Spring 2003

The BRIDGE

LINKING ENGINEERING AND SOCIETY

NATIONAL ACADEMY OF ENGINEERING
OF THE NATIONAL ACADEMIES

Promoting the technological welfare of the nation by marshalling the knowledge and insights of eminent members of the engineering profession.

Volume 33, Number 1 • Spring 2003

The BRIDGE

LINKING ENGINEERING AND SOCIETY

Editorial

Features

NAE News and Notes

(continued on next page)

Rapid changes in computing will continue for the foreseeable future.

Computing Meets the Physical World

Butler Lampson is Distinguished Engineer at Microsoft Corporation. He is an NAE member.

Butler Lampson

The field of computing has always changed rapidly, and it is still doing so. The changes are driven, more than anything else, by Moore's law. Many people think the pace of change is slowing, or even that because we already have the Internet and Google, there is not much left to do. I hope these papers will convince you that this view is entirely wrong.

For the last 50 years, new applications of computers have followed a pattern, as one manual activity after another has become automated. In the 1940s, it became possible to automate the calculation of ballistic trajectories and in the 1950s of payrolls and nuclear weapon simulations. By the 1970s, it was possible to create reasonably faithful representations of paper documents on computer screens. In the 1990s, we had the equivalent of a telephone system for data, in the form of the Internet. In the next two decades we will have embodied computers, machines that can interact with the physical world.

Hardware and Software

The factor that determines whether or not an activity can be automated is whether the hardware is up to it. According to Moore's law, the cost performance of computers improves by a factor of 2 every 18 months, or a factor of 100 every 10 years; this applies to processing, storage, and communication. Moore's law is not a law of physics, but it has held roughly true for several

decades and seems likely to continue to hold true for at least another decade. Indeed, today some things are developing much faster than that. Storage capacity, for example, is doubling every 9 months, not every 18 months. Wide-area communication bandwidth is also improving faster than Moore's law. Sometimes, with speech recognition and web search engines, for example, the cheaper cycles or bytes can be applied directly. Often, however, by spending more hardware resources, we can minimize programming effort; this is true for applications that use web browsers or database systems.

Hardware is the raw material of computing, but software gives it form. Our ability to write software is limited by complexity. People have been complaining about the "software crisis" at least since the early 1960s, and many people predicted in the 1960s and 1970s that software development would grind to a halt because of our inability to handle the increasing complexity of software. Needless to say, this has not happened.

The software "crisis" will always be with us, however (so it isn't really a crisis). There are three reasons for this:

- As computing hardware becomes more powerful (at the rate of Moore's law), new applications quickly become feasible, and they require new software. In other branches of engineering the pace of change is much slower.
- Although it is difficult to handle complexity in software, it is much easier to handle it there than elsewhere in a system. Therefore, it is good engineering to move as much complexity as possible into software, and engineers are busily doing so.
- External forces, such as physical laws, impose few limits on the application of computers. Usually the only limit is our inability to write programs. Because we have no theory of software complexity, the only way to find this limit is by trial and error, so we are bound to overreach fairly often.

A lot of software today is built from truly gigantic components: the operating system (Windows or Linux), the database (Oracle or DB2), and the browser (Netscape or Internet Explorer). These programs have 5 million to 40 million lines of code. By combining them with a little bit of new code, we can build complex applications very quickly. These new applications may use a hundred or a thousand times the hardware resources custom-built programs would use, but they can be available in three months instead of five years. Because we have plenty of hardware resources, this is a good way to use them. It is programmers and time to market that are in short supply, and customers care much more about flexibility and total cost of ownership than about the costs of raw hardware.

Another way to look at this is that today's PC is about 10,000 times bigger and faster than the 1973 Xerox Alto, which it otherwise closely resembles (Thacker, 1988). A PC certainly doesn't do 10,000 times as much, or do it 10,000 times faster. Where did these cycles go? Most of them went into delivering lots of features quickly, which means that first-class design had to be sacrificed. Software developers traded reductions in hardware resources for shorter time to market. A lot of cycles also went into integration (for example, universal character sets and typography, drag and drop functions, spreadsheets embedded in text documents) and compatibility with lots of different hardware and lots of old systems. Only a factor of 10 went into faster responses.

Applications

There have been three broad waves of applications for computers, about 25 years apart (Table 1). Currently, the communication wave is in full flood, and the first signs of embodiment (relatively unrestrained interactions with the physical world) are starting to appear. Of course the earlier waves do not disappear, simulation continues to be an important class of applications.

TABLE 1 Applications for Computers

Category	Starting Date	Examples
Simulation	1960	Nuclear weapons, payroll, games, virtual reality technology
Communication (and storage)	1985	E-mail, online airline tickets, books, movies
Embodiment	2010	Vision, speech, robots, smart dust

Usually a computer application begins as a fairly close simulation of a manual function. After 10 or 20 years, people begin to explore how the computer can do the job in a radically different way. In business, this is called "business process reengineering." The computer no longer does the same things as a bookkeeper; instead, it makes it possible to close a company's books two days after a quarter ends. Boeing builds airplanes in a very

different way because computers can model every mechanical detail.

The earliest computers in the 1950s were used for simulation. Simulations of nuclear weapons, astrophysics, protein folding, payrolls, project scheduling, games, and virtual realities all fall comfortably into this category.

The next great wave of computer applications, embodiment, is just beginning.

The communication wave became apparent outside of research laboratories around 1980, and we are now in the middle of it. Today, we have e-mail, search engines, and the ability to buy airline tickets, books, movie tickets, and almost anything else online. TerraServer, gives us access to publicly available satellite telemetry of the world. The Library of Congress' catalog is online, and you can buy any one of a million and a half books on Amazon.com. Conduct a search on Google today, and in half a second you can research a database of about 3 billion pages that is updated every two weeks—and will soon be updated in real time.

