INTERNET VOTING

The February article on Internet voting totally ignores one of the major problems with any kind of e-voting (G.E.G. Beroggi, "Secure and Easy Internet Voting," pp. 52-56). While it describes efforts to ensure data security, a discussion of the issue of physical security was missing.

What is to prevent an abusive spouse from forcing an "acceptable" vote? Where are the safeguards preventing a domineering employer from requiring a "friendly" vote as a condition of employment? How do you prevent a ward boss from buying votes or just the PIN numbers?

Yes, these abuses are illegal, but with no third-party observation of the act of voting, they are all too likely to occur. And, yes, I realize that absentee ballots are subject to these problems as well. However, at this time, that is a rather small percentage of the electorate. Furthermore, it would probably be easy to detect clusters of questionable votes.

I can see no practical technological solution to this problem. Thus, universally applied Internet voting might bring many more serious problems than it solves.

Lyle Greg Lisle
L.G.Lisle@ieee.org

Giampiero Beroggi responds:

Switzerland introduced absentee voting by postal mail in 1994, which back then was also very controversial for the same concerns offered here regarding e-voting.

Despite these initial concerns, up to 80 percent of the votes cast for the federal parliament elections in 2007 were absentee ballots sent by postal mail. In fact, the Swiss population and government have had positive experiences with absentee voting by postal mail over the past 13 years. Thus, they have strong confidence in their voting system, both for its technology as well as for the voters' maturity to freely express their opinions.

The same should also hold for e-voting in the near future.

giampiero@beroggi.net

DIGITAL EVOLUTION

In reading "Harnessing Digital Evolution" (P. McKinley et al., Jan., pp 54-63), I was particularly interested to observe the lack of attention paid to early work on learning intelligent systems.

In the 1970s, evolution in adaptive artificial intelligence systems was applied by the University of Canterbury group run by John Andraeus, which developed PURR and PUSS and subsequently demonstrated that they could learn any algorithm a Turing machine could compute. In the 1980s, computer viruses had limited evolutionary capabilities, and the theoretical results showed that a virus could evolve any algorithm that a Turing machine could compute. In the computer virus arena, experimental systems were used to evolve toward network computation optimization in the late 1980s and early 1990s.

I am highly supportive of the interest in evolutionary digital systems, but we stand on the shoulders of giants, and we should pay proper homage to their efforts in putting our own in context.

Fred Cohen
dr.cohen@mac.com

Philip McKinley responds:

We certainly agree that our research activities are supported by a foundation of excellent research in evolutionary computation, machine learning, robotics, and related areas conducted over the past several decades by many individuals and research groups. We highly value these important contributions. Unfortunately, it was infeasible to review all related research in our article. Instead, the two sidebars attempt to provide a basic background for the interested reader, including references to general works in evolutionary computation and artificial life, pointers to relevant conferences and journals, and a listing of a subset of the groups conducting related work.

mckinley@cse.msu.edu

THE SLIDE RULE AND THE STETHOSCOPE

In "The History of the Computing Profession" (The Profession, Jan. 2008, pp. 112, 110-111), Neville Holmes offers some constructs that might prove helpful to me in my work. I would, however, like to raise a question about the analogy between the slide rule and the stethoscope. Being a physician as well as a computer scientist, this analogy does not work for me for the following reasons.

The slide rule is a computational device. Even when it was the best device available, it suffered from some recognized problems:

- the accuracy was directly proportional to the length,
- the final result often depended on interpolation,
- there was a possibility of uneven expansion of the elements as the ambient temperature increased,
- it was slow (relatively),
- it could not produce a hard-copy record of its results, and
- understanding and utilizing all of its capability required training.

The slide rule's advantages include portability and low cost.

As newer computational devices have become available, abandonment of the slide rule was inevitable. In fact, engineers abandoned it with striking rapidity.

The stethoscope, on the other hand, is a transducer; the human ear is the computational device. The stethoscope can be an imperfect transducer because

- the frequency response differs depending on which style of diaphragm or bell is employed and the pressure applied, and
- the tubing can introduce sound artifacts.

The human ear also has its shortcomings, most notably the loss of high-frequency response with age.

