be king, on top of standards and protocols that allow customization by people who aren't deep programmers. Actual code-based program-

ming will be driven deeper into the computing layers.

Now, I can speculate about what kinds of systems these might be. Notice what I'm not speculating about. Massive attempts to program parallel computing have failed in the past 30 years. Semantics and logic programming have never caught on, but there are existence proofs that they could, and many of us have described in detail what might happen in this space. The Internet of Things will probably create massive new opportunities. But I decline to speculate more on these developments. Rather I want to say something very specific about what might develop based on the notions of the fundamental book-keeping capabilities and ECs if raw computer power, inferential power, and semantic standards continue to develop as expected.

### Everyone is a Service

I've made many predictions in this space about how we might end up with a lot of networked services. Marty Tenenbaum predicted a "sea of services" back in the late '80s, which still has yet to arrive but is now obviously possible. But now I want to be more specific.

I do so by revisiting a prediction I made in the late '80s. I said that along about now, most companies would have fewer employees, and most people would be self-employed. I wish some copy of this internal memo had survived somewhere. It was roundly denounced at the time as "just more expert systems." Obviously, I wasn't sufficiently clear. But never mind, I will make an even more specific prediction now.

I had a technical rather than an economic reason for thinking that the world would outsource. I was sure that by, say 2010, we'd have made a variety of tasks into commodities that would have standardized descriptions, subject to machine inference, so that we could match our skills and

current capacity to outstanding tasks and bid for them: like a Craigslist of job descriptions as a super temp agency, with bidding on eBay. That hasn't quite happened yet, but it's clear that there's a strong economic motivation for distribution of work, just as there was for the failure of planned economies and mainframes.

The task and skill descriptions I had in mind bore a strong resemblance to the kind of Semantic Web services we seem so close to achieving (but yet remain so far away.) If we look at human services today, we can see different types that require different descriptions.

If the skill set required or offered is artistic or creative, such as a rock and roll drummer, personal assistant, or semantic researcher, then the standardized description is a filter for a set of candidates that require a human interview. Temp agencies offer commodity skills, such as bookkeepers or receptionists, that can have a standardized description sufficient for hiring. In this case, the task and skill set tend to converge: be a receptionist for 10 days or prepare the income taxes for a typical small business. These begin to look more like services capable of being described in such a way that the demand and capabilities could be matched with little human intervention.

More interesting are commodity tasks that can be done remotely but which imply some ongoing service. For instance, setting up a call center at home also can be standardized at the outset, but monitoring the service provision might require a human to review the tapes. Insurance requirements have become very standardized, to the point of being an automated service with good semantics, at least for the bid and acceptance. The semantics are less clear for the ongoing service, when there's a claim, for instance. Then the definition of wind versus water dam-

age might require a political interpretation, until we agree on formal semantics for the fine print.

Some services have very much become commodities, where even if there's a process, monitoring it can also be a standardized service, such as shipping, airline travel, book buying, and car rental. Computers might perform some services entirely, such as stock trades and bank transfers. Such services can be fully automated in both demand and capability matching and provision. In many cases, we're just waiting on the right kinds of security and quality of service mechanisms for these to become standard Semantic Web services, possibly supplied by third-party services, for which the business model hasn't been developed yet.

In all cases, whether the service is done by a computer or a computer program, the more of a standardized description of both requirements and functionality, and the more semantics, and the more sophisticated the matching algorithms for matching demand to supply, the less "friction" and the more automation. Then, a computer program can put these services together to achieve a larger goal as well as track and coordinate them. Fortunately, most tasks and services tend to become commoditized, even as new cycles of innovation begin.

### Coordinating Services

I predict, in the next 10 years, people will assemble their collective skills and current tasks to form "flash companies" for nonroutine work composed of commoditized tasks and skill sets that's currently done by large, permanent companies, and also work that isn't yet possible. Auctions will be the basic pricing model. Contracts among individuals will be simple, comparable to tearing the wrapper off purchased software today. There will be no HR. Large firms won't have the efficiency to

compete with on-demand assemblies of service providers.

Affinity groups and social networks will be important. If you've ever worked in a large company, you know there's the official process for getting something done, and you know that the best way to get things done is to go out-of-channel and ask your buddies for a favor. The new formal coordination support would take this into account and become the new best practice as people route work to their friends.

This will dramatically decrease the time and cost of complex projects, especially ad hoc ones. Let me trot out my old example of building construction. Right now, in the office of every contractor on a large project is a bulletin board of 200 Post-its, the largest number the contractor can handle, of changes and notices that the contractor must convey to other people along with what it means to them. The contractor has no computer support for this kind of change notification.

And the contractor's workflow is a project plan, based on the critical path method, which is irrelevant in about two weeks because of all the changes. Thus, large projects take a lot of time and are expensive. A lot of mistakes are made, and there are surely complex projects we just can't do today by muddling through somehow.

Providing flexible distributed workflows and coordination for distributed tasks will revolutionize not only construction but all of the large non-routine endeavors of mankind. We'll be able to do things we can't imagine now because we're too constrained by the technology we use today to manage complexity. Much less are we able to perform adequate knowledge archiving because so much of the knowledge is informal process-based information that isn't captured, which is why we always have to start from scratch, instead of, for example, just restarting the Saturn V program.

