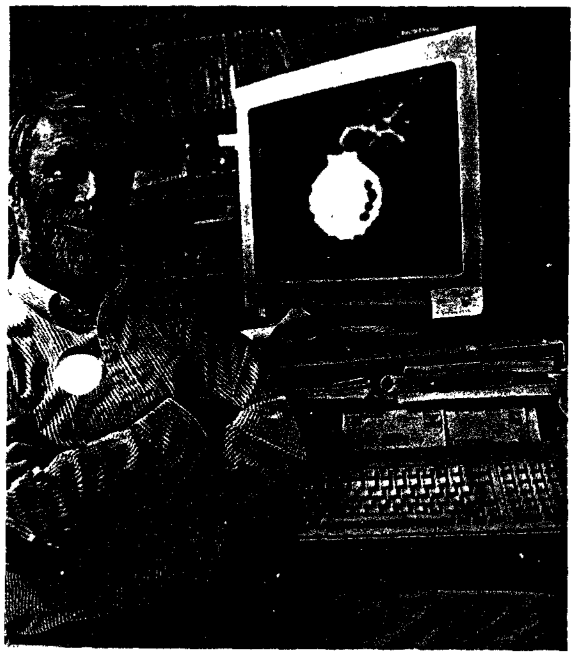
The Dean of Disaster

Plane crashes, nuclear reactor accidents, explosions at chemical plants – if computers were at fault, Peter Neumann knows all about it.

By Simson L. Garfinkel

He's a fantastic storyteller, he's always ready with a pun, and he can play two recorders at once – simultaneously piping out both melody and accompaniment – while he beats the rhythm with his foot. But to hundreds of thousands of people around the world, Peter G. Neumann is best known for moderating RISKS-Forum, one of the Internet's most widely read electronic forums.



What are computer RISKS? Any use of computers that might accidentally lead to loss of life, property, or money. They are dangers as simple as sending credit card numbers by e-mail (which could bounce into unauthorized hands) and as deadly as bugs in medical equipment. Disasters are a mainstay, including numerous plane crashes, nuclear reactor accidents, and explosions at chemical plants – all brought about, in part, by faulty computer systems.

ou put in more omplexity trying ɔ add reliability, nd that complex- y itself is sus- ect, and hence ɩore risky.

Over the years, the readers of RISKS have cast a wide net, sending contributions to Neumann on everything from space missions that have been scrubbed because of typos to the risks of remote-control garage-door openers and answering machines. RISKS readers are big on privacy: Some of the earliest descriptions of the National Security Agency's (NSA) proposed Clipper encryption chip appeared in RISKS-Forum, quickly followed by technical, social, and political discussions about the dangers posed by the government-sponsored encryption standard.

Unlike other online forums, RISKS maintains a consistently high level of discussion and a low level of noise. "It's a forum of discussion that doesn't just run wild and rampant," says Dorothy Denning, chair of computer science at Georgetown University. Equally impressive is the number of postings from highly respected members of the computer science community. "It's a very good source of information," Denning says.

Besides the mailing list, Neumann edits the journal *Software Engineering Notes* and has a monthly column on the last page of *Communications of the Association for Computing Machinery*, the journal of the ACM. He's also putting the finishing touches on a book about software safety and risks. Its tentative title? "*RISKS: The Book* – as opposed to RISKS the movie and RISKS the game," Neumann jokes.

Neumann got his start with computers in 1953 as an undergraduate at Harvard. There he worked on the Harvard Mark I – the same computer that was incapacitated by the first "bug" (a moth that flew into a relay). After earning a doctorate in applied mathematics at Harvard and a doctorate from the Technische Hochschule in Darmstadt, Germany, he headed Bell Labs's participation in the Multics project – one of the earliest attempts to build a reliable and secure computer system.

Working on Multics taught Neumann the futility of building risk-free systems: Every time he tried to design a system that had no weak links and no security flaws, new ones would appear.

Today, Neumann is at SRI International's Computer Science Laboratory in Menlo Park, California, where he has worked on numerous projects for government and industry. Despite his work in software safety, Neumann says that music is his life's great passion: In addition to playing piano, bassoon, and recorders, Neumann sings madrigals and is a trustee of the Greenwood Music Camp in Cummington, Massachusetts.

The RISKS mailing list started in 1985. At the time, some members of the Association for Computing Machinery's executive council wanted the ACM to go on record decrying then-President Reagan's Strategic Defense Initiative, or "Star Wars," as too risky. The idea didn't go over well with the rest of the members of the council. As a compromise, ACM's president, Adele Goldberg, asked Neumann to head the Committee on Computers and Public Policy and create a public forum for discussing risks to the public caused by the use of computers. "An online newsgroup seemed like the most effective way to do that," Neumann recalls. I caught up with Neumann by phone and e-mail and asked him about his favorite topic.

SG: How many people read RISKS?
PN: I wish I could tell you... It's clearly one of the most widely read Internet news groups. The answer is probably somewhere around 100,000, but I have no idea. I have no way of guessing. All I know is that I keep getting mail from people I've never heard of, and the distribution list keeps growing and growing.
SG: What are the risks of running a large mailing list?
PN: The biggest problem is the barf mail – fielding ten new

Deth to teh typos!

