Grand Challenges Conference Application

Making Computing Tractable for Everyday People
A Grand Challenge for Computer Science and Engineering

Mary Shaw
School of Computer Science
Carnegie Mellon University
mary.shaw@cs.cmu.edu

Many discussions of Grand Challenges focus on heroic challenges -- technological breakthroughs that provide new capabilities for discrete, newsworthy tasks such as decoding the human genome, forecasting weather, communicating at extremely high rates, creating sophisticated images, and the like. These challenges typically have explicit success criteria and a distinct product. Often neglected, but equally important, are challenges with softer edges and widespread impact. I propose a challenge of this second type: *Make computing and information technology tractable for everyday people*.

Despite extraordinary progress in making computers intelligible to people who are not computer professionals, it remains very difficult for an everyday person to configure a computer system, to select and install an individually-tailored collection of applications and data, to establish and maintain good communications, to tailor an application to individual needs, and -- especially -- to recover gracefully when anything at all goes wrong. True, a consumer can order a completely-configured package from a retailer, but this yields a one-size-fits-many solution. Further, there's little help afterwards with backup, security, critical patches, reconfiguring communications, recovery after failure, or other system administration problems that arise regularly (one could say "routinely", but this should not be routine).

Currently,

- Computers and software have a rapid release/obsolescence cycle, and differences between versions are often hard to understand and manage.

- The internet provides a wealth of useful resources, but with highly variable accuracy and dependability; an increasing share of everyday activity relies on these resources, and security (including privacy) is a serious and growing issue.

- Legislation affecting public and private computing is proposed and debated with scant appreciation of the technical opportunities and risks.

- There is little widespread public understanding of the associated issues of trust and protection, dependability or product selection and correspondingly little capability for everyday people to control their own electronic destinies.

The result is that raw computing and communication power is cheap and abundant, but it is not packaged or delivered in such a way that everyday people can *evaluate* the available resources, *select* those that meet their needs, *configure* a system with resources from diverse sources, *adapt* resources to meet their individual needs, *produce* well as consume content, and know how much to *trust* individual resources and compositions of resources.

Industry is not likely to solve this problem. Their interests are aligned with treating the public as consumers of their information content, preferably locked in to particular standards, rather than with empowering the public to create, adapt, and even export their own custom solutions.

Nor is there hope in the computer-as-appliance (or appliance-as-computer) model. Here the premise is that computers will disappear into everyday objects and we won't have to think about them. This is attractive as far as it goes, but we see increasingly that those everyday objects can useful interact with each other and their owners, and denying the owner any discretion about configuration and control gives up much of the potential of the technology.

A key underlying problem is our growing dependence on complex computing systems composed by incremental evolution from parts that we only partially understand; important properties of these systems are difficult to predict.

Let's take as a grand challenge the task of civilizing the electronic frontier sufficiently to enable everyday people to evaluate, select, configure, adapt, and produce their computing and information resources, and to know how much they can trust the result. In this way we may secure the benefits of the technology to the advantage of all our citizens.

What will this entail? A great deal, including research opportunities that depart from the current common agenda

Evaluate and select
Current approaches to evaluating software emphasize reasoning from first principles. However, this style of reasoning for systems assumes more precise specifications of components than are available in practice; further, it does not deal well with emergent properties of systems. It is unlikely that bottom-up well-founded discrete reasoning will suffice for the complex systems of interest even in the hands of experts, let alone that everyday people will be able to apply it. Aggregate approaches have long been discouraged by the argument that our phenomena are discrete rather than continuous, but I don't find that compelling -- gas laws describe the aggregate behavior of large numbers of molecules, insurance creates pools from large numbers of distinct individuals, and telephony does load analysis aggregating over large numbers of callers, who are individually as discrete and discontinuous as anything in software. *Let us, then, explore the possibility of aggregate models of reasoning about complex software systems*.