The stethoscope is used primarily for the qualitative assessment of the functional state of various internal organs. It is true that for specific conditions and scenarios, it might be preferable to use alternatives such as ultrasound, x-ray, or a CT scan. However, the availability of these alternatives does not invalidate the information that can be gathered with the stethoscope nor can they completely replace it.

The most recent demonstration of the continuing importance of the stethoscope is the difference in systolic blood pressure detected when measured with a fancy automated fuzzy-logic device versus a mercury sphygmomanometer and a stethoscope in a patient with a slow heart rate. The automated device read 98 mmHg systolic; the physician/stethoscope detected the first Korotkoff sound (the true systolic pressure) at 113 mmHg (http://en.wikipedia.org/wiki/Korotkoff_sound).

With sufficient contemplation, I might be able to come up with a better medical analogy to the slide rule, but so far I haven't. So my conclusions are: (1) the analogy offered does not work, and (2) the article would not suffer at all, and might be less distracting to nitpickers like me, if the analogy sentence was simply omitted.

Daniel Essin, MD
essin@ieee.org

Neville Holmes responds:

Thank you for your comments and for your interesting observations on the role of the stethoscope, in particular its continuing importance given the publicity accorded recently to fancy replacements based on digital technology.

However, I had not intended my likening of the stethoscope to the slide rule to be taken as referring to its technical function. Rather, just as the wearing of a stethoscope is still seen popularly as de rigueur for a doctor, so in earlier years was the wearing of a slide rule visibly projecting from an engineer's pocket. When I started my study of engineering in the early 1950s, I was required to bring a slide rule along on the first day, and it was used throughout my course and taken into the end-of-year examinations.

neville.holmes@utas.edu.au

THE IDEAL COMPUTER

I have been programming since the Stone Age of computers when their speed was 100 operations per second, and I fully agree with Simone Santini's observations in "Making Computers Do More with Less" (The Profession, Dec. 2007, pp. 124, 122-123).

The main problem is that computers, which originally were computing devices, are now designed and sold primarily as entertainment or business tools because that's where manufacturers make most of their money. The scientific users are secondary—they're supported, but they have to pay for the excesses other people want.

In "The Inevitable Cycle: Graphical Tools and Programming Paradigms" (*Computer*, Aug. 2007, pp. 24-30) my coauthor and I described the cycle of using graphical tools in programming. I think we are at the peak of this cycle, and the use of pictures and possibly color in programming should fade out soon.

Standard libraries in all OO languages force programmers to write code that is inefficient in execution speed and data footprint and has overhead in allocating additional objects because it uses array-based collections.

All this happened only because the templates (or generics) do not support two or more cooperating classes. We described a temporary solution in *Dr. Dobb's Journal* ("Reusable Associations"; www.ddj.com/architect/202401093). This topic also was the subject of an OOPSLA 2007 workshop (www.codefarms.com/OOPSLA07/workshop).

Jiri Soukup
jiri@codefarms.com

Simone Santini responds:

It is true that computer companies make most of their money selling their computers as business and entertainment tools. It might be true that you do need a 2-GHz CPU to run a videogame, but I think most business applications could be drastically simplified and run on a computer and an operating system like those I envision in my article. You don't need a 2-GHz CPU and 5 Gbytes central memory to run a word processor, a spreadsheet, and a presentation program. VisiCalc ran in a machine with 64 Kbytes of RAM. I'm not saying we should go back to its limitations, but I don't see any reason why a spreadsheet should not run in, say, 10 Mbytes of memory.

I agree that standard libraries in most OO languages are poorly made. The problem is that, unlike libraries in functional or structured languages, in OO there is a strong temptation—which designers apparently can't resist—to impose a structure to the program that uses the libraries. Many times, in order to use a library, you have to make your objects part of a hierarchy defined by the library, you have to inherit from some library object, and so on. This means that the application designer is not completely free to choose the best structure for the problem at hand and this, I believe, has a strong influence on the poor quality of the software that is being generated.

Unfortunately, it seems that the current generation of programmers is being trained to make heavy use of libraries. I have seen many young programmers who, faced with a new problem, start immediately searching the Internet for all possible libraries that can approximately cover any part, no matter how small, of the problem. Instead, they should begin by analyzing the structure of a solution and later—but only later—looking for possible libraries that can be used without disrupting the structure they have designed.

simone.santini@uam.es

We welcome your letters. Send them to computer@computer.org.