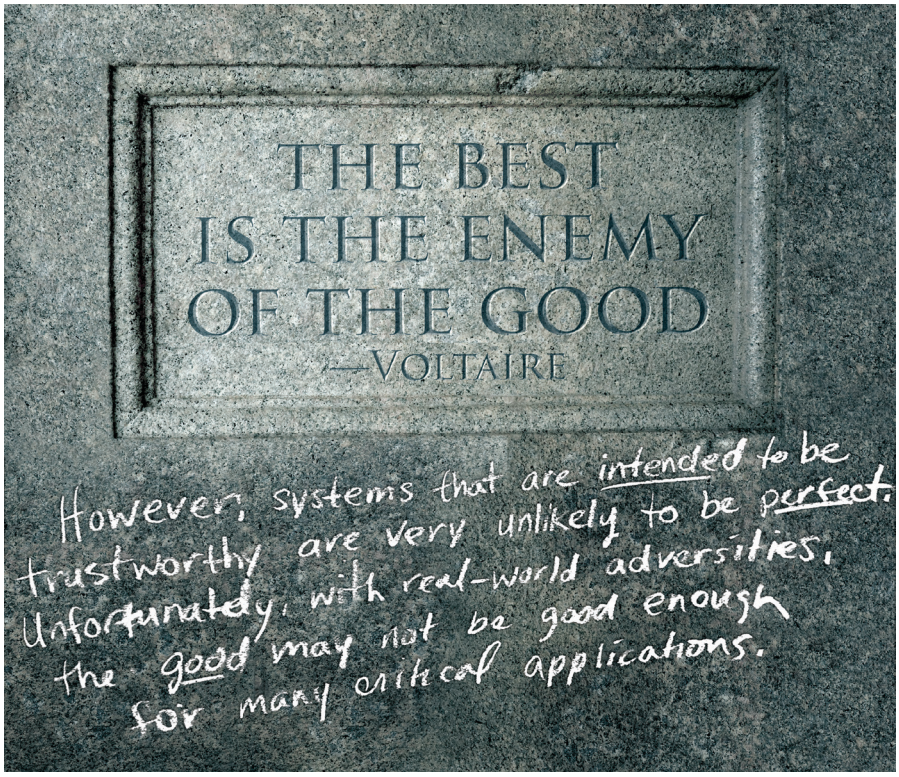
DOI:10.1145/2347736.2347746

Peter G. Neumann

Inside Risks
The Foresight Saga, Redux

Short-term thinking is the enemy of the long-term future.



THE PRIMARY MESSAGE of this column is that optimizing computer-related alternatives in the short term may be seriously detrimental in the long term, particularly where trustworthy behavior is essential—with respect to satisfying requirements for security, reliability, resilience, human safety, human usability, and so on. In general, we know from experience that it can be very difficult to retrofit systems with new implementations to make them trustworthy, especially if they were developed on insecure platforms without the benefit of security-aware development practices. Thus, a well-reasoned understanding of the trade-offs is essential before potentially sacrificing possible future opportunities in an effort to satisfy short-term goals. One complicating factor is that much more knowledge of the past and the present—and appreciation of the effects of possible futures—is needed to intelligently making such trade-offs.

This column is intended for a diverse audience relating to computer-related risks, including notably researchers, system developers, curricula creators, teachers, politicians, and many others. The starting point is that almost everything that we necessarily must depend on relating to information systems, networks, and national infrastructures is today for the most part riddled with flaws that can be exploited or triggered as a result of willful or accidental misuse, hardware and software failures, environmental hazards such as power outages and earthquakes, and so on. Similarly, there is often an inherent dependence on system administrators, with the hope that they are nearly infallible.

For example, software vendors tend to release new system versions with known bugs, realizing that they can economize by letting users and application developers discover the bugs! Critics might argue that this can lead to problems that could be avoided if the designers analyzed long-term consequences—including anticipating attacks exploiting serious vulnerabilities. In addition, some software tends to outlive hardware, with flawed software sometimes lingering on. Worse yet, bad interfaces tend to persist—due to commitments on backward compatibility. Furthermore, easy programmability and a desire for lowest common denominators often trumps the advantages of functional abstractions, strong typing, controlled memory management, formal reasoning, and other somewhat more farsighted approaches.