The next great wave, which is just beginning, is embodiment. Of course, computers have been used in process-control systems for a long time, but that is comparatively uninteresting (albeit of considerable economic importance). We are now seeing the first computer systems that can function effectively in the real world—computerized cars, robots, smart dust. They are still in their infancy, but the most interesting developments in computing in the next 30 years will be in this domain.

A Boston company called iRobot has just introduced what seems to be the first plausible domestic robot, a vacuum cleaner that crawls around a room in a vaguely spiral pattern, bouncing off of things (see it at *www.roombavac.com*). The price is $199. In fact, with only 14k bytes of ROM and 256 bytes (not kilobytes) of RAM, it's barely a computer.

What's Next?

In a recent paper, Jim Gray (2003) countered the widely held perception that most of the important developments in computing have already happened and that the future holds little more than refinements and cost reductions. Gray predicted that the next 50 years would be much more exciting than the last 50, both intellectually and in practical applications. Here are some of the challenges he raises.

Win the impersonation game. The classic Turing test asks whether a person sitting at a keyboard and display can distinguish between a conversation with a computer and a conversation with another person. To win, roughly speaking, a computer must able to read, write, think, and understand as well as a person. The computer will need some facility with natural language and a good deal of common sense. Anyone who has tried using natural language to interact with a computer knows that we still have a long way to go; and we don't even know how far.

Hear, speak, and see as well as a person. Meeting this challenge will be much more difficult. Today's best text-to-speech systems, given enough data, can do a pretty good job of simulating a person's voice, although they still have trouble with intonation. In a quiet room, you can dictate to a computer a little faster than a person can type, at least if, like me, the person types fairly fast but makes a lot of errors. If there is any background noise, however, the computer does much worse than a person. To see as well as a person is even more difficult. People first learned to parse two-dimensional images on the retina and construct a model of a three-dimensional world so they could detect tigers in the jungle and swing from tree to tree. Today's best systems do a fair job of recognizing buildings on a city street, but not in real time.

Answer questions about a text corpus as well as a human expert. Then add sounds and images. A computer can't yet read and absorb Google's 3 billion web pages and then answer questions about them in a sensible way. It can find documents where words occur or documents with a lot of other documents pointing to them, but it can't understand content.

Be somewhere else as observer (tele-past), participant (tele-present). Videoconferencing represents the first feeble step in this direction. Can virtual presence equal real presence? We don't really understand what makes real presence good, so this is an open question—one that has implications for medicine, transportation, education, and social relations. Remote surgery is just one valuable, but extremely demanding application.

Devise a system architecture that scales up by 10^6. Computer systems on the Internet often serve millions of users, sometimes hundreds of millions, and the demand can change rapidly. After September 11, for example, the main web news sites collapsed because traffic was 10 to 100 times higher than normal. In addition, the same architecture must be used across a wide range of systems to ensure compatibility and consistency. The Internet has met this challenge for transporting data, but storage, processing, and coordination over such a range of sizes are problems yet to be solved.

Given a specification, build a system that implements it. Do it better than a team of programmers. Writing an adequate specification is a daunting task, as anyone who has tried it knows. Automatically building a system to implement it means converting it into a form that can be executed reasonably efficiently. Most teams of programmers don't do this terribly well, so if we can create a system that can do it at all, we have a good shot at doing it better than a team of programmers. Today, we can do it in very limited domains; the canonical examples are certain spreadsheet and database query applications, in which the specification and the program are almost indistinguishable.

Build a system used by millions that can be administered by half a person. The operating costs of most computer systems dwarf their hardware costs. Configuration, backup, repair, expansion, and updates require a lot of human attention. There is no reason in principle why this work can't be done by machines, except for the small effort required to set policy (e.g., telling the system who the authorized users are and which tasks are most important).

Common Themes

Three themes common to these challenges are central to the way computing will develop in the next few years and decades: information, uncertainty, and ubiquity.

Information. Very soon it will be technically feasible to put everything we have online and remember it forever. But making the most of this capability will require that the information be meaningful to the machine in some sense. Even though today's web is feeble by this standard, it has already had a tremendous impact on our lives. Machines that can answer questions about the information they store and relate different pieces of information to each other would be able to do much more for us.

Uncertainty. Interacting with the physical world necessarily involves dealing with uncertainty. The computer needs a good model of what can happen in the part of the world it is interacting with, and boundaries that tell it when the model no longer applies. This is often called common sense, and it is essential not only for sensors and robots, but also for natural user interfaces, such as speech, writing, and language. For each of these, the machine often has to guess meaning; it needs to guess well, and the user needs to know what to do when the guess is wrong.

Ubiquity. Computers are getting so cheap and so small that we can begin to think about having a computer on every fingernail, a computer inside every manufactured physical object. We could have guardian angels, for example, that monitor the state of our health and safety, call for help when it's needed, and so forth. Every manufactured object in the world could respond to us and interact with its fellows. How can all of this be done reliably and conveniently? How can people tell all of these computers what to do?

These are just a few examples of the opportunities before us. The papers that follow focus on physical ways computers might interact with the world.

References

Gray, J. 2003. What next?: a dozen information-technology research goals. Journal of the ACM 50(1): 41–47.

Thacker, C. 1988. Personal Distributed Computing: The Alto and Ethernet Hardware. Pp. 267–290 in A History of Personal Workstations, edited by A. Goldberg. Reading, Mass.: Addison-Wesley.