For this vision, we need computer-mediated coordination of the kind I've described previously.<sup>7</sup> In essence, this is just various kinds of book-keeping happening over a network where individuals customize individual task nodes that comprise the collective work. A proactive network will support individuals by tracking changes in the work and the conditions that affect it, and this support will empower them to change the process as desired. It's just book-keeping, and it can be done with an EC that will let people know how their individual task completions affect each other, and coordinate the changes, enforcing Pareto optimality and preventing cycles of distributed change, for instance. This vision will happen because there's a strong economic basis for it.

There may be various motivations to belong to a large company, such as a feeling of security or, in the US, the bizarre situation that having health insurance depends upon employment by a large company. But there are powerful market forces for distributing work: cutting costs while improving productivity and even capabilities.

There will always be a need for some large companies to perform routine large jobs, such as coal mining. But for many of the enterprises active today, having large companies perform them simply implies a lack of coordination technology, to be developed in the near future.

### Plenty of Room for Research

Some coordination ECs will develop naturally in an ad hoc manner on the Web. We can see nascent coordination ECs happen as we build travel Webs to let us know when our colleagues will visit the same city and as smart phone apps let us know when our friends are nearby. Craigslist will develop a folksonomy of skill sets and someone will develop a phone app with a standard way of matching task



and skill requirements. Some version of this vision is inevitable.

However, for large project coordination, we need a more disciplined set of standards and protocols. Providing this technology is a challenge to the computer science community. Let me sketch out one near-term implementation: the “world wide workflow.” This would be a semiprescribed workflow that consists of packets of task descriptions shipped to qualified providers who have won the bid to do the work. These tasks would essentially be semantic services.

Each packet would let the provider not only perform the work in the most flexible manner, but also modify the control part of the packet, and thus the workflow, as needed. There might be some workflow monitors and some way of handling conflicts between preferred constraints and what the individual providers want to change. But it will be dynamic and conformable by individual task performers. Packets and status notices will be distributed by the communications channels of choice at the time. Competing providers will provide the overall platform as a metaservice.


This is but one way this could happen to provide the fundamen-

tal service of automatic coordination of an active network that will let people know how they’re affected by other people’s work decisions, as well as exogenous contingencies. Coordination can be augmented by process synthesis on demand, as well as some projection of effects of changes, perhaps with simulation, so that individuals, as well as the collective, can see what kind of processes could result from various task and workflow modifications. We don’t yet know the best way to do this: it’s going to take research in *coordination engineering*.

I mentioned more requirements and issues about this kind of coordination previously.<sup>7</sup> Some of the issues are important and have no clear answer. Most have to do with understanding the dependencies among tasks, in the abstract, the best data structures and algorithms for managing them in a distributed environment, and the best methods for providing navigation notices to the distributed task performers in order to design, configure, and build complex systems. Is ensuring only Pareto optimality the best we can do? What is the most elegant representation of dependencies that can be extended in task-specific ways? There are many open questions in process synthesis, simulation, and look-ahead as well. I invite the computer science community to investigate the fundamental issues of coordination engineering.

Not only will distributed ECs take over work that large firms are doing today, they’ll also do work that’s not even possible today. There’s plenty of room for self-employment outside the firm, out in a new space of ECs with coordination links, supported by a new kind of bookkeeping. And there’s plenty of room in the research space for enabling this coordination space.

We’re already building some

of the pieces of this vision today. Achieving some version is inevitable, and doing it well could be necessary for our survival. 


**Acknowledgments**

Thanks to Christoph Bussler, Paul Hofmann, Michael Huhns, and Munindar Singh for valuable comments on very early drafts of this column.

**References**

1. G. Bell, “The Laws of Predictions,” talk at Association for Computing Machinery Conf. (ACM 97), 1997; <http://research.microsoft.com/en-us/um/siliconvalley/events/acm97/gbnovid.ppt>.
2. L. Bowden, “The Language of Computers,” *Am. Scientist*, vol. 58, 1970, pp. 43–53.
3. K. Rebello, A. Cortese, and R. Hof, “Inside Microsoft,” *Business Week*, 15 July 1996; [www.businessweek.com/1996/29/b34842.htm](http://www.businessweek.com/1996/29/b34842.htm).
4. C. Petrie, “The WWW Proposal: How It Really Happened,” *IEEE Internet Computing*, Jan./Feb. 1998; [www.computer.org/ic-cailliau/](http://www.computer.org/ic-cailliau/).
5. C. Petrie, “Emergent Collectives for Work and Play,” *Société de Stratégie, AGIR Revue Générale de Stratégie*, Nos. 20–21, Jan. 2005; [www-cdr.stanford.edu/~petrie/revue](http://www-cdr.stanford.edu/~petrie/revue).
6. M. Krakovsky, “Hotel Case Study: Peer Pressure’s Impact on the Environment,” *Scientific Am.*, Nov. 2008; [www.scientificamerican.com/article.cfm?id=hotel-case-study](http://www.scientificamerican.com/article.cfm?id=hotel-case-study).
7. C. Petrie, “Collective Work,” *IEEE Internet Computing*, Mar./Apr. 2008, pp. 80–82.

**Charles Petrie** has been a senior research scientist at Stanford University since 1993. His research interests include concurrent engineering, virtual enterprise management, and collective work. Petrie has a PhD in computer science from the University of Texas at Austin. He is EIC emeritus and a member of *IC*’s editorial board. Contact him at [petrie@stanford.edu](mailto:petrie@stanford.edu).

 Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.