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

As another example, hardware architectures that enabled trustworthy layered enforcement of fine-grained least-privilege access policies are no longer in the mainstream, which tends to limit the trustworthiness that can be achieved in software. Also, problematic hardware instructions tend to survive, again for backward compatibility.

The resulting shortfalls with respect to desired system, network, and enterprise behavior can be very serious. That has been noted repeatedly in the preceding 227 Inside Risks columns (including a similarly titled column[5]), and does not need much further elaboration here.

Some inspiration for writing this column came from the ACM Turing Centenary Celebration this past June, which attempted to look back at the past and to consider what might happen in the future. The talks by the Turing laureates and other invited participants ranged from near to far into both the past and the future—reminding us of some of the laureates' important past contributions, while at the same time giving diverse perspectives on the future.

Consider some of the general guidance that has emerged from our collective pasts. This may seem similar to earlier Inside Risks columns, but bears repeating because it is not widely observed in practice.

Requirements. We should anticipate the long-term needs that a system or network of systems must satisfy, and plan the development to overcome potential obstacles that might arise, even if the initial focus is on only short-term needs. This might seem to be common wisdom, but is in reality quite rare. Common requirements for security, reliability, fault tolerance, resilience, diagnostic ability, adaptability, human safety, interoperability, long-term evolvability, trustworthiness, and assurance evaluations are generally much too weak. Furthermore, highly distributed control with highly networked or cloud-dependent systems demands much greater foresight. Also, refining requirements on the fly often causes serious development problems.

System development. We can gain significantly by using effective design methodologies, basic principles, well-reasoned system/network architectures, horizontal (modular) and vertical (layered) abstraction with encapsulation and strong typing, predictable composability, use of formal methods for assurance where most effective, suitable choices of languages for requirements, specifications, programming, and so on—compatible with the sophistication of the requirements and the expertise of the developers.

There is much to be gained from farsighted thinking that also enables short-term achievements.

Research. Solving problems more generally with preplanned evolution, rather than just barely attaining short-term requirements, can be very advantageous. With some foresight and care, this can be done without losing much efficiency. Often a slightly more general solution can prove to be more effective in the long run. There is much to be gained from farsighted thinking that also enables short-term achievements. Thus, it seems most wise not to focus on one without the other. Some new clean-slate approaches are emerging in response to the needs for much greater system and enterprise trustworthiness, as are executable hardware-software co-design languages (for example, see Dave[1]). Such efforts have long-term goals, but can also have significant short-term results—especially in an ongoing formally based hybrid capability-based hardware-software architecture,[6,7] which allows legacy software to coexist securely with newly developed highly trustworthy hardware-software.

Roles of science and engineering. Computer science has evolved into a very useful collection of scientific principles and methods, with significant advances in many areas—although the use of systemwide metrics and evaluations of trustworthiness still have significant room for advances. On the other hand, the so-called field of software engineering is still sorely lacking in engineering foundations and discipline, and therefore unlike well-established engineering fields. Theoretical bases and supporting tools can be very helpful to engineering practice, in simplifying and analyzing complex systems, and especially when it comes to long-term thinking. Metatheories enhancing the predictable composition of requirements, subsystems, and measures of trustworthiness enabling evaluations of emergent properties of entire systems would be extraordinarily valuable in facilitating long-term thinking.

Experimentation. Experimental computer science and engineering tend to lurk in the background somewhat as orphan stepchildren. Exploration of alternative system architectures, easily reconfigured testbeds, parameterizable symbolic analyses and simulations, layered composable evaluations, and so on would all be very supportive of long-term thinking.

Education. We must do better at teaching principles and theoretical bases, not just how to write programs or use tools for development and analysis. Thinking to support long-term advances requires much more than rote learning, lowest-common denominator standards and academic evaluation criteria, and short-sighted elimination of historically relevant departments deemed of less commercial value. In a down economy, farsighted education remains a most essential need for the future.

People. Accepting that to err is human, and that malicious misuse is here to stay on an increasingly more pervasive scale, we need to spend more effort on anticipating the potential shortcomings of system interfaces for human-system interactions, and design systems that are much more people tolerant and people resilient. The burden on today's system administrators is enormous, but could also be greatly reduced by the presence of more long-term thinking in system design and implementations.

