Three Great Challenges for Half-Century-Old Computer Science

FREDERICK P. BROOKS, JR.

University of North Carolina at Chapel Hill

1. *Quantification of Structural Information*

Shannon and Weaver [1949] performed an inestimable service by giving us a definition of information and a metric for information as communicated from place to place, negentropy. This metric,

$$\mathrm{H} = -\Sigma\, p_i \ \mathit{log}\ p_i$$

and the associated concept of noise, have proved rich sources of further theory and of applications galore.

We have no theory, however, that gives us a metric for the information embodied in structure, especially physical structure. We know that an automobile is a more complex structure than a rowboat. We cannot yet say it is x times more complex, where x is some number. Yet we know that the complexity is related to the Shannon information that would be required to specify the structures of the car and the boat.

I consider this missing metric to be the most fundamental gap in the theoretical underpinnings of information science and of computer science. Recent developments, however, make it timely to address it. The fundamental biological structures are rich enough to repay study and yet simple enough that there is hope of making real progress on an information theory of structure. (Rowboats and automobiles are much too hard.) The coding of genetic information by DNA is apparently simple enough that it can be handled with existing communication theory. The folded structure of proteins is not. Entropic and energetic considerations are necessary, but not yet sufficient, for explaining and predicting that structure, even after the amino acid sequence is known. Yet proteins are relatively simple structures.

A young information theory scholar willing to spend years on a deeply fundamental problem need look no further.

2. *Software Estimation*

Given specific functional, reliability, and performance specifications for a software system, we do not yet know how to estimate the effort required to build it. Nor, given the product specifications and a quite specific description of the skills of a particular team, can we reliably estimate how long it will take to grow the product.

The first book on computer software, by Wilkes et al. appeared in 1951. Now, a half century later, there have been many, many advances in concept and technique. Not the least of these is Boehm's monumental *Software Engineering Economics* [Boehm 1981] the COCOMO model set forth therein, and the subsequent software economic models stimulated by that work. Nevertheless, we still don't know what we are doing, unless it is very similar to something we have done before.

The challenge is to make software engineering as predictable a discipline as civil or electrical engineering. I still do not expect any radical breakthrough, any silver bullet, to solve this problem [Brooks 1986]. But the accretion of many contributions has already made much progress, and I believe continued careful research, ever validated by real practice, will bring us to that goal.

3. *User Interface Design for Computer Systems*

Today, the design of the user interface for an operating system or an application program is still an art, not yet an engineering discipline.

Much research has been done on human perception, human cognition, and on the human factors of output from the mind to the bodily effectors. On the other side of the human—computer interface, much research and development has advanced interface technologies: computer graphics, sound synthesis, speech synthesis, speech recognition, and haptics.

Substantial as these research corpora are, we seem still to lack any systematic or disciplined way of integrating them when designing a new hardware-software-user interface. We do not have reliable ways even of predicting whether a proposed specific interface design will be good, that is,

—Intuitive for the novice;
—Efficient in perception and motion for the expert;
—Robust under misuse;
—Facilitating in recovery from cognitive or manipulative mistakes;
—Helpful in diagnosing errors and suggesting corrective action;
—Rich in incrementally learnable functions, like the alphabetical shortcuts on the Mac interface.

The challenge is to integrate the relevant but disparate bodies of knowledge into a discipline of design.

ACKNOWLEDGMENTS. Conversations with colleagues over the years have stimulated these ideas. I am especially indebted to Professor Jurg Nievergelt, whose thoughts on these challenges especially resonated with mine.

REFERENCES

BOEHM, B. 1981. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, N.J.

BROOKS, F. P. 1986. No silver bullet—Essence and accident in software engineering. *Information Processing 1986*. Reprinted as Chapter 16 of Brooks, F. P. 1995. *The Mythical Man-Month, Anniversary Edition*, Addison-Wesley, Reading, Mass.

SHANNON, C. E., AND WEAVER, W. 1949. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Ill.

WILKES, M. V., WHEELER, D. J., AND GILL, S. 1951. *The Preparation of Programs for an Electronic Digital Computer, with Special Reference to the EDSAC and the Use of a Library of Subroutines.* Addison-Wesley, Reading, Mass.