Nothing kills your image faster than looking dumb in print. But who has time to check every spreadsheet, e-mail message, and memo that gets created in a day's time? *Thunder 7*,™ that's who.

Thunder 7 is an extraordinarily clever spelling checker that works in *all* your Macintosh® applications and desk accessories—not just your word processor. *Thunder 7* watches your spelling as you type, alerting you the instant you make a mistake, and helping you fix it *right then*—not later, if you find the time.

Once you've started using *Thunder 7*, you'll probably find it indispensable for other reasons as well. *Thunder 7's Shorthand Glossary* expands your abbreviations instantly as you type—and *automatically* corrects your most persistent typos, like the dreaded "teh!" *Thunder 7's Thesaurus* puts useful synonyms at your fingertips. *Ghostwriter* saves your keystrokes to disk as a last-resort backup in case of power failures or crashes. Automatic *Smart Quotes* keep your "desktop publishing" looking like "professional typesetting."

Thunder 7—available *now* from your favorite Macintosh software source. Suggested retail price, $99.95.

★★★★½—*MacUser* Magazine
(Version 1.0 reviewed; version 1.5 shipping.)

Baseline
PUBLISHING

For more information, call
800-926-9677
or 901-682-9676, Facsimile: 901-682-9691

pieces of rejected mail every day. Every time I put out an issue (between two and four times a week), I get six or ten addresses that suddenly don't work. Some of them work again the next day, most of them just stop working for periods of time. Then a month later you get an angry message from somebody asking 'Why am I not getting RISKS?' "

SG: Can we trust computers?

PN: Read my book [which should appear in 1994]. It's very mixed in its conclusions. It gives a great deal of evidence why you shouldn't trust computers or the people who work with them, and yet it offers some hope. If we were able to know in advance what the requirements were – and we really had them correct, and we were able to design something that was consistent with those requirements, and we had really gifted people who could implement the system in such a way that was consistent with its design, and we had gifted people who would operate the system, remembering what the original requirements were, so they wouldn't compromise, and we had a user community that was fairly intelligent – then we might have a chance at having computer systems that we might be able to trust.... There are an awful lot of things that can go wrong.

SG: What's your favorite case of something going wrong?

PN: The ARPANET collapse of 1980. There was a combination of problems: You had a couple of design flaws, and you had a couple of dropped bits in the hardware. You wound up with a node contaminating all of its neighbors. After a few minutes, every node in the entire network ran out of memory, and it brought the entire network down to its knees. This is a marvelous example because it shows how one simple problem can propagate. That case was very similar to the AT&T collapse of 1990, which had exactly the same mechanism: A bug caused a control signal to propagate that eventually brought down every node in the network repeatedly. Both of those cases are beautiful examples of what can go wrong, because they involve a confluence of circumstances.

SG: In the first issue of *Software Engineering Notes* (1976), you wrote that "the state of the art of software engineering has been horrendous, but seems to be improving." Do you still think that?

PN: I think that it's still improving, but it hasn't lived up to expectations. It's very frustrating trying to deal with large systems. They never seem to come out the way they're supposed to.

SG: Why is software so hard to do right?

PN: Because there are so many things that can go wrong. If we look at one of the telephone collapses, there was a three- or four-line code patch that screwed up, and brought down large numbers of systems, including a number of airports. Everything just closed up because of one code bug that was installed without adequate testing. On the other hand, if you try to design something with no weak links you end up spending an enormous amount of your effort on redundancy and reliability. There are quite a few systems where over half of the code is devoted to redundancy management. A lot of that code never gets run in normal operations, so it is untested. The more complex the system is, the more likely it is to fail.

SG: So it's a Catch-22?

PN: Yes. You put in more complexity trying to add reliability, and that complexity itself is suspect, and hence more risky.

SG: Should programmers be licensed?

PN: A chapter in the book addresses that. I'm ambivalent. It's one of these double-edged swords. The licensing process is often lowest-common-denominator stuff. In order to get the certification process through, you end up with the minimum set of skills that people need to have. And yet, if they are dealing with life-critical systems, they need to have a tremendous amount of experience, creativity, imagination, a sense of what won't work, and a conservative attitude towards development. There is no way you can establish certification procedures that will ferret out those traits. My bottom line is that certification procedures would be wonderful if they could be made to work, but I don't think that they can be made to work – especially for critical systems.

SG: So what is the answer?

PN: The answer is to try to stick to simple systems. Do things as reliably as you can. Use intelligent people. You shouldn't have people with limited experience writing life-critical systems. I keep trying to put a positive spin on things, yet I'm very frustrated by the difficulties involved in getting something to work correctly. I've spent most of my professional career trying to make things work better. And yet, knowing that people can screw up, and hardware can screw up, and designs are typically flawed, and implementations are almost always flawed, leads me to the conclusion that it is a losing battle. So I'm kind of skeptical of some of the really critical uses of computers in life-critical situations. ■ ■ ■

Simson L. Garfinkel (simsong@nextworld.com) is a computer consultant, science writer, and a senior editor at Nextworld *magazine.*