Cognitive vs. subconscious thinking. A recent book by Daniel Kahnemann[2] revisits some of the earlier studies of left-brain (logical, linear, methodical) versus right-brain (intuitive, subconscious, out of the box) thinking. Our educational systems tend to prod the former, while in some cases neglecting the latter. Kahnemann's fast thinking (more or less right brain) tends to be checked or modulated by slow thinking (more or less left brain). What is important in the present context is that long-term thinking inherently requires a well-integrated combination of both[4] as applied to computer system development). A holistic balance of human intelligence, experience, memory, ingenuity, creativity, and collective wisdom, with slow and fast thinking, is extremely valuable in exploring the trade-offs between short-term gains and long-term potentials within some sort of holistic big-picture foresight.

Innovation. New computing technologies tend to introduce new security vulnerabilities, as well as reintroduce earlier ones. This has occurred over many decades, with buffer-overflow attacks, man-in-the-middle attacks (for example, Needham-Schroeder), distributed denial-of-service attacks, physical attacks on crypto and mobile devices, spam, and both large-scale and targeted social engineering. When security problems continually recur, it might be time to do something different. Perhaps this tendency can be overcome with formally based architectures and developments.

High assurance. Formal methods have always had enormous promise, but have been very difficult to use. As they become more integrally and seamlessly embedded in the development process and more diverse (encompassing a variety of solvers), they may finally become more viable. The biggest remaining challenge may involve designing and analyzing systems that are composed from components that themselves have been thoroughly analyzed and whose analyses are themselves composable.

Reliance on the marketplace. Market success in hardware and software tends to produce winners that may not be adequately trustworthy. The alternative of clean-slate developments is generally unpopular and difficult to pursue in commercial enterprises, but currently reborn in several DARPA research and development programs such as CRASH (Clean-slate design of Resilient, Adaptable, Survivable Hosts) and MRC (Mission-oriented Resilient Clouds).

Interactions with other disciplines. All of the considerations noted here can be much more effective if motivated by real applications such as medical information systems, telerobotic surgery systems, real-time control systems, low-power multipurpose mobile devices, and so on. The application of long-term thinking to such applications is essential to ensure satisfaction of their critical requirements. An earlier article[3] characterizes the holistic nature of energy, agriculture, and health care—each of which requires a deeper understanding of the need for long-term thinking—and contrasts them with various computer-related risks issues.

A few illustrative requirements:

Computer system security and integrity. The relative ease of perpetrating certain attacks such as viruses, worms, and exploits such as Conficker and Stuxnet suggests that long-term concerns have been largely ignored. With particular attention to critical national infrastructure systems, we seem to have arrived at lowest-common-denominator systems and have had to live with them, in the absence of better alternatives. The standards for acceptable levels of security and best practices are typically much too simplistic and basically inadequate. Relevant efforts of various research and development communities seem to be largely ignored.

Computer-aided elections suffer from all of the aforementioned difficulties, plus more. The proprietary nature of commercial systems is a serious obstacle to meaningful oversight, as is the lack of constructive integrity throughout the entire election cycle, the lack of incisive audit trails, extensive opportunities for insider misuse of technology, and manipulation of the externalities—for example, registration, authentication, disenfranchisement, and so on.

The financial crises of the past few years present another example in which

the almost total absence of realistic long-term thinking and oversight contributed to worldwide economic problems. Optimizing for short-term gains often tends to run counter to long-term success (except for the insider investors, who having taken their profits have little interest in the more distant future). Although this may not seem like a typical Inside Risks case, it is certainly illustrative of the main theme here.

The Future

The ACM Risks Forum has helped dramatize past experiences with the effects of design faults, system security vulnerabilities, system failures and errors, and the pervasive roles of people throughout. Previous columns have highlighted the importance of understanding these experiences and applying them diligently in the future.

Social engineering (exploiting human weaknesses) is a significant factor in system penetrations, inadvertent insider misuse, and above all the spread of malicious malware and other forms of malicious misuse combined with the existence of vulnerable systems whose exploitation permits email and Web-based scams to facilitate online identity fraud and other forms of malfeasance. For example, innocently clicking on a seemingly legitimate link is a common failing. Ultimately, no matter how trustworthy individual hosts, servers, and networks might become, users will need much greater help in detecting and avoiding scams and deception. For example, human frailties such as greed, gullibility, and obliviousness to the risks will clearly persist. Thus, widespread computer literacy is an urgent goal, involving both short-term and long-term aspects. Nevertheless, designing systems, networks, and applications that wherever possible effectively mask the overall complexity and concerns for trustworthiness must also be a long-term goal.