Configure and adapt
Current software development models are essentially closed-shop models in which the developer is assumed to have control over the development process and associated components. This leads to a model of consumer software in which applications only interact well if the manufacturer has chosen to support that interaction via explicit integration or external standards. However, the wealth of resources available on the internet provide an irresistible opportunity for everyday people to synthesize diverse resources to serve their own needs. This sort of synthesis is radically different from closed-shop development – resources are under-specified, they may change without notice, they often do not interoperate gracefully, they may be used for some incidental functionality rather than their intended purpose. The results are much more like coalitions than like fully integrated systems. *Let us explore ways for everyday people to synthesize their own applications from independent resources, and to understand the dependability of the result.*

Produce

Current business models are driving the content model of the Internet to an asymmetry in which most content is controlled and distributed by large suppliers, with individuals acting principally as consumers and occasional originators of incidental content for personal use. These models are not sympathetic to the idea that everyday consumers could synthesize significant new content by refining, remixing, or elaborating on existing resources. Intellectual property concerns are a major impediment to such derivative works, but technology offers the possibility of enabling micro payments for micro uses, thereby (perhaps) providing net benefit to the intellectual property owners as well as opportunity to everyday people to produce new content. *Let us explore ways to restore the original copyright concept of a limited monopoly in exchange for public benefit and develop business models that encourage, rather than discourage, everyday producers of creative adaptations.*

Trust

Current dependability research focuses on critical systems in which the objective is obtaining the highest dependability possible and the cost of failure is large enough to justify substantial investment. Current fault-tolerance models distinguish normal, failure, and sometimes degraded states and consider transitions among these states. Everyday software, however, has more relaxed and situation-specific requirements. First, the definitions of degradation and failure may be application-specific. Second, everyday dependability may be able to exploit types of evidence that are too informal to be considered for high-dependability analysis. Third when components are under-specified, specifications of fine structure are unlikely to be better. A possible alternative to state-based dependability is to define gradients of quality, make situation-specific judgments about acceptable levels, and add homeostatic mechanisms to implementations that strive to improve quality whatever its level. A possible alternative to classis fault tolerance is to detect failure and provide compensation through insurance or warranty. *Let us explore ways to establish how much trust an everyday user needs to place in a system (or coalition), to determine whether a system is sufficiently dependable for a specific situation, and to include compensation for failure as an alternative to prevention.*

Biography

Mary Shaw is the Alan J. Perlis Professor of Computer Science, Co-Director of the Sloan Software Industry Center, and member of the Institute for Software Research International and the Human Computer Interaction Institute at Carnegie Mellon University. She has been a member of this faculty since completing the Ph.D. degree at Carnegie-Mellon in 1972. From 1992 to 1999 she served as the Associate Dean for Professional Education. In 1997-98 she was a Fellow of the Center for Innovation in Learning. From 1984 to 1987 she served as Chief Scientist of CMU's Software Engineering Institute. She had previously earned a B.A (cum laude) from Rice University and worked in systems programming and research at the Research Analysis Corporation and Rice University.

Her research interests in computer science lie primarily in the areas of programming systems and software engineering, particularly software architecture, programming languages, specifications, and abstraction techniques. Particular areas of interest and projects have included software architectures (ORCa, Vitruvius, UniCon), technology transition (SEI), program organization for quality human interfaces (Descartes), programming language design (Alphard, Tartan), abstraction techniques for advanced programming methodologies (abstract data types, generic definitions), reliable software development (strong typing and modularity), evaluation techniques for software (performance specification, compiler contraction, software metrics), and analysis of algorithms (polynomial derivative evaluation).

She has participated in developing innovative curricula in Computer Science from the introductory to the doctoral level, including the Immigration Course (1971), a complete undergraduate curriculum design (1984), a system of professional masters programs (1993-8), and courses in data abstraction (1979), software engineering (1990), software architecture (1995), and software engineering research (2000).

Dr. Shaw is an author or editor of seven books and more than one hundred forty papers and technical reports. In 1993 she received the Warnier prize for contributions to software engineering. She is a Fellow of the Association for Computing Machinery (ACM), the Institute for Electrical and Electronics Engineers (IEEE) and the American Association for the Advancement of Science (AAAS). She is also a member of the Society of the Sigma Xi, the New York Academy of Sciences, and a member emeritus of Working Group 2.4 (System Implementation Languages) of the International Federation of Information Processing Societies. She is a past member of the Computer Science and Telecommunications Board and DARPA ISAT study group. In addition, she has served on a number of advisory and review panels, conference program committees, and editorial boards.