Conclusion

Although this column has barely scratched the surface of an iceberg-like collection of problems, it addresses the need for urgently developing compelling logical and realistic justifications for embedding long-term thinking into our planning. Commodity hardware/software aims at the mass market; on the other hand, what is needed for certain critical applications such as national infrastructures, secure and resilient cloud servers, and so on is the existence of meaningfully trustworthy networked systems. Thus, there is a major disconnect that requires some long-term thinking to overcome. Unfortunately, the real-world arguments for short-term optimization are likely to continue to prevail unless significant external and internal efforts are made to address some of the long-term needs. C

New computing technologies tend to introduce new security vulnerabilities, as well as reintroduce earlier ones.

References

1. Dave, N. A Unified Model for Hardware/Software Codesign. Ph.D. thesis, MIT, Cambridge, MA, 2011.
2. Kahnemann, D. *Thinking, Fast and Slow.* Farrar, Strauss and Giroux, 2011.
3. Neumann, P.G. Holistic systems. *ACM ACM SIGSOFT Software Engineering Notes 31*, 6 (Nov. 2006), 4–5; http://www.csl.sri.com/neumann/holistic.pdf.
4. Neumann, P.G. Psychosocial implications of computer software development and use: Zen and the art of computing. In *Theory and Practice of Software Technology*, D. Ferrari, M. Bolognani, and J. Goguen, Eds., North-Holland, 1983, 221–232.
5. Neumann, P.G. The foresight saga. *Commun. ACM 50*, 9 (Sept. 2006); http://www.csl.sri.com/neumann/insiderisks06.html#195.
6. Neumann, P.G. and Watson, R.N.M. Capabilities revisited: A holistic approach to bottom-to-top assurance of trustworthy systems. Fourth Layered Assurance Workshop, U.S. Air Force Cryptographic Modernization Office and AFRL (Austin, TX, Dec. 2010); http://www.csl.sri.com/neumann/law10.pdf.
7. Watson, R.N.M. et al. CHERI: A research platform deconflating hardware virtualization and protection, runtime environments/systems, layering, and virtualized environments. RESoLVE workshop (London, U.K., Mar. 3, 2012); http://www.csl.sri.com/neumann/2012resolve-cheri.pdf.

Peter G. Neumann (neumann@csl.sri.com) chairs the ACM Committee on Computers and Public Policy and moderates the ACM Risks Forum (http://www.risks.org). He is very grateful to his committee members for their long-standing incisive feedback on Inside Risks columns—including this one!

Calendar of Events

October 15–18
31st International Conference on Conceptual Modelings,
Florence, Italy,
Contact: De Antonellis Valeria,
Email: valeria.deantonellis@ing.unibs.it

October 15–19
ACM SIGUCCS Annual Conference,
Memphis, TN,
Sponsored: SIGUCCS,
Contact: Carol Rhodes,
Phone: 812-856-2007,
Email: csrhodes@indiana.edu

October 15–19
Conference on Systems, Programming, and Applications: Software for Humanity,
Tucson, AZ,
Sponsored: SIGPLAN,
Contact: Gary T. Leavens,
Phone: 407-823-4758,
Email: leavens@eecs.ucf.edu

October 21–24
Conference on Systems, Programming, and Applications: Software for Humanity,
Tucson, AZ,
Sponsored: SIGPLAN,
Contact: Gary T. Leavens,
Phone: 407-823-4758,
Email: leavens@eecs.ucf.edu

October 22–24
The 14th International ACM SIGACCESS Conference on Computers and Accessibility,
Boulder, CO,
Sponsored: SIGACCESS,
Contact: Matt Huenerfauth,
Phone: 646-639-3815,
Email: matt@cs.qc.cuny.edu

October 22–26
International Conference on Network and Service Management,
Las Vegas, NV,
Contact: Medhi Deep,
Email: dmedhi@umkc.edu

October 22–26
5th International Conference of Security of Information and Networks,
Jaipur, India,
Contact: Manoj Singh Gaur,
Email: gaurms@